CSC326 Project Report

Name: Hsuan Hsu
Student #: 1000389288

Name: Yefei Li
Student #: 998920413

1. Introduction

The CSC326 Search Engine project, completed by two people, uses the knowledge acquired from the course to build a fundamental web search engine using Python as backend. In addition to using Python knowledge from the course, the team also had the opportunity to explore other web programming technologies such as HTML, CSS, and JQuery/JavaScript. The project is divided into two major parts as with many other projects in the industry: a frontend part and a backend part. At the fundamental level, the frontend portion of the project is responsible for displaying an aesthetically pleasing user interface, processing search queries, and redirecting to pages to display the search results. The backend includes a pagerank algorithm that determines the scores of URLs based on the user's search string. Based on that score, the backend script sends the orders and the links to the frontend to display the result.

2. Search Engine Enhancement

In order to elevate the state of the web search engine the team implemented significant improvements to both the frontend and the backend code of the project. We chose the route where it gave more front end and back end features and left performance alone as we believe that our baseline performance from lab 3 was efficient enough. Although there were definitely more improvements to be made for performance, the status quo is sufficient.

Frontend

The fronted and is significantly improved and is up to par with modern web development technologies. First, Twitter Bootstrap is used to improve the style of the frontend to reflect contemporary web features. Twitter Bootstrap allowed the website to have a convenient navigation bar placed at the top of the page. The navigation bar contains a miniature version of the site's logo and the left end, and several useful links on the right end of the bar. One unique feature that the navigation bar has is that it stays on top of the user's browser as the user scrolls down, which is an important feature that improves the user experience of the website. In addition to the fixed navigation bar, the links on the navigation bar also highlights as the user hover overs it with their cursor. This adds another layer of user friendliness to the search engine. On the search result page, the navbar provides similar functionalities but with a search bar embedded. This search bar allows the user to search further without having to go back to the homepage. It includes the same search functionalities as the search bar on the main page, which will be described next.

Next is the search bar improvement. The search bar has been improved so that it takes zero mouse clicks for a user to complete a search. This means that once the user reaches the website he/she doesn't have to click on the search bar to start typing; the user can just simply start typing on their keyboards and the letters will appear on the search bar. Once the user starts typing, the search bar provides smart suggestions that matches the search query against a list of words in our database. So, the user can use the arrow keys on their keyboards to select the search word without ever using a mouse click. Lastly, to search, the user can press enter/return on their keyboards and the page will be redirected to the search results. The total number of clicks for the user to get to the search results page is zero.

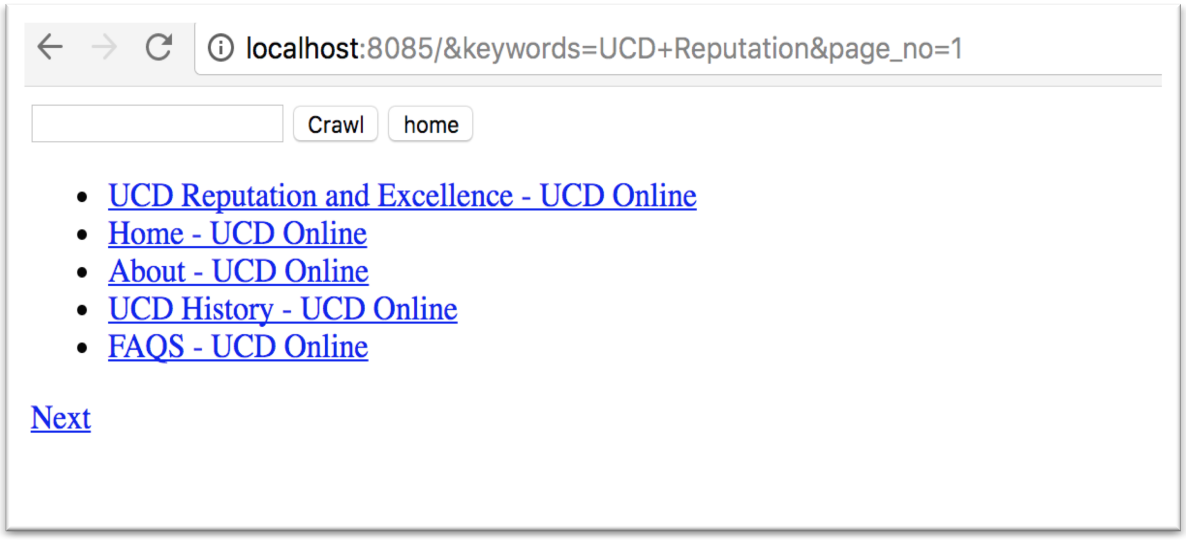
Moving on to the overall aesthetics, the frontend logo and the search bar uses the "animate.css" library to display animations. The logo drops down from the top while the search bar fades in. This makes the website look stylish and trendy, which provides a visual appeal to the users. The users will be more likely to revisit and use the search page again. Adding these animations provides the "wow" factor that distinguishes the team's website from the others.

Note that the Google login feature has been temporarily removed for the submission of this project because the submission requires a "One-Click Deployment" script as mentioned by the course instructor. This means that it would be impossible to know the new Amazon redirect URI at the time of the deployment script for it to register on Google. However, this function can be easily implemented with Google, Facebook, and etc.

Backend

The backend of the search engine application contains the web server that processes the user request and the crawler which indexes the webpages, store the indexes in the database. The crawler craws the links in the text file recursively by looking at all the important tags of the HTML document and following the links in the page with a specified depth. Along with crawling the webpage, it creates an inverted index which is a mapping from word to the urls that contains this word. We store this data in the MongoDB database. The crawler also calculates a score for each page that representing how important this page is which is known as the page rank score.

The reference pagerank algorithm computes a page's score by only looking at the number of links that pointing to it and the page rank scores of the pages pointing to it. It does not depend on user's search query. Our improved algorithm compares uses ulr text match and page title match to boost the page rank score temporarily according to the search query. Thus, we are able to show the webpages with url or title matching one or more of the query token at the beginning.



This Screenshot of the improved algorithm shows that, we were able to provide more relevant links to the user.

We have also considered to use word frequency as a matric to boost the page rank score. However, this approach tends to be noisy compared to using the title and url text. We might have pages with query words appearing a significant of times and its focus is not on any of the query token.

3. High-Level Documentation of Design

To improve the functionality of the web search engine, we had to include externally available public APIs. Fortunately, these APIs are easy to include and only required minimal to no installation. With this simple yet efficient style of implementation, the frontend code does not depend on the backend code at all as all the necessary data is available in our MongoDB database. The backend populates the database while the frontend retrieves data from the database after all the data has been loaded.

Frontend Features

All fronted end features are implemented in “frontend.py” and “frontend.tpl”. “frontend.py” includes search string parsing in the homepage (the *search_page()* function). This parses the search string and sends the words to the *search_result(keywords, page)*. This function retrieves the search results from our MongoDB database and implements *pagination* here. Here it makes sure that each search result page only displays 5 results per result page and if there’s more than 5 results, subsequent pages are added with “Next” and “Previous” buttons that allow the user to navigate between search result pages. In addition, we have several dictionaries populated from our MongoDB database at the top of “fronted.py” as global variables. This way, we can retrieve data from our MongoDB databse only once which leads to a higher efficiency. Next, the search suggestion functionality along with other css improvements are implemented in the “frontend.tpl” file. At the top (header section), we included the necessary CSS dependencies

that make our code work. At the bottom, in the script section, we have JQuery code that allows the page to autofocus on the search bar without typing and the search suggestion code.

Backend Features

In the find_urls function of frontend.py, we try to increment the relevant pages' page score by looking at each of the query word. URL_TEXT_MATCH_WEIGHT and TITLE_TEXT_MATCH_WEIGHT are the max weights we add to a link if all the query words are contained in the url text and the page title. If matching happens partially for a page, we assign the boost of a weight of the total score divided by the number of query words and multiplies by the number of matched words. This logic is implemented in the get_pg_score function.

4. Difference Between Current Design and Proposed Design

Initially, we hoped for this website to only achieve basic functionalities (i.e. minimal CSS used and no JavaScript/JQuery). However, we decided to improve the aesthetics and the backend algorithm to challenge ourselves and to test the knowledge that we learned from this course. The improvements were described above in earlier sections.

5. Testing Strategy

To ensure that the functions work as intended, several approaches to testing were employed. First, the team developed unit tests that would test for corner cases of the functions throughout the course of developing the website. These unit tests were helpful as it made sure the basic functionalities worked as intended and no problem was encountered during deployment. The second method used for testing was benchmarking. We used "Apache Bench" to determine the speed and the performance metrics of the AWS server and our scripts. This is particularly useful because it allowed us to determine which parts of our server/algorithm needed to be improved. Once we determined the parts that needed to be improved, we were able to focus on those areas and improve our code. The last testing method that we used was to get as many people to test out our front end as possible. This is important because the user's perspective is always different from the developer's perspective. Something that the developer finds convenient might not be convenient for the users. Thus, after getting people to use our website, we were able to improve the search engine in such a way that optimized user friendliness. Overall, the combination of these testing methods made our web search engine more efficient and user friendly.

6. Lessons Learned

The team believes that the lessons learned from completing this project is invaluable. Since we were able to learn and practice Python and many other web programming languages, the team members were able to get exposure to the full-stack development process. Also, the team realized that communication and time-management are extremely important in completing a project of this skill. Communication is important because the team members could have

different opinions on certain functions or algorithms. It is crucial to hash these conflicts out as early as possible so everyone is on the same page and focus on one common end goal. If team members do not communicate, it will be extremely difficult to meet the deadline as people would have other courses and deadlines to meet. Most importantly, the most important lessons learned were the technologies that we were exposed to during the development process. The project not only required the use of Python concepts learned in class and tutorials, but it also required extensive research on the internet in order to take the project above and beyond. There were many problems that were encountered and required countless hours of research to resolve. Since front end development is not exactly easy, it was probably the hardest to learn as HTML and CSS are not like conventional programming languages. All in all, the lessons learned from completing this project will definitely help the team members in future projects as well as in the industry.

7. Things would do differently

While the team members are extremely proud of the results of the project, the only thing that the team would do differently is spare more time to do each part of this project. Since we all had different commitments, it was difficult for us to meet frequently for this project. So, we had to divide the parts up evenly and work on them individually. In hindsight, it was probably a better idea to start earlier on each lab and complete them with better confidence. Development wise, the team exhausted all their knowledge on this development and the only way to improve is to know some web development before starting this project; however, that is impossible. Having prior experience with web development would definitely help and make this process easier.

8. Usefulness of Course Material on this Project

Although we learned how to program in Python in CSC326 and the project requires the use of Python, only the syntax and algorithm parts of the course helped with the project. The other concepts in the course were not particularly useful for the project as the project requires concepts from not just python. Also, since we had the freedom of development of the project, we did not necessarily have to implement the concepts that were taught in-class. Overall, the concepts taught in this course will definitely be useful in the future, however, for the purposes of this lab, only the syntax taught was the most useful.

9. Time Spent

Our group completed most of the lab without the help of the lab session. We spend 20 hours for each lab on average. Time was spent mainly on debugging and researching on how to implement various algorithms and improving the frontend aesthetics.

10. Useful Parts of the Lab

We think most part of the lab is useful. We think we might spend more time on learning AWS's other useful features.

11. Useless Parts of the lab

We think the deployment script from the lab2 is a bit useless because we chose to do manual deployment using the AWS console.

12. Other Feedback and Recommendation

The project handout and the course syllabus can be clearer. And it's better to let student know about the marking scheme as early as possible. Overall, we think it's a good course for students to gain experience in python programming and web development.

13. Work/Responsibility Breakdown

One member focus on the frontend development while the other focused on the backend. Frontend development includes modules like UI development and design and web logic. Backend development focused on the implementation of PageRank and persisting the data to database.