CENG 778 Spring 2015
Programming Assignment
Due Date: 15 April 2015

In this programming assignment, you will simulate the document-at-a-time (DAAT) query processing with WAND optimization. In particular, for each query, you will fetch the posting lists (from the disk) that correspond to the query terms from the index. Then, as we assume disjunctive semantics (which means that we compute the ranking score for each document that includes at least one of the query terms), you will first do the typical DAAT processing until you score the first *k* documents and insert them into a heap. From this point on, you will switch to WAND processing, which means that you will fully compute the score of a document *only if* it can exceed the score threshold (i.e., the minimum score in the heap) based on the term upper-bounds; and otherwise, skip it (please see the lecture notes and references at the end for more details). You have to keep the counts of the evaluated and skipped posting elements per query, and report the average counts over the query set.

Index: You are given two files for the index: a wordlist file (text) and postings lists file (binary). The format of the *wordlist.txt* is as follows:

furkan 38 11.982635
furkanaytac 1 15.620221

where the first field is the term, a string with the max length of 100 characters (note that, the numbers are also considered as terms in our system), the second field (an integer) is the length of its postings list (i.e., the number of documents that include term *t*) and the last field is the pre-computed IDF value (a floating point value). The wordlist file includes 3,560,655 terms sorted in alphabetical order.

The index file (*postings.bin*) includes pairs of integers (i.e., each posting is a <*doc-id, term frequency*> pair) and each integer takes 4 bytes. The entire file includes 62,292,689 postings. The doc-ids start from 1; and goes up to 2,236,050 (Remark: some –a few thousand- documents do not appear in any of the postings lists). While accessing a term's posting list, you should seek to the beginning of its list (which means to get the posting list lengths of all preceding terms and multiply by posting size) and then read the postings that belong to this term. Note that, this binary file is produced by C in a Linux system, hence if you attempt to read it in Windows, you may need to assume "*Little Endian*".

There is also a third text file that stores the vector lengths of the documents. The file is in the format <*doc-id*><*doc-length*> per line, where the first entry is the integer document id (from 1 to 2,236,050) and the second entry is the vector length, a floating number.

Query Log: The query log includes *500* queries. Each line includes a single query (terms are sorted alphabetically for convenience):

com yahoo
com google
com myspace

If the query includes terms that are not included in the wordlist, just discard these terms (no problem as we anyway assume disjunctive semantics!).

Scoring Function: When you evaluate a document *d* for a given query *q*, compute the *score(d, q)* using simple Cosine metric with tf-idf weighting. That is, compute the term-frequency (obtained from the posting list) times IDF (obtained from the wordlist file) for each term in the query, add them, and finally divide by the document *d*'s length (obtained from the document length file). Note that, we do not weight query terms for the sake of simplicity.

Term upper-bounds: To apply WAND; you need to compute an upper-bound for each term, which is the highest possible contribution to the score from any of the documents in this term's postings list. Ideally, for a given term, this is the highest term-frequency value in the term's posting list times the term's IDF. For simplicity again, assume that the IDF score of a term serves as its upper-bound value.

What should you report?: For each query, you have to keep track of the EvalPostingNo (i.e., the number of posting list elements that are used in the score computation), without and with WAND optimization. Notice that, for the former case, EvalPostingNo is simply the sum of the lengths of the posting lists for each term in the query.

Then, first report the average EvalPostingNo with and without WAND for the entire query log, along with the percentage of improvement. Secondly, compute the average EvalPostingNo over the queries of length 1 (i.e., including only one term), length 2, etc., and report in a table.

Run your simulations for 2 different cases, where *k* is set to **2 and 10** (recall that *k* is the number of documents retrieved at the end, and hence, the size of the heap) and report the above results for each value of *k*.

What and how to submit? You have to submit:

a) A zipped folder of your source code (do **not** include the data files)

b) A 1-2 pages report that include *i)* a brief discussion of implementation details (what kind of data structures did you use, how did you manage certain functionality, etc.), and *ii)* the results as discussed above. Reports must be *spell-checked*.

References:

[1] Andrei Z. Broder, David Carmel, Michael Herscovici, Aya Soffer, Jason Y. Zien: Efficient query evaluation using a two-level retrieval process. CIKM 2003: 426-434 (the main paper, but has some extra details than what I discussed in the class)

[2] Craig Macdonald, Iadh Ounis, Nicola Tonellotto: Upper-bound approximations for dynamic pruning. ACM Trans. Inf. Syst. 29(4): 17 (2011) (The example covered in the lecture is in page 5; its snapshot is at the COW page, see **WAND-example.jpg** at the Additional Resources section).

Data: You can download the **data files** (query log: hw500queries.txt , wordlist: wordlist.txt, index: postings.bin and document length files: doc_lengths.txt) from the following address (Note that the total size of the files is around 0.6 GB):

http://www.ceng.metu.edu.tr/~altingovde/ceng778/

Disclaimer: Since we are using real queries and terms from a Web collection; please be aware that some of the expressions may be highly offensive. The data provided here should be used **exclusively** for this homework and should not be published & posted anywhere by any means.