**Indexing and Retrieval**

Implement a small search engine. The main steps are:

* Read in the already stemmed document collection provided in the file tccorpus.txt inside [tccorpus.zip](http://www.ccs.neu.edu/course/cs6200f15/tccorpus.zip). This is an early standard collection of abstracts from the *Communications of the ACM*.
* Build a simple inverted indexer that reads the corpus and writes the index. You should invoke it with
* indexer tccorpus.txt index.out

* Implement the BM25 ranking algorithm, and write a program to provide a ranked list of documents for a file with one or more queries. You should pass parameters for the index file, the query file, and the maximum number of document results, and return documents on the standard output, like so:
* bm25 index.out queries.txt 100 > results.eval
* Submit the output from this run, with the top 100 document IDs and their BM25 scores for each test query according to the following format:
* query\_id Q0 doc\_id rank BM25\_score system\_name

The string Q0 is a literal used by the standard TREC evaluation script. You can use any space-free token for your system\_name.

* Also, submit your code, instructions for compiling it, and a short report describing your implementation.

**Tokenized Document Collection**

* The provided tccorpus.txt file is in the format:
  + A # followed by a document ID
  + Lines below the document ID line contain stemmed words from the document.
* For example:
* # 1
* this is a tokenzied line for document 1
* this is also a line of document 1
* # 2
* from here lines for document 2 begin
* ...
* ...
* # 3
* ...
* For tokenization, simply break the character sequence at any run of whitespace (space, newline, etc.).
* Also, because this each document ends with a list of numerical cross references, you can, for this assignment, ignore any tokens that contain only the digits 0-9.

**Building an Inverted Index**: The following data structures are required for BM25 computation:

* Term frequencies (tftf) are stored in inverted lists: word -> (docid, tf), (docid, tf), ...
* For this assignment, you don't need to consider term positions within documents.
* Store the number of tokens in each document in a separate data structure.
* You may employ any concrete data structures convenient for the programming language you are using, as long as you can write them to disk and read them back in when you want to run some queries.

**BM25 Ranking**

1. Retrieve all inverted lists corresponding to terms in a query.
2. Compute BM25 scores for documents in the lists.
3. Make a score list for documents in the inverted lists.
4. Accumulate scores for each term in a query on the score list.
5. Assume that no relevance information is available.
6. For parameters, use k1=1.2k1=1.2, b=0.75b=0.75, k2=100k2=100