IEMS 5730 Spring 2019 Homework 2

Release date: Feb 1, 2019

Due date: Mar 11, 2019 (Monday) 11:59am (i.e. noon-time)

The solution will be posted right after the deadline, so no late homework will be accepted!

Every Student MUST include the following statement, together with his/her signature in the submitted homework.

I declare that the assignment submitted on Elearning system is original except for source material explicitly acknowledged, and that the same or related material has not been previously submitted for another course. I also acknowledge that I am aware of University policy and regulations on honesty in academic work, and of the disciplinary guidelines and procedures applicable to breaches of such policy and regulations, as contained in the website http://www.cuhk.edu.hk/policy/academichonesty/.

Signed (Student________________________) Date:____________________________

Name_______________________________ SID______________________________

Submission notice:

- Submit your homework via the elearning system

General homework policies:

A student may discuss the problems with others. However, the work a student turns in must be created COMPLETELY by oneself ALONE. A student may not share ANY written work or pictures, nor may one copy answers from any source other than one's own brain.

Each student **MUST LIST** on the homework paper the **name of every person he/she has discussed or worked with**. If the answer includes content from any other source, the student **MUST STATE THE SOURCE**. Failure to do so is cheating and will result in sanctions. Copying answers from someone else is cheating even if one lists their name(s) on the homework.

If there is information you need to solve a problem but the information is not stated in the problem, try to find the data somewhere. If you cannot find it, state what data you need, make a reasonable estimate of its value, and justify any assumptions you make. You will be graded not only on whether your answer is correct, but also on whether you have done an intelligent analysis.

Q1 [30 marks]: Pig Setup and Basic Operations

You are required to perform some simple analysis using Pig on the n-grams dataset of Google books. An ‘n-gram’ is a phrase with n words. The dataset lists all n-grams present in books from books.google.com along with some statistics.

In this question, you only use the Google books bigram (1-grams). Please go to Reference [1] and [2] to download the two datasets. Each line in these two files has the following format (TAB separated):

bigram year match_count volume_count

An example for 1-grams would be:

```
circumvallate   1978   335   91
circumvallate   1979   261   95
```

This means that in 1978(1979), the word "circumvallate" occurred 335(261) times overall, from 91(95) distinct books.

(a) **[10 marks]** Install Pig in your cluster. You can either install the Pig service under your Hortonworks cluster or refer the following link to install the 0.17.0 version of Pig under your Hadoop cluster:
http://hadooptutorial.info/pig-installation-on-ubuntu/

(b) **[5 marks]** Upload these two files to HDFS and **join** them into one table.

(c) **[5 marks]** For each unique bigram, compute its average number of occurrences per year. In the above example, the result is:
circumvallate (335 + 261) / 2 = 298
Notes: The denominator is the number of years in which that word has appeared
Assume the data set contains all the 1-grams in the last 100 years, and the above records are the only records for word ‘circumvallate’. Then the average value is
(335 + 261) / 2 = 298,

instead of
(335 + 261) / 100 = 5.96

(d) **[10 marks]** Output the **20** bigrams with the highest average number of occurrences per year along with their corresponding average values sorted in the descending order. If multiple bigrams have the same average value, write down any one you like (that is, break ties as you wish).

You need to write a Pig script to perform this task and save the output into HDFS.

Submit your output together with Pig script in one SINGLE pdf file.

Hints:

- This problem is very similar to the word counting example shown in the lecture notes of Pig. You can use the code there and just make some minor changes to perform this task.

Q2 [30 marks]: Hive Setup and Basic Operations

In this question, you are asked to install hive on the top of MapReduce on your Hadoop master machine, Then compare the performance between Hive and Pig. You can either install the Hive service under your Hortonworks cluster or refer the following link to install the 2.3.4 version of Hive under your Hadoop cluster:

https://cwiki.apache.org/confluence/display/Hive/GettingStarted

Write a Hive script to perform exactly the same task as that of Q1 with the same datasets. Rerun the Pig script in this cluster and compare the performance between Pig and Hive in terms of overall run-time and explain your observation.

Submit your output, explanation and your Hive commands/ scripts in one SINGLE pdf file.

Hints:

- Hive will store its tables on HDFS and those locations needs to be bootstrapped:
```
$ hdfs dfs -mkdir /tmp
$ hdfs dfs -mkdir /user/hive/warehouse
$ hdfs dfs -chmod g+w /tmp
$ hdfs dfs -chmod g+w /user/hive/warehouse
```
- While working with the interactive shell (or otherwise), you should first test on a small subset of the data instead of the whole data set. Once your Hive commands/ scripts work as desired, you can then run them up on the complete data set.

Q3 [40 marks]: Word Counting on a Storm Cluster

In this question, you are required to implement the word-counting algorithm under the Storm platform. The dataset is available at:

http://mobitec.ie.cuhk.edu.hk/ierg4330Spring2019/homework/StormData.txt

The basic idea is to first define a FileReaderSpout class to read an input file and emit a sentence every time. Then you can implement a SplitSentenceBolt class to split every sentence into words and emit the word to the next WordCountBolt class, which needs to compute the frequency of each word. The word count result should be dumped to a persistent storage. You are allowed to refer to (or even borrow codes) from publicly available

Storm examples/source-codes AS LONG AS you clearly list and acknowledge your sources in your submission.

(a) [20 marks] Multi-node Storm cluster setup.
Install Storm in your cluster. You can either install the Storm service under your Hortonworks cluster or refer to the following link [9] [10] to install the 1.0.x version of Storm with 4 VMs. In the second case, you just need to set up a single-node Zookeeper service in your master node. The Master node also needs to run Nimbus as well as Storm UI service.

After setting up the multi-node Storm cluster, examples can be found in [7] under the *Storm_HOME/examples/storm-starter* directory. Run the *StatefulTopology* example to validate the successful installation of your multi-node cluster as the following command line:

```
$ ./bin/storm jar examples/storm-starter/storm-starter-topologies-1.0.4.jar \
storm.starter.StatefulTopology
```

(b) [20 marks] Find Frequent Word: At-most-once model
Write a word counting program without any processing guarantee, i.e. under the at-most-once model. Your program should read the input file, count the frequency of words and dump the final result to the persistent storage (e.g., Linux file system or HDFS). Find out the Top 10 most frequently used words in the dataset and their corresponding frequency. You are also required to implement the function to count the number of tuples emitted, number of tuples acked, number of tuples failed in your word counting program.

Q4 [Bonus 20 marks]: Discover Twitter Trending Hashtags using Storm

In this questions, you are required to implement an algorithm to identify Twitter trending hashtags using the Apache Storm. The basic idea is to design a TwitterSpout class to collect and emit tweets by leveraging the twitter4j library[6]. You will also need to implement additional bolt classed, e.g. to extract hashtags from every tweets, as well as to count and determine the most popular hashtags in an ongoing basis. You are allowed to refer to (or even borrow codes) from publicly available Storm examples/source-codes AS LONG AS you clearly list and acknowledge your sources in your submission.

(a) [20 marks] Find popular hashtags using the multi-node Storm cluster
Design and implement an algorithm to determine the list of popular hashtags among tweets under the topic of “Trump” and keep printing out the results once every 10 minutes. Specifically, you will need to:

- Using the TwitterStream API in the twitter4j library [6] to continuously ingest live tweets containing the keywords “Trump”, ”trump” and “TRUMP”.
- You can use the getHashtagEntities() function to extract the hashtags from these tweets.

- Every 10 minutes, you need to identify and report (print out) all the popular hashtags and their corresponding frequencies. For this homework, a hashtag is said to be popular if it appears in more than one percent of all the tweets received since the last report (printout).
- Your Storm program should be able to operate continuously until it is terminated by the user.

Hints:

- You are recommended to use Hortonworks to provision the Storm cluster [6] [7] [8].
- You are recommended to use the t2.large instance type for your VMs. Each instance has 2 CPU cores and 8GB memory.
- You are recommended to use a stable version of Storm, e.g., Storm-1.0.x. or Hortonworks 2.6.x.
- You are allowed to post-process the result dumped from Storm.
- You **are NOT allowed to use Trident.**
- You are recommended to use Maven [10] to compile and build the executable Jar.
- To facilitate debugging, you can find logs under Storm_HOME/logs/directory upon any errors.
- DO NOT emit the entire file in one nextTuple to avoid failure caused by network congestion. You can emit one line for each *nextTuple( )* call.

Reference:

[1] Google Books 1:
http://storage.googleapis.com/books/ngrams/books/googlebooks-eng-all-1gram-20120701-a.gz
[2] Google Books 2:
http://storage.googleapis.com/books/ngrams/books/googlebooks-eng-all-1gram-20120701-b.gz
[3] Twitter real-time analytics under apache storm in udacity
https://github.com/udacity/ud381
[4] Example for Twitter real-time analytics
https://github.com/Blackmist/TwitterTrending
[5] Twitter Stream API examples
http://twitter4j.org/en/code-examples.html
[6] twitter4j library
http://twitter4j.org/javadoc/twitter4j/TwitterStream.html#filter-java.lang.String...-
[7] How to run examples in storm-starter
https://github.com/apache/storm/tree/master/examples/storm-starter
[8] How to install a distributed apache Storm cluster
http://knowm.org/how-to-install-a-distributed-apache-storm-cluster/
[9] Setting up a Storm Cluster
http://storm.apache.org/releases/1.0.2/Setting-up-a-Storm-cluster.html
[10] Storm Maven http://storm.apache.org/documentation/Maven.html