CIS Algorithm and Applications of Data Mining

Assignment 1

**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 for this assignment).

b. You are required to only use Spark RDD

2.2 Programming Environment

**Python 3.7 and Spark 3.0**

**3. Yelp Data**

In this assignment, you are provided with two datasets (i.e., reviews and businesses) extracted from the Yelp dataset for developing your assignment.[[1]](#footnote-1) You can access and download the datasets from the Google Drive: <https://drive.google.com/drive/folders/1XRJxIGqwbaU12G48Ybhw3EvRgbPGPttf?usp=sharing>

**4. Tasks**

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

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

**4.1 Task1: Data Exploration (3pts)**

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** (0.5pts)

C. The number of distinct users who have written the reviews (0.5pts)

D. Top **m** users who have the largest number of reviews and its count (0.5pts)

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 (1pts)

4.1.2 Execution commands

$ python task1.py <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/m/n – see 4.1.1

4.1.3 Output format:

You need to 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.



Figure 2: An example output for task1 in JSON format

**4.2 Task2: Exploration on Multiple Datasets (2pts)**

4.2.1 Task description

In task2, you will explore two datasets together (i.e., review and business) 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 (1pts). 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

$ python task2.py <review\_file> <business\_file > <output\_file> <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 need to 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.



Figure 3: An example output for task2 in JSON format

**4.3 Task3: Partition (3pts)**

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** (1pts). At the same time, you need to show the number of partitions for the RDD and the number of items per partition with **either default or customized partition function** (1pts). You should design a customized partition function (like a hash function) (1pts).

4.3.2 Execution commands

$ python task3.py <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, either “default” or “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 need to 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 result is a list of pairs **[business, count]** (no need to sort).

A screenshot of a cell phone

Description automatically generated

Figure 4: An example output for task3 in JSON format

1. <https://www.yelp.com/dataset> [↑](#footnote-ref-1)