DSCI 553 Foundations and Applications of Data Mining

Fall 2022

Assignment 1

Deadline: Sep 20, 23:59 PM PST

1. Overview of the Assignment

In assignment 1, you will work on three tasks. The goal of these tasks is to get you familiar with Spark operation types (e.g., transformations and actions) and explore a real-world dataset: Yelp dataset (https://www.yelp.com/dataset).

If you have questions about the assignment, please ask on Piazza, which will also help other students.

You only need to submit on Vocareum, NO NEED to submit on Blackboard.

2. Requirements

2.1 Programming Requirements

a. You must use **Python** to implement all tasks. **You can only use standard python libraries** (i.e., external libraries like numpy or pandas are not allowed). There will be a 10% **bonus** for each task if you also submit a Scala implementation and both your Python and Scala implementations are correct.

b. **You are required to only use Spark RDD** in order to understand Spark operations. You will not get any points if you use Spark DataFrame or DataSet.

2.2 Programming Environment

Python 3.6, JDK 1.8, Scala 2.12, and Spark 3.1.2

We will use these library versions to compile and test your code. There will be no point if we cannot run your code on Vocareum.

On Vocareum, you can call `spark-submit` located at `/opt/spark/spark-3.1.2-bin-hadoop3.2/bin/spark-submit`. (Do not use the one at /usr/local/bin/spark-submit (2.3.0)). We use `--executor-memory 4G --driver-memory 4G` on Vocareum for grading.

2.3 Write your own code

Do not share code with other students!!

For this assignment to be an effective learning experience, you must write your own code! We emphasize this point because you will be able to find Python implementations of some of the required functions on the web. Please do not look for or at any such code!

TAs will combine all the code that can be found from the web (e.g., Github) as well as other students' code from this and other (previous) sections for plagiarism detection. We will report all detected plagiarism.

2.4 What you need to turn in

We will grade all submissions on Vocareum, the submissions on blackboard will be ignored. Vocareum produces a submission report after you click the "Submit" button (It takes a while since Vocareum needs to run your code in order to generate the report). Vocareum will **only grade Python** scripts during the **submission phase** and it will grade **both Python and Scala** during the **grading phase**.

a. [REQUIRED]three Python scripts, named: (all lowercase)

task1.py, task2.py, task3.py

b1. [OPTIONAL, REQUIRED FOR SCALA] three Scala scripts and the output jar file, named: (all lowercase)

hw1.jar, task1.scala, task2.scala, task3.scala

c. You don't need to include your results or the datasets. We will grade your code with our testing data (data will be in the same format).

3. Yelp Data

In this assignment, you will explore the Yelp dataset. You can find the data on Vocareum under resource/asnlib/publicdata/. The two files business.json and test_review.json are the two files you will work on for this assignment, and they are subsets of the original Yelp Dataset. The submission report you get from Vocareum is for the subsets. For grading, we will use the files from the original Yelp dataset which is **SIGNIFICANTLY** larger (e.g. review.json can be 5GB). You should make sure your code works well on large datasets as well.

4. Tasks

4.1 Task1: Data Exploration (3 points)

You will work on test_review.json, which contains the review information from users, and write a program to automatically answer the following questions:

A. The total number of reviews (0.5 point)

B. The number of reviews in 2018 (0.5 point)

C. The number of distinct users who wrote reviews (0.5 point)

D. The top 10 users who wrote the largest numbers of reviews and the number of reviews they wrote (0.5 point)

E. The number of distinct businesses that have been reviewed (0.5 point)

F. The top 10 businesses that had the largest numbers of reviews and the number of reviews they had (0.5 point)

Input format: (we will use the following command to execute your code)

Python:

spark-submit --executor-memory 4G --driver-memory 4G task1.py <review_filepath> <output_filepath>

Scala:

spark-submit --class task1 --executor-memory 4G --driver-memory 4G hw1.jar <review_filepath> <output_filepath>

Output format:

IMPORTANT: Please strictly follow the output format since your code will be graded automatically.

a. The output for Questions A/B/C/E will be a number. The output for Questions D/F will be a list, which is sorted by the number of reviews in the descending order. If two user_ids/business_ids have the same number of reviews, please sort the user_ids /business_ids in the alphabetical order.

b. You need to write the results in the JSON format file. You must use **exactly the same tags** (see the red boxes in Figure 2) for answering each question.

```
{
    "n_review": 11111,
    "n_review_2018": 11111,
    "n_user": 11111,
    "top10_user": [["ABCDEFGHIJKLMNOPQ", 1111], ...., ["BCDEFGHIJKLMNOPQR", 111]],
    "n_business": 11111,
    "top10_business": [["QPONMLKJIHGFEDCBA", 1111], ...., ["RQPONMLKJIHGFEDCB", 111]]
}
```

Figure 1: JSON output structure for task1

4.2 Task2: Partition (2 points)

Since processing large volumes of data requires performance decisions, properly partitioning the data for processing is imperative.

In this task, you will show the number of partitions for the RDD used for **Task 1 Question F** and the number of items per partition.

Then you need to **use a customized partition function to improve the performance** of map and reduce tasks. A time duration (for executing Task 1 Question F) comparison between **the default partition** and **the customized partition** (RDD built using the partition function) should also be shown in your results.

Hint:

Certain operations within Spark trigger an event known as the shuffle. The shuffle is Spark's mechanism for redistributing data so that it's grouped differently across partitions. This typically involves copying data across executors and machines, making the shuffle a complex and costly operation. So, trying to design a partition function to avoid the shuffle will improve the performance a lot.

Input format: (we will use the following command to execute your code)

Python:

spark-submit --executor-memory 4G --driver-memory 4G task2.py <review_filepath> <output_filepath> <n_partition>

Scala:

spark-submit --class --executor-memory 4G --driver-memory 4G task2 hw1.jar <review_filepath> <output_filepath> <n_partition>

Output format:

A. The output for the number of partitions and execution time will be a number. The output for the number of items per partition will be a list of numbers.

B. You need to write the results in a JSON file. You must use **exactly the same tags**.

```
{
    "default":{
        "n_partition": 2,
        "n_items":[1111,2222],
        "exe_time": 0
    },
    "customized":{
        "n_partition":2,
        "n_items":[3332,1],
        "exe_time":0
    }
}
```

Figure 3: JSON output structure for task2

4.3 Task3: Exploration on Multiple Datasets (2 points)

In task3, you are asked to explore two datasets together containing review information (test_review.json) and business information (business.json) and write a program to answer the following questions:

A. What are the average stars for each city? (1 point)

1. (**DO NOT** use the stars information in the business file).
2. (**DO NOT** discard records with empty "city" field prior to aggregation).

B. You are required to compare the **execution time** of using two methods to print top 10 cities with highest stars. Please note that this task – (Task 3(B)) is **not graded.** You will get full points only if you implement the logic to generate the output file required for this task.

1. You should store the execution time (start from loading the file) in the json file with tag "m1" and "m2".
2. Additionally, add a "reason" field and provide a hard-coded explanation for the observed execution times.

   Method1: Collect all the data, sort in python, and then print the first 10 cities

   Method2: Sort in Spark, take the first 10 cities, and then print these 10 cities

Input format: (we will use the following command to execute your code)

Python: spark-submit --executor-memory 4G --driver-memory 4G task3.py <review_filepath> <business_filepath> <output_filepath_question_a> <output_filepath_question_b>

Scala: spark-submit --class task3 --executor-memory 4G --driver-memory 4G hw1.jar <review_file> <business_file> <output_filepath_question_a> <output_filepath_question_b>

Output format:

a. You need to write the results for Question A as a file. The header (first line) of the file is "city,stars". The outputs should be sorted by the average stars in descending order. If two cities have the same stars, please sort the cities in the alphabetical order. (see Figure 3 left).

b. You also need to write the answer for Question B in a JSON file. You must use **exactly the same tags** for the task.
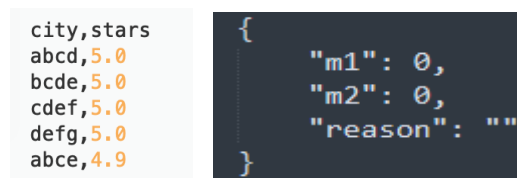

Figure 3: Question A output file structure (left) and JSON output structure (right) for task3

5. Grading Criteria

(% penalty = % penalty of possible points you get)

1. You can use your free 5-day extension separately or together https://forms.gle/h4t46LCahrtDk9rVA

    1. This form will record the number of late days you use for each assignment. We will not count late days if no request is submitted.

1. There will be a 10% bonus if you use both Scala and Python and get expected results.

2. We will combine all the codes we can find from the web (e.g., Github) as well as other students' code from this and other (previous) sections for plagiarism detection. If plagiarism is detected, there will be no point for the entire assignment and we will report all detected plagiarism.

3. All submissions will be graded on the Vocareum. Please strictly follow the format provided, otherwise you can't get the point even though the answer is correct. You are encouraged to try out your code on Vocareum terminal.

4. We will grade both the correctness and efficiency of your implementation. The efficiency is evaluated by processing time and memory usage. **The maximum memory allowed to use is 4G, and maximum processing time is 1800s for grading.** The datasets used for grading are larger than the ones that you use for doing the assignment. You will get *% penalty if your implementation cannot

generate correctness outputs for large files using 4G memory within the 1800s. Therefore, please make sure your implementation is efficient to process large files.

5. Regrading policy: We can regrade your assignments within seven days once the scores are released. Regrading requests will not be accepted after one week.

6. There will be a 20% penalty for late submission within a week and no point after a week. If you use your late days, there wouldn't be a 20% penalty.

7. Only when your results from Python are correct, the bonus of using Scala will be calculated. There is no partial point for Scala. See the example below:

Example situations

Task	Score for Python	Score for Scala (10% of previous column if correct)	Total
Task1	Correct: 3 points	Correct: 3 * 10%	3.3
Task1	Wrong: 0 point	Correct: 0 * 10%	0.0
Task1	Partially correct: 1.5 points	Correct: 1.5 * 10%	1.65
Task1	Partially correct: 1.5 points	Wrong: 0	1.5

6. Common problems causing fail submission on Vocareum/FAQ

(If your program runs successfully on your local machine but fail on Vocareum, please check these)

1. Try your program on Vocareum terminal. Remember to set python version as python3.6,

```
export PYSPARK_PYTHON=python3.6
```

And use the latest Spark

```
/opt/spark/spark-3.1.2-bin-hadoop3.2/bin/spark-submit
```

2. Check the input command line formats.

3. Check the output formats, for example, the headers, tags, typos.

4. Check the requirements of sorting the results.

5. Your program scripts should be named as task1.py task2.py etc.

6. Check whether your local environment fits the assignment description, i.e. version, configuration.

7. If you implement the core part in python instead of spark, or implement it with a high time complexity (e.g. search an element in a list instead of a set), your program may be killed on the Vocareum because it runs too slow.

8. You are required to only use Spark RDD in order to understand Spark operations more deeply. You will not get any points if you use Spark DataFrame or DataSet. Don't import sparksql.

9. Do not use Vocareum for debugging purposes, please debug on your local machine. Vocareum can be very slow if you use it for debugging.

10. Vocareum is reliable in helping you to check the input and output formats, but its function on checking the code correctness is limited. It can not guarantee the correctness of the code even with a full score in the submission report.

7. Running Spark on Vocareum

We're going to use Spark 3.1.2 and Scala 2.12 for the assignments and the competition project. Here are the things that you need to do on Vocareum and local machine to run the latest Spark and Scala:

On **Vocareum**:

1. Please select JDK 8 by running the command "**export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-amd64**"

2. Please use the spark-submit command as "**/opt/spark/spark-3.1.2-bin-hadoop3.2/bin/spark-submit**"

On your **local machine**:

1. Please download and set up **spark-3.1.2-bin-hadoop3.2**, the setup steps should be the same as spark-2.4.4

2. If you use Scala, please update Scala's version to **2.12** on IntelliJ.

8. Tutorials for Spark Installation

Here are some useful links here to help you get started with the Spark installation.

Tutorial for ubuntu: https://phoenixnap.com/kb/install-spark-on-ubuntu

Tutorial for windows:
https://medium.com/@GalarnykMichael/install-spark-on-windows-pyspark-4498a5d8d66c

Windows Installation without Anaconda (Recommended):
https://phoenixnap.com/kb/install-spark-on-windows-10

Tutorial for mac: https://medium.com/beeranddiapers/installing-apache-spark-on-mac-os-ce416007d79f

Tutorial for Linux systems:
https://www.tutorialspoint.com/apache_spark/apache_spark_installation.htm

Tutorial for using IntelliJ:
https://medium.com/@Sushil_Kumar/setting-up-spark-with-scala-development-environment-using-intellij-idea-b22644f73ef1

Tutorial for Jupyter notebook on Windows:
https://bigdata-madesimple.com/guide-to-install-spark-and-use-pyspark-from-jupyter-in-windows/