SMART SEARCH

Intelligent UIC based Search Engine

Sriram Veturi
Computer Science
University of Illinois at Chicago
Chicago, Illinois, USA
sriram.tutu@gmail.com

ABSTRACT

This projects to build an efficient, fast and highly scalable search engine which would search the web with respect to the user's query and provide the search results. The high level idea is to build a search engine just for the University of Illinois at Chicago. The search engine would yield results for any query from the user with respect to the University of Illinois at Chicago. For any search engine, a database of the web pages is required. For this project, a database of pages from the UIC domain was created and thus, the search results are restricted to the university. The user can enter the query and the top ten results matching the query would be shown to the user. The project involves various popular information retrieval and search algorithms like TF-IDF scheme, Cosine Similarity, Google's Page Rank, etc. Overall, the project was built to get efficient, reliable and most matching results across the web about UIC.

METHODOLOGY

CRAWLER

Search Engines could be broadly classified into three types as Robot-Driven Search Engines, Directory based Search Engines, and Meta-Search Engines. The search engine developed here is a Robot-Driven Search Engine which used a 'robot' or 'worm' to crawl the web and indexed the pages on web. As of 2018, there are 1.8 Billion web sites available which is a really huge number. The crawler or the spiderbot visits the pages, get the data present in the pages and most importantly, gets the information about the outgoing links of the pages. Many web pages do not allow the crawlers to visit the pages and limit their functionality by adding a file called 'robot.txt'. The whole purpose of this file is to limit any bot or crawler to get information of the page or the children pages that could be accessed ahead. The web graph is representation of the web sites and their connected websites so that there is link between the connected components. The link is basically is a hyperlink pointing to the sites. The crawler used in this project only crawls in the UIC domain and a maximum of 5000 pages. The algorithm used for crawling is a Breadth First Search approach (queue implementation). The process of crawling starts from the initial website (https://cs.uic.edu). It visits the site and gets all the hyperlink references in the site and adds it to the queue. A site is popped from the queue and crawled, respective hyperlinks pushed into the queue again. This process happens until the queue is empty, which is unlikely, given the amount of pages on the web in the UIC domain, or a maximum of 5000 pages are crawled, whichever comes first. This is main functionality of the crawler in this project.

TEXT EXTRACTION

As the crawler visits the pages in the Breadth First Search manner, it not only gathers the hyperlinks but it also extracts the textual data present in the site from the relevant tags like body, paragraph, div, etc. It ignores the text present in the unnecessary tags like style, etc. As it extracts the data, it stores the textual data in a separate folder. In this way, the documents initial database is generated.

DATABASE

The content extracted from each page is then preprocessed in a number of ways. The stop words were removed from the text which carry no meaning with them. All the special characters, numbers, etc were eliminated which only keeps the strings in the ASCII format. Now, at this point, it is good to stem the entire corpus using Porter Stemmer. The text preprocessing heavily relied upon the NLTK library. The text was preprocessed at this point. Since, we have the list of words present in a page, a word count mapping was done in order to get the count of repetitions of the words in the page. Now, next major step is to decide which hyperlinks to crawl. There were several condition which were discovered after experimenting with the crawling process. First, only to crawl those sites which are responsive (200 OK on sending request). Second, when you are dealing with the hyperlinks, one parent page with numerous sections, can have multiple hyperlinks pointing to particular sections in the page. In that case, it is wise to consider only the parent page and hyperlinks which point to the particular sections in the same page. Also, it is important to ignore the dead ends in the web graph which include pages with PDF, DOC, GIF, TAR, etc file. The list of extensions all these unnecessary files are listed in the 'CHALLENGES' section below. When a user sends multiple requests to a domain, certain sites have a tendency to throw an error which does not allow multiple requests in a timeframe. To counter this problem, random user agents were generated to hit the sites, to make it seem that the requests are coming from different users. Finally, the list of all outgoing links was generated. The final database is generated in the JSON format where each page has a separate file. Each page's file contain three keys namely, 'URL', 'OUTGOING_LINKS', and 'WORD_COUNT_MAP'. The 'URL' field is nothing but the hyperlink of the page visited. The 'OUTGOING_LINKS' field is an array or list of the children hyperlinks in the page. Finally, the 'WORD_COUNT_MAP', as discussed in the text preprocessing part in the same section, is a mapping of word and its count in the document. All these files are stored in a folder which is considered as the main database for the further operations.

USER INPUT

The user input was obtained from the user interface developed, which is described in the 'USER INTERFACE' section below. The textual query entered by the user also goes through the same preprocessing steps like stop words, removal, text cleaning, stemming, etc as discussed in the previous section.

TF-IDF OPERATIONS

As the database is generated at this point, let's discuss the core algorithm used in this project for building the Vector Space model using the inverted index data structure. TF-IDF (Term Frequency - Inverse Document Frequency) scores were calculated for the documents and the words in them. The term frequency gives an idea about the frequency of the term in the document. It is the count of the term divided by the total number of words in the document. It increases as the terms appears more times than other words. The inverse document frequency damps the words that appear over all the documents which do not convey any meaningful information.

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{i,j}} \qquad idf(w) = log(\frac{N}{df_t})$$

The final TF-IDF score could be calculated with the formula below:

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

$tf_{i,j}$ = number of occurrences of i in j
df_i = number of documents containing i
N = total number of documents

In the same way, get the TF-IDF scores of the users queries if the queries are saved as documents again. Once the TF-IDF scores are calculated for both the documents and the queries, we need to fetch the most similar documents based on the user's query. So, to get the similarity score, Cosine Similarity was used. The formula for the Cosine Similarity is shown below. The top 200 hundred pages were retrieved based on the user's query using the Vector Space Model. Later, using the Page Rank discussed below, would be used to get the top ten results.

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A}\cdot\mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum\limits_{i=1}^{n} A_i B_i}{\sqrt{\sum\limits_{i=1}^{n} A_i^2}\sqrt{\sum\limits_{i=1}^{n} B_i^2}}$$

USER INPUT

According to my research, other important measures of distance include Jaccard/Tanimoto. Jaccard/Tanimoto distance is used to evaluate the distance between two objects whose feature set is sparse. Traditional distance measures would consider two objects similar if they had 5 features with non-zero values that were not similiar if they both also contained 1000 zero records each. Jaccard accounts for the fact that sparse data sets have false similarity by only considering features which are present in either object for the distance calculation rather than all the features. Domains where this is useful include market basket data where a typical customer buys only a small set of the total items available in a grocery store. Without Jaccard any two customers would be very similar. Cosine distance is a calculation of the similarity in the direction of two vectors and completely ignores the magnitude of them.

The similarity measure is the cosine of the angle between the two feature vectors.

PAGE RANK ALGORITHM

PageRank is a very popular algorithm used by google with several modifications to rank the websites based on the number of inlinks to the website. The algorithm output the probability distribution about the likelihood of a person visiting a page after some random clicks. This algorithm is also probably the best known ranking algorithm till date. The formula for the page is shown below.

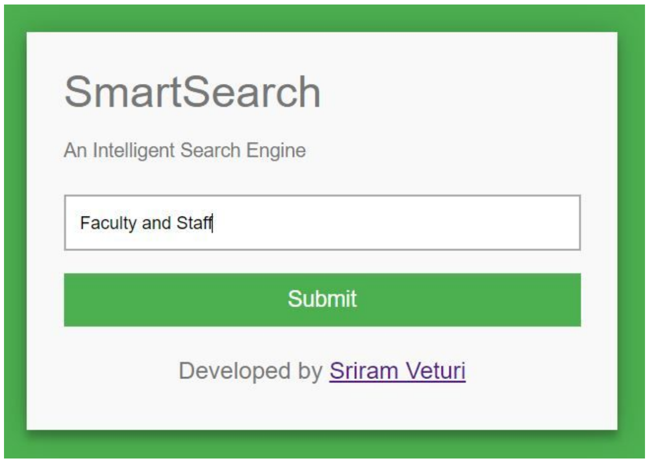
$$S(A) = \frac{\epsilon}{n} + (1-\epsilon) \sum_{(B,A)\in G} \frac{S(B)}{out(B)}$$

The page ranks of all the websites in the web graph are calculated and stored in a separate file. So, out of the top 200 pages already filtered, get the best 10 web pages based on the Page Rank scores generated from the algorithm.

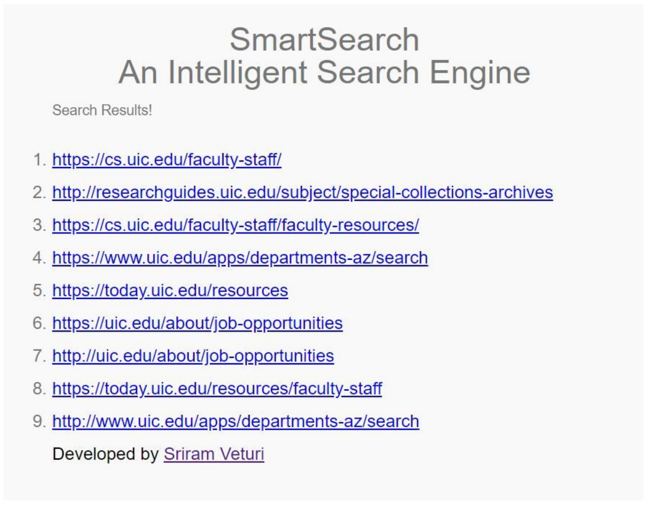
USER INTERFACE

The user interface involves two pages which mainly involves a landing page and then the results page. The entire search engine backend was developed in Python but the user interface was developed using very basic HTML and CSS. The search engine was deployed using Flask API on the local host. When the user runs the main Python file, and opens the address on the browser, a page appears where the user is allowed to enter the query in a form and after submitting, a new page appears where the search results are displayed to the user in the hyperlink format. The user can then view the websites in the hyperlinks shown.

Landing Page:



Search Results Page:


SmartSearch
An Intelligent Search Engine

Search Results!

1. https://cs.uic.edu/faculty-staff/
2. http://researchguides.uic.edu/subject/special-collections-archives
3. https://cs.uic.edu/faculty-staff/faculty-resources/
4. https://www.uic.edu/apps/departments-az/search
5. https://today.uic.edu/resources
6. https://uic.edu/about/job-opportunities
7. http://uic.edu/about/job-opportunities
8. https://today.uic.edu/resources/faculty-staff
9. http://www.uic.edu/apps/departments-az/search

Developed by Sriram Veturi

The website for the search engine currently could be deployed on the localhost but as an extension to this projects, I plan to deploy it Heroku servers or using Amazon Web Services.

CHALLENGES

This was a very challenging project as it involved many dead sites, sites with cookies, sites which only accepted a particular number of requests in a timeframe. Also, the user interface development was very new to me which made it challenging task for me. The problem of sites accepting only a particular number of requests in a period of time was resolved by using a different and random user agent for every request that was sent the page. Since, the user agent is different for every request, the server thought the requests were generated by different people at the same time. In this way, I tried to resolve this issue.

There were many unresponsive, dead, and broken links in the web graph. To handle those websites, I checked for the status code of the response. If the status code was 200 OK or anything in 200 which generally means successful, only those sites were crawled. All the other sites were not crawled as they entered the except clause in the program.

It was very challenging to identify the sites which had no outgoing links particularly, the sites which had documents with extensions as follows:

- pdf
- doc
- docx
- xls
- avi
- mp4
- xlsx
- jpg
- png
- gif
- gz
- rar
- tar
- rv
- tgz
- zip
- exe
- js
- css
- ppt

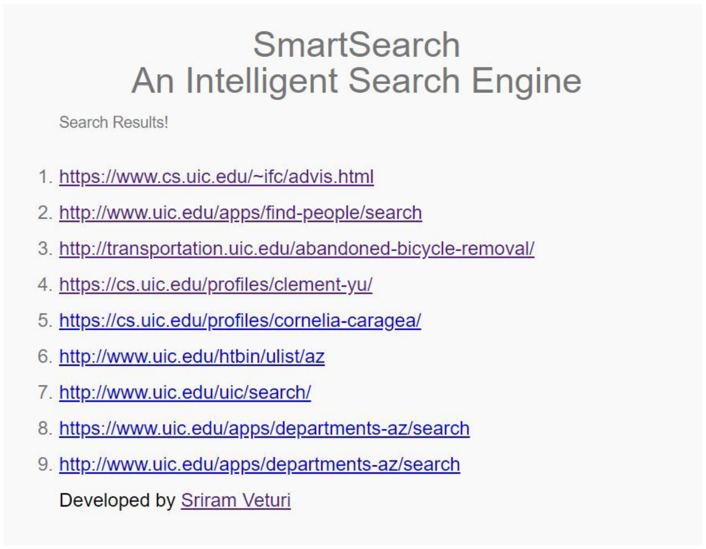
Although, the occurrence ratio of these files is very low but, to build a comprehensive search engine, it was important to remove such files from the web graph.

There were also many websites which require the user to accept the site cookies. If the bot does not accept the site cookies, the request gets into an infinite indirection loop which eventually becomes a timeout session. So, to handle the situation with cookies, I created a CookieJar which temporarily stores the cookies and flushes them as soon as the request is completed successfully.

MANUAL EVALUATION

I tried various queries for the search engine and analyzed the results and decided the relevance of the search results with respect to the user query. Following are some of the queries which I tried and evaluated upon.

1. The query was "information retrieval" and there was a result which had the Professor Cornelia Caragea's faculty profile who teaches the Information Retrieval course at UIC. So, I think the retrieved results were accurate enough.



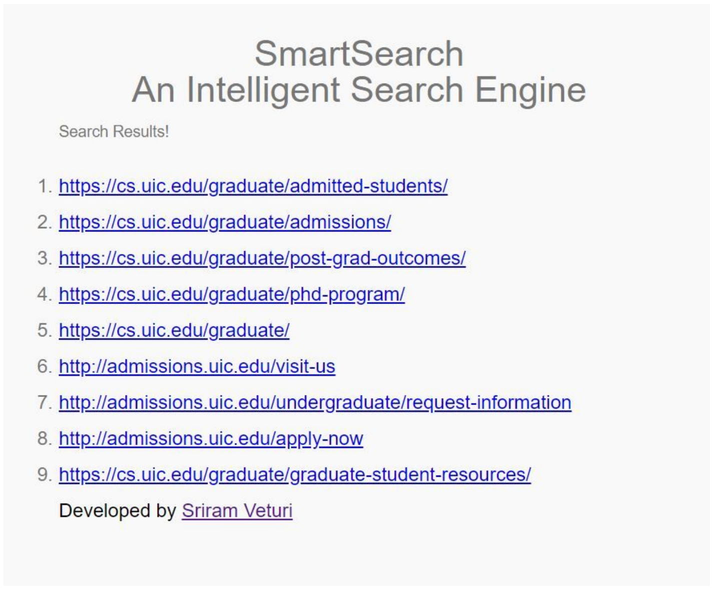
SmartSearch
An Intelligent Search Engine

Search Results!

1. https://www.cs.uic.edu/~ifc/advis.html
2. http://www.uic.edu/apps/find-people/search
3. http://transportation.uic.edu/abandoned-bicycle-removal/
4. https://cs.uic.edu/profiles/clement-yu/
5. https://cs.uic.edu/profiles/cornelia-caragea/
6. http://www.uic.edu/htbin/ulist/az
7. http://www.uic.edu/uic/search/
8. https://www.uic.edu/apps/departments-az/search
9. http://www.uic.edu/apps/departments-az/search

Developed by Sriram Veturi

2. For the query "graduate school", the results were https://cs.uic.edu/graduate/admitted-students/, https://cs.uic.edu/graduate/admissions/ in the top two results. I think these two were pretty accurate results for a random users.



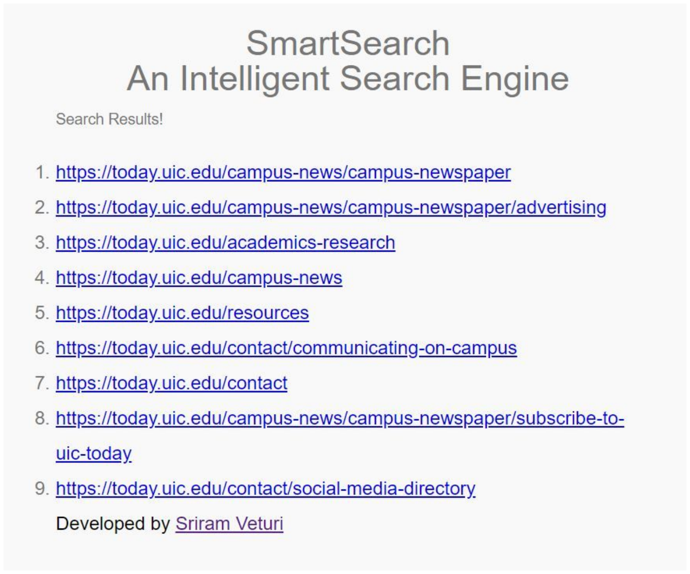
SmartSearch
An Intelligent Search Engine

Search Results!

1. https://cs.uic.edu/graduate/admitted-students/
2. https://cs.uic.edu/graduate/admissions/
3. https://cs.uic.edu/graduate/post-grad-outcomes/
4. https://cs.uic.edu/graduate/phd-program/
5. https://cs.uic.edu/graduate/
6. http://admissions.uic.edu/visit-us
7. http://admissions.uic.edu/undergraduate/request-information
8. http://admissions.uic.edu/apply-now
9. https://cs.uic.edu/graduate/graduate-student-resources/

Developed by Sriram Veturi

3. For the query “uic news”, all the top 5 sites were in the domain today.uic.edu which represented the news happening in UIC.

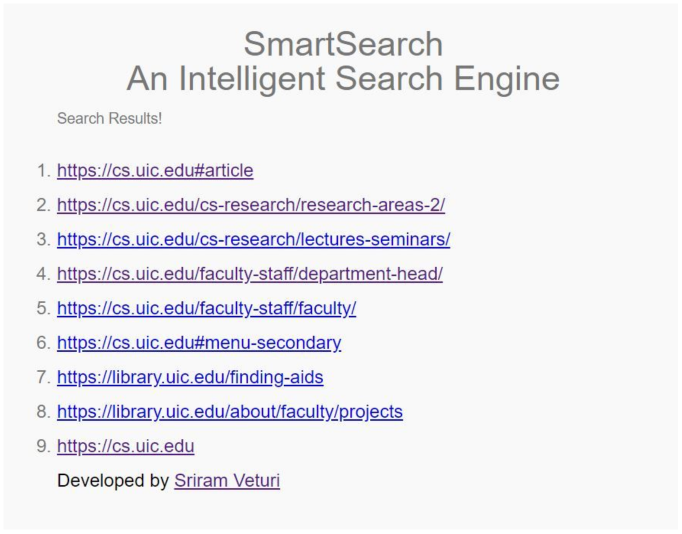

SmartSearch
An Intelligent Search Engine

Search Results!

1. https://today.uic.edu/campus-news/campus-newspaper
2. https://today.uic.edu/campus-news/campus-newspaper/advertising
3. https://today.uic.edu/academics-research
4. https://today.uic.edu/campus-news
5. https://today.uic.edu/resources
6. https://today.uic.edu/contact/communicating-on-campus
7. https://today.uic.edu/contact
8. https://today.uic.edu/campus-news/campus-newspaper/subscribe-to-uic-today
9. https://today.uic.edu/contact/social-media-directory

Developed by Sriram Veturi

4. For the query “Robert Sloan 12346”, the site https://cs.uic.edu/faculty-staff/department-head/ was retrieved which has information about the department head of computer science. So, for this result, I feel the precision should be 1.

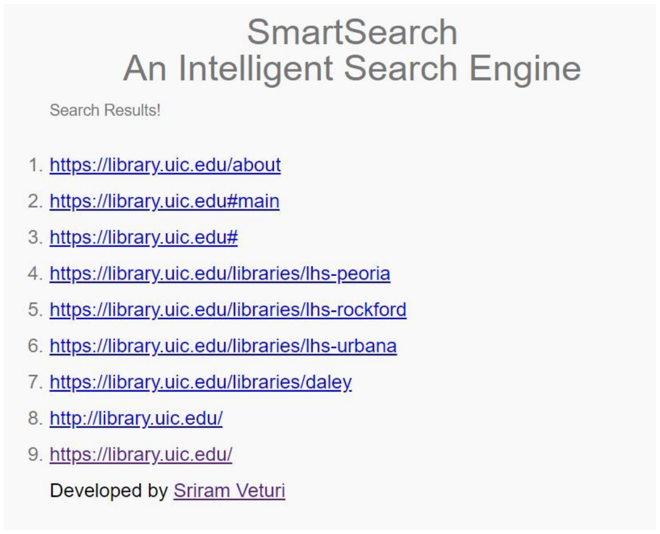

SmartSearch
An Intelligent Search Engine

Search Results!

1. https://cs.uic.edu#article
2. https://cs.uic.edu/cs-research/research-areas-2/
3. https://cs.uic.edu/cs-research/lectures-seminars/
4. https://cs.uic.edu/faculty-staff/department-head/
5. https://cs.uic.edu/faculty-staff/faculty/
6. https://cs.uic.edu#menu-secondary
7. https://library.uic.edu/finding-aids
8. https://library.uic.edu/about/faculty/projects
9. https://cs.uic.edu

Developed by Sriram Veturi

5. For the query “uic library”, all the urls retrieved were in the library.uic.edu sub domain which were expected.


SmartSearch
An Intelligent Search Engine

Search Results!

1. https://library.uic.edu/about
2. https://library.uic.edu#main
3. https://library.uic.edu#
4. https://library.uic.edu/libraries/lhs-peoria
5. https://library.uic.edu/libraries/lhs-rockford
6. https://library.uic.edu/libraries/lhs-urbana
7. https://library.uic.edu/libraries/daley
8. http://library.uic.edu/
9. https://library.uic.edu/

Developed by Sriram Veturi

ERROR ANALYSIS

I tried to analyze the results of the search engine by different means. The most important way was to compare it with Google’s search results. But Google has a database of millions of indexed pages. So, I tried to limit the search results to the UIC domain and then, analyzed the results.

Google’s search results had many PDF and WORD documents in the top results. Since, I did not index the pages with such extensions, my search engine did not display such pages.

After experimenting with different queries, I realized that I had calculated the page ranks of the entire indexed web graph initially and then, used Cosine Similarity to filter out the top 200 pages. So, a lot of highly ranked pages even if they were not very similar to the user’s query were retrieved. So, this could be the reason for some unrelated pages in the search results.

RELATED WORK

There are many projects developed in the similar lines. I could find many research articles related to intelligent Search Engines. Some of them are shown below.

- http://www.academicinfo.net/subject-guides
- http://education.iseek.com/iseek/home.page
- http://www.virtuallrc.com/
- https://www.refseek.com/
- https://academic.microsoft.com/home

FUTURE WORK

For the search engine to be more comprehensive and reliable, I plan to extend the project by adding many more indexed pages to the database and would try to get the PageRank scores from the webgraph created after the Cosine Similarity scores are calculated. In the future releases, this would be my major contribution to the project. Currently, the website is hosted on the localhost which plan to deploy using a public IP so that this project could be shared with as many people as possible.

REFERENCES

https://en.wikipedia.org/wiki/Web_crawler

https://www.researchgate.net/publication/235286070_The_evaluation_of_WWW_search_engines

https://medium.freecodecamp.org/how-to-process-textual-data-using-tf-idf-in-python-cd2bbc0a94a3

https://www.geeksforgeeks.org/page-rank-algorithm-implementation

http://mines.humanoriented.com/classes/2010/fall/csci568/portfolio_exports/bhoenes/similarity.html

https://www.lifewire.com/how-to-search-specific-domain-in-google-3481807

https://www.lowcountrygradcenter.org/the-6-best-search-engines-for-academic-research/