Milestone 3 Report

Team: akatsuki
Shehrin Fowzia, 60751818
Yasna Nekooei, 34021380
Luis Najera, 79038761
Juan Pablo Zavala, 47913183

Test queries

1. Cristina Lopes

- Very good time performance around 10 ms but poor ranking performance. The user probably expects to see Professor Lopes' faculty profile website in the top results but a lot of web pages include the name a lot as bibliographic references. Changed the ranking computation to use cosine similarity so larger documents are not heavily favored.

2. Professor Wong Ma

- Bad ranking performance despite using cosine similarity to score each webpage. Changed the code to also index title words from webpages and gave them a high weight when ranking. Using this heuristic improved the quality of the results and websites at the top are likely to be the most relevant for the user.

3. ACM

4. Machine learning

5. Master of computer science

- Ranking performance was not ideal for this query. The MCS website did not appear in top results due to the high number of webpages that had 'computer

science’. To solve this, headers and bold words were also indexed and saved in the postings as important text to be used in ranking and this fixed the issue.

6. Master of software engineering

- Ranking performance was good but the response time jumped to 450 ms with this query. We implemented a function that processes the query to remove stopwords and achieve better performance. In this case, removing ‘of’ from the query improved the response time to 250 ms.

7. artificial intelligence uci

8. Ice cream

9. arachnophobia

10. uci computer science

11. uci donald bren school of information and computer science

- This query still did very poorly in terms of efficiency with 600ms. Despite removing stopwords, the bottleneck occurred when calling the eval() function many times to turn each string representation of the posting list into a python object. Eval() was changed with json.loads() and it improved runtime performance significantly to 150 ms.

12. Where can I find the nearest ice cream shop near Donald Bren Hall

- Very fast performance but very low results due to intersection pruning. Although the query contains many tokens and each have a large posting list, the amount of postings that contain all words turned out to be zero. Changed the code of the intersection function to look for postings with most/one of the query terms if the amount of contenders is below 10.

13. To be or not to be

- Poor ranking performance since the entire query was removed using a library that filtered stopwords. To deal with this, we decided to improve our stopword filter function to only modify queries with two or more tokens, and we established our own list of stopwords instead of using a library. The list only includes the most common articles, prepositions, and conjunctions (no verbs).

14. The to of in on

- Special case since all tokens are in our stopword list so it was removed when it ran. Modified our stopword filter function to only keep the first token under the assumption that a query full of articles and prepositions is actually not a meaningful phrase in any case.

15. In

- Poor runtime performance just above 300 ms. Changed the ranking function to handle cases in which the query is just one stopword. For these, finding the intersection of postings is skipped.

16. The

- In this case, the runtime performance was still barely above 300 ms due to the length of the posting list for this word. Changed the ranking function code to use a priority queue with heap implementation, instead of saving the scores in a dictionary and sorting the large list of scores. This improved runtime to 230 ms.

17. We

18. the be to of and a in that have I

19. I want to know about the master of software engineering program

- Did well in terms of efficiency but the quality of results was questionable. Adjusted the weights/coefficients for title and header words to improve ranking

performance while preserving the effectiveness of the search engine with previous queries.

20. I am wondering maybe this is the longest query that someone would type but one cannot ever know for sure