CS 121 Winter 2021
Assignment 3 Milestone 3 Report

Seong Su Park 63888411
Huikang Huang 58039679
Anuj Ramaswamy 39134771

List of 20 Testing Queries:

1. cristina lopes
2. machine learning
3. ACM
4. master of software engineering
5. which uci professor is the best among all the ics professors in uci 2021
6. to be or not to be
7. Richard Pattis
8. information retrieval
9. Provide implementation
10. After the second colon
11. Such algorithms have practical implications
12. Our research crosses countries and continents on a regular basis, for instance comparing the impact of innovation practices and resulting technological advances in different countries and societies
13. Department of Computer Science
14. he is and will be one of the best
15. as of is to the
16. a world-class research environment that goes well beyond the core areas of computer science
17. An introduction to Web information retrieval, including crawling, indexing, retrieval, and concepts of classification and clustering of text documents
18. Search engine
19. Alex Thornton
20. Science, technology, engineering, and mathematics

While we are developing our search engine, queries with a lot of stop words and really long queries, for example query **#5, #6, #12, #14, #15, #16, #17,** were doing poorly in terms of query time. We figured that the main reason for their long query time was due to their large number of postings our program has to retrieve from our index file. Specifically, since stop words are very common, they would have the most postings in their posting list. The more postings a term has, the longer it takes to retrieve all the postings, and hence longer query time.

As a solution, we decided to cache those common stop words in memory. First, we came up with another text file that contains all the common stop words that we found. Then, before our program asks for queries from the user, the program would build an in-memory dictionary that retrieves and stores the posting lists of all the common stop words mentioned in the new text file. Thanks to the in-memory dictionary, the query time becomes much faster for longer queries and queries with a long of stop words. Instead of retrieving the posting list from the index text file, if a term is in the cached dictionary, its posting list will be retrieved directly from memory, saving us a lot of time. On top of common stop words, we also added some of the common words found in the given ics.edu domains.

As for search performance wise, we also had queries, such as query **#4, #8, #9, #10, #11, #13, #20,** that were doing poorly in the beginning. For those queries, the search result would show web pages that contain those query terms, but those query terms were not necessarily next to each other. For example, if our query is “master of software engineering”, a result web page could contain all of these four terms, but they would appear separately in the web page, which is not so accurate.

In order to improve our search performance, we have implemented bi-gram indexing and utilize the bi-grams during searching. So in addition to the regular unigram terms, we have also added bigram terms in our inverted index. During search time, when the program receives a query, the program would break the query down into two sets, one with bigram terms, and one with unigram terms. The program will first try to compute the searching result using the bigram terms, if the program cannot find anything using the bigram terms (for example the query is a single term), then the program will use the unigram terms to compute the result. Implementing the bigram helped us improve our result significantly. For the same query “master of software engineering”, the result web pages would contain this exact phrase of four words connected next to each other.