Josie Libbon

Project 5 Report

Processing files implementation: **Option 2 - Automatic reading of arbitrary number of documents**

My implementation uses a simple hashmap that reads in an arbitrary number of text files from a designated directory. A main function that first calls the training process to read all the documents and create the hash table/populate it with all the words from the text files. It then removes words that occur in all of the files (stop words). Once the training phase is over, it enters the search (retrieval) phase to search for the keywords and find the documents that contain these keywords. Main prompts the user for the query set (search phrase entered on one line) or prompt to exit: "Enter Search String or X to Exit". If the user enters "X," main will exit the program, otherwise it calls the read_query function to read the query set. The program then computes the score, prints out the documents in order of relevance (i.e., descending order of scores), prints out the document *tf-idf* scores to search_scores.txt, and returns to the main prompt (i.e., to prompt for another search or to exit).

How the Algorithms interact:

https://lucid.app/lucidchart/e2b4a5bc-ffdd-40f8-ae75-5f224bb93e0c/edit?viewport_loc=-1941%2C-1383%2C4878%2C2400%2C0_0&invitationId=inv_d1193fea-a2fd-40ec-af8a-4e838e0ee75d

(also pictured on next page)

```
hash_table_removeatPosition
              ↑
         findPosition
              ↑
       hash_table_remove
              ↑
        num_of_removes
              ↑
            dFreq
              ↑
          stop_word
              ↑
hash_table_insert ← createNewNode ← contains ← readFIle ← training
                                                              ↑
                                          main → get number of buckets
                                           ↓
                                      initializeTable
                                           ↓
                                      if input is X, exit program
                                       training
                                           ↓
                                       readquery
                                           ↓
                      rank → tf_idf → getCount → dFreq
                                           ↓
                                       relevance
                                           ↓
                                     printRelevance
                                           ↓
                                      bubbleSort
                                           ↓
                                     resetRanking
```