DSCI553 Foundations and Applications of Data Mining

Summer 2021

Assignment 1

Deadline: June. 1st 11:59 PM PST

1. Overview of the Assignment

In assignment 1, you will complete three tasks. The goal of these tasks is to help you get familiar with Spark operations (e.g., transformations and actions) and MapReduce, which we learned in Week1 (May 20).

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).

b. There will be **10% bonus** for Scala implementation. **You can get the bonus only when both Python and Scala implementations are correct.**

c. **You are required to only use Spark RDD**, i.e., no point if using Spark DataFrame or DataSet.

2.2 Programming Environment

Python 3.6, Scala 2.11.8, and Spark 2.3.0

We will use Vocareum to automatically run and grade your submission, Vocareum is compatible with these versions. **Please make sure to install the versions listed above.**

Here are the suggestions about development:

1. You can write and test scripts on your local machine.
2. You **MUST** test your scripts in the Vocareum terminal to avoid potential problems.
3. Your homework is ready to submit for grading after steps 1 and 2 can successfully produce results.

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 we can find from the web (e.g., GitHub) as well as other students' code from this and other (previous) semesters for plagiarism detection. We will report all detected plagiarism to the university without any negotiation.

3. Input Datasets

In this assignment, you are provided with two datasets (i.e., reviews and businesses) extracted from the Yelp dataset for developing your assignment.[1] You can access and download the datasets either under the directory on Vocareum: **resource/asnlib/publicdata/** or in the Google Drive: https://drive.google.com/drive/folders/1iLmxTEcqZCvsogMKc2RQhrIC6EXJ3lIR?usp=sharing
(you can download from Google Drive by using your USC email account.)

Here are examples of the contents of the datasets.

Dataset Name	**Example**
review.json	{"review_id": "zJCXxmBCmXaasFC8lltNBw", "user_id": "kn4lbh5vEsqeo5SbHWS_HA", "business_id": "NjDC1Evc2Wa6ykj8xVpoUw", "stars": 4.0, "text": "This was my first time here. I tried the salmon enchiladas with brown rice and Cajun beans.\n\nSalmon is a tricky fish as I have mentioned in previous reviews. I was happy to taste well cooked and tender salmon. The red enchilada sauce was spicy, but not unbearable. \n\nThe Cajun beans were also spicy but tender. They were made with white beans. These beans had the right kind of tenderness to indicate they were not canned beans.\n\nThe brown rice was regular wheat rice. This is a great place for some Baja fish.", "date": "2017-01-02 05:43:58"}
business.json	{"business_id":"sAX1q1kLvOnXKURq_MP_Wg","name":"Park Road Books","address":"4139 Park Rd","city":"Charlotte","state":"NC","postal_code":"28209","latitude":35.1752678,"longitude":-80.8492694,"stars":4.0,"review_count":33,"is_open":1,"attributes":{"BusinessAcceptsCreditCards":"True","BikeParking":"True","RestaurantsPriceRange2":"2","DogsAllowed":"True"},"categories":"Bookstores, Books, Mags, Music & Video, Shopping, Newspapers & Magazines","hours":{"Monday":"9:0-21:0","Tuesday":"9:0-21:0","Wednesday":"9:0-21:0","Thursday":"9:0-21:0","Friday":"9:0-21:0","Saturday":"10:0-18:0","Sunday":"10:0-18:0"}}

We generated these datasets by random sampling. These given datasets are only for your testing. **We will use different sampled subsets for grading.**

Besides reviews and businesses datasets, we provide a stopwords file to help you remove stop words in the reviews. The stopwords file is stored in the same directory with two datasets above in Vocareum and Google Drive.

4. Tasks

You need to submit the following files on Vocareum: **(all lowercase)**

A. Python scripts: **task1.py**, **task2.py**, **task3.py**

B. [Bonus] Scala scripts: **task1.scala**, **task2.scala**, **task3.scala**; Jar package: **hw1.jar**

[1] https://www.yelp.com/dataset

Task	Points (Scala bonus)
Task1: Data Exploration	4.5pts (0.45pts)
Task2: Exploration on Multiple Dataset	4.0pts (0.4pts)
Task3: Partition	4.0pts (0.4pts)

4.1 Task1: Data Exploration (4.5pts)

4.1.1 Task description

You will explore the **review dataset** and write a program to answer the following questions:
A. The total number of reviews (0.5pts)
B. The number of reviews in a given year, **y** (1pts)
C. The number of distinct users who have written the reviews (1pts)
D. Top **m** users who have the largest number of reviews and its count (1pts)
E. Top **n** frequent words in the review text. The words should be **in lower cases**. The following punctuations “(”, “[”, “,”, “.”, “!”, “?”, “:”, “;”, “]”, “)” and the given stopwords are excluded (1pts)

4.1.2 Execution commands

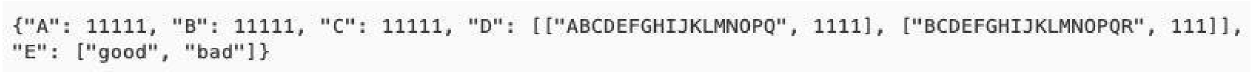
```
Python: $ spark-submit task1.py <input_file> <output_file> <stopwords> <y> <m> <n>
Scala:  $ spark-submit --class task1 hw1.jar <input_file> <output_file> <stopwords> <y> <m> <n>
```

Params:

input_file – the input file (the review dataset)
output_file – the output file contains your answers
stopwords – the file contains the stopwords that should be removed for Question E
y – the given year (see 4.1.1 B)
m – top m users (see 4.1.1 D)
n – top n frequent words (see 4.1.1 E)

4.1.3 Output format:

You must write the results in the JSON format using **exactly the same tags** for each question (see an example in Figure 2). The answer for A/B/C is an integer. The answer for D is a list of pairs **[user, count]**. Counts in the pairs should be integers. All answers should be sorted by the count in the descending order. If two users/words have the same count, please sort them in the alphabetical order. The answer for E is a list of frequent words (**no need to sort but all low cases**). **Please strictly follow the output formats.**



```
{"A": 11111, "B": 11111, "C": 11111, "D": [["ABCDEFGHIJKLMNOPQ", 1111], ["BCDEFGHIJKLMNOPQR", 111]],
"E": ["good", "bad"]}
```

Figure 2: An example output for task1 in JSON format

4.2 Task2: Exploration on Multiple Datasets (4pts)

4.2.1 Task description

In task2, you will explore the two datasets together (i.e., **review and business**) and write a program to compute the average stars for each business category and output top **n** categories with the highest average stars. The business categories should be extracted from the "categories" tag in the business file. **The categories** should be **split by comma** and **removed leading and trailing spaces**. **No other operations needed** to process contents in the "categories" tag in the business file. Stars are extracted from the review dataset. Two datasets are joined by "business_id" tag. You need to implement **a version without Spark (2pts)** and **a version with Spark (2pts)**. You could then compare their performance yourself (not graded).

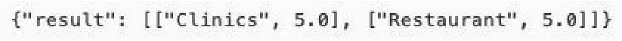
4.2.2 Execution commands

Python: $ spark-submit task2.py <review_file> <business_file > <output_file> <if_spark> <n>
Scala: $ spark-submit --class task2 hw1.jar <review_file> <business_file> <output_file> <if_spark> <n>
Params:

review_file – the input file path (the review dataset)
business_file – the input file path (the business dataset)
output_file – the output file path storing your answers
if_spark – use Spark or not, {"spark", "no_spark"}
n – top n categories with highest average stars (see 4.2.1)

4.2.3 Output format:

You must write the results in the JSON format using **exactly the same tags** (see an example in Figure 3). The answer is a list of pairs **[category, stars]**, which are sorted by the stars in descending order. If two categories have the same value, please sort the categories in **alphabetical order**. Stars in pairs are one decimal. **Please strictly follow the output formats.**



```
{"result": [["Clinics", 5.0], ["Restaurant", 5.0]]}
```

Figure 3: An example output for task2 in JSON format

4.3 Task3: Partition (4pts)

4.3.1 Task description

In this task, you will learn how partitions work in the RDD. You need to compute the businesses that have more than **n** reviews **in the review file**. Other than the default way of partitioning the RDD, you should also design a customized partition function to improve computational efficiency. The "partition_type" is a hyperparameter in your program to decide which partition method to use. For either the **default** or the **customized** partition function, you need to show the number of partitions for the RDD, the number of items per partition, and the businesses that have more than n reviews **(1pts for each partition type)**. Your customized partition function should improve the computational efficiency, i.e., reducing the time duration of execution **(2pts)**.

4.3.2 Execution commands

Python: $ spark-submit task3.py <input_file> <output_file> <partition_type> <n_partitions> <n>

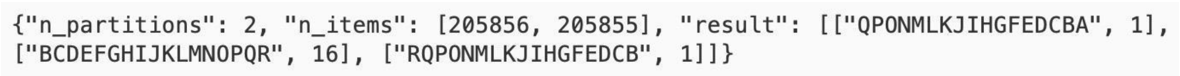
Scala:$ spark-submit --class task3 hw1.jar <input_file> <output_file> <partition_type> <n_partitions> <n>

Params:

input_file – the input file (the review dataset)
output_file – the output file contains your answers
partition_type – the partition function, {“default”, “customized”}
n_partitions – the number of partitions (only effective for the customized partition function)
n – the threshold of the number of reviews (see 4.3.1)

4.3.3 Output format:

You must write the results in the JSON format using **exactly the same tags** (see an example in Figure 4). The answer for the number of partitions is an integer. The answer for the number of items per partition is a list of integers. The answer for the result is a list of pairs **[business, count]** (no need to sort). Businesses in the pairs are the exact same contents and formats as the corresponding tag in the review dataset. Counts in the pairs should be integers.

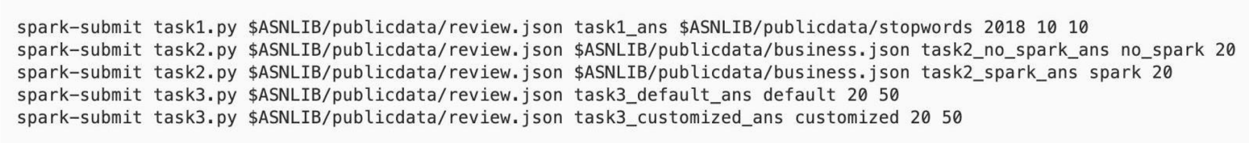


```
{"n_partitions": 2, "n_items": [205856, 205855], "result": [["QPONMLKJIHGFEDCBA", 1],
["BCDEFGHIJKLMNOPQR", 16], ["RQPONMLKJIHGFEDCB", 1]]}
```

Figure 4: An example output for task3 in JSON format

5. About Vocareum

1. The purpose of Vocareum is for you to test if your code can be executed properly on Vocareum and can produce output in the correct format. **We do not accept the regrading request if the submission cannot be run or generate correct output formats on Vocareum.**
2. You can use the provided datasets under the directory: asnlib/publicdata/ (for Vocareum terminal, the directory is $ASNLIB/publicdata/). So you do not need to upload the dataset to Vocareum.
3. You should upload the scripts under your workspace (under directory work/)
4. Once you click on “Submit”, all your code is submitted, and the submission script is automatically run on Vocareum. You will receive a submission report after Vocareum finishes executing your scripts. The **submission report** should include the running time and score of each task for the Python implementation. You can submit scripts on Vocareum as many times as you want. We will grade your last submission before the deadline.
5. You first test your scripts on your machine, and then on Vocareum terminal, and then submit to Vocareum if the testing both on your machine and Vocareum is successful. The submission script may not test all the aspects of your codes, so you **MUST** test your code in the Vocareum terminal as well. Here are the testing commands in the Vocareum terminal for your reference:



```
spark-submit task1.py $ASNLIB/publicdata/review.json task1_ans $ASNLIB/publicdata/stopwords 2018 10 10
spark-submit task2.py $ASNLIB/publicdata/review.json $ASNLIB/publicdata/business.json task2_no_spark_ans no_spark 20
spark-submit task2.py $ASNLIB/publicdata/review.json $ASNLIB/publicdata/business.json task2_spark_ans spark 20
spark-submit task3.py $ASNLIB/publicdata/review.json task3_default_ans default 20 50
spark-submit task3.py $ASNLIB/publicdata/review.json task3_customized_ans customized 20 50
```

6. You could add *--driver-memory 4g --executor-memory 4g* to your spark-submit command to limit its memory usage, in case your code could work properly in your local (with more resources) but would run into memory error.
7. You can find a tutorial about Vocareum in the Week1 folder in D2L.

6. Grading Criteria

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

1. We do not have partial credits. For example, you will get **0** although your result covers 80% answer. **You will also get 0 if your outputs do not follow the format requirements (such as descending order or alphabetical order).**
2. You can use your free 5-day extension separately or together. You must submit a late-day request via https://forms.gle/syyUsyyTM684vf4K6. This form is recording the number of late days you use for each assignment. By default, we will not count the late days if no request submitted. Please see the detail of late requests at Piazza. Also, you are not able to use 5-day extension for the final HWs(i.e., HW4).
3. There will be 20% penalty for the late submission within one week and no point after that. If you use your late days, there wouldn't be the 20% penalty.
4. There will be a 10% bonus for each task (i.e., 0.45pts, 0.4pts, and 0.4pts) if both your python Scala implementations are correct. **The Scala bonus will not be calculated if your Python results are not correct.** There is no partial point for Scala.
5. There will be no point if your programs cannot be executed on Vocareum Please start your assignment early! You can resubmit on Vocareum. We will grade your last submission.
6. There is no regrading. Once the grade is posted on the Blackboard, we will only regrade your assignments if there is a grading error. No exceptions.
7. There will be no point if your submission falls into the following situations:
   a. The submission cannot be executed on Vocareum. Each task will be run five times and graded on the best run.
   b. The execution failure on Vocareum is because of the script naming issue, output file formats issue.