MET CS 755, Spring 2018

Assignment – 1 (20 points)

Java 8 Programming - Streaming

Due date: February 13, 6:00 pm

1. Description

In this assignment we will analyze text data from Wikipedia pages. We want to analyze the Wikipedia data and extract some detail knowledge about it and ask specific research questions, like we want to know what are the top 5 Thousand words of English Wikipedia, or what are the Wikipedia pages that use most of these words.

2. Dataset

You will be dealing with a data set that consists of around 102 million text documents. This is a dataset download as data dump from Wikipedia which includes 102059798 wikipedia pages (12G of Text data). The data is stored in a simple plain text file, each line of the file is a Wikipedia page.

You'll see that the contents are sort of a pseudo-XML, where each text document begins with a

<doc id = ... > tag, and ends with </doc>.

The main text of each Wikipedia text document is stored as values inside XML tags of <doc> and </doc>. Each Wikipedia page has a unique ID that can be used as key to address documents, a title and a Web URL.

We have prepared data sets with different file sizes for your use.

1. A small data set of this large file (1000 lines includes 1000 Wikipedia pages – **5.9MB**)

https://s3.amazonaws.com/metcs755/WikipediaPages_oneDocPerLine_1000Lines_small.txt

You should download and look at the WikipediaPages_oneDocPerLine_1000Lines_small.txt

file before you begin. **Each line of the file is one Wikipedia Page.**

Or as direct S3 address, so you can use it using **AWS Java SDK for Amazon S3**

s3://metcs755/WikipediaPages_oneDocPerLine_1000Lines_small.txt

S3 bucket name : "**metcs755**"

file name: "**WikipediaPages_oneDocPerLine_1000Lines_small.txt**"

For your implementation, you should use first the small dataset to implement your assignment. After you have debugged your code and you are sure that your code is correct, run it on the larger dataset which is the whole Wikipedia Dataset.

2. The larger data set of 1 million text documents (2.2G).

https://s3.amazonaws.com/metcs755/WikipediaPages_oneDocPerLine_1m.txt

or as direct S3 address

s3://metcs755/WikipediaPages_oneDocPerLine_1m.txt

S3 bucket name : "**metcs755**"

file name: "**WikipediaPages_oneDocPerLine_1m.txt** "

3. Assignment Tasks:

We have prepared for your use a ***Java Maven project template*** in the following Github repository, https://github.com/kiat/metcs755/

3.1. Task 1 (10 points)

First, you need to write Java 8 code to create a data processing pipeline that builds a dictionary including the 5,000 most frequent words in the Wikipedia corpus. This dictionary can be essentially a String Array or a String List that has the word as the value, and the relative frequency position of the Word as the index of the array/list. For example, the value is zero for the most frequent word, and 19,999 for the least frequent word in the dictionary.

Convert the words to uppercase before further processing.

To **get credit for this task**, print out the frequency position of the words "**during**", "**and**", "**time**", "**protein**", and "**car**". These should be values from 0 to 4,999, or -1 if the word is not in the dictionary, because it is not in the to 5,000 Words.

3.2. Task 2 (10 Points)

We want to know which top 20 Wikipedia articles (in our 1 million page Wikipedia datase) that include the most words of the top 5k dictionary words which we create the in the task 1. You are asked to write

a Java 8 Code that process the 1 million Wikipedia pages and counts up how many dictionary words are used in each documents, sort the pages and get the top 20 Wikipedia pages. Each word has the same weight, for example if you have a page that uses 899 words of the 5k words, then it gets the ranked 899. The top ranked pages are pages that use all of the 5k words. You may have multiple pages in the same rank. If a page does not have any of the words it gets the rank zero.

To **get credit for this task**, provide the list of top 20 Wikipedia pages.

3.3. Task 3 (Optional for more Advanced Groups)

(**Instead of doing the task 2** you can do this task, you can implement this task.)

You will convert each of the documents in the data set to a TF ("Term Frequency") vector that has 20,000 entries. For example for a particular document, the entry in the 177th value in this vector is a double that tells us the frequency, in this document, of the 177th most common word in the corpus. Likewise, the first entry in the vector is a double that tells us the frequency, in this document, of the most common word in the corpus.

It might make sense to store only the non-zero entries in this vector, just to keep the RAM requirements low.

Create your own S3 buckets and store the final result of your computation in a CSV file (Comma Separated Volume file).

The next step is to calculate the Inverse Document Frequency (IDF) and Term Frequency Inverse Document Frequency (TF-IDF) https://en.wikipedia.org/wiki/Tf%E2%80%93idf .

You may want to try the whole Wikipedia data set.

The original large data set of 102 million text documents (102059798 Wikipedia pages - **12G**).
https://s3.amazonaws.com/metcs755/WikipediaPages_oneDocPerLine.txt

or as direct S3 address

s3://metcs755/WikipediaPages_oneDocPerLine.txt

5. Important Considerations

5.1. Execution of Implementation

Start one instance of Amazon AWS EC2 machine. You should use machine type **"t2.xlarge" which has 4 cores and 16GB of RAM.** You should use AMI **"Ubuntu Server 16.04 LTS (HVM), SSD Volume Type"**

As you can see on EC2 Price list , this costs around 18.8 cents per hour. That is not much, but **IT WILL ADD UP QUICKLY IF YOU FORGET TO SHUT OFF YOUR MACHINES**. Be very careful, and stop your machine as soon as you are done working. You can always come back and start your machine or create a new one easily when you begin your work again. Another thing to be aware of is that Amazon charges you when you move data around. To avoid such charges, do everything in the **N. Virginia** region. That's where data is, and that's where you should put your data and machines.

- You should document your code very well and as much as possible.

- You should use the Google Java Style Guide (https://google.github.io/styleguide/javaguide.html)

- You code should be compilable on a unix-based operating system like Linux or MacOS.

5.2. Academic Misconduct Regarding Programming

In a programming class like our class, there is sometimes a very fine line between "cheating" and acceptable and beneficial interaction between peers. Thus, it is very important that you fully understand what is and what is not allowed in terms of collaboration with your classmates. We want to be 100% precise, so that there can be no confusion.

The rule on collaboration and communication with your classmates is very simple: you cannot transmit or receive code from or to anyone in the class in any way---visually (by showing someone your code), electronically (by emailing, posting, or otherwise sending someone your code), verbally (by reading code to someone) or in any other way we have not yet imagined. Any other collaboration is acceptable.

The rule on collaboration and communication with people who are not your classmates (or your TAs or instructor) is also very simple: it is not allowed in any way, period. This disallows (for example) posting any questions of any nature to programming forums such as **StackOverflow**.

As far as going to the web and using Google, we will apply the **"two line rule"**. Go to any web page you like and do any search that you like. But you cannot take more than two lines of code from an external resource and actually include it in your assignment in any form. Note that changing variable names or otherwise transforming or obfuscating code you found on the web does not render the "two line rule" inapplicable. It is still a violation to obtain more than two lines of code from an external resource and turn it in, whatever you do to those two lines after you first obtain them.

Furthermore, you should cite your sources. Add a comment to your code that includes the URL(s) that you consulted when constructing your solution. This turns out to be very helpful when you're looking at something you wrote a while ago and you need to remind yourself what you were thinking.

6. Turnin

We'll be asking for a copy of your results, and a copy of implementation code.

Create a single PDF or Word document that has results for all tasks (if large provide links to the results files).

Turn in this document as well as all of your code. Upload your PDF document and ZIP code of your implementation to the blackboard system.

Make also Screenshots of EC2 CPU Utilization while your tasks are running on EC2.

Advanced students may install Ganglia Monitoring System http://ganglia.sourceforge.net/ to collect CUP utilization data and report it in form of figures.