CS433-CS533 Project Part I
(Due: 4:00pm, February 24, 2020)

The goal of the course project is to develop a small IR system and to practice implementing different components of an IR system. This project will be carried out in 3 parts. This is Part I of the project.

The goal of Part I is to build a dictionary for the terms of a small document collection. We will start with a very small collection consisting of 200 titles of news articles. The 200 news titles are included in the file 200_title.txt in the Project Part I folder. Each row of the file contains one title. Your project should treat each title as a separate document. Later a larger collection will be provided for this project.

The project is a team project with two members of your own choice.

Your project for this part needs to accomplish the following tasks:

1. Tokenize the documents. Your tokenizer should include removing special symbols and punctuations (apostrophes, hyphens, periods in numbers, space between numbers, parentheses, etc.), case-folding (convert everything to lower case), and stemming (you are allowed to use Porter's stemmer). The resulting tokens will be called terms.
2. Generate (term, docID) pairs for all the terms produced by your tokenizer.
3. Sort the (term, docID) pairs by term in alphabetic order (numbers will be treated as strings, so 22 will be sorted before 3).
4. Merge (term, docID) pairs that have the same term into a structure of the format: (term, df, postings-list), where df is the document frequency of the term and the postings-list is a sequence of the pairs of the format: (docID, tf), where tf is the term frequency of the term in the document (i.e., the number of times the term appears in the document) identified by the docID. The postings list for each term is sorted in ascending docIDs.
5. Generate output. The output of this part of the project consists of two files: dictionary.txt and postings.txt. Each line in dictionary.txt corresponds to one term and it has three fields: (term, df, offset), where offset refers to the location of the postings list (the first line of the postings list of this term) in the postings.txt. For example, if the first term in dictionary.txt has df = 2 and points to the first line in postings.txt, then the offset for this term is 0; furthermore, the second term in dictionary.txt will have an offset = 2 (i.e., pointing to the third line in postings.txt) because the first term has two postings (df = 2) which will use the first two lines in postings.txt. Each line in postings.txt has two fields: docID and tf. If the postings list of a term has k postings, it will have k consecutive lines in postings.txt sorted by docIDs.

Programming language requirement

You may write the code using any programming language but it is required that your program must compile on harveyv.binghamton.edu. No exceptions. It should be purely a command line program. NO GUI will be accepted. This enables the TAs to run your code using test scripts. DO NOT assume that since your program compiled and ran correctly on your laptop it will also compile and run correctly on harveyv.binghamton.edu

What to submit?

You need to send a [file_name].*tar.gz* file to the TA by the due time. The file name should contain the last names of the team members. When the file is unzipped it should contain a directory with the same name as the zip file. The directory should contain the following files:

1. A page with the following and signed by all team members: "We have done this assignment completely on our own except for the software/tools acknowledged in the project report. We have not copied it, nor have we given our solution to anyone else. We understand that if we are involved in plagiarism or cheating we will have to sign an official form that we have cheated and that this form will be stored in our official university record. We also understand that we will receive a grade of 0 for the involved assignment and our grades will be reduced by one level (e.g., from A to A- or from B+ to B) for the first offense, and that we will receive a grade of "F" for the course for any additional offense of any kind."
2. The source code of your implementation plus possibly a make file. The code should be reasonably commented.
3. A readme.txt file on how to run your code.
4. A status report of your project. It should report what programming language is used, how complete your implementation is (especially for students who could not fully complete the project), whether correct results are obtained and the number of terms in your dictionary.
5. dictionary.txt
6. postings.txt

Plagiarism Check

All your code will be subject to check for similarity with other submissions using Moss. So you are advised not to look at each other's code.