PD2 2023

HW7

CSIE@NCKU 2023

Deadline: 2023/6/21 11:59pm

Homework Description:

In HW6, we only calculate the similarity based on idf_sum. In fact, another important concept, namely 'term frequency' is also introduced to measure the discrimination power of documents to match user need (query). Basically, term frequency of a keyword k_i is generally defined as the occurrence of the keyword in the document d_j, says $tf(k_i, d_j)$. For example, for document d_1:

"Quantum computing is the future way of computing. In addition, Quantum is also an important technique for scientific computation".

In d_1, tf('quantum', d_1) = 2, tf('computing', d_1)=2, tf('important', d_1)=1.

Generally, we can measure the similarity by combining tf and idf, that is: $rank(d_j, q_m)$=sum(tf*idf) for each keyword in query q_m.

However, it is known that a long document tends to be selected because it contains more keywords with high term frequency. So that the measurement using $rank(d_i, q_m)$=sum(tf*idf) is fragile against SEO (Search Engine Optimization, which is generally used to make a website have a high score to be ranked as top one in the search engine result even though it is not a good website for the given query).

Therefore, in HW7, we specifically try to "normalize" the important of tf of k_i in d_j:

new_tf(k_i, d_j)=(#k_i_in_d_j / #all_words_in_d_j).

In addition, to minimize the impact that long documents generally have more different keywords and are likely to easily get higher ranks, we also try to summarize only some top idf-ranked keywords without taking all keywords in query q_m into account.

In HW7, we fix to select top-3-idf keywords of q_m in d_j.

Formally, our measurement of rank d_j to q_m is:
$rank(d_j,q_m)=sum(idf(t_x, d_j)*new\_tf(t_x, d_j))$, where t_x belongs to keywords of q_m and is top-3-idf keywords of q_m in d_j.

For the output, we 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 "rank" in the descending order.
Here the rank should be calculated according to $rank(d_j,q_m)$ aforementioned.

For example, suppose that we have the query q_m as "quantum computing for each student". The following strings may get idf_sum rank as:

1. "Quantum computing is the future way of computing. In addition, Quantum is also an important technique for scientific computation."
2. "Students should follow the same concept of computing. But the perceptive of each student will be different for each concept."

Suppose idf(Quantum)=4, idf(computing)=3, idf(for)=1, idf(each)=2, idf(student)=3. For S_ID 1 (i.e., d_1), top-3-idf keywords are 'Quantum, computing, for' and for S_ID=2 (i.e., d_2), top-3-idf keywords are 'student, computing, each'.

Note that #words in d_1 is 19, and #words in d_2 is 20.
$new\_tf(Quantum, d_1)=2/19$, $new\_tf(computing, d_1)=2/19$, $new\_tf(for, d_1)=1/19$.
The $rank(d_1,q_m)=(2/19)*4+(2/19)*3+(1/19)*1 \sim 0.79$

$new\_tf(student, d_2)=1/20$, $new\_tf(computing, d_2)=1/20$, $new\_tf(each, d_2)=2/20$.
The $rank(d_2,q_m)=(1/20)*3+(1/20)*3+(2/20)*2 \sim 0.5$

When you need to return top-1 result, you will return S_ID set {1}.

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.

When 2 keywords have the same idf, and we need to select one of them to be included in top-3-idf keywords, please select the one with larger new_tf*idf.

If the k-th string and (k+1)-th have the same rank value, 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 rank value, please sort them according to their S_ID in the ascending order.

In this HW, a query won't contain a keyword more than once.

Please only consider to output results with rank > 0. Suppose that for some situations, top-k results contain cases with rank =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 rank = 0 for all strings.

For some queries, we cannot get k results with rank >0. For such cases, please use "-1" in the output.
For example, suppose that you only get rank >0 with S_ID=5, and you want to return top-3 results. You will return:
5 -1 -1
If no any string with rank >0 for top-3 results, you return:
-1 -1 -1

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 HW7) if you can pass at least 3 small testcases within 30 mins.

The Input Argument: corpus_file query_file value_of_k

Deadline:

2023/6/21 11:59pm

HW7 should be submitted before the deadline. No excuse to submit your code after the deadline. TA will copy your code at 2023/6/22 00:01. If your code is not in the hw7 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, 10].
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 rank.
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 "**hw7**" 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 hw7.cpp ktchuang@140.116.246.230:~/hw7
```

How to Submit:

Please pay attention to the following instructions when submitting homework:

1. Under your account folder, create the folder named "**hw7**".
   (*Please note that you must pay attention to the correct capitalization. If we cannot correctly copy the foler hw7, you will get score '0'.)

2. Put every necessary files under the folder, including :
   i. Your **main program**, such as main.cpp, hw7.cpp, main.h, program.h, etc.
   ii. **makefile** (*Name your executable file as "hw7")
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/hw7$ pwd
/home/vs6112030/hw7
netdb@2023pd2:/home/vs6112030/hw7$ ls
hw7  main.cpp  makefile
```

How we execute:

```
netdb@2023pd2:/home/vs6112030/hw7$ make all
```

```
netdb@2023pd2:/home/vs6112030/hw7$./hw7 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.