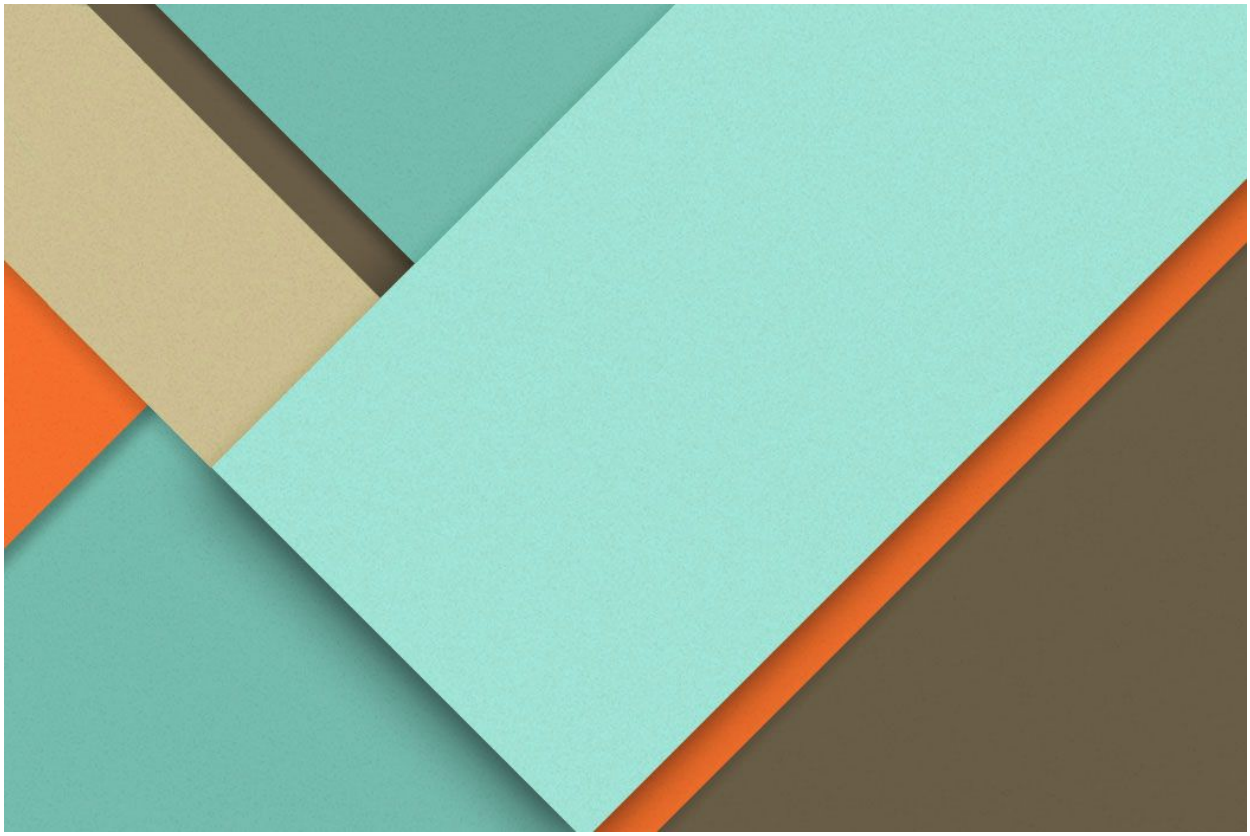

Joogle Feature Report (Lab 4)

11.30.2018

Lin Xuan Zhang: 1001278850

Jorge Ramos: 1001484663

Describe the design of your enhanced search engine in detail. If you enhanced an algorithm, describe the different candidate algorithms and how they are different from the baseline implementation in Lab 3, and describe the quantitative metric you use to judge the merits of the candidates and how you chose your final candidate.

- **Multi word search**
    - We devised our own algorithm for multi-word search. This can't be compared to our implementation in lab3 since it was only for single word search.
    - Here is the algorithm: Given a string of words, we first check if any document contains all the words being searched, not necessarily in sequential order. The resulting pages that contain all the words will have their pageRank score boosted by 10*(number_of_words).
    - The factor of 10 was determined experimentally to be a good candidate.
    - In the next step, we remove a word from the end of the search string, and look for a pages that contain all the remaining words. The resulting pages that contain all these words will have their page Rank boosted by 10*(number_of_words - 1).
    - We remove words and repeated the above procedure until there is only 1 word left.
    - We then go through every individual word and find the pages that contain each of these words, this concludes the algorithm.
    - If a page has already been found and inserted into our results data structure, it won't be considered again in later searches. This is to ensure we don't have duplicates in our search result.
    - We chose to implement multi-word search because it's a basic functionality each search engine should have. A search engine that only support one word search is not very useful.
- **Autofill**
    - When the user starts typing a word, a dropdown will appear showing possible words to complete the word the user is typing.
    - This is a new feature that didn't exist in lab3.
    - Here's the algorithm: every time user starts typing, the word typed so far is recorded, and a javascript callback function is invoked. This javascript callback will query a dictionary and find 5 words that begin with what the

user has typed so far. The 5 options will be shown in a dropdown. Ther user can click on it or use the arrow+Enter to complete the autofill.
    - The dictionary comprises of approximately 1500 words.
- **Math expression interpretation using the Shunting Yard Algorithm**
    - The Shunting Yard Algorithm is a popular method invented by Edgar Dijkstra for parsing mathematical expressions in infix notation into a postfix expression
    - It uses a stack to read / hold operators and numbers
    - We chose this algorithm because it is a popular algorithm, it is relatively simple to implement, and Jorge has past experience implementing this algorithm.
- **Animated Logo using SVG**
    - SVG stands for Scalable Vector Graphics
    - SVG defines vector-based graphics in XML format
    - Every element and every attribute in SVG files can be animated
    - Advantages of using SVG over other image formats (like JPEG and GIF) are:
        - SVG images can be created and edited with any text editor
        - SVG images can be searched, indexed, scripted, and compressed
        - SVG images are scalable
        - SVG images can be printed with high quality at any resolution
        - SVG images are zoomable
        - SVG graphics do NOT lose any quality if they are zoomed or resized
        - SVG is an open standard
        - SVG files are pure XML
    - We chose to implement this because the Logo is one of the first things that the user sees. We want to make a good first impression to the user.

Indicate the difference of your proposed design and completed design if there is any. If the search engine is completed differently than the proposed design, explain why.

- We finished what we proposed. The only difference is autofill. Due to the content size of an actual dictionary (~270k words), it would make our app extremely slow and impossible to use. The reason for the slow query is because we use a linear search. To make the query faster, we could store in a tree structure instead, but

we ran out of time. As a result, we limited the dictionary to the 1500 most popular words in English.

Explain your testing strategy during the development. Describe how you identify the corner cases.

- Python testing script:
  - Call the crawler and assert that the inverted index value are set probably by using know correct values.
- Manual testing
  - We would pick words that we know belong to certain webpages and verify our search engine's result by searching them.

Lessons learned from this project.

- Technical
  - Learned Bottle framework
  - Session management using Bottle framework
  - Web crawling by learning from the provided crawler.py file
  - Google's pageRank algo using the provided sample pageRank.py file
- Non-technical:
  - Communication. Since we splitted up the work for lab 3 and 4, we had to communicate constantly to keep each other updated on the progress. Otherwise repeat work could potentially be done. We also discussed all the possible implementations for all the features and made a decision together. Useful skills in industry to keep each other in the loop and increase productivity.
  - We used Git for VCS, industry standard tool
  - Got stuck too long on Google login during lab2, should've skipped and did other parts first.

Describe what you would do differently if you had to do it again. What would you do if you had more time. Did any parts take longer than you thought, and Why?

- What we would do differently:
  - Split up work more. We didn't because we both wanted to learn, but school is busy and we don't have that much time to spend on the labs.
- Which part took longer?
  - Google login on AWS, because there was no explanation or good documentation online on how the redirect should work.

How the material from the course help you with the project.

- The course material provided us with basic knowledge of Python. Otherwise, the course and labs were not that related.

How much time it takes for you to complete each lab outside the lab sections.

- Lab 1: 3 hours
- Lab2: 15 hours
- Lab3: 8 hours
- Lab4: 10 hours

Which part of the project you think is useful and you believe the labs should spend more time on it.

- The PageRank algorithm was useful. It was interesting to learn how it works in order to use it in the application. It forces students to learn it.

- The labs should spend some more time on front-end. Not much emphasis was placed on front-end, but you should. It's what users see and is a useful skill

Which part of the project you think is useless and you think it should be removed from the labs when this course is being offered in the future.

- Google login on Amazon AWS (even the TAs didn't know how to do it). Either TAs figure out how it works or remove it.

Other feedback or recommendations for the course.

Please update lab instructions. It's a consensus that they're unclear and outdated. Also, have the TAs actually do the lab. It seemed like they are understanding the lab for the first time when they are in the lab rooms.

Responsibilities of each member. If you believe that workload is distributed unequally in your group, you may describe the situation in this section.

Work was evenly distributed.

- First 2 labs completely together.
- Lab 3: Jorge did more frontend, Martin more backend
- Lab 4: Jorge did shunting yard and animation, Martin did multi-word search and autofill