CS6220 BDS 2024 Fall

Homework Assignment 1

(Programming Category)

Student Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Student Session: cs6220-A

You are given three types of programming problems. You only need to choose one of the 3 problems for your first programming homework.

* **Problem 1** is designed to get hand-on experiences on learning Distributed/Parallel Big Data Computing with Hadoop and/or Spark MapReduce. (page 1-3).
* **Problem 2** is designed to get Hand-on experience on solving a real world data analytic system problem using real world datasets, instead of benchmark datasets. (page 3-4)
* **Problem 3** is designed for those students who want to learn hand-on experience with LLM prompt engineering techniques (page 5-6).

For the problem with multiple options, you are asked to choose only one option. This provides sufficient diversity for students with different knowledge and educational backgrounds. Feel free to choose any of your favorite programming languages, such as Java, C, Perl, Python, and so forth.

Top performing HW1 programming option should not only complete the itemized requirements and go beyond the above to demonstrate your deeper understanding of the problem.

**Post Date**: Monday of Week 2

**Due Date**: midnight on Friday of Week 3 (Sept 6)

(with no penalty extension to 9am of Sept 7 (Saturday, hard deadline).

**Problem 1. Learning Distributed/Parallel Big Data Computing with Hadoop and/or Spark MapReduce**

This problem has 2 options. You only need to choose one option.

The first option is for the beginner of Hadoop/Spark MapReduce.

The second option is designed for those students who are familiar with Hadoop MapReduce programming or Spark and would like to try something more challenging.

Deliverable (for each of the three options):

1. Source code and executable with readme. (provide URL if you use open source code packages)
2. Screen Shots of your execution process/environments
3. Input and output of your program
4. Runtime measurements in excel plots or tabular format.
5. Two MapReduce configuration scenarios (e.g., varying #mapers/#reducers, JVM queue sizes, different datasets, different sizes of dataset) to run the same MapRedue program and compare their performance and elaborate your results.
6. Two different MapReduce programs solving the same problem with the same dataset. Hint: some problem may be solved by two MapReduce Jobs or three MapReduce Jobs, or by different ways to design map/reduce code.
7. Provide your experimental comparison and analytical report.

Hint:

* + If you use two or more datasets of different sizes, you report/analyze the scaling effect of your program.
  + If you use two different implementations of MapReduce programs running on the same dataset, you report the runtime performance optimization aspects of your two programs.

**Option 1.1**

*Suitable for students who are the beginner of Hadoop/Spark MapReduce*

* Install HDFS and Hadoop MapReduce on your laptop.
* Run the word count map-reduce program, and report the runtime for three different sizes of datasets, two of the datasets should be larger and need multiple reducers and all three will be larger and need multiple mappers.
* Using MapReduce to solve another problem. You may choose one of the following.
  + Consider a dataset of 100 files, print the top 30 words occurring in the most files. Scaling your solution to 1000 files, 5000 and optionally 10,000 files. Make sure to meet deliverable requirements (e) and (f).
  + Solving K-means clustering problem with a dataset given in Spark ML lib, R, Hadoop Mahout ML library, or Scikit-learn. Scaling your solution to two larger datasets, the size of the two scaling datasets should exceed the JVM queue size you used to run the K-means cluster. Also run at least 3 scenarios by varying the hyperparameters of K-means algorithm: 3~5 different K settings with the same centroid selection, 3-5 different centroid selection with the same K. These requirements can substitute some of those in deliverable (e) and (f). Note that you are required also to meet deliverable requirements, especially (e) and (f).
  + Solving Logic Regression problem with a dataset given in Spark ML lib, R, Hadoop Mahout ML library, or Scikit-learn. Scaling your solution to two larger datasets, the size of the two scaling datasets should exceed the JVM queue size or Virtual machine memory you have used with the default allocation to run the LR program you write or you download (acknowledge with reference and URL). Make sure to meet deliverable requirements, especially (e) and (f).

You are encouraged to use excel file to generate your runtime statistics plot or organize the performance measurement data in a tabular format.

You are encouraged to learn by observing the runtime performance of Hadoop/Spark MapReduce program through different ways of programming the same problem and show their impact on the runtime performance of the MapReduce job.

**Option 1.2:**

This option is designed for students who are familiar with Hadoop MapReduce or Spark and are interested in hand-on comparison of them through example data problems and configuration tuning.

**Comparing Hadoop MapReduce with Spark MapReduce**

* Install HDFS, Hadoop MapReduce and Spark MapRedue on your laptop.
* Run the word count map-reduce program, and report the runtime for three different sizes of datasets, three of the datasets should need increasing # reducers and increasing # of mappers. Report also average map performance and average reducer performance in addition to total runtime. Make sure also to meet the deliverable requirements, especially (e) and (f).
* Run your MapReduce program on both Hadoop and Spark to compare the performance results. Using MapReduce to solve another problem of your choice. You may choose one of the following.
  + Consider a dataset of 100 files, print the top 30 words occurring in the most files. Scaling your solution to 1000 files, 5000 and optionally 10,000 files. Make sure to meet deliverable requirements, especially (e) and (f).
  + Solving K-means clustering problem with a dataset given in Spark ML lib, R, Hadoop Mahout ML library, or Scikit-learn. Scaling your solution to two larger datasets, the size of the two scaling datasets should exceed the JVM queue size you used to run the K-means cluster. Also run at least 3 scenarios by varying the hyperparameters of K-means algorithm: 3~5 different K settings with the same centroid selection, 3-5 different centroid selection with the same K. These requirements can substitute some of those in deliverable (e) and (f). Note that you are also required to meet deliverable requirements, especially (e) and (f).
  + Solving Logic Regression problem with a dataset given in Spark ML lib, R, Hadoop Mahout ML library, or Scikit-learn. Scaling your solution to two larger datasets, the size of the two scaling datasets should exceed the JVM queue size or Virtual machine memory you have used with the default allocation to run the LR program you write or you download. Make sure to meet deliverable requirements (e) and (f).

You may use excel file to generate your runtime statistics plot or organize the performance measurement data in a tabular format.

You are encouraged to learn by observing the runtime performance of Hadoop/Spark MapReduce program through different ways of programming the same problem and show their impact on the runtime performance of the MapReduce job.

**Problem 2: Hand-on experience on solving a real-world data system problem.**

This problem is designed for those students who are interested in using Hadoop MapReduce to solve problems using a real world dataset.

You are recommended to use software packages from Spark ML library, HadoopMahout directly or use Hadoop MapReduce or Spark as the underlying platform and running some of your favorite ML library on top, such as Scikit-learn (http://scikit-learn.org/), Weka (<http://www.cs.waikato.ac.nz/ml/weka/>) or R https://bigdata-madesimple.com/10-r-packages-machine-learning/.

**Option 2.1 Mining the Kaggle million songs dataset**

The million songs dataset is a big data mining challenge. One of the earlier Kaggle challenges. It contains (1) the full listening history for 1M users, (2) half of the listening history for 110K users (10K validation set, 100K test set). You are asked to predict the missing half.

http://www.kaggle.com/c/msdchallenge

Statistics of the dataset:

* Large Data Set
  + 1,019,318 users
  + 384,546 MSD songs
  + 48,373,586 (user, song, count)
* Kaggle Competition: offline evaluation
  + Predict songs a user will listen
  + Training: 1M user listening history
  + Validation: 110K users

This is a well-known big data challenge and there are software postings in the Public domain. Feel free to use any of them you found helpful.

Deliverable:

1. URL of the dataset
2. The subset of the data you used in your program or model training and testing.
3. The software packages and tools or library you use in your programs
4. The open source code you leveraged in developing your MillionSongMiner.
5. The experiments you conducted to report your mining accuracy, ideally over three different datasets extracted from the million-songs dataset in three different sizes (e.g., 2x, 5x, 10x of your initial smaller dataset, say 1000, or 10,000 songs.)
6. Analysis of your hand-on experiences, with three lessons learned or three observations you wish to make.

**Option 2.2 Mining the White House visitor Log dataset**

The White House Visitor Log dataset available in the directory of programming assignment in CSV. It was an older version in 2009 from

<http://www.whitehouse.gov/files/disclosures/visitors/WhiteHouse-WAVES-Released-0827.csv> which was replaced by recent datasets. Feel free to use the recent datasets whitehouse.gov.

You are required to write an efficient program (such as using MapReduce) to find the following information:

(i) The 10 most frequent visitors (NAMELAST, NAMEFIRST, NAMEMID) to the White House.

(ii) The 10 most frequently visited people (visitee\_namelast, visitee\_namefirst) in the White House.

(iii) The 10 most frequent visitor-visitee combinations.

(iv) Some other interesting statistics that you can think of.

Throughout this programming assignment, do not limit your programs to run with a single design choice (such as one reduce task only).

Deliverable:

1. URL of the dataset
2. The software packages and tools or library you use in your programs
3. The open source code you leveraged in developing your visitor log miner
4. The experiments you conducted to report your mining results and time, ideally over three different datasets in three different sizes (e.g., 2x, 5x, 10x of your initial dataset.)
5. Analysis of your hand-on experiences, with three observations you wish to make.

**Problem 3: Hand-on Experience with Large Language Models (LLMs)**

This problem is designed for those students who are interested in using Pretrained LLMs such as GPT family of models, like GPT-3.5, ChatGPT through their APIs or the open source Pretrained LLMs, such as LLaMA, Gemini. You can find many pre-trained LLMs from Huggingface LLM leader board. Feel free to choose your favorite LLMs for this problem.

You are asked to design two learning tasks that can be solved by leveraging pre-trained LLMs to generate your final solutions. Hint: You can use Huggingface LLM evaluation/finetuning datasets or Kaggle LLM competition datasets.

1. For each task, you will need to design at least 5 different queries, including different types of questions and different in-context phrases of similar/same questions (queries), e.g., zero shot, one-shot, few-shot prompting, CoT prompting, ToT prompting, to name a few.
2. For each of the 5 queries in one of your two learning tasks/problems, you are asked to analyze the results obtained independently from at least three independent pretrained LLMs.
3. For each query and each chosen pre-trained LLM, you are asked to use at least 3 different LLM hyperparameter settings, such as 3 temperature settings, or 3 settings of presence-penalty, or 3 settings of two or more hyperparameter combos.
4. For item (2) and (3) under each of the two learning tasks, you are asked to compare the results from the 5 queries on the same learning task one by one. For each of the 5 prompt queries, you are asked to provide the correct response/output for the question, and elaborate the reasoning of your results with analysis and critique.

Deliverable:

1. For each problem you design, provide your thought on why you choose this problem. NOTE: if you choose any problem from the Public domain, acknowledge the source with URL and full reference. Most of the prompt queries in the public domain were tested on one or two pretrained LLMs. You will need to run all tests with three pretrained LLMs for this HW problem.
2. For each problem, list all 5 queries you plan to ask and why and how they are different syntactically but semantically similar or the same. Note
3. For each query in each of your two problems, print your LLM hyperparameter settings in screen shots or plain text, including the default LLM settings you inherit from each pretrained LLM. Also print /screenshot all the results you obtain for each of the 5 prompt queries. Finally, provide the final result/output you will give and elaborate with analysis/ comparison.
4. For each problem, compare each of the 5 queries and their results under all three pretrained LLMs. Elaborate on your observations with respect to each LLM and with each learning task/problem.
5. For each of the three pretrained LLMs of your choice, compare the two learning tasks/problems, using the 5 prompt queries with different LLM settings or in-context phrases. Discuss how this LLM works on these two problems, and pros and cons you have observed by working with the two problems using three different pre-trained LLMs.