Restaurant Recommender System for Yelp.com

Panagiotis Karagiannis PKARAGIA@UCSC.EDU

Juraj Juraska JJURASKA@UCSC.EDU

Noujan Pashanasangi NPASHANA@UCSC.EDU

Konstantinos Zampetakis KZAMPETA@UCSC.EDU

1. Describing focus

In this project we will explore some of the capabilities of machine learning to extract useful information from the Yelp dataset. More precisely, we will deal with the prediction of whether a user likes or dislikes a previously unvisited business. The prediction will be performed based on the ratings of other similar customers who have visited the given business. The similarity of customers will be determined by how their ratings of other businesses correspond to each other. We intend to focus on a subset of the businesses, namely the restaurants. Our results can then be used for recommending restaurants to the user.

There are two principle ways on how to approach the recommendation process. One utilizes the feature similarity of the individual restaurants, while the other is based on the similarity of users, which assumes that similar users rate restaurants similarly. Hence, the primary data that we are going to use are the ratings of the restaurants (i.e. the stars given by the users to the restaurants), included in the review JSONs. Other features of the reviews (e.g. the date), and of the restaurants (e.g. the attributes), may later serve to enhance the recommendations.

2. Data

The restaurant ratings will be extracted from the review JSONs into a utility matrix with rows corresponding to the users and columns corresponding to the restaurants. Obviously, this will produce a sparse matrix, as each user has only rated a small fraction of all restaurants.

At the current stage, our program reads and parses the relevant JSON files and creates an internal structure representing the matrix. We extract the following features from the review JSONs: `business_id`, `user_id`, `stars`, and `date`. As mentioned above, the date property is only extracted for a potential future use, such as enhancing the prediction by reducing the importance of outdated reviews.

Besides creating the utility matrix, we filter only the restaurants, i.e. the businesses having `Restaurants` as one of their categories. We store these in a separate data structure for a future reference, when we require additional information about the restaurants. The `business_id` property serves as a key to match the corresponding restaurant, and we currently store the name of the restaurant in the structure.

In order to normalize the ratings across the user database, the average rating for each user will be calculated and subtracted from the individual ratings of that user. Although there is an average rating property provided (see `average_stars` in the user JSON), it is calculated over all the businesses in the dataset, but we only consider restaurants in our work.

3. Evaluation Criteria

The greater part of the ratings will be used as the training set for our learning algorithm, and the rest of them as the testing set. The data will be partitioned into these sets at random, however, keeping at least a certain proportion of each users ratings. After training the recommendation system we will have it predict the ratings for the user-restaurant pairs in the testing set. Subsequently we will evaluate the closeness of the rating predictions to the true values using the root mean square error (RMSE).

4. Method Specification

As we have mentioned in the previous report we are going to use collaborative filtering method to predict the rating that a user would give to a specific restaurant. In this method we need to compare different objects, users and restaurant. There are two main approaches to do so, neighborhood approach and latent factor models. A successful combination of the two mentioned methods is mentioned in [1]. We focused on the latent factor model for starting our work, following [1].

Using the latent factor models such as Singular Value Decomposition (**SVD**) we would be able to compare users and restaurant together, and would not have the problem of comparing different objects anymore. We associate each user and each restaurant with f dimensional vectors. which should be regarded as a vector of different factors inferred from users ratings. These factors, such as the cost of the restaurants for example, may or may not be explicit, and it is difficult to be exploited from the users profile and the restaurant description.

We will focus on the **SVD** method for the user-restaurant matrix. A typical model associates each user u with a factor vector $p_u \in \mathbb{R}^f$, and each restaurant with a factor vector $q_i \in \mathbb{R}^f$. The prediction is done by taking an inner product of p_u and q_i. Conventional **SVD** is undefined when knowledge about the matrix is incomplete. Only addressing the few rating that we have or inserting not fully considered numbers in the place of the unknown rating, could cause overfitting or inaccurate results. Instead, we only use the ratings given and we introduce regularizing to overcome overfitting.

Some users tend to give higher ratings than average and some restaurants tend to get higher rating than others. We take these facts into account by using baseline estimates to adjust the rating that we have in the training set (see following table for notation).

r_{ui}	:Given rating of user u for the restaurant i
$\hat{r_{ui}}$	:Predicted rating of user u for the restaurant i
μ	:Overall mean of all given ratings
b_u	:Baseline estimate for u user
b_i	:Baseline estimate for i item
b_{ui}	:Total baseline estimate for u user and i item
$\mathcal{K}$	:Set of all given ratings r_{ui}

To be more precise the prediction will have the following form:

$$\hat{r_{ui}} = b_{ui} + p_u^T q_i.$$

As we mentioned above, the loss function we use in order to learn the parameters p_u, q_i and b_{ui} will be the RMSE:

$$\sum_{(u,i)\in\mathcal{K}} (r_{ui} - b_{ui} - p_u^T q_i)^2.$$

in which we add a regularization term:

$$\sum_{(u,i)\in\mathcal{K}} (r_{ui} - b_{ui} - p_u^T q_i)^2 + \lambda(\|p_u\|^2 + \|q_i\|^2 + b_u^2 + b_i^2).$$

where λ is a regularization constant, chosen heuristically.

To minimize the previous expression over p_u, q_i, b_u and b_i we use the gradient descent method. We implemented this method from scratch, using the following updates for the parameters. Each update is done by iterating to the whole set $\mathcal{K}$:

$$\begin{aligned}
p_u &\leftarrow p_u + \gamma^*((r_{ui} - \hat{r_{ui}})q_i + \lambda^* p_u) \\
q_i &\leftarrow q_i + \gamma^*((r_{ui} - \hat{r_{ui}})p_u + \lambda^* q_i) \\
b_u &\leftarrow b_u + \gamma^{**}((r_{ui} - \hat{r_{ui}})) + \lambda^* b_u) \\
b_i &\leftarrow b_i + \gamma^{**}((r_{ui} - \hat{r_{ui}})) + \lambda^* b_i)
\end{aligned}$$

where γ^*, γ^{**}, λ^* are the steps for each update. At this point we have omitted estimation of baseline parameters.

5. Testing on a simple example

In implementing the test case, we consider a very simple scenario where we only assume three users: (u_0, u_1, u_2) and three restaurants (i_0, i_1, i_2). The rating of each user for a particular restaurant can be summarized in the following table:

T	i_0	i_1	i_2
u_0	5	4	5
u_1	1	2	2
u_2	5	4	4

The entry $r_{uj} = s$ signifies that user u_i has rated restaurant i_j with s stars.

Now we conceal some of the data and try to predict the values in order to fill in the table and test the accuracy of our algorithm.

T	i_0	i_1	i_2
u_0	5	4	5
u_1	1	2	x_1
u_2	5	x_2	4

This example is deliberately constructed in such a way as to be able to intuitively guess the missing entries in table **T**, even without knowing the true values. More precisely, we see that user u_0 andu_1 have opposite taste in restaurants while u_0 and u_1 seem to be very similar. Finally, from the data available to us it would seem that u_1 and u_2 have differences in the way they would rate the same restaurant.

Therefore, given this data one would intuitively predict that x_2 should be relatively high, since u_0 has rated the same restaurant as high. Nevertheless, x_1 would be relatively low since both u_0 and u_2 have rated the business high.

Predicting by using the above algorithm we deduce vectors p_u and q_i s.t. $p_u^T q_i$ gives the rating of user u for restaurant i.

Let r_{ui} be as above. Then after running our algorithm we get that:

$$r_{12} = x_2 = 2.2857$$
$$r_{21} = x_1 = 3.7857$$

calculating the RMSE error we get that

$$RMSE = \sqrt{(2.2857 - 2)^2 + (3.7857 - 3)^2} = 0.69$$

which shows that our prediction is accurate.

6. Accuracy of our implementation

As we said above we used RMSE to evaluate our predictions. First we separated our data into two different parts, train data and test data. To do so we have been careful about users having only a few ratings, we made sure that we do not take all of their ratings into the test data. We approximately take, uniformly at random, one tenth of our data as test data. Using a subset of all our data comprising $10,000$ ratings we got the RMSE to be 1.2. We expect the error to be reduced when the baseline estimates added.