Assignment 1: Corpus Indexing

phuonglh@gmail.com

February 27, 2023

1 Problem

In this assignment, you will develop a program which indexes a corpus. The input is a corpus of M documents, each document has an id. The output is N posting lists where N is the size of the vocabulary. You can use any programming language or programming styles (object-oriented, functional) that you prefer to do this assignment. We will work on an English corpus – The Reuters Corpus. This corpus contains 10,788 news documents totaling 1.3 million words. In this assignment, we restrict to the test portion of this corpus, which contains 3,020 documents. Download the file "*reuters_test.zip*" from the course website and unzip it to a directory on your local machine. Each document is a text file; we use its name as *docId*. For example, the document *14826* has the following content:

```
ASIAN EXPORTERS FEAR DAMAGE FROM U.S.-JAPAN RIFT
  Mounting trade friction between the
  U.S. And Japan has raised fears among many of Asia's exporting
  nations that the row could inflict far-reaching economic
  damage, businessmen and officials said.
      They told Reuter correspondents in Asian capitals a U.S.
  Move against Japan might boost protectionist sentiment in the
  U.S. And lead to curbs on American imports of their products.
      But some exporters said that while the conflict would hurt
  them in the long-run, in the short-term Tokyo's loss might be
  their gain.
  ...
```

Your program needs to read the content of this corpus, build the vocabulary, find posting lists corresponding to tokens in the dictionary and save the lists to an output text file named "*index.txt*". Each line of this output file should be in the format as follows:

```
token <tab> docId1 <space> docId2 <space> ...
```

If the vocabulary has N tokens then there will be N lines in the output file. *The vocabulary should be sorted alphabetically.*

2 Guide

As demonstrated in the lecture, you should follow the following steps:

1. Read the data folder and get all text files, take the name of a file as its docId; the whole multi-line content of a file can be considered as a (long) string.
2. Tokenize the content of all documents using spaces (space, tab, newline) and punctuations. Each document is an array of tokens. Make all tokens lowercase.
3. Run a stemmer (for example the Porter Stemmer) to map all tokens to their root form. (*)
4. Develop a word shaper to tranform some common entities to their representative token. For example, "*118*" → [NUM], "*100.89*" → [NUM], "*10,000*" → [NUM], etc. (*)
5. Build the vocabulary.
6. Construct posting lists and save them to the *index.txt* text file, in the requested format.

It's OK if you cannot complete steps 3 and 4 (in red color). You can skip these two language processing steps and submit your assignment anyway.

3 Submission

If you develop your program in Scala, Python, or Java, submit your source file *YourFullName.scala*, *YourFullName.py*, or *YourFullName.java*, respectively; for example *NguyenPhuTrong.scala*.

- If you have multiple source files then put them in a directory and zip it to *YourFullName.zip* and submit.
- Do not submit a Notebook (say **.ipynb*). Your teacher will not run any notebook server for evaluation of this assignment.
- You need to submit only your source code, no data is included to save space. You can assume that a user needs to provide a data folder (i.e., *.../data/reuteurs/test*) as an argument for your program to run.
- Submit your file to Classroom before the due date. You have 7 days to hand in.

Further Developments*

This section is not required for this assignment. For students who want to develop further their program, here are some extended processing steps to consider:

1. For each token in a posting list, associate it with the number of documents it appears in. For example, instead of listing

   $$jerry \rightarrow List(d1,\ d3,\ d10)$$

   we use

   $$(jerry,\ 3) \rightarrow List(d1,\ d3,\ d10)$$

   This frequency number will be useful for searching at a later step.

2. Write a front-end application which allows users to interact with your back-end program more friendly. This front-end app can be desktop-based or web-based and has some functionalities. For example, it has an GUI which allows a user:

   (a) Click a button to select the data folder that contains input documents;

   (b) Click a button to start indexing the corpus;

   (c) Show built the vocabulary once the indexing finishes;

   (d) Enter a token into a textbox and get the posting list associated with this token.