Comp 479/6791 **Project 3** — amended **Fall 2020**

Objectives: Implement SPIMI. Implement ranking of returns. Test and analyze your system, discuss how your design decisions influence the results.

Due Date: 17.11.2020 — **extended to 19.11.2020 at 11am**

Data: Use Reuters21578 for testing and if needed, continue your text scrubbing skills for the final project. Note that the text preprocessing should be secondary in this project.

Description: this project consists of two subprojects that build on each other. Each subproject should be very simple to execute, discuss with your peers and during Lab Q&A if there are any hurdles.

Subproject I: Implement SPIMI using your Project 2 Subproject 1 system. In particular:

1. (Project 2 Subproject I item 1:) develop a module that while there are still more documents to be processed, accepts a document as a list of tokens and outputs term-documentID pairs. Instead of appending new term-docID pairings **(since you are not to compress the index, a matched token and docID, not a pair data structure. Omit punctuation.)** to a global list, do:
2. SPIMI:
   - in the following, replace K with 500 for submitting your first and last block for grading
   - replace K with 10000 for comparison with naive indexer

   for K term-docIDs, create a new hash key for the term if necessary and/or append the docID to the postings list associated with the hashed term **if it is not already listed in the postings list or if it is already listed, augment a term counter to calculate tf.**
3. when the block is full (representing K term-docIDs), collect the index, sort, and "store" in consecutively labelled `BlockX`
4. disk block merging: when all term-docID pairs of your input are stored in block-sized indices, merge the mini indeces into a global index. You can hold the merged index in memory
5. compare timing with the naive indexer **(for 10000 term-docID pairings).**
6. compile an inverted index for Reuters21578 without using any compression techniques

docID hint: Use the NEWID values from the Reuters corpus to make your retrieval comparable.

Subproject II: Convert your indexer into a probabilistic search engine

1. using the assumptions made in Chapter 11 about independence of terms and documents etc. and
2. using the BM25 formula (11.32),
3. rank the documents your SPIMI implementation returns and
4. for a given query, return a ranked list of results.

Notes: experiment with different values for the parameters k_1 and b as described in the textbook.

Test queries:

1. design **four** test queries:
   (a) a single keyword query, **to comapre with Project 2**
   (b) **a query consisting of several keywords for BM25**
   (c) a multiple keyword query returning documents containing all the keywords (AND), **for unranked Boolean retrieval**
   (d) a multiple keywords query returning documents containing at least one keyword (OR), where documents are ordered by how many keywords they contain), **for unranked Boolean retrieval**
2. run your **four** test queries to showcase your code and comment on the results in your report

Deliverables:

1. individual project
2. well documented code
3. well documented sample runs for your queries on the information needs:
   (a) Democrats' welfare and healthcare reform policies
   (b) Drug company bankruptcies
   (c) George Bush
4. any additional testing or aborted design ideas that show off particular aspects of your project
5. a project report that summarizes your approach, illustrates your design and discusses what you have learned from the project. Note that a summary and commentary on your sample runs has to be included in the report

Submissions: submit on Moodle

Marks:

Spimi implementation	2pts	Attr 1
Final inverted index	1pt	Attr 1
Ranking	1pt	Attr 1
Multiple keyword AND queries **test and development**	1.5pt	Attr 1
Multiple keyword OR queries **test and development**	1.5pt	Attr 1
Project report	1pt	Attr 1,6