PD2 2023

HW6

CSIE@NCKU 2023
Deadline: 2023/5/29 11:59pm

Homework Description:
Same as HW5, we have the same format of input, including corpus files and query files. We also follow the same principles in HW5 to process corpus and queries.
For the output, we change to output "top-k" results for each query. k is in the range of [1, 3].
Also same as previous homeworks, use 'cout' to output in the console. We will use shell operator ">" to copy output to a created file named "result". For the example input, the output should be:

```
1 5 4
3 4 5
5 3 1
```

In this case, k is set of 3. And each line contains k S_ID, sorted by "similarity" in the descending order.
Here the similarity should be calculated by word vectors according to IDF (Inverse Document Frequency) value. Suppose we have totally 100,000 strings in the corpus, and now a keyword, such as "quantum", appears in 100 difference S_IDs. The keyword "quantum" have its IDF value equals to lg(100000/100) = 3. You can google/ask chatgpt 'Inverse Document Frequency' for the details.

As we can find, for a keyword with high frequency appearing in the corpus, it means that the keyword is relatively common and cannot give good discrimination to indicate if it really matches user's need.
For example, if you give a query "quantum for students", we could expect that the keyword "quantum" is more important than "students", and "quantum" and "students" are certainly both more important than "for". It is because we may get idf(quantum) = 3 and idf(students)=2 and idf(for)=1.

In HW6, we would like to "sum" idf values of all query terms if it appears in a corpus string, and return the top-k strings (shown its S_ID) with maximum idf_sum.
Different from HW5, it could be "partially matching", meaning that some query keywords may not appear in the top-k strings. For example, suppose that we have the query term "quantum for students". The following strings may get idf_sum:

1. "Quantum computing is the future way of computation" -> idf_sum = 3.
2. "Students should follow the same rules" -> idf_sum = 2.
3. "Quantum could be good for computing" -> idf_sum = 4.

Suppose you need to return top-1 result, you will return S_ID set {3}. Note that it is not fully matching.

You don't need to remove stop words. In addition, no word stem needs to be considered. Please let "capital letter" be equal to its small letter. ~~For each query, the result should be ordered according to S_ID in the ascending order.~~ (This statement is incorrect!)

If the k-th string and (k+1)-th have the same idf_sum, please output the one with smallest S_ID. Therefore, we expect that we always output k results for each query.

If the top-k result contains strings with the same idf_sum, please sort them according to their S_ID in the ascending order.

Note that sometimes a keyword may appear more than once in one string of the corpus. But you should skip such cases and consider it only appears once in the string. We won't count the term frequency, meaning that we only consider whether a keyword appears in the string or not.

For some queries that contain the identical keywords (because query terms are randomly selected from the corpus, we may face this situation), please count the same keyword "multiply" according to the its frequency in the query. It means that you can just consider each keyword in the query as different keyword without considering their differences.

Please only consider to output results with idf_sum > 0. Suppose that for

some situations, top-k results contain cases with idf_sum=0. For example, a keyword may only exist in the corpus in one string with S_ID=100, and the top-3 result for queries containing this keyword should only report {100}. We won't need to consider the case that idf_sum = 0 for all strings.

Starting from this HW, the system resource, such as execution time is limited. We will terminate all executions with time larger than 30 mins. Of course, you will get basic score (60% score of HW6) if you can pass at least 3 small testcases within 30 mins.

The Input Argument: corpus_file query_file value_of_k

Deadline:
2023/5/29 11:59pm (Monday).
HW6 should be submitted before the deadline. No excuse to submit your code after the deadline. TA will copy your code at 2023/5/29 00:01. If your code is not in the hw6 Folder, you will get score '0'.

Environment:
1. uname -a
   Linux version 5.15.0-67-generic (buildd@lcy02-amd64-116) (gcc (Ubuntu 11.3.0-1ubuntu1~22.04) 11.3.0, GNU ld (GNU Binutils for Ubuntu) 2.38) #74-Ubuntu SMP Wed Feb 22 14:14:39 UTC 2023
2. IP: 140.116.246.230
3. Please remember you should connect to the server with a NCKU IP. Make sure you use NCKU VPN or connect to the server in our school.

Spec:

Note:
1. It is encouraged that you can "use chatGPT".
2. A string may contain more than 200 words.
3. Any punctuation mark can be removed.
4. The corpus may be larger than 100 MB.
5. The range of k is [1,3].
6. More than 10,000 queries could be used.
7. In each query, the number of keywords will be in [1,3].

8. STL map or hash_map is a good implementation for indexing.
9. Check the TRIE structure if you have time.
10. S_ID is not always ordered, but it is a positive integer value.
11. Remember the strict output ordering policy. We will examine your correctness by our shell script without any excuse. In this homework, the output is sorted by their idf_sum.
12. Please list the result to console. We will use the shell operator ">" to copy all your homework output to a created file named "result".
13. You need to declare the executable file named "**hw6**" which will be generated by using your makefile with "make all".
14. The csv file name will be given as the input argument without any exception of file handle error.
15. For the convenience of checking your homework, **output** must not contain any exceptional character, otherwise your score will be deducted.
16. The execution efficiency will also be counting. Top 10% submissions will get the bonus of 20% score.
17. The memory usage will also be counting. Top 10% small usages will get the bonus of 20% score.
18. If you know how to use scp (you may use Windows-based PowerShell), you could try scp in powershell like:

```
scp hw6.cpp ktchuang@140.116.246.230:~/hw6
```

How to Submit:

Please pay attention to the following instructions when submitting homework:

1. Under your account folder, create the folder named "**hw6**".
   (*Please note that you must pay attention to the correct capitalization. If we cannot correctly copy the foler hw6, you will get score '0'.)

2. Put every necessary files under the folder, including :
   i. Your **main program**, such as main.cpp, hw6.cpp, main.h, program.h, etc.
   ii. **makefile** (*Name your executable file as "hw6")
3. Make sure it works normally under the folder and all files are in the correct path.

Examples of files in your folder:

```
netdb@2023pd2:/home/vs6112030/hw6$ pwd
/home/vs6112030/hw6
netdb@2023pd2:/home/vs6112030/hw6$ ls
hw6  main.cpp  makefile
```

How we execute:

```
netdb@2023pd2:/home/vs6112030/hw6$ make all
netdb@2023pd2:/home/vs6112030/hw6$./hw6 corpus.txt query.txt 3 >
result
```

Optional: You could consider to change your data structure. Let every keyword be encoded as an integer and comparing “integer equivalence” is more efficient than “string equivalence”. Finally, HashMap will be highly necessary for you to process this HW.