Task 1: Building a Business Recommendation System

Introduction

Nowadays almost everyone has at least one social media account, and people spend a lot of time on it. As social media becomes more and more popular, aside from advertisements, businesses like Yelp may take advantage of that and recommend restaurant and other businesses to users based on users' activities and account information. For the first task, we are building a business recommendation system using Yelp's dataset with 2 different approaches: model-based collaborative filtering and memory-based collaborative filtering.

Data Setup

For the first task, we will be using Yelp's user.json, business.json and review.json. We will be using Python 3 as our programming language and json and pandas DataFrame will be used to process data.

First, we loaded user.json, business.json and review.json as our raw data into Python using Python's json module. Next, we selected data from business.json, we do not want the data size to be too big or too small, and we want to have businesses that have enough review count,, thus we selected Toronto and Phoenix as our target cities, and stored all businesses in those cities into separated pandas DataFrame, each business we selected only has Business ID as attribute.

After getting businesses from business.json, we set user review count threshold to N, and selected users that has at least N reviews, and stored them into a pandas DataFrame, each user has Review Count and User ID as attributes.

Next, we collected reviews from review.json, this time we took a slightly different approach. Because review.json is significantly larger than user.json and business.json, parsing review.json directly will take a lot of computational power and time to complete. Therefore we used a dictionary to parse review.json. We initialized 2 dictionaries first, user_dict and business_dict, the key of each dictionary will be User ID and Business ID that we read from user DataFrame and business DataFrame, and set their values to be 0. As we read through review.json entry by entry, if one review entry's User ID is in our user DataFrame and its Business ID is in our business DataFrame, we added 1 to both the value of that User ID in user_dict and the value of that Business ID in business_dict. Finally we store this review entry into an initial review DataFrame,

After getting the initial review DataFrame, we check each entry that the reviewer wrote at least K_1 reviews (K_1 will be the value of the corresponding User ID in user_dict) and the business received at least K_2 reviews (K_2 will be a threshold we manually set at the beginning). We stored all qualified review with User ID, Business ID and Stars as attributes into a pandas DataFrame as our final review data. We split our review data into training set and testing set that 75 percent of reviews will be training set and 25 percent will be testing set.

Finally, we created a new dictionary to store reviews ratings from one user to one business. We read through our final review DataFrame, for each entry, we use User ID + Business ID as its key and stars as value. We made a numpy matrix to store all the ratings with its columns to be Business ID and its rows to be User ID, and each cell will be the corresponding rating from that specific user to that specific business.

Methodology

Now that we have the review data that satisfies our pre-defined thresholds, we can build a N by M matrix based on the data. N represents the number of users and M represents the number of businesses. Each element (i, j) in the matrix represents review score that user i gives to business j.

Collaborative Filtering

We are trying to build a recommending system to recommend restaurant to the Yelp users. Traditionally there are two ways to build a recommender system: content-based method and collaborative filtering (CF). Content-based recommender relies on the features about the users and businesses, which in our case are hard to extract form the limited data. Besides, content-based recommender also suffers if the input data is very sparse. We will shift our focus onto collaborative filtering which is capable of learning the data features all by itself and dealing with sparse data properly. Two main CF techniques are applied in our recommender: memory-based and model-based recommender.

Memory-based collaborative filtering approach is mainly based on similarity and can be easily divided into two sections: user-item filtering and item-item filtering. The ideas behind this approach are that if two users are similar, they may like similar restaurants and if two restaurants are similar, they may get similar reviews. In other words, for user-item filtering if user A and user B share similar ratings on some restaurants, they may also have similar options on other restaurants. In contrast item-item filtering will take a restaurant, find similar restaurants and then predict the ratings based on the similarity.

However, there are several limitations for the memory-based CF techniques, such as the fact that the similarity values are based on common items and therefore are unreliable when data are sparse and the common items are therefore few. To achieve better

prediction performance and overcome shortcomings of memory-based CF algorithms, model-based CF approaches have been investigated. Model-based collaborative filtering approach uses pure rate training data and tries to figure out the patterns behind the data.
Matrix Factorization (MF) model is an unsupervised learning method for latent variable decomposition and dimensionality reduction. The goal of MF is to learn the latent preferences of users and the latent attributes of items from known ratings to predict the unknown ratings. The main idea is that because the training data is of high dimension but low rank thus being very sparse, we can actually fit the whole data with the product of some low rank matrix.

Algorithm Implementation

Memory-based

For memory-based CF, it naturally divides into two classes: user-based and business-based, because we can compute both the similarity between the users and businesses. The key part of the algorithm is to is to train a similarity matrix for the given training data. For each pair of the users or businesses, we compute their similarity based on their cosine similarity. So user similarity matrix will be a N by N matrix in which element at (i, j) represents the similarity between user i and user j. Same rules apply to the M by M business similarity matrix in which element at (i, j) represents the similarity between business i and business j.

With the similarity matrix, we can use the weighted similarity sum of review scores from other users or businesses. Different users may have different score patterns: some usually gives higher scores while some are quite conservative for their ratings, we need to normalize the data by removing the mean review scores for each user if we want to eliminate this effect in the user-based method. At the predicting stage, we shall add back their mean values.

Model-based

For model-based CF, a well-known matrix factorization method is Singular value decomposition (SVD). The basic idea of SVD is to divide the original matrix into three small matrices:

$$M = U\,S\,V^T$$

M: original matrix N by M
U: N by r matrix
S: r by r matrix with non-negative real numbers on the diagonal
V: M by r matrix

Because r is a much smaller number than N and M, typically between 20 to 100, we successfully decomposes a big matrix into smaller low rank matrices. Then we can use the product of these matrices to recover the review scores.

Evaluation

As we have our algorithms implemented, we shall then consider the evaluation methods. Commonly used evaluation methods would be Root Mean Square Error (RMSE) and Mean Absolute Error (MAE). So we will apply both measurements to see how our algorithms work.

To begin with, we pick two cities: Toronto and Phoenix, because both cities have approximate 1,5000 businesses in town. Only Las Vegas has more restaurants than the two cities and the rest cities has far less restaurants.

Besides, our algorithms perform differently for different data sparsity, so we need to see the performances under different conditions to get a good measurement. For the sparse condition, we pick the users with more than 20 reviews and businesses with more than 5 reviews, while for the less sparse condition, we pick the users and businesses with more than 100 reviews.
The evaluation results are show in the following tables.

		User-based	Business-based	Model-based
Toronto	RMSE	1.054	3.685	1.054
	MAE	0.841	3.518	0.841
Phoenix	RMSE	1.137	3.895	1.138
	MAE	0.893	3.708	0.895

User review # >= 20, Business review # >= 5
Toronto sparsity = 99.38%, Phoenix sparsity = 99.37% (whole data set)

		User-based	Business-based	Model-based
Toronto	RMSE	0.940	3.272	0.954
	MAE	0.744	3.127	0.755
Phoenix	RMSE	0.906	3.315	0.920
	MAE	0.692	3.156	0.704

User review # >= 100, Business review # >= 100
Toronto sparsity = 83.18%, Phoenix sparsity = 80.91% (whole dataset)

Conclusion

It is clear from the above tables that user-based CF and model-based CF have done a good job in predicting the review scores of the users and their performances are quite similar. But the business-based CF does not perform to the similar level. Another observation is that with more data available, the better performance all these algorithms will achieve.
The reason for the bad performance of the business-based CF may be caused by the sparsity of our input data. It is very possible that two restaurants have very few reviews by the same user thus leading to a very low similarity or even no similarity. This will affect prediction based on the weighted review scores of other restaurants.

The limitations of our approach for task 1 are mainly in three aspects.
First, with very sparse data we cannot further improve the performance of the business-based algorithm. With small overlap of different businesses, their similarity will be quite different and then we will have no guarantee on the weighted sum based on the similarity matrix.
Second, we only use the data with pure review content. It is important for a recommending system to include some features of the users and businesses to perform a hybrid CF. But the data released by the yelp is quite limited so we did not put the user and business features into consideration. If more data about the users and businesses are available, it is interesting to see how hybrid CF could perform.
Third, our CF approach cannot solve the cold start problem. We can only handle well for the users who have done some reviews to some restaurants. If a total new user joins the data, we can do nothing but just give them recommendations based on the average scores of the restaurants without any personalization.