Insert Your Title Here∗

Search Engine

Yumeng Yang†  
 Engineering Department  
 University of Illinois at Chicago  
 Chicago IL U.S  
 yyang244@uic.edu

DSCRIPTION

Big data has become an important and popular topic. A following question is that how to retrieve data that people need from big data sets effectively. Search Engine like Google, Baidu and Bing are built to help people get information accurately and efficient. This project is about how to build a basic search engine.

The project consists of three parts. First, information gathering. This project implements a BFS crawler to scrape information on the internet. Second, retrieve relevant document from the data according to a query. Last, optimization. In this project, I use PageRank to optimize the performance of the search engine.

Bellowed are the detail of these three parts.

1 Web Crawler

There are two basic method to implement a web crawler: Depth First Search and Breadth First Search. Compared to Depth First Search, BFS requires more space to store the node. But the problem for DFS is that, it might go too deep along one path. So, when we have a large number of nodes, DFS is not appropriate. So, here I choose BFS to implement the Web Crawler.

There are several key points: the first one is that some website may prevent too frequent request from a robot, so our request may be rejected. The second one is how to parse a web page to get the content and extract the link from it. The third one is how to achieve BFS algorithm.

Normally, server will check User Agent in Headers to check who is visiting the website. The default parameter of User Agent will indicate ‘Python’ which will be treated as robots by server. So, we need to modify the parameter to hide the identity of crawler. Here I use the library fake-useragent. It will generate different agents randomly.

However, we can parse the page manually or we can download the whole page and then extract the content we need. But another useful and powerful library is available, which is BeautifulSoup4. BeautifulSoup4 has substantial function to help us parse the web page, download the content we need and extract the link. But when I extracted links using BeautifulSoup4, I found that many of the links extracted by BeautifulSoup4 were not a http format. I check the source code and compared those links, then I realized that those wrong links started with ‘/’ instead of “[https://](NULL)” or “[http://](NULL)” eliminate the root file in links. Then I do some change on my code to modify those links starting with ‘/’ to the format of http.

BFS is Breadth First Search, which means we need to travel the graph level by level. Thus, I need a queue to help me achieve traveling node in this order since queue is in first in first out order. First, we poll a node in queue. After we visited the page and download the content to out computer, we extract links on that webpage and put them into queue. In addition, a graph is different from a tree. Circles might exist in a graph. Keeping a visited node set can prevent visit those nodes we’ve already visited. Every time after we visited a page, we put the link into visited. Every time before we visited a page, we check if we visited this page already.

2 Vector Space Model

There are two retrieval models in general: Boolean models, Vector Space Models. Boolean Models require exact match, exactly the same order, exactly the same words. Although the result of this method can be 100% correct, but it may require user to follow a pattern to input their query. Different from Boolean Model, Vector Space Model can return a ranking list. But meanwhile, it is not that efficient. In this project I choose Vector Space Mode.

2.1 Preprocess

Before we do some calculation, we need to do something on the pages we download to make later step easier. Punctuations have nothing to do with information retrieval. So, they need to be removed. Different morphology of one word can be treated as the same. Thus, we need stem to just keep the root of the word. Noticed that some words are meaningless when we doing information retrieval such as the, a, and and so on. Those are called stopwords and should be remove from the documents. Meanwhile, there exist some other noisy words we need to remove. After doing this, out document will be clean and representative.

2.1 Weighting Scheme

I choose TF-IDF as the weighting scheme. TF-IDF is the combination of local weighting and global weighting. TF is the frequent of term i in document j. IDF represents the inverse document frequency of term i. the basic idea of TF-IDF is that a word i appear more often in one document j but appear less in the whole collection is important in this document and can gain higher weight.

2.2 Similarity Measure

After we get the TF-IDF value of every word in document and query, we can calculate the similarity of the given query and every document. I choose Cosine coefficient as my similarity measure.

It measures the cosine of the angle between two vectors. Cosine similarity is the result of inner product normalized by the vecter lengths. In this project, we have the weight of every word in document wj, and the weight of every word in query wj, we need to calculate the cosine similarity of the same two words appearing in document and query. Add CosSim of all those matching words. This result is the similarity of a query and a document. Then we rank the document according to the CosSim.

3 PageRank

Although Vector Space Model is a powerful model to retrieve information. But we need to take another factor into consideration: authority. The authority of a webpage represent the number of inlinks of this webpage. That is to say, the more inlinks it gets, the higher rank it should be. But how to get the authority of a webpage? PageRank and HIT can solve this problem. PageRank only cares about the inlinks of a page while HIT need to deal with inlinks and outlinks Unlike HIT will be executed every time user input their query, PageRank can be done off line. So it is a query independent algorithm. Additionally, it requires less calculation. PageRank is an idea algorithm which can reduce the response time.

The basic idea is that we keep a directed graph. If there is a link from one page to another, we add such an edge on the graph. After building the graph, we run Page Run on this graph repeat. Every time we calculate the new weight for every page. In case we jump into a page whose number of outlinks is 0 and we get nowhere to go, we assign a same probability 1/n to every single page. That means, every loop, we can jump to another page randomly with probability a or we can jump along an outlink. This is called teleporting. It prevents getting stuck locally.

In this project, I represent the web network by array instead of graph. If there is an edge from v to u, we set array[v,u] 1, otherwise 0. We need another array to keep the current PageRank value S. Every time we update S by the formula below until the algorithm converge.

RESULTS

The five inputs of this project are:

cs master student

graduation deadlines

phd research

happy holiday

career fair

**output of query: cs master student**

**-------------without PageRank---------------**

http://catalog.uic.edu/ucat/course-descriptions

https://www.cs.uic.edu/our-department

https://www.cs.uic.edu/Main/AboutCS

http://catalog.uic.edu/gcat/course-descriptions

http://catalog.uic.edu/all-course-descriptions

https://www.cs.uic.edu/masters-student-receives-honorable-mention-at-ubicomp-2018

https://www.cs.uic.edu/masters-student-receives-honorable-mention-at-ubicomp-2018/#respond

https://www.uic.edu/apps/departments-az/search?dispatch=letter&letter=M

http://www.uic.edu/apps/departments-az/search?dispatch=letter&letter=M

https://catalog.uic.edu/ucat/colleges-depts/engineering/cs/joint-bs-ms

**-----------with PageRank----------**

https://www.uic.edu/apps/departments-az/search?dispatch=letter&letter=M

https://today.uic.edu/uic-establishes-new-graduate-degree-in-city-design

https://www.cs.uic.edu/our-department

http://www.uic.edu/apps/departments-az/search?dispatch=letter&letter=M

http://engineering.uic.edu/college-news-2013

https://www.cs.uic.edu/Main/AboutCS

https://www.cs.uic.edu/evl-to-be-featured-in-chicago-new-media-1973-1992-exhibition/#respond

https://www.cs.uic.edu/ms-students-win-mobility-hackathon-chicago-with-feel-your-city-platform/#respond

https://www.cs.uic.edu/masters-student-receives-honorable-mention-at-ubicomp-2018/#respond

https://catalog.uic.edu/ucat/colleges-depts/engineering/cs/joint-bs-ms

**output of query: career fair**

**--------------without PageRank--------------**

http://grad.uic.edu/graduate-college-new-student-orientation

http://grad.uic.edu/petitions

https://jobs.uic.edu/job-board/job-details?jobid=104249

https://www.cs.uic.edu/employment

https://today.uic.edu/tag/awards

https://today.uic.edu/tag/technology

http://studentemployment.uic.edu

http://www.uic.edu/depts/st\_empl

http://dos.uic.edu/student-veterans-affairs/faqs

https://www.cs.uic.edu/News/WebHome?name=arch2009

**-----------with PageRank----------**

https://www.cs.uic.edu/News/WebHome?name=arch2009

http://www.uic.edu/depts/st\_empl

https://jobs.uic.edu/job-board/job-details?jobID=105417&job=2018-2019-ece-open-rank-tenured-tenure-track-faculty-search

https://www.cs.uic.edu/employment

http://studentemployment.uic.edu/events/internship-part-time-job-fair

http://grad.uic.edu/graduate-college-new-student-orientation

http://studentemployment.uic.edu/news-and-events

https://jobs.uic.edu/job-board/job-details?jobid=104249

https://jobs.uic.edu/job-board/job-details?jobID=104024&job=clinical-track-teaching-faculty-computer-science

http://studentemployment.uic.edu

EVALUATION

What I need to do to evaluate this project is to look the top 10 result and check if it is relevant to my query. While, how to define relevant is tricky. I’ll define a page is irrelevant when it is completely different from the query. Otherwise, it is relevant.

As we can see in table1, in general, result with PageRank is better than the result without PageRank. For the third query “phd research”, result without PageRank is higher. The top 10 pages retrieved are all Phd students’ or researchers’ personal pages. these pages are highly relevant to the query, but usually these pages have lower authority, which may lead to lower rank with PageRank.

Another interesting thing is this search engine is straightforward. Supposed we have a query “cs master student”, those pages about “cs graduate student” will not get a higher rank even these two sentences are the same. This is because we didn’t deal with synonym.

At first, among the retrieved pages, I ranked top 100 by their PageRank. This led to an even worse result. Many irelavant pages were given higher rank and were returned. I checked the results from UIC website and found out that normally after the top 30-50 result, the pages would become irrelevant to the query. So, the top 100 pages I retrieved may contains many pages that has higher PageRank but are irrelevant to the query. This may explain why after applied PageRank, the result was worse. Then I chose to rank the top 20 retrieved pages by their PageRank.

As chart1 indicates, as we choose to rank more retrieved pages according to their PageRank, the number of irrelevant pages will become bigger.

This lead to a conclusion that if we want to apply PageRank, we need to make sure that there are enough relevant pages to our query. Otherwise, PageRank algorithm may return a lot of irrelevant pages with higher authority.

|  |  |  |
| --- | --- | --- |
|  | No PageRank | With PageRank |
| cs master student | 10 | 10 |
| graduation deadlines | 5 | 6 |
| Phd reasearch | 10 | 3 |
| Happy holiday | 5 | 7 |
| Career fair | 4 | 8 |

Table 1. number of relevant pages among the top 10 returned pages

Chart1. Performance of PageRank

In order to better evaluate the result, I input the same query on UIC website and get a list of results. Some webpages are expired or irrelevant, so I picked 10 pages according to the returned list. This might be unfair because even GOOGLE and BAIDU they are possible to give different rank of list according to the same query. But this will give me a better look on what should I improve.

This is the result on UIC website of input “cs master student”

https://www.cs.uic.edu/graduate-programs/

https://www.cs.uic.edu/graduate-admissions/

https://www.cs.uic.edu/masters-student-receives-honorable-mention-at-ubicomp-2018/

https://catalog.uic.edu/gcat/colleges-schools/engineering/cs/ms/

https://catalog.uic.edu/ucat/colleges-depts/engineering/cs/joint-bs-ms/

https://www.cs.uic.edu/graduate-programs/cs-graduate-program-degree-requirements/

https://www.cs.uic.edu

https://www.cs.uic.edu/our-department/

https://www.cs.uic.edu/Main/AboutCS

https://www.cs.uic.edu/graduation-guidelines/

As we can see, the result returned from UIC website is totally different from the result we get using out search engine. This can be explained from 3 perspectives. First, the number of returned pages are too small. We only compare the top 10 pages, it is reasonable that such difference exits. Second, we use different retrieval models or different optimization algorithm. This will also lead to different result. Third, our search engine is a basic search engine. We need add other features to make it more efficient.

CHALLENGE

Because the data set is much bigger than before, so the biggest challenge is how to retrieve information effectively. Taking this into consideration, I decided to use PageRank as the intelligent part. First of all, this algorithm is precise enough. The most important is that PageRank is query independent. So, if user input a query, the system will not need to spend time on calculate inlinks and outlinks and then the ranking.

Apart from efficiency, another challenge is how to design the algorithm. PageRank and Vector Space Model all need a lot of calculation, especially we have so many data. What kind of data structure to choose to make the algorithm straightforward and faster becomes vital.

RELATED WORK

Search Engine like Google and Baidu are all efficient and powerful. They maintain a huge data set to index the information and can return result to user within 0.1 second. Apart from that, Both Google and Baidu can scrap the website dynamically and can support different languages.

But, Google has made great progress on webpage dupilication Sometimes, a same article may appear in several different webpages. But user do not wish to see the same article twice. Google implement an algorithm Simhash to do web page duplication. It mapped document to a 64 bits binary number sequence. Compare the difference between 2 sequences to represent the different of 2 documents.

Another Search Engine has gain people’s attention these years – GoGoDuck. The reason that why GoGoDuck is getting popular is that it protect users’ privacy. Normally, a search engine will save users’ information and put ads on website according to the information. But website security has become a great concern. For GoGoDuck, it will not keep users’ information nor identify users by Cookies.

FUTURE WORK

For web crawler, my next move is to implement a multithreading web Crawler. This project requires no less than 3000 pages. Plus all the parsing and processing time it took me around an hour to finish scraping. So, this is one thing that I can improve a lot.

Another thing is that there is still a lot to do on preprocessing. I only do some basic things. Some Unicode characters in the document, no matter how hard I tried, still couldn’t find be removed in a proper way without influence their neighbor words.

Then for page rank, it is still a little bit slow, it will take more than 1 minute to calculate PageRank for the whole collection. While, a minute seems not that bad. But notice that this is not the real world. We only need to process 3000 pages this time. But an actual search engine will need to handle billions of pages. How to calculate them fast is important. My plan is to represent the whole network by graph in python instead of arrays. If I apply graph, I only need to store 3000 nodes and maybe 100 \* 3000 edges. But for array, we need 3000 \* 3000 space to store the whole array. This will also lead to slow calculation definitely.

In terms of the final result, from my observation, it is not accurate and comprehensive enough because this project doesn’t take synonym into consideration. For example, if the input is “CS master student”, when we search on UIC website, many results will be about “CS graduate student”. But in our system, it will only return pages have higher similarity with “CS master student”. In fact, “CS master student” is the same with “CS graduate student”

CONCLUSION

Basically, Search Engine is consisted of three parts. Web Crawler, retrieval model and optimization. Although, this search engine can achieve basic functionality, it still has a lot to be improve. I can apply more optimization algorithm to make the result more accurate or some speed accelerate algorithm to make it faster.

REFERENCES

[1] <https://sspai.com/post/40117>

[2] <https://www.reliablesoft.net/top-10-search-engines-in-the-world/>

[3] <https://en.wikipedia.org/wiki/Google_Search>

[4] https://en.wikipedia.org/wiki/DuckDuckGo

Conference Name:ACM Woodstock conference

Conference Short Name:WOODSTOCK’18

Conference Location:El Paso, Texas USA

ISBN:978-1-4503-0000-0/18/06

Year:2018

Date:June

Copyright Year:2018

Copyright Statement:rightsretained

DOI:10.1145/1234567890

RRH: F. Surname et al.

Price:$15.00