**Python files for data preprocessing:**

Python file 1: ‘convert\_json\_to\_csv.py’

The data from yelp are huge json files, we use this python file to convert json files to csv files that we can process in python notebook or editors like pycharm.

Python file2: ‘restaurants\_AZ.py’

The ‘business.csv’ file contains all the merchants including non-restaurants like shopping mall or auto repair shops, so we use the file to choose restaurants from all the merchants. Since we want to make our recommendation more specific, we will let the user choose a specific state, say AZ, and make local recommendation system.

Python file3: ‘user\_top1000.py’

The data has millions of users, so it usually takes a long time for us to go through the whole user data file. Among them over 95% users only have less than 20 reviews, which will show much less accuracy when we do nlp analysis. Thus, we rank these users by their review accounts in ‘user.csv’ file, and choose the top 1000 users as sample to do further analysis. Actually, all the analysis and results can apply to other users, so we think it’s ok to narrow our user sample.

Python file4: ‘review from user top1000.py’

According to the statement above, we choose top 1000 users as sample, so correspondingly, we have to get their reviews. We just use this python file to get reviews of these ‘sample’ users.

**Python files for recommendation edition1:**

Python file5: ‘recommendation edition1.py’

Recommendation edition1 is based on the idea of collaborative filtering. In this file, we will collect the stars each user give for each restaurant and get a very sparse matrix. Since stars can be identified as the preferences of users, by normalizing and decomposing this matrix, we can make recommendations for users.

**Python files for recommendation edition plus:**

**filter building:**

Python file 6: ‘categories\_filter.py’

User can choose the categories he/she likes to have for this meal, and our system will this file to filtrate restaurants meeting the category requirement.

Python file7: ‘filter.py’

Combine the categories filter and open time filter (Rebecca has uploaded it). User can filtrate restaurants by choosing

the categories and time he/she would like for this meal

**NLP analysis:**

Python file8: ‘nlp\_user.py’

In order to make recommendations, we want to get the features of users, more specifically, we want to get the features the user like. So for each user, we use this file to choose the restaurants he/she gives no less than 4 stars (hopefully have positive review), collect all the reviews the user gives for these restaurants and do NLP analysis.

Python file9: ‘nlp\_restaurant.py’

In order to make recommendations, we want to get the features of restaurants. We use this file to collect all the reviews the restaurant receives and do NLP analysis.

**Features for ranking:**

**We will have three features for restaurants ranking:**

**Global star**

**Similarity**

**Distance**

Python file10: ‘similarity\_calculation.py’

In order to make recommendations, we want to match the users and the restaurants, so we use this file to get the similarity between the features of each user and each restaurants.

Python file11: ‘geo\_feature\_normalization.py’

We have the function that can get the distance of the user and the restaurants (Arpit has uploaded it). The distances we get are usually in a large number, in order to make it a feature for ranking, we use this file to normalize these distances.

Python file12: ‘unsupervised learning.py’

Now for each user, we have a filtered restaurants list based on global star, similarity and distance, but we have no labels to test, so we use this file to do unsupervised learning.(however, we will only get clusters not ranking, that’s a problem)

**Cold Start Problem**

For cold start problem, the idea is that, we will let the new user A to choose the categories he/she likes (including cuisine and food), and we will also get the preferred categories the each old user like. If the new user’s preferred categories are very similar to one old user B, we will say A and B are similar users, and we will recommend A what we will recommend to B.

Python file13: ‘categories\_AZ.py’

We use this file to let all the restaurants categories

Python file14: ‘categories\_user\_like.py’

For each user, we will use this file to collect the categories of the restaurants receiving no less than 4 stars from him/her, and take them as his/her preferred categories

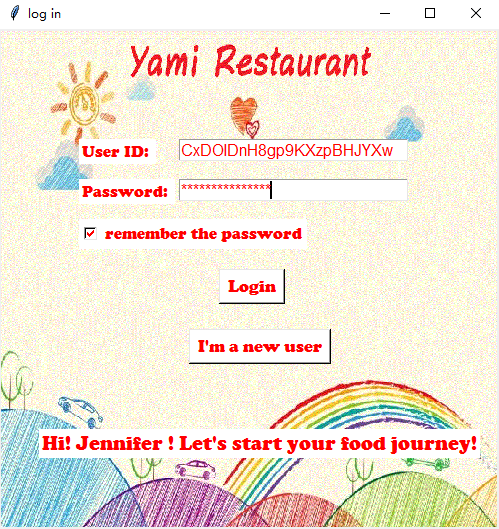
Python file15: ‘new\_user.py’

In this file, we will let the new user input some of his/her information to sign up, then based on the information, find the similar user.

**User Interface:**

We use tkinter to build all the interfaces based on the code given above. (some codes are changed to fit the grammar of tkinter and some are optimized to get a shorter run time)

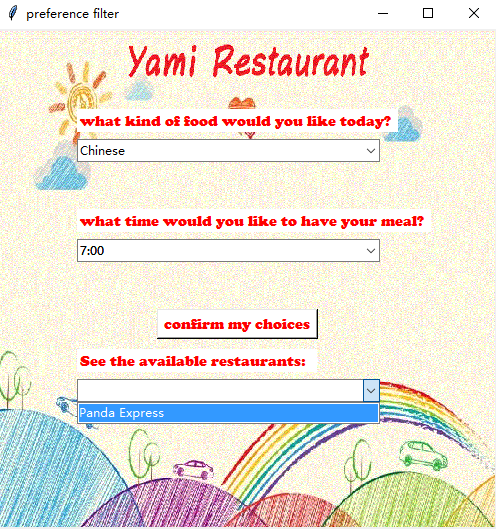
Python file16: ‘login\_interface\_final.py’



Python file17: ‘recommendation\_edition1\_interface\_final.py’



Python file18: ‘filter\_interface\_final.py’



Python file19: ‘new\_user\_interface.py’

