CSCI 5523 Introduction to Data Mining

Spring 2023

Assignment 3 Part II (7 points)

Deadline: Apr. 3rd 11:59 PM CDT

1. Overview of the Assignment

In Assignment 3, you will complete three tasks. You will first implement Min-Hash and Locality Sensitive Hashing (LSH) to find similar businesses efficiently. Then you will implement two types of recommendation systems.

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 use 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

For this assignment, we have generated sample review data from the original Yelp review dataset using some filters, such as the condition: "state" == "CA". We randomly took 80% of sampled reviews for training, 10% for validation, and 10% as the test dataset. (We do not share the test dataset.) You can access and download the data on Google Drive:

a. train_review.json

a. val_review.json – containing only the target user and business pairs for prediction tasks

b. val_review_ratings.json – containing the ground truth rating for the pairs in val_review.json

c. TAs will use the test dataset for grading

4. Tasks

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

a. Python scripts: task2build.py, task2predict.py

Similar to task1.py, the main function has been provided to you. Please do not change the content in the screenshot in your submission. You need to implement the **main()** function.

b. Model files: task2.case1.model, task2.case2.model

c. Result files: task2.case1.val.out, task2.case2.val.out

d. [OPTIONAL] You can include other scripts to support your programs (e.g., callable functions).

4.1 Task1: Min-Hash + LSH (3pts) (see Link)

4.2 Task2: Collaborative Filtering Recommendation System (7pts)

4.2.1 Task description

In this task, you will build collaborative filtering (CF) recommendation systems using *train_review.json*. After building the systems, you will use the systems to predict the ratings for **EVERY** user and business pair in the test file. You are required to implement 2 cases:

- Case 1: Item-based CF recommendation system (5pts)

You will build a recommendation system by computing the Pearson correlation for the business pairs with **at least M co-rated users** (default M=3). During the predicting process, you will use the system to predict the rating for a given pair of user and business. You must use **at most N business neighbors** who are the top N most similar to the target business for prediction (default N=3).

- Case 2: Item-based CF recommendation system with Min-Hash LSH (2pts)

You will combine the Min-Hash and LSH algorithms in your item-based CF recommendation system since the number of business pairs might be too large to compute. You will rerun Task1 to identify the potential business pairs whose **Jaccard similarity >= 0.01**, to serve as the business similarity (neighbor weights) used in the predicting process. You must use **at most N business neighbors** who are the top N most similar to the target business for prediction (default N=3).

4.2.2 Input Format (Please make sure you use exactly the same input parameters names!)

We use the "**argparse**" module to parse the following arguments.

Parameters in task2build.py

1. Input file path (**--train_file**): The path to the train file including path, file name and extension
2. Model file path (**--model_file**): The path to the model file including path, file name and extension
3. Time file path (**--time_file**): The path to the time file including path, file name and extension
4. Threshold (**--m**): The threshold for co-rated users for the business pairs (default: 3)

Execution example:

$ python task2build.py --train_file <train_file> --model_file <model_file> --time_file <time_file> --m <co_rated_thr>

Example: $ python task2build.py --train_file train_reviews.json --model_file task2.model --time_file task2.time --m 3

Parameters in task2predict.py

1. Input file path (**--train_file**): The path to the train file including path, file name and extension
2. Input file path (**--test_file**): The path to the test file including path, file name and extension
3. Model file path (**--model_file**): The path to the model file including path, file name and extension
4. Model file path (**--output_file**): The path to the output file including path, file name and extension
4. Time file path (**--time_file**): The path to the time file including path, file name and extension
5. Threshold (**--n**): The threshold for the number of business neighbors (default: 3)

Execution example:

$ python task2predict.py --train_file <train_file> --test_file <test_file> --model_file <model_file> --output_file <output_file> --time_file <time_file> --n <n_weights>

Example: $ python task2predict.py --train_file train_reviews.json --test_file test_reviews.json --model_file task2.model -- output_file task2.out --time_file task2.time --n 3

4.2.3 Output format

Model Format:

You must write a business pair and its similarity in the JSON format using exactly the same tags like the example in Figure 1. Each line represents a business pair, i.e., "b1" and "b2". You can truncate decimals for 'sim' if the model file is larger than 100M, that is the maximum file size on Gradescope.


```
{"b1": "cYwJAgA6I12KNsd2rtXd5g", "b2": "Fid2ruy5s600SX4tvnrFgA", "sim": 0.032448377581120944}
{"b1": "7zecrDCEugcx8bgFn9LbLQ", "b2": "1VvxsddAoINg8TJXOZgEfg", "sim": 0.018867924528301886}
```


Figure 1: An example output for model in the JSON format

Prediction Format:

You must write the prediction for **EVERY** target pair in the JSON format using exactly the same tags like the example in Figure 2. Each line represents a predicted pair of ("user_id", "business_id"). You can truncate decimals for 'stars' if the result file is larger than 100M.

```
{"user_id": "1vXJWH7Lsdzsd8aU3SOsdA", "business_id": "ZzvfffV9kFY3ysdSgyRUBQ", "stars": 3.607958829899405}
{"user_id": "2svfwyX1hn2lsdjv5Sn36w", "business_id": "JAmQCczUclsDUfsdjNdjQA", "stars": 1.442154461436827}
```

Figure 2: An example output for task3 in JSON format

4.2.4 Grading

We will grade your model based on the number of valid business pairs and the execution time. To get the full score, your model should contain more than 1,200,000 pairs (1pt) and the execution time on Gradescope should be less than 400 sec (1pt).

Setting ID	Settings	Number of Pairs	Time
1	co_rated_thr=3	1,200,000	400s

During the submission period, your prediction will be graded by comparing against the ground truth in val_review.json. We use RMSE (Root Mean Squared Error) defined in the equation below to evaluate the prediction performance,

$$RMSE = \sqrt{\frac{1}{n}\sum_i w_i (Pred_i - Rate_i)^2}$$

where $Pred_i$ is the prediction for the i-th pair of ("user_id", "business_id") and $Rate_i$ is the true rating for the pair. n is the total number of the user and business pairs.

Your script must predict for **ALL** the pairs in the given test file. We will compute **(1) the overall RMSE** and **(2) the RMSE for the pairs whose true ratings are in {1, 2, 5}** because we want to examine the model capability in predicting good (rating=5) and bad (rating=1,2) reviews. To get the full score, your results should achieve lower RMSE (1pt each) than the below table and the execution time on Gradescope should be less than 40 sec (0.5pt each).

Case	Settings	RMSE	RMSE_125	Time
Case 1	n_weights=3	0.85	0.95	40s
Case 2	n_weights=3	0.85	0.95	40s

During the grading period, TAs will grade your code with the test dataset, which has the same data distribution as the val dataset.

5. 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.
4. Homework assignments are due at 11:59 pm CT on the due date and should be submitted on Canvas. Late submissions within 24 hours of the due date will receive a 30% penalty. Late submissions after 24 hours of the due date will receive a 70% penalty. Every student has FIVE free late days for homework assignments. You can use these free late days for any reason, separately or together, to avoid the late penalty. There will be no other extensions for any reason. **You cannot use the free late days after the last day of the class.**

6. Submission

The Gradescope submission will be available on **Mar 20th**.