UIC Search Engine

Indra Sai Kiran Valluru

University of Illinois at Chicago, Department of Computer Science, Chicago, Illinois 60607 USA

Abstract: This report serves as the final course project for the University of Illinois at Chicago's CS582: Information Retrieval course for the Fall 2022 term. The project's objective is to create and put into operation an online search engine for the UIC domain. A web crawler, preprocessing, and indexing of web pages, as well as an IR system that uses the vector-space model to retrieve web pages pertinent to a user query, are all included in the search engine.

1. Software

All functions are modularized in the software's Python(3.8) design so they can be further enhanced for usage in other projects and future work. It takes the search engine about three hours to crawl, gather, and preprocess 7000 webpages from the UIC domain. The 'PickleFiles' folder contains all the pickle files required to run the function, allowing you to do so without first browsing the UIC domain. The search engine can be launched immediately by running the search query.py script file from the terminal. However, python scripts are also offered to carry out web crawling and website preprocessing.

1.1. Web Crawler

The program uic crawler.py runs a web crawler. Crawling is limited to the UIC domain (https://www.uic.edu/) after starting at the UIC-CS department page (https://www.cs.uic.edu/). The UIC-CS page serves as the root node for the crawler's breadth-first search (BFS) approach, which uses a First in First Out queue to record the URL links it will visit next. The HTML content is downloaded and parsed using the Beautiful Soup library for each web page that is browsed, and all links that are present in the page are extracted.

Only if a link is found to be in the UIC domain, does not contain an unnecessary file extension, and is not already in the queue is it added to the FIFO queue.

Since a deque may remove elements from the head in constant time, employing a deque to implement the FIFO queue improves crawler performance. In contrast, a list can only delete elements in constant time from the tail. Deque was chosen because, in the BFS technique, links are popped from the front of the queue.

Holographic imaging in the soft X-ray (SXR) and extreme ultraviolet (EUV) have been demonstrated in several experiments realized using EUV/SXR lasers and synchrotron sources. These include the first realization of soft X-ray laser holography at Lawrence Livermore National Laboratory using a large laser facility, and the holographic recording of biological samples and sub-micron structures using soft X-ray radiation from synchrotrons, among other experiments. A key idea in these experiments is to use coherent short wavelength illumination to achieve a spatial resolution beyond the reach of visible light.

Because they required a long time to download and did not link to legitimate web pages that could be parsed and processed, 22 file extensions were flagged as irrelevant and ignored for web traversal. The following file extensions are disregarded: ".pdf," ".doc," ".docx," ".ppt," ".pptx," ".xls," ".xlsx," ".css," ".js," ".aspx," ".png," ".jpg," ".jpeg," ".gif," ".svg," ".ico," ".mp4", ".avi," ".tar," ".gz," .

Before the connections were added to the FIFO queue, they were processed. Every URL added

to the queue now points to a different web page because the query parameters indicated by ”?” and the intra-page anchors indicated by ”” have been deleted. If the crawling limit is reached or the FIFO queue becomes empty, the BFS traversal of links is stopped. The crawling of 7000 pages took close to three hours.

Every website page is downloaded to the ”FetchedPages” folder. Links to downloaded and processed web pages are saved in dictionaries. The name of the file that contains the link’s downloaded content serves as the key for each link. The dictionary is pickled and stored on disk for future use.

1.2. Preprocessing

The preprocessor.py script is used to preprocess the downloaded web pages. A Beautiful Soup object is created from scratch for each page that is downloaded. All of the content on the webpage is first extracted, and any text that is never displayed in a browser is subsequently disregarded. This is accomplished by excluding the text from the HTML tags ”script,” ”meta,” and ”style.” After that, the retrieved text is subjected to the usual preprocessing procedures in order to tokenize it, including the elimination of any punctuation, digits, and stop words. Each token is stemmed using the Poster Stemmer program from the nltk package. Any remaining stop words and tokens with a length of 1 or 2 characters were eliminated after stemming. Each downloaded webpage’s tokens are saved in a dictionary.

The text preparation is finished, tokens are created, and an inverted index is calculated. It is a dictionary among dictionaries, this inverted index. The dictionary of websites where each token appears is the equivalent value for each key in the vocabulary of the corpus of web pages. The internal dictionary’s key is represented by the downloaded web page’s filename, and its matching value is represented by the number of tokens on that specific page.

It takes roughly 20 minutes to preprocess each retrieved web page and calculate the inverted index. The inverted index and the dictionary of tokens are therefore saved to disk for subsequent use, just like the crawler’s dictionary of all pages crawled.

1.3. Vector Space Model

The search query.py program is used to process user queries and retrieve the UIC domain’s most pertinent web pages.

The inverted index, tokens of webpages, and all pickle files for links crawled are first unpickled and extracted.

The Inverse Document Frequency for each webpage- token pair is calculated using the inverted index and stored as a separate dictionary. The frequency of the most frequent token in each webpage is calculated, which will be required to calculate the Term Frequency of each token for its corresponding webpage. Then finally, the TF-IDF value for each token in the corpus is calculated and stored as a dictionary of dictionaries like the inverted index.

To create document vectors for each webpage, the document length of each web page is determined. The user enters a query, which is preprocessed and tokenized just like the tokens on the webpage. The cosine similarity between each downloaded webpage and the query is then calculated. The viewer is then shown a list of the URLs in decreasing order of cosine similarity scores.

2. Challenges

Since I had never done web crawling before, it took me some time to grasp the fundamentals of it. Compared to the rest of the project, it took a lot of time because we had to crawl 7000 pages, and there were many strange URLs that needed to be handled when they were found in order to avoid edge situations.

With limited Python knowledge, I found it difficult to design the software, divide it into various modules based on its functionality, and add Object Oriented programming to improve the

readability of the code.

I had a lot of problems when I first created the engine with the basic crawler, including the inverted index table having a large number of javascript code phrases.

The Beautiful Soup object was returning a wide variety of terms from each downloaded webpage while parsing and extracting the text from them. The vector space model's performance may have been harmed by these non-English words because they were producing trash tokens.

Since I am preprocessing everything to avoid delays, I had to rerun the crawling part multiple times in order to try with other similarities. I could have designed the project to avoid this

3. Weighting Scheme and Similarity Measure

3.1. Weighting Scheme

The straightforward TF-IDF of words on webpages was the weighting technique employed in the project. Even though it is a fairly plain and uncomplicated weighting scheme, it has shown to be one of the most successful and efficient weighting methods over time. This project called for a fairly objective accounting of the significance of tokens in each document, which was ideal.

3.2. Similarity Measure

The Cosine Similarity was the similarity metric used to rank pertinent webpages. The query's length as well as the document's length are taken into account by cosine similarity. Furthermore, the tokens that are absent from the query or the document have no impact on their cosine similarity. Typically, the query is brief, and as a result, its vector is very sparse, making it possible to calculate similarity scores quickly.

3.3. Alternate Similarity Measures

In addition to cosine similarity, other similarity metrics include the inner product, dice coefficient, and jaccard coefficient. The Inner Product does not take the length of the document or query into account, hence when employed, it does not produce good results. The Inner Product metric has been modified to create the Cosine Similarity measure. The other similarity measurements for carrying out this project have not been explored by me. Rank relevant websites. Cosine similarity takes into account the length of both the document and the query. Furthermore, their cosine similarity is unaffected by the tokens that are missing from either the query or the document. The query is often short, and as a result, its vector is highly sparse, allowing for speedy calculation of similarity scores.

4. Evaluation

The IR system was evaluated using five custom queries and the top 10 webpages retrieved for each query.

4.1 Query: Information Retrieval

```
 ***  UIC Web Search Engine  ***

Enter search query: Information Retrieval


1 https://cs.uic.edu/profiles/cornelia-caragea
2 http://cs.uic.edu/profiles/cornelia-caragea
3 https://it.uic.edu/news-stories/9-2-9-5-technology-solutions-migrating-red-website-infrastructure
4 https://engineering.uic.edu/letter-from-the-dean-spring-2020
5 https://library.law.uic.edu/circulation-services.php
6 https://transportation.uic.edu/abandoned-bicycle-removal
7 https://financialaid.uic.edu/aid-process/verification
8 https://registrar.uic.edu/student-records/transcripts
9 https://registrar.uic.edu/student_records/transcripts
10 https://cs.uic.edu/news-stories/students-in-break-through-techs-cs-100-course-focus-on-networking-in-final-week-of-class
```

Each retrieved web page has the query terms in them.
Precision = 1.0

4.2 Query: Course Catalog

```
 ***  UIC Web Search Engine  ***

Enter search query: Course catalog


1 http://catalog.uic.edu
2 https://catalog.uic.edu
3 https://catalog.webhost.uic.edu/ucat/cat1113archive
4 https://catalog.uic.edu/ucat/archive-links
5 https://catalog.webhost.uic.edu/ucat/cat1315archive
6 https://catalog.uic.edu/search
7 http://catalog.uic.edu/gcat/degree-programs
8 https://catalog.uic.edu/gcat/degree-programs
9 https://catalog.webhost.uic.edu/ucat/catalog/pdf/index.html
10 http://catalog.uic.edu/ucat
```

All the retrieved results are about the course offerings available. The precision for this query is 1.0

4.3 Query: Assistantships

```
 ***  UIC Web Search Engine  ***

Enter search query: Assistantships


1 http://grad.uic.edu/assistantships
2 https://childrenscenter.uic.edu/employment
3 http://grad.uic.edu/graduate-funding-overview
4 https://grad.uic.edu/funding-awards/graduate-funding-overview
5 https://grad.uic.edu/graduate-funding-overview
6 https://hr.uic.edu/hr-staff-managers/hiring/grad-assistants
7 http://financialaid.uic.edu/types-of-aid/waivers-assistantships
8 https://ahs.uic.edu/kinesiology-nutrition/admissions-and-programs/ms-in-kinesiology/before-you-
9 https://socialwork.uic.edu/academics/phd-in-social-work/phd-financial-aid-information
10 https://ahs.uic.edu/applying/tuition-and-aid
```

All results are related to some form of assistantships or financial aid. The precision for this query is 1.0

4.4 Query: AI Research

```
 ***  UIC Web Search Engine  ***

Enter search query: AI Research


1 https://dentistry.uic.edu/dentistry-research/research-training-and-education
2 https://dentistry.uic.edu/dentistry-research
3 https://undergradresearch.uic.edu/for-students
4 https://undergradresearch.uic.edu/programs
5 https://undergradresearch.uic.edu/for-faculty
6 http://uic.edu/research
7 https://uic.edu/research
8 https://dentistry.uic.edu/dentistry-research/student-research-center
9 https://dentistry.uic.edu/dentistry-research/dentistry-research-areas
10 https://today.uic.edu/uic-research-update
```

Each retrieved web page has some(research) query terms in them.
Precision = 1.0

4.5 Query: Career Fairs

```
 ***  UIC Web Search Engine  ***

Enter search query: Career Fairs


1 https://ecc.uic.edu/employers/career-fairs
2 https://researchguides.uic.edu/copyright/fairuse
3 https://careerservices.uic.edu/students/career-programs-events
4 https://law.uic.edu/experiential-education/clinics/fairhousing/attorneys
5 https://law.uic.edu/experiential-education/clinics/fairhousing/letter-seng-bethel
6 https://law.uic.edu/experiential-education/clinics/fairhousing/students
7 https://engineering.uic.edu/news-stories/september-career-fair-helps-kick-off-employer-recruitment
8 https://law.uic.edu/experiential-education/clinics/fairhousing/fair-housing-tester
9 https://engineering.uic.edu/news-stories/engineering-career-fair-provides-access-to-success
10 https://law.uic.edu/experiential-education/clinics/fairhousing
```

Of all the webpages retrieved seven results are related to career fair events, tips, and descriptions career fairs.
Precision = 0.7

5 Results

The top 3 most relevant webpages returned are indeed the top results for all searches, as can be observed from the evaluation of the sample queries, and the precision for all queries is fairly strong. The query results also show that in a vector space model IR system based on the TF- IDF scheme, words that are more common in a text play a significant influence in defining the topic of that particular document. Not all documents contain words like ”internship,” ”career,” and ”job,” but those that do have them frequently. As TF is a key factor in document ranking, this lessens

the uncertainty in getting the most comparable documents.

But it is also evident from the results of the queries that the query results returned might not always be relevant to the context intended by the query. For instance, in query 2, the context the query 'Information Retrieval' could be to search for the course' CS582: Information Retrieval'. But the pages returned have no connection with the course 'Information Retrieval'. Except for the first query result returned, all other webpages are just webpages that have the occurrences of words 'inform' and 'retriev' (the stemmed query tokens).

An error that can be noticed when evaluating the query results was that some links are repeated in the results. Even though the crawler checks for duplicate links during link traversal, there are some links that are present in webpages with both 'http' and 'https' extensions. The crawler at present treats the links which point to the same page but have different HTTP format as distinct links. This can be worked on in the future.

Related Work

I read a few online resources to learn how to perform link traversal and web crawling, implementing Breadth- First Search in Python, parsing webpages' text, and efficiently calculating cosine similarities.

I also found a couple of papers, one discussing the integration of the Text Rank algorithm in a vector search model and the other discussing how to return results relevant to the context of the search query in a search engine

Future Work

Given that it was created from scratch, the implemented online search engine's precision is quite respectable and adequate.

But by including a variety of other features into the search engine as a whole, the performance of the search engine can be greatly enhanced.

The current page's URL can be used to convert relative URLs for the web crawler, who can also use Multi-Threaded Spidering and Directed Spidering techniques like Topic-Directed or Link-Directed Spidering.

The integration of the Text Rank algorithm and a query context determining algorithm with the IR system will increase accuracy even more and return pages that are much more precise.

In order for the search results to be updated, the crawled pages must occasionally be updated.

The search engine's front end could be a GUI program.