Web-Search Engine

Sri Vishnu Prasanth Maddala
Master's in Computer Science
University of Illinois at Chicago
smadda2@uic.edu

ABSTRACT

This document is a report for the final project of CS 582 Information Retrieval at University of Illinois at Chicago. This project is about web-crawling and searching using the similarity measure between search query and pages. Results of the queries that are tested were almost accurate and it also provided a lot of insight and scope for future work.

INTRODUCTION

The software is written in Python3 in a more user readable format to make it easily extensible for future work. In order test the software and makes it less time consuming, all the necessary web-pages were preprocessed and the import variables such as Inverted Index, Document lengths, individual tokens for each document, URLs are stored in pickle files. All the pickle files were shared to test the performance of the software.

CRAWLER

The script that does the crawling is '*crawler.py*'. It works in traditional BFS fashion, starting at the root node which is domain of UIC-CS (*https://www.cs.uic.edu/*) and extracts all the web-pages which are in UIC domain(uic.edu) and adds those links to the queue. Each link is dequeued and it's HTML content is downloaded using 'Beautiful soup' library, all the links of this page are extracted and added to the queue if all the necessary conditions were met.

There were few links with the extensions ('.avi','.ppt','.gz','.zip','.tar','.tgz','.docx','.ico', '.css', '.js','.jpg', '.jpeg', '.png', '.gif', '.pdf', '.doc', '.JPG', '.mp4', '.svg') which are not considered as they takes lot of time to download.

Crawling will be terminated when the queue gets empty or a certain threshold was met. Here I used the threshold value of 3000.

During the crawling all the links were stored in a dictionary as values and it's corresponding file name as key.

This dictionary is dumped using pickle for the future use.

PREPROCESS

The script that does the preprocess of the downloaded content is '*preproces.py*'. It takes each downloaded page from the local

directory and extract all the content of the '<body>' and '<Title>' tags.

After the extraction necessary preprocessing steps like 'stemming', 'stop-word removal', 'removing duplicates, numbers, special characters' are applied and creates raw inverted index("Idf values are not considered").

EXTRACTION

The final script is 'extraction.py', which loads the raw inverted index from pickle and updates the each value by multiplying with IDF of that word and also calculates cosine similarity with query and each file using updated inverted index. Based on the similarity values appropriate pages(links) will be displayed.

CHALLENGES

At first, I started implementing the crawler. I have no idea how to do this. So I started to learn about the libraries that are used to extract HTML content.

When I wrote my first crawler, it took few hours to download some 100 pages. Then I carefully saw the links where it's getting stuck, I found few links with extensions ['.avi','.ppt','.gz','.zip','.tar','.tgz','.docx','.ico','.css', '.js','.jpg', '.jpeg', '.png', '.gif', '.pdf', '.doc', '.JPG', '.mp4', '.svg']. When I excluded these links my crawler started downloading the pages more quickly.

I didn't find much difficulty in extracting hyper links from HTML, but the hectic part is during the content extraction. I implemented many logics which everything turned out be on the negative side.

I got stuck in this part almost for 2-3 days finally implemented some logic which gave better results.

MEASURES AND WEIGHTING SCHEME

The weighting scheme that I used was the simple **TF-IDF** of words. I have no second thought it has been proved to be one of the most effective and also accounts for the importance of words in each document in the correct way. I didn't spend much time on this as this is the best weighting scheme when it comes to web-search engines.

The similarity measure used to rank documents was the **Cosine Similarity**. I started by implementing inner product first and then converted it into cosine similarity which hardly takes two lines of extra code. It's better to use cosine similarity because it takes both the document length and query length into consideration.

Even for this I have no second thought as it was already proved to be the best similarity measure.

ALTERNATIVES

Instead of cosine similarity another measure is inner product similarity which doesn't take length of query and document into account, which may lead to bad results. So it is better to use cosine similarity instead of inner product similarity.

One more approach is, the integration of similarity scores with page rank scores. This approach will increase the accuracy because page rank scores are the probabilities of pages. These scores are based on in-links and out-links of that page. So if a page has more in-links, probability of that page will be high when these score are integrated with similarity scores the page with low similarity but with high probability could be on top.

EVALUATION AND RESULTS

I experimented on 5 random queries and printed top 10 links of those web-pages.

Here are the results:

Query: UIC career fair

```
Enter your search:UIC career fair
https://ecc.uic.edu/engineering-career-fair/
https://www.ois.uic.edu/
http://uic.edu/about/job-opportunities
http://life.uic.edu/
https://registrar.uic.edu/transfer-assistance/
https://careerservices.uic.edu
http://careerservices.uic.edu/upcoming-career-events/
https://careerservices.uic.edu/upcoming-career-events/
https://studentemployment.uic.edu/events/on-campus-job-fair/
https://engineering.uic.edu/student-organizations/
```

Query: UIC events

```
Enter your search:UIC events
https://today.uic.edu/events/categories/event-special-event
https://today.uic.edu/events/2019-uic-it-community-conference
http://www.facebook.com/sharer/sharer.php?u=https://today.uic.edu/attn-uic-grads&t=Attn: UIC grads
https://today.uic.edu/events/categories/event-lecture
https://today.uic.edu/events/categories/event-health-medicine
https://today.uic.edu/events/grant-writing-workshop-2
https://today.uic.edu/events/categories/event-conference
https://engineeringalumni.uic.edu/events/
http://engineering.uic.edu/deans-list-fall-2018/
https://today.uic.edu/events/categories/event-featured
```

Query: UIC cornelia caragea

```
Enter your search:UIC cornelia caragea
https://www.ece.uic.edu/faculty-staff/
https://cef.uic.edu/
https://today.uic.edu/campus-news/campus-newspaper
http://advance.uic.edu/alumni-association/
http://www.uic.edu/apps/find-people/search
https://www.uic.edu/apps/departments-az/search
http://www.uic.edu/depts/accc/lan/server_services/AASindex.html
http://www.uic.edu/depts/accc/pclabs/index.shtml
https://www.uic.edu/apps/departments-az/search?dispatch=find&orgid=99883
http://www.uic.edu/apps/departments-az/search
```

Query: UIC internships

```
Enter your search:UIC internships
https://www.honors.uic.edu/
https://oia.uic.edu/about/oia-programs/international-agreements/active-uic-internatic
https://rockford.medicine.uic.edu/research/research-opportunities-students/summer-sci
https://ecc.uic.edu/students/graduation-survey-results/
https://ecc.uic.edu/events/graduating-soon-claim-your-gift-for-completing-the-uic-fir
https://ecc.uic.edu/career-toolbox/go-search/
http://ecc.uic.edu/search-uiccareers/
https://engineering.uic.edu/student-organizations/
https://studentemployment.uic.edu/events/internship-part-time-job-fair/
https://engineering.uic.edu/senior-design-expo/
```

Query: UIC

```
Enter your search:UIC
https://today.uic.edu/campus-news/campus-newspaper
http://advance.uic.edu/alumni-association/
http://www.uic.edu/apps/find-people/search
https://www.uic.edu/apps/departments-az/search
http://www.uic.edu/depts/accc/lan/server_services/AASindex.html
http://www.uic.edu/depts/accc/pclabs/index.shtml
https://www.uic.edu/apps/departments-az/search?dispatch=find&orgid=99883
http://www.uic.edu/apps/departments-az/search
http://snap.uic.edu/
https://www.uic.edu/apps/departments-az/search?dispatch=find&orgid=99287
```

DISCUSSION ON RESULTS

We can see that in almost all the queries above, at least 6-7 results were coming good. It means we can say that precision is 60% or above.

I think if a word appears more number of times in a document like UIC, career fair., it is giving more good results. Where as if a word appears less number of times it is giving ambiguous results this is the major error in this software.

So by seeing the results, we can say that term frequency is playing a major role in search.

DISCUSSION ON RELATED WORK

Saw few articles on web-crawlers, how efficiently the crawling is done.

Also the papers on page-rank algorithm, that focuses on integration of page-rank with similarity measure.

FUTURE WORK

The overall performance of the search engine was pretty good. To make this search engine more accurate we need to improve the similarity values. We can do that by integrating cosine similarity with page rank. So this would be my future work.

How to Run:

There is no need to run '**crawler.py**' and '**preprocess.py**', as those results are already stored in a pickle files and shared with you. So to test the software, run '**extraction.py**' with all the pickle files downloaded in the folder where python file was downloaded.

Note:Please keep 'extraction.py' and all the pickle files that are shared in one folder and also make sure all the necessary packages are installed using 'pip' before running.

Command to run:

In command prompt go to the path where 'extraction.py' was stored and run

'py extraction.py'