CMSC 676 Phase 4 REPORT Akarsh Kashamshetty

Preprocessing: Used the same preprocessor from the previous phases which converted the text to lower case and removed any unnecessary chars.

Algorithm:

1. I have used the dictionary list and postings file from the phase3 and loaded them from disk to memory. And created two dictionaries (HashMap), the dictionary list dictionary contains key as word and value as list of its freq in whole corpus and its first occurrence in postings file and created a dictionary from postings file where key is the counter or line number, and value is a list of document id and its weight in that document.
2. Now I used a list to store the words from the command line.
3. Next traversing through each word in word list from command line. I have done the preprocessing and searched for it in the index dictionary (dictionary list). If it is not present in the dictionary I continue to next word. Else, I get the value of the freq and location from the dictionary and then using a while loop I have inserted the freq number of document id and weights list into a new dictionary of words where the key is the word and value is a list of lists to hold the document id and weights of the term.
4. Once I have populated my word dictionary with all the document id and weights with respective to words. I traversed through all the values of the word dictionary and created final document HashMap with document and sum of the weights of the word.
5. Sorted the final hashmap by the sum of weights in descending order and displayed the top 10 in the output.

Extra Credit : Algorithm(query term weights):

I used the keyword “Wt” for differentiating the queries. Then if query term weights are passed then I multiplied the weights of the terms with given weights. And rest of the algorithm is same.

Data Structure used: HashMaps, Lists.

Output(example):

A picture containing text

Description automatically generated

Text

Description automatically generated with medium confidence

Text, application

Description automatically generated with medium confidence

We can see the difference in list of documents when weights are given giving priority to words with high weight.

Extra Credit : The query engine works even with multiple terms given. For example

Text

Description automatically generated

Given multiple terms – diet, international, affairs ….etc

Time complexity: O(nm) where n is the number of terms given and m is (term \* freq). As the dictionary and postings are constant sized dictionaries.

Extra Credit : Comparing the weights with log and without log.

with log

Text

Description automatically generated with medium confidence

without log

Chart

Description automatically generated with medium confidence

log

Text

Description automatically generated

without log

Text

Description automatically generated

we can see the difference in the weights. For example for file 28.html with log we are getting result as 0.33 and with out log we see weights as 10.99.