CS 47300: Web Information Search and Management
Project 2 - Collaborative Filtering *Last updated 10/28 12:30pm*

Due 11:59pmEDT Friday, 06 November, 2020

Overview

The task is to implement collaborative filtering on a food recipe dataset. You will perform a systematic evaluation of various methods. Please read the instructions carefully before dive into answering questions. *Note that this is a long project, and we strongly encourage you to start early.*

Dataset

The dataset needed for the project is available in the `/homes/cs473/project2/data/` folder, accessible from CS teacing lab machines (we suggest mc17 and mc18.cs.purdue.edu). It consists of 3 files:
1. `/homes/cs473/project2/data/dishes.csv`: This file consists of IDs, names, and ingredients of the dishes; 1 indicates the ingredient's presence in a dish, and 0 indicates its absence.
2. `/homes/cs473/project2/data/user_ratings_train.json`: This file contains the ratings of 500 users. Each user has a list of tuples corresponding to the dishes the user has already rated. The first element of the tuple is the dish_id, and the second item is the user's rating for that dish. You need this file to train.
3. `/homes/cs473/project2/data/user_ratings_test.json`: This file contains rating information about 100 users which is a subset of users in train users (but with different dishes). The file format is same format as the `user_ratings_train.json`. You will use this file to test your models.

Tasks and Evaluation

. You are going to perform two types of tasks.

- Task 1: Predicting users' ratings: The task is to predict a user's rating for dishes where they have not rated yet. For this task, you will use Mean Absolute Error (MAE), which is the average absolute deviation of the predicted ratings on items' actual ratings. For details, see slide `https://www.cs.purdue.edu/homes/clifton/cs473/CollabFilterMod.pdf`.
- Task 2: Recommending documents to users: You need to recommend dishes to users. The goal is to recommend dishes that the users will like the most. You can assume that if a user rates a dish higher, they are most likely like the dish. You can rank the unrated dishes based on its predicted rating and present it to the user in descending order of predicted rating. You will use rank-based metrics such as Precision, and Recall to report results on this task.

Implementation Details

- The user rating is given in the wide-scale range: 1-5. To calculate MAE, you need to consider `user_ratings_test.json`. This file contains the ground truth information. For MAE, you need to predict rating per dish id and compare it with the ground truth.
- For the rank-based metrics precision, and recall, a binary rating should be sufficient (like/dislike). The wide-scale range can be converted to binary rating as follows: If *rating* ≥ 3, then like (1), otherwise dislike (0). You can consider that the liked (1) dishes are ground truth relevant to users while calculating the rank-based metric scores. All other dished with dislike (0) or not rated are non-relevant.

Part 1: Collaborative Filtering (50%)

You need to implement any **one** of the following two options:

Option 1: Memory-based Collaborative Filtering

For this part, you need to implement Task 1 and Task 2 using a memory-based collaborative filtering method. Compute the performance of this method on corresponding evaluation metrics.

Implementation . You can implement the memory-based approach that we discussed in class to predict users' ratings for dishes. As for the measure of similarity, feel free to use any similarity you find appropriate. See the discussion of Pearson Correlation Coefficient and Vector Space Similarity at `https://www.cs.purdue.edu/homes/clifton/cs473/CollabFilterMem.pdf`. Please clearly mention which measure(s) of similarity you use to report the results. It is a good idea to try several choices and see how they perform. Note that, you only use `dish.csv` and `user_ratings_train.json` files to train. You will only use `user_ratings_test.json` to report *testing* performance.

Option 2: Model-based Collaborative Filtering

The idea is to incorporate a content-based clustering and see how it impacts Task 1 and Task 2. We leave it open to what model-based approach you use. Please refer to the class slide for details discussion on model-based approaches at `https://www.cs.purdue.edu/homes/clifton/cs473/CollabFilterMod.pdf`.

Here are some hints which you might find helpful. You can do a content-based soft clustering of documents, then use a memory-based approach to give each cluster a "recommendation score" to each user. A document recommendation is based on a weighted average of the cluster scores that document clusters in to. A simple way to give a recommendation score is to average all the available ratings of a user. However, you can think of a more effective way to get the "recommendation score" of clusters.

What to include in the report .

1. Briefly describe your implementation in text and refer to your code's file(s) name. The file locations should be relative to the folder you submit through turnin.
2. Report the MAE for Task 1 (user rating) on testing user data. You can use the provided test user file, but you may want to define your own testing data as well to get a more effective test. But be careful not to "test on the training data", as this would not be a valid test.
3. Report the Precision@10, Precision@20, Recall@10, and Recall@20 for Task 2 (dish recommendation). Again, you can develop additional test data or perform something beyond simply using the provided test user file if you think it is appropriate.
4. Based on these results, briefly summarize your findings. One or two paragraphs should be adequate.

Part 2: Build your own model (25%)

For this part, you will build a second model to compare with the one you implemented in Part 1. You should try something you think will result in improved results over what you did in Part 1.
Here are a few things to consider.

- If you chose Option 1 in Part 1, you can use Option 2 for this question's requirement. Similarly, you can use Option 1 if you already did Option 2.

- For full credit, at least one of the parts should involve clustering based on the “text” (ingredient list) of the recipes.
- If you find yourself pressed for time, you can earn partial credit by making a change to your part 1 solution that you think will make an interesting difference in performance (by which we mean the quality of the results.) For example, if you select Option 1 and use vector space similarity in part 1, you can choose a different similarity measure (e.g., Pearson Correlation Coefficient) as part 2.

What to include in the report.

1. Briefly describe your implementation in text and refer to your code’s file(s) name. The file locations can be relative to the folder you submit through turnin.
2. Report the MAE for Task 1 (user rating) on testing user data. Again, you can define your own test data, but remember that in Part 3 you will want a fair comparison between Parts 1 and 2.
3. Report the Precision@10, Precision@20, Recall@10, and Recall@20 for Task 2 (dish recommendation).
4. Based on these results, briefly summarize your findings. One or two paragraphs should be adequate.

Part 3: Comparison (25%)

In this part, you will compare and analyze the best performing model. Use the results you obtain in Parts 1 and 2. Use relevant evaluation metrics to support your claims.

What to include in the report .

1. Include at least one Table or Figure comparing approaches in Part 1 and Part 2. Add any other figures/tables you find relevant.
2. Briefly analyze the results in the text. This would mostly be describing the tables/figures you add to the report. One or two paragraphs should be good enough.
3. Report two examples (one for Task 1, and one for Task 2) where Part 1 approach performs better than Part 2. Briefly explain why.
4. Report two examples (one for Task 1, and one for Task 2) where Part 2 performs better than Part 1. Briefly explain why.
5. Compare what happens for the two models if you add a new recipe to the dataset (one for which you have no user ratings, just a list of ingredients.)

Part 4: Other Things to Consider

If you want to test your models on larger datasets, use the users data available at `/homes/cs473/project2/data/big_1000`, and `/homes/cs473/project2/data/big_2000`. You can use the same `dishes.csv` file. See how the performance changes when you add more data for training. Your code might take a longer time (should be in minutes) to run, and the memory requirements might be higher. Make sure you complete other parts before trying this. Include your analysis in the report if you want to get feedback.

You might already guess that the instruction we provide is a recipe for writing a scientific article to compare different methods systematically. We hope you find the recipe useful. If you find yourself developing novel ideas/algorithms/methods, you may want to write a research article describing your results. It so happens there is an appropriate workshop with a December 14 deadline: `http://research.nii.ac.jp/`

`decor/decor2021.html`. If you do want to do this, feel free to reach out to us. We will be happy to assist you in putting together a submission. However, you will likely need to go beyond the ideas in Options 1 and 2 to have something that would be accepted, but it is worth a try. While this part will not be graded, we would be excited to see you participate.

Submission

Step 1: Submit Code. Submit a single folder named project2/ using turnin. The folder must contain two files, MODELPART1 and MODELPART2, that, when executed, run your codes. MODELPART1 runs the Part 1 model, and MODELPART2 runs the Part 2 model. Each model should output as follows:

```
Task 1 MAE: <value>
Task 2 Precision@10: <value>
Task 2 Precision@20: <value>
Task 2 Recall@10: <value>
Task 2 Recall@20: <value>
```

These are the values you generate on testing your models on `user_ratings_test.json` file.
Use the following command to submit the project:
turnin -c cs47300 -p project2 <path-to-submission-folder>
You may submit any number of times before the deadline. Only your most recent submission would be recorded.

To view submitted files, use:
turnin -c cs47300 -v -p project2
Don't forget the -v option, else it will overwrite your submission with an empty folder.

Step 2: Gradescope. Submit the report.pdf file on Gradescope. The Report should includes instructions to run your code and your observations. Please refer to the instructions of different parts to see what needs to be included in the report.