CSCI 5523 Introduction to Data Mining

Spring 2023

Assignment 1 (10 points)

Deadline: Feb. 15th 11:59 PM CDT

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.

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

2.2 Submission Platform

We will use a submission platform to automatically run and grade your submission. We highly recommend that you first test your scripts on your local machine before submitting your solutions. **We will Gradescope to grade your submissions**

2.3 Programming Environment

Python 3.9.12, and Spark 3.2.1

2.4 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) sections for plagiarism detection. We will report all detected plagiarism to the university.

3. Yelp Data

In this assignment, you are provided with two datasets (reviews.json and business.json) extracted from the Yelp dataset for developing your assignment.[1] You can access and download the datasets from the "data" folder on Google Drive. **We generated these datasets in a random sampling way. These given datasets are only for your testing. During the grading period, we will use different sampled subsets of the Yelp datasets.**

4. Tasks

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

Python scripts: task1.py, task2.py, task3_default.py, task3_customized.py

PDF file: task3.pdf

4.1 Task1: Data Exploration (4 points)

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
B. The number of reviews in a given year, **y**
C. The number of distinct users who have written the reviews
D. Top **m** users who have the largest number of reviews and its count
E. Top **n** frequent words in the review text. The words should be in lower cases. The following punctuations i.e., "(", "[", ",", ".", "!", "?", ":", ";", "]", ")", and the given stopwords are excluded. You can access the file that contains stopwords from the "data" folder on Google Drive.

4.1.2 Execution commands

You need to use the following piece of code for reading the parameters:

```
parser = argparse.ArgumentParser(description='A1T1')
parser.add_argument('--input_file', type=str, default='./backup/data/hw1/review.json', help='the input file ')
parser.add_argument('--output_file', type=str, default='./backup/data/hw1/a1t1.json',
                    help='the output file contains your answers')
parser.add_argument('--stopwords', type=str, default='./backup/data/hw1/stopwords',
                    help='the file contains the stopwords')
parser.add_argument('--y', type=int, default=2018, help='year')
parser.add_argument('--m', type=int, default=10, help='top m users')
parser.add_argument('--n', type=int, default=10, help='top n frequent words')
```

You can execute task1.py using the following command line:
$ python task1.py -- input_file <input_file> --output_file <output_file> --stopwords <stopwords> --y <y> --m <m> --n <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

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

y/m/n – see 4.1.1

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 a number. The answer for D is a list of pairs **[user, count]**. The answer for E is a list of frequent words. 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.

```
{"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 (3 points)

4.2.1 Task description

In task2, you will explore the two datasets together (**review and business datasets**) and write a program to compute the average stars for each business category (i.e., first joining datasets then taking average) and output top **n** categories with the highest average stars. The business categories should be extracted from the "categories" tag in the business file and split by comma (also need to remove leading and trailing spaces for the extracted categories).

4.2.2 Execution commands (using argparse as 4.1.2)

Python: $ python task2.py --review_file <review_file> --business_file <business_file> --output_file <output_file> --n <n>
Params:

review_file – the input file (the review dataset)
business_file – the input file (the business dataset)
output_file – the output file contains your answers
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 the descending order. If two categories have the same value, please sort the categories in the alphabetical order.

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

Figure 3: An example output for task2 in JSON format

4.3 Task3: Partition (3 points)

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**. At the same time, you need to show the number of partitions for the RDD and the number of items per partition. You will implement the function using **(1) the default**

partition function with the default partition number; (2) a customized partition function with an input n_partitions, i.e.. design a customized partition function (like a hash function). You need to describe the customized partition function you use, compare the execution time of the two methods (default vs. customized), and examine how the number of partitions affect the execution time. You need to justify the result with one or two sentences (write your answer in a PDF file "task3.pdf").

4.3.2 Execution commands

$ python task3_default.py --input_file <input_file> --output_file <output_file> --n <n>
Params:
input_file – the input file (the review dataset)
output_file – the output file contains your answers
n – the threshold of the number of reviews (see 4.3.1)

$ python task3_customized.py --input_file <input_file> --output_file <output_file> --n_partitions <n_partitions> --n <n>
Params:
input_file – the input file (the review dataset)
output_file – the output file contains your answers
n_partitions – the number of partitions (e.g., 10)
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 a number. The answer for the number of items per partition is a list of numbers. The answer for the result is a list of pairs **[business, count]** (no need to sort).

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

Figure 4: An example output for task3 in JSON format

6. Grading Criteria

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

1. You can use your free 5-day extension separately or together. During grading, TAs will count the number of late days **based on the submission time.** For example, if the due date is 2023/09/10 23:59:59 CDT and your submission is 2023/09/11 05:23:54 CDT, the number of late days would be 1. **TAs will record grace days by default (NO separate Emails)**. However, if you want to save it next time and treat your assignment as late submission (with some penalties), **please leave a comment in your submission.**
2. There will be no point if your submission cannot be executed on the grading platform. **We will provide more information about the submission platform early next week.**
3. There is no regrading. Once the grade is posted on Canvas, we will only regrade your assignments if there is a grading error. No exceptions.