CPSC 5330 Spring 2024
Lab 4: Hadoop on AWS

Part 0, Style Points (5 points)

(These are easy points to get, just by following instructions and handing in good clean work!)

Was the output clean and well presented? Was the text and explanations well written? Were all the files there and named according to instructions?

Part I – Better Term Conversion (20 points)

Recall that the previous policy of "termifying" words in a document was to

(a) split the line on whitespace

(b) downcase all letters

(c) filter out all non-letters.

That policy leads to some tokens that don't intuitively look like words (for example very long terms that came from word sequences separated by hyphen characters) that would be useful in a search engine. Maybe we can do better.

You will use the following rules to *termify* the text.
For each text line do the following steps *in order*:

1. Convert letters to lower case
2. Split on whitespace *and punctuation* to get a set of *words*. So the line
   `"hello world's best-ever fair is so_NICE!"`
   would produce this list of words:
   `['hello', 'world', 's', 'best', 'ever', 'fair' 'is', 'so', 'nice']`

   HINT: the underscore character is tricky. The regular expression metacharacter `\W` matches letters, digits, *and the underscore character*, so it isn't exactly what you need. You want to consider underscore to be a punctuation character too.
3. Remove words that have 1 or fewer characters
4. Remove words that are sequences of digits (Filter words that are only digits, like `'123'`, but not strings that contain digits like `'a1b2c3'`)
5. Remove "stop words"

Stop words are common words like ‘a’ and ‘the’ that appear frequently in most documents and are rarely useful in determining query relevance. The common method for processing stop words is to have a stop word list (really a stop *term* list), and simply remove tokens that appear in the list. Stopwords for this assignment are in a file `stopwords.py` – that file contains code you should put in your mapper code.

You will then gather some statistics about the document terms. Your reducer output will have one record per document, with the following tab-separated fields

1. Document ID -- same as previous lab. For example, `'austen-persuasion'`
2. The total number of *words* in the document, prior to any processing (i.e. after splitting on whitespace and punctuation, but before any filtering)
3. The number of stopword terms in the document
4. The total number of terms in the document after all processing
5. The number of *unique* terms in the document after all processing

Here is a sample output record. There should be one record per document in your reducer output file(s)

```
test-document      30271   15122   13509   4409
```

Part 2 – Document Frequency (20 points)

The previous lab computed “term frequency,” which was the number of times a term appeared in a document divided by the number of terms in the document.

To finish our search engine we will need one more statistic: *document frequency.* Document frequency is a tuple `(term, count)` where `count` is the number of documents in the corpus in which the term appears one or more times.

Recall in the previous lab the first Map Reduce job produced a data set `term-count-doc-and-term` with tuples of the form `(document_id, term, count)`. The output of that Map Reduce job can be used to produce a data set document-frequency with tuples of the form `(term, count)` as described above.

The Teams site contains the data set `term-count-doc-and-term` which is the result of a sample run from the Term Frequency job. You will upload this data set (directory) to S3. You will write streaming Hadoop Python code that will read that data set and write the data set `term-frequency`.

Sample output

```
test-document    13509
```

Submission and AWS Details

- We will review and evaluate your two applications directly on AWS.
- You will create an S3 bucket `<name>-week5` where `<name>` is your username (email to the left of the @). It will hold all your work for this lab.
- The bucket will contain two directories, `code` and `data`
- The `data` directory will contain
    - A directory `textcorpora` with the documents. You will copy that directory into S3 from the Teams site
    - A directory `term-count-doc-and-term` containing the Step 1 output from Lab 3. You will copy that directory from the Teams site into S3
- The code directory will contain
    - Two files `term-statistics-mapper.py` and `term-statistics-reducer.py` which is your code for Part 1
    - Two files `document-frequency-mapper.py` and `document-frequency-reducer.py` which is your code for Part 2
- After running your two applications, the `data` directory will also contain
    - A directory `term-statistics`, which is the output of your Map Reduce job from Part 1
    - A directory `document-frequency`, which is the output of your Map Reduce job from Part 2
- You should have two separate (terminated) clusters, one which successfully ran your code for Part 1, and one which successfully ran your code for Part 2

Submission

We will review your submission directly on AWS, so you will submit a single PDF file containing the following information

1. The IDs of the two EMR clusters that produced the output for the two parts of the lab
2. A retrospective report a reflection on the assignment, with the following components
    a. Your name
    b. How much time you spent on the assignment
    c. If parts of the assignments are not fully working, which parts and what the problem(s) are

d. Were there aspects of the assignment that were particularly challenging? Particularly confusing?
e. What were the main learning take-aways from this lab – that is, did it introduce particular concepts or techniques that might help you as an analyst or engineer in the future?