University of Toronto
Faculty of Applied Science and Engineering
CSC326: Programming Languages
Project Report

Project Title: Search Engine
December 03, 2017

Name:	Student #:
Jiyang Zhang	1004654304
Yalan Zhang	1002220857
Renxiao Chen	1002057739

1.0 Abstract

- Search Engine is an important tool for our daily life. The design project that the team worked on is search engine that is able to search using keyword input. Results will be computed and displayed as urls. Besides, map searching and image searching are also implemented. This document will first introduce detailed project design and features, and then give commands relates to the project and this course.

2.0 Detailed Design Description

- The project is composed of frontend, backend and deployment. This section will conduct detailed analysis related to the design of the frontend, backend and deployment of this project.

2.1 Frontend Design

2.1.1 Search Suggestion

- The search engine will dynamically give suggestions based on the matching between current input and user search history.
- *e.g: If you once searched for something related to "engineering", the next time you use this search engine, when you type in the "en", it will automatically give the word "engineering" as a guessing.*

2.1.2 Animated Logo

- The logo is implemented to rotating on the main qurey page.

2.1.3 Search Box

- The number of clicks for each search is minimized since search box is also displayed on the result page for user to start a new search.

2.1.4 Map & Image modes

- The user can click on the "All", "Map" and "Image" to choose different modes.
- *e.g: After choose the map mode, if you type in a destination, it will automatically redirect to the google map page with the destination you just typed.*

2.2 Backend Design

2.2.1 Search algorithm

- After analysing the crawler algorithm carefully, we found that, the storing part costs most time and memory. We searched for the better algorithm and got to know the 'Google mapreduce'.(https://static.googleusercontent.com/media/research.google.com/en//archive/mapreduce-osdi04.pdf) Therefore, we improve the creating inverted index algorithm with the help of multiprocessing (reduce_map.py) which simulates part of the function of the Google mapreduce algorithm.
- After testing: algorithm implementing the reducemap took 12s to crawler the http://www.eecg.toronto.edu/. While using the normal algorithm, the performance is 15.3s, which is not much longer than the mapreduce one. We assumed that because the crawling depth is one and we just tested one url, it may not perform much better as we had expected. However for huge search engine like Google, it will run faster and efficiently.

2.2.2 Update the database

- To have a more powerful search engine, we update the database to store the images. While crawling the urls, we store not only the html tag<a>, but also store the url in the html tag <img>. If the user click on the image mode and type the words, the search engine will search the database for the images' urls related to these words and show on the frontend.

2.2.3 Redirect to the google map

- We add the map mode to our searching engine, once the user clicks on the map mode and type the keyword, our searching engine will redirect to the google map result page concerning with the keyword.

2.2.4 Better deployment performance

- In the lab3, while testing the performance using benchmarking, the results are as below:

C = 5   n = 100

time taken for tests	104.805 seconds
total transferred	195900 bytes
requests per second	0.95[#/sec]
time per request	1048.046 [ms]
transfer rate	1.83 [Kbytes/sec] received

- To improve the performance, we decided to use load balancer: '*A load balancer accepts incoming traffic from clients and routes requests to its registered targets (such as EC2 instances) in one or more Availability Zones. The load balancer also monitors the health of its registered targets and ensures that it routes traffic only to healthy targets. When the load balancer detects an unhealthy target, it stops routing traffic to that target, and then resumes routing traffic to that target when it detects that the target is healthy again.*' ref: http://docs.aws.amazon.com/elasticloadbalancing/latest/userguide/how-elastic-load-balancing-works.html

  As for testing, we launch two instances in different subnets(`us-east-1a & us-east-1d`) and use the load balancer. The results are below:

n =100   c = 100

time taken for tests	69.611 seconds
total transferred	1357000 bytes
requests per second	1.44 [#/sec]
time per request	302.046 [ms]
transfer rate	1.09 [Kbytes/sec]

n = 2000 c = 2000

time taken for tests	298.954 seconds
total transferred	326140 bytes
requests per second	6.69 [#/sec] (mean)
time per request	298954.429 [ms] (mean)

transfer rate	1.07 [Kbytes/sec] received

At the final deployment script, we launch 4 instances and each subnets 2 instances in order to make full use of the load balancer.

3.0 High Level Diagram of Project's Code

3.1 Backend part

- **Relating files:** Backend_lab4_new.py**,** pagerank.py, reduce_map.py, storage.py.
- While running Backend_lab4_new.py, the crawler will craw the urls in the 'urls.txt'. Since the mapreduce algorithm does not perform much better in this case we decided not to use it. While crawling the urls, the backend computes the score of each url and the storage.py will store the inverted_index, document index, lexicon and page rank into the database(in our project the database's name is 'eecgdata').

3.2 Deployment part

- **Relating files**: deployment.py, config.txt, urls.txt, termination.py
- User should write the aws access key id in the first line and aws secret access key in the second line. While running the deployment.py it will automatically get the access key and secrete key and create security group, create key pair, launch instances, upload all the files and then install the libraries. After that, it will run backend to update the database and run frontend finally. At last it will create the load balancer and add them to it. (creating load balancer part runs well in my own laptop while fails in the school computers, I think it may because of the different edition of 'boto' module)

3.3 Frontend part

Relating files: FrontEnd_lab4.py, getdata.py

- When user inputs the keyword, the frontend will get data from database with the help of getdata.py

4.0 Difference

- I think at the beginning of the project, our proposed searching engine is just a simple one and can just search simple words. However, at end, we implemented many exciting features to make the frontend much more friendly to users.

5.0 Testing Strategy

- This section will explain the strategies our team used for testing and debugging, and how we avoid corner cases.

5.1 Frontend Testing Strategy

- When tested the usability of user interface, in order to improve the effectiveness of testing and debugging, we customized test cases for each function. For example, when tested the pagination, we created a fixed url list to see to result of pagination function. Fixed url list avoided setting up connection between frontend and backend every time to compute the url list. This method saved time and gave a intuitive judgment of the functionality of pagination.
- To avoid corner cases, the method our team used to test frontend was carefully test through each route and all possible user input.

5.2 Backend Testing Strategy

- In terms of testing the database, we launched website written by ourselves which ran in the localhost and used the crawler to crawl it. Since the contents on the testing website were not too many and it is easy to figure out the correctness of the results.
- Just the same with what we have stated above, in order to understand the improvements after implementing the load balancer, we used the benchmarking tool to analyze the performance of the server with different concurrence and numbers of requests. And it proves to be satisfactory.

6.0 Lessons Learned From This Project.

- **Doing projects as a team must use Github!!!!**
  Without using git decreases the efficiency of the team. Our team didn't use merge tracing tools for this project. Hence, every one spent a lot of time on finding the difference of each release of each file.
- Using wise testing strategy could improve the efficiency of testing. As mentioned in the section above, our team used various test methods for this project. For example, since this project consist of many files, write different test cases for each file first instead of combine all files together increase the debug efficiency a lot.
- Communication is a prominent part of this project, and teamwork plays a delicate role within this part. It either promotes productivity and a trustworthy relationship, or it backfires and fails everyone involved. Our team have a good communication and team dynamic. Especially for the frontend fart, both teammates show respect to each other's idea and suggestion. This results a good team relationship.
- Ability of self-learning is important for this project. Material from the course only gives a basic knowledge of coding using python. When it comes to algorithm (backend) and web design such as html, css (frontend), we need to learn these by ourselves. Our abilities of self-learning are quite improved through this project.
- We should have good coding habits and skills such as how to quickly find the bugs ,how to get help from internet etc.

7.0. What Would I Do Differently

- Organizing everything among teammates is extremely hard only using email and lack of communication made it difficult to understand each other's codes. If we had a second chance, we would prefer to reserve some time slots every week to work together and utilize github to manage all materials.

- If we had more time on this project, we may consider to add some virtual advertisements to make our application more commercialized and realistic. Also, we want to add some useful plug-ins to our searching engine program, such as extracting QR codes and recognizing words or views in pictures. It is also under consideration that we can put the quick links of some commonly used website like "Youtube" and "Facebook" to broaden the usage of our app and increase the usability. Finally, we need to pay more attention to HTML stuff and make our app more aesthetic and reliable.

- In terms of the backend, we may improve the page rank algorithm. In my opinion, the score of each web page is not only concerned with how many websites it is linked to, but also the frequency a particular keyword appears.
- Secondly, we may try to implement the phrases searching, for example, get the score of each word in the phrases and then post the urls list based on the page rank algorithm.
- Moreover, we may try to enhance the server by using the elastic group to launch instances depending on the number of requests and concurrency.

8.0 Material From The Course Related To Project

- **frontend**
- As mentioned before, meterial form the course are mainly about coding in python. Those knowledge are very correlated to the bottle framework and some algorithm (such as pagination) used for frontend. However, we were not taught anything about web design such as HTML, CSS and JavaScript in class. Even the basic knowledge were introduced during tutorials, obviously they are not enough to fulfill all the requirements for the web design. For example, animated logo, it is difficult to implement it only use HTML and python knowledge introduced in lectures and tutorials.
- **Backend:**
- As far as I am concerned, the basic python coding knowledge is very important to backend, since the list, dictionary and other data structure are very fundamental and quite useful.
- Secondly, the persistence related knowledge such as sql database is also of great importance to us since we have to deal with data base in the project.
- Thirdly, the object-oriented programming in python which is helpful because while we learn the new python module or library, it is a class at most time, so learning object oriented programming is of great help to our project as well.

9.0 Time Consumption

- About 20 hours for lab1,2,3;
- 30 hours for lab4 (actually more hours are needed)

10.0 Project Feedback

- **Which part of the project you think is useful and you believe the labs should spend more time on it.**
- I think the backend and algorithm part is useful because if the project requires the running time and memory the backend use, it may be more challenging and good for us to understand the data structure and algorithm
- **Which part of the project you think is useless and you think it should be removed from**
- Deployment on aws is useless because it just using aws and no help to our coding ability. What’s more, it may cost a lot of money (e.g. I have spent 30 USD on aws)

11.0 Course Feedback

- More knowledge about web design may be introduced during lecture and tutorials to lessen students burden of learning after class on their own to finish the project.

12.0 Responsibilities Of Each Member

Our team has a good teamdynamic and works of this project were divided equally. The table below will roughly introduce the responsibilities of each member.

	Frontend	Backend	Deployment	Report	Testing
Jiyang Zhang	●	●	●	●	●
Yalan Zhang	●			●	
Renxiao Chen	●				