Yelp Recommendation System Based on Collaborative Filtering

Sainan He, Jiaoyang Fu, Yameng Li

Electrical and Computer Engineering, University of Waterloo, Canada,
s66he, j45fu, y949li@uwaterloo.cau

Abstract. ¡¡¡¡¡¡¡ HEAD Based on Yelp Data Challenge dataset, we aim to develop a predictive personalized recommendation system on users review star rating for restaurants, applying collaborative filtering algorithms. In particular, we implement and compare the performances of four algorithms including baseline, User-based and Item-based collaborative filtering and Singular Vaule Deccomposition (SVD). We evaluate our results by comparing our predicted rating to the actual rating using Root Mean Squared Error(RMSE) and Mean Absolute Error(MAE) metrics.

Keywords: Recommendation System, Collaborative filtering, Singular Vaule Deccomposition (SVD), Yelp Data Challenge

1 Introduction

With rapid development of advanced technology, people nowadays can achieve the things that they desired faster and more effectively than ever. While, at the same time, the requirements for accurate, personalized and convenient services are increasing. Fortunately, Yelp contributes to providing reasonable recommendations of various businesses to users, e.g., restaurants.

Yelp collected review dataset which records how well each user rates for limited amount of restaurants. Based on these review history, it suggests favored restaurants to users. The problem is that Yelp seems to offer similar recommendations that are popular for various users. In fact, people may have diverse preferences to food. For instance, some users may prefer Asian food, while others may favor Mexican food. In fact, yelp did not consider these factors too much. Therefore, it seems still have some room to improve the recommendation system in the personalization aspect.

Collaborative filtering, a method making predictions based on a large dataset used by some recommendation system like Yelp, Amazon, and Netflix. It automatically predict about the interests of a user by collecting preference or tastes information from other users. Furthermore, there are numerous collarborative filtering methods, such as baseline, K- nearest neighbor, and matrix decomposition. This paper aims to compare how accurate each methods applying on Yelp Data Challenge through using Root metric Mean Squared Error(RMSE) and Mean Aboslute Error(MAE).

2 Literature Review

There are generaly three types of recommendation systems: content-based filtering, collaborative filtering and hybrid approaches. Content-based recommendation systems work with users profile and items characteristics. The feature used to bulding profiles are often a set of keywords. For example, a music recommendation system[**?**] implemented with content-based filtering, each song is assigned an attribute manually. If a users profile shows interests in songs with particular atrtibutes, similar songs will be recommended to the user. The limitations of these systems is that it always recommends similar items to user that he has already purchased and its difficult to recommend items for new users.

Collaborative filtering try to predict the utility of items for a particular user based on the items previously rated by other users with similar tastes and preferences. In[**?**] the author builds a recommendation system for a retail store using three kinds of collaborative filtering algorithms - memory based approach, matrix factorization and bigram matrix method. Collaborative filtering also has limitations that it is difficult to recommend items for new users, to recommend items which have not been rated before, and to recommend when rating information is insufficient.

Hybrid approach combines multiple techniques to overcome the limitations of individual systems. A restaurant recommendation system for yelp user[**?**] adopts hybrid approach by extracting collaborative and content-based features to identify customer and restaurant profiles. A hybrid cascade of K-nearest neighbor clustering, weighted bi-partite graph projection, and several other learning algorithms are proposed.

3 Data

We collect our data from Yelp recommendation Kaggle competition[**?**],This dataset contains 11,537 businesses, 8282 check-in sets, 43873 users, and 229907 reviews. Each user has a unique user id as well as business. We target Restaurant in the city of Phoenix as it is more reasonable to recommend restaurants for users in the same city and Phoenix has the most amount of review records among all the cities.

3.1 Data Processing

The original data file is in json format. We firstly parse the raw data of imformation from users, businesses and reviews, respectively and merge them into one dataframe. Then we extract the records with features of Restaurants and Phoenix for futher analysis. After this approach, we have 17145 users, 1454 business and 52749 reviews.

Because many users only review a few restaurants, our dataset yields a sparsity of 99.79% which can be visualized from figure 1. To solve this problem, we create a smaller dataset which only consider restaurants reviewed by more than 50 users.

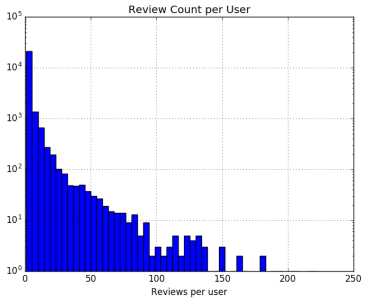

Fig. 1. User review count

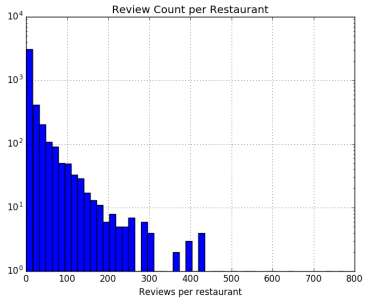

Fig. 2. Restaurants review count

3.2 Traing and Testing sets

In order to evaluate the performance of our recommendation system, we divided our data into training and testing datasets. We extract a list of all restaurants each user has rated, take 80% of this list as training set and 20% as test set.

4 Methods

4.1 Baseline

Our baseline model is similar to the model implemented by[**?**] which is a mean predictor and accounts for the user and item effects.

$$b_{ur} = \mu + b_u + b_r \ . \tag{1}$$

Here μ is the mean rating of reviews of all business by all users. The parameter b_{ur} indicates the difference between the average rating of user u and μ. The parameter bi indicates the difference between the average rating of business i and μ. This will normalize the widely noticed tendency of some user giving higher rating than others and some restaurants getting higher ratings than others.

4.2 User-User Collaborative Filtering

User-user collaborative filtering, also know as *k-NN collaborative*,was the first of the automated CF methods. It find other users whose past rating behavior is similar to that of current user and use their ratings on that item to predict what the current user will like. To predict Mary's preference for an item she has not rated, user-User CF looks for other users who have high agreement with Mary on the items they have both rated.These users ratings for the item in question are then weighted by their level of agreement with Marys ratings to predict Mary's rating on that item.

4.2.1 Computing Predictions

To compute predictions or recommendations for a user u, user-user CF firstly needs to determine the number N of neighbors will be used to generate the result. Then computing the weighted average of the chosen neighboring users' rating i by using similarity as weights. The formula is given as below:

$$p_{u,i} = \bar{r}_u + \frac{\sum_{u' \in N} s(u,u')(r_{u',i} - \bar{r}_{u'})}{\sum_{u' \in N} |s(u,u')|} \tag{2}$$

In order to eliminate the differences in users's use of the rating scale, subtracting the user's mean rating $\bar{r}_{u'}$ to compensate is necessary. The parameter $p_{u,i}$ is predicated rating on item i for user u. $\bar{r}_{u'}$ is average rating on all items rated by user u.The parameter $r_{u',i}$ indicates the rating of user u' on item i.$s(u,u')$ is similarity between user u and u'. N is the number of neighbors chosen for user u.

4.2.2 Computing User Similarity

An critical parameter used to calculate predications is similarity function.Among many proposed similarity functions we only choose Cosine Similarity to give a detailed introduction.

In Cosine Similarity model, users are represented as $|I|$-demensional vectors of rating on $|I|$ items.Similarity is measured by the cosine distance between two rating vectors. The formula is given below indicating how to calculate the Cosine Similarity between user u and v.

$$s(u,v) = \frac{r_u \cdot r_v}{\|r_u\|_2 \|r_v\|_2} = \frac{\sum_i r_{u,i} r_{v,i}}{\sqrt{\sum_i r_{u,i}^2}\sqrt{\sum_i r_{v,i}^2}} \tag{3}$$

Unknown ratings are considered to be 0. r_u is rating vector of user u. $\|r_u\|_2$ is the Euclidean norm of rating vector r_u.

4.3 Item-Item Collaborative Filtering

Item-Item Collaborative Filtering uses similarity between the rating patterns of items.If two items tend to have the same users like and dislike them,then they are similar and users are expected to have similar preferences for similar items.

4.3.1 Computing Predictions

To generate predictions or recommendations for user u on item i, item-item CF firstly determine a set S of items most similar to i. Then computing the weighted average of the user's rating on item j from set S. The formula is given as below:

$$p_{u,i} = \frac{\sum_{j \in S} s(i,j) r_{u,j}}{\sum_{j \in S} |s(i,j)|} \tag{4}$$

The parameter $p_{u,i}$ is predicated rating on item i for user u. S is a set of items most similar to item i. $s(i,j)$ is the similarity between item i and j. $r_{u,j}$ is the rating of user u on item i.

4.3.2 Computing Item Similarity

As in user-user collaborative filtering, a variety of methods can be used for computing item similarity. In this section, we focus on Cosine Similarity, the most popular similarity metric, as it is simple, fast, and produces good predictive accuracy.

$$s(i,j) = \frac{r_i \cdot r_j}{\|r_i\|_2 \|r_j\|_2} \tag{5}$$

Unknown ratings are considered to be 0. r_i is rating vector of item i. $\|r_i\|_2$ is the Euclidean norm of rating vector r_i.

4.4 Singular Value Decomposition

Singular Value Decomposition(SVD) is a latent factor method which popularly applied in Netflix Prize competition. A matrix can be approximated with SVD by multiplying two generated feature matrices P and Q with rank k.

$$\hat{R_{ij}} = P_i Q_j^T \tag{6}$$

In this paper, the matrix R is a ratings matrix with rows representing all the users, and columns representing businesses. For example R_{ij} represents the rating from user i to restaurant j. Then, minimizing the squared differences between the rating matrix R and the SVD approximated matrix $\hat{R}$. Thus,projecting user and restaurants rating matrix into two lower dimension feature matrices P as user feature matrix and Q as restaurant matrix. In order to improve the SVD prediction accuracy, subtract the global average rating, μ.

$$\hat{R_{ij}} = P_i Q_j^T + \mu \tag{7}$$

$$\underset{P,Q}{\operatorname{argmin}} \sum_{(i,j)\in R} \left(R_{ij} - \mu - P_i Q_j^T\right)^2 \tag{8}$$

We also add a regularization to avoid overfitting to our training set.

$$\underset{P,Q}{\operatorname{argmin}} \sum_{(i,j)\in R} \left(R_{ij} - \mu - P_i Q_j^T\right)^2 + \lambda(\|P_i\|^2 + \|Q_j\|^2) \tag{9}$$

In order to find optimum matrices for P and Q, we implement the alternating least squares method as described by Cichocki and Zhou[**?**],[**?**].

After obtaining the predicted rating by matrices P and Q via SVD, add the global mean back to each predicted rating. We take k as 20 and λ as 0.01.

5 Experiments and Results

5.1 Evaluation Metrics

Our goal is to predict the rating a user would give to a restaurant. We predict the rating that user has not rated in the training dataset, but the true rating is stored in the test dataset. We use the root-mean-square error and mean-absolute error for evaluation.

$$RMSE = \sqrt{\frac{\sum \left(r'_{u,i} - r_{u,i}\right)^2}{N}} \tag{10}$$

$$MAE = \sqrt{\frac{\sum \left|r'_{u,i} - r_{u,i}\right|}{N}} \tag{11}$$

Here $r_{u,i}$ is the predicted rating from user u on item i and $r'_{u,i}$ is the true rating; N is the size of test dataset.

5.2 Implementation

Our work was implemented by Python. We have seven python files in total. They are main, loadData, Baseline, User-based CF, Item-based CF, SVD and Evaluation. The main program is the controller of the whole recommendation system. At first, it invoked the loadData file for data parsing. Then it invoked Baseline, User-based CF, Item-based CF and SVD predicting ratings using these four algorithms. Each time the predicted rating and true rating were stored together and the Evaluation program will calculate the RMSE and MAE errors for the predictions.

5.2.1 Data Structure

We used Pandas for data processing which is an open source, data analysis library. The original json files were loaded into three dataframe storing information of users, business and reviews respectively. We joined the business and reviews dataframes based on business id and then joined with the users dataframe on user id. After parsing the raw data, we used the key Category and City to extract records of Restaurants and Phoenix. As to rebost sparsity problem, we computed the count of reviews grouped by restaurant id and filtered the restaurants getting rid of the restaurants which had fewer than 50 reviews.

After re-organizing the data, we used dictionary data structure to build the user-item matrix. For each records in the joined dataframe, we extract user as the key, the value of each key is a nested dictionary with business id as key and review rating as value, e.g. $\{user\_id : \{restaurant\_id : rating\}\}$. Meanwhile, we also built another item-user matrix which can be indexed by restaurant id, e.g. $\{restaurant\_id : \{user\_id : rating\}\}$.

For dividing data into training and testing data, we only consider users with more than 30 reviews. That is for each key in the user-indexed dictionary, if the

length of the value set is greater than 30, we took 20% of the entry out of the set and stored them into the test dictionary. The remaining 80% became the training data.

After this approach, we have 281 restaurants, 13820 users in training set and 91 users in test set.

5.3 Baseline

The joined dataframe was used to calculate overall average ratings μ that all users reviewed on all restaurants, average ratings b_u of every users grouped by user id and average ratings b_r of every restaurants grouped by restaurant id. These values were applied in equation(1) for the baseline prediction.

The performance of our baseline predictor is $RMSE = 0.9014$, $MAE = 0.8397$.

5.4 User-based CF and Item-based CF

5.5 SVD

5.6 Comparison

Table 1 shows the RMSE and MAE of each applied algorithms.

Table 1. Performance of four algorithms for recommendation system

Method	RMSE	MAE
Baseline	0.9014	0.8397
User-based	0.9754	0.8745
Item-based	1.1106	0.9217
SVD	0.9175	0.8403

6 Discussion and Future Work

From table 1 we can see that the baseline which is a mean predictor yields the best result. It is not surprising that collaborative filtering does not work very well with the yelp dataset. As we cited in the data section, the number of users who have reviewed enough restaurants for calculating similarity scores with other users is quite low. Collaborative filtering usually works not well when data are sparse, and has cold start problem for new users and items.

When applying SVD method to the yelp data, we did not minus the global average rating value first, the RMSE result is 1.0568, which indicates the method works not that well comparing to other methods. To improve SVD, we subtracted

the global average rating value first to generate the metrics P and Q, after generating the predicted matrix, adding the global average rating value to each element. The RMSE result improves to 0.9175. Analyzing this phenomenon, it may note that in the origin matrix of rating between users and restaurants contains too many latent value. Thus, if we do not use global average value to control the latent value, it will badly influence the SVD performance measured by RMSE.

In the future work, we would like to analyze the review text which is provided in the review json file and use this information to create comprehensive user and restaurant profiles. Also, we would explore further hybrid approaches and evaluate their performances.