Project Report: Recommender System

Raquel Aoki
Phd Thesis
Computing Science
raoki@sfu.ca

Biswarup Ghosh
MSc Thesis
Computing Science
bghosh@sfu.ca

Mohsen Katebi
MSc Thesis
Computing Science
mkatebi@sfu.ca

Abstract

Project report submitted to Data Mining class taught by Professor Ke Wang on Fall 2017. The objective of this project was create a Recommender System to Yelp Dataset.

1 Introduction

In the past years, an area that has received more attention in Data Mining field is Recommender Systems. The goal is very simple: make better recommendations of items considering a previous behavior of items to users. How these recommendations are made, however, is a little more complicated. There are several algorithms and approaches to construct a Recommender System, and choosing the best one in each situation is part of the challenge. Another factor that can be challenging is dealing with huge datasets. Many applications demand extra strategies to implement those models due to the large amount of structured and unstructured(text) data [Sarwar et al., 2001].

In this course project, the goal is predict the rating that a user would give to a business id on Yelp dataset. Yelp is an American corporation focusing on providing business ratings and reviews, and they have a subset available for personal, educational and academic purposes. The training set has 2,038,130 ratings from 693,209 users in 145,303 different business ids. The test set has 158,024 ratings from 75,541 users in 50,017 business ids. The first challenge of this project was dealing with the dataset size, which demands to adopt different strategies in some cases, such as using sparse matrix and not the matrix directly in the Matrix Factorization approach. Another challenge is the *cold start* of some users or business. The cold start happen when we don't have much previous information about an user or/and business to use in the model and make predictions. Follows in this text an analysis of the dataset in Section 2 and the methods applied on this project on Section 3. The Section 4 shows the experiments made and a conclusion about this project is made at Section 5.

2 Exploratory Data Analysis

Before diving into to model building we wanted to see how the data is distributed, and if there are any interesting patterns that can help us in refactoring the model. The training set has ratings between Oct/2004 and Dec/2016, while the test set's ratings are between January and July 2017. The average rating on training set is 3.715, in a range between 1 and 5. We found that there are significantly more 5 and 4 ratings compared to 1 and 2 (Table 1).

Table 1: Rating distribution on training set

Rating	**1**	**2**	**3**	**4**	**5**
Count	267494	176537	255154	509647	829298
Percent	13.12%	8.66%	12.52%	25.01%	40.69%

Also, it's worth noting that lion's share of rating came after 2009 as we can see in Figure 1. To check if there is any seasonal variance in ratings across the 12 months of the year, we made Figure 2. The data does not show any clear pattern and appear to be distributed evenly across the months.

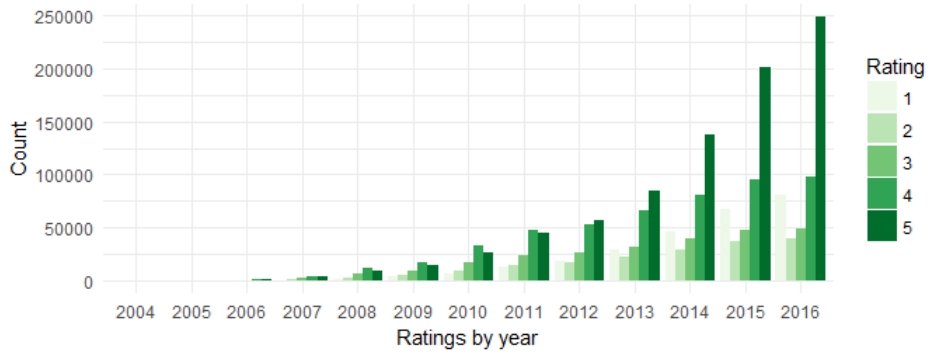

Figure 1: Rating distribution by year

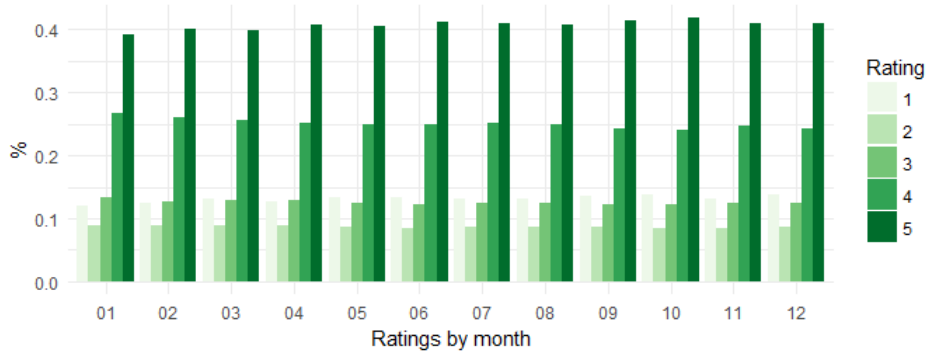

Figure 2: Rating distribution in months

3 Approaches

The choice of a good recommender depends mainly on dataset characteristics and its evaluation on training sets [Leskovec et al., 2014]. The Yelp training set has 2,038,130 $(user\_id, business\_id, rating)$ tuples. From those users, 75% rated at most 2 $business\_id$, which might turn user-user recommendation a bad choice, because most of the users will not have much in common to calculate similarity.

The training set has 145,303 business id. Although real problems are much larger than this, these number of users and business ids demand adaptations to use some techniques. Item-Item correlation, for example, demands a triangular matrix 145303×145303, what is more than which most regular laptops can handle. In this case, we need to work with a sparse representation of this matrix.

Besides those collaborative-filtering approaches, there are some content-based techniques that use business or/and user profile. For this course project, for example, we had available user comments. However, we did not use this dataset because we focused on collaborative filtering methods and matrix factorization based methods. The lack of features also makes working with decision trees or regression models difficult, once both of them need features in the training phase. We could had work with some features handmade based on business or user correlations, but constructing such features based on rating might not yield better result.

Also, we have worked to include a preference score based on the ratings' date. In this case, more recent ratings have more importance compared to the older ones. We tried different approaches. In

one of them, we calculated weights based on the time has passed from the rating divided by fifteen years. Therefore, we normalized the ratings after multiplying the weights. We tried different ways of normalizing in a way that the whole process preserves the average of all ratings. However, these approaches do not significantly decrease the RMSE.

Finally, based on our problem and dataset characteristics, we have decided to use collaborative filtering techniques and matrix factorization based methods which are explained in more details in Subsections 3.1 and 3.2.

3.1 Collaborative Filtering Methods

Collaborative filtering method is a widely used recommender system in industry as they are fairly easy to use and relatively give high accuracy when applied on dense data sets. We have implemented user-based and also item-based collaborative filtering in this project. In user-based collaborative filtering, to predict an item for a user, first we find all the users that have rated that item and compute their similarity using Pearson correlation (equation 1), and then we use this similarity to predict the rating for the user (equation 2). This is formalized as below[Leskovec et al., 2014]:

$$sim(a,b) = \frac{\sum_{p \in P}(r_{a,p} - \bar{r}_a)(r_{b,p} - \bar{r}_b)}{\sqrt{\sum_{p \in P}(r_{a,p} - \bar{r}_a)^2}\sqrt{\sum_{p \in P}(r_{b,p} - \bar{r}_b)}} \tag{1}$$

$$pred(a,p) = \bar{r}_a + \frac{\sum_{b \in N} sim(a,b) * r_{b,p} - \bar{r}_b}{\sum_{b \in N} sim(a,b)} \tag{2}$$

Item-based collaborative filtering is similar to user-based collaborative filtering and is based on item-item similarity (in our case business id).

However, the dataset is not very well suited for collaborative filtering based methods, as we found that 44798 businesses have less than two ratings, and there are 20501 business which rated by only one user. This is a classic case of cold start, where we don't have enough data for the user or the business to make a similarity decision with other users.

3.2 Latent Factor Models

The Latent Factor Models assume that users and items can be represented by latent factors [Koren et al., 2009]. Hence, the rating matrix $R_{u \times i}$ with u users and i items can be re-write as a product of two matrix: $Q^T_{d \times i} P_{d \times j}$ that represent item(business) and user latent factors. After obtaining those two matrices, each rating $\hat{r}_{u,i}$ can be calculated as a dot product $q_i^T p_u$, that captures the interaction between user u and item i.

The challenge is obtain those two matrices Q and P, as define the ideal d dimension. In this project we use Singular Value Decomposition(SVD) to extract those latent factors. We use Surprise library [Hug, 2017] and two different functions implemented:

- SVD: The prediction is set as $\hat{r}_{ij} = \mu + b_u + b_i + q_i^T p_u$; If user u is unknown, then the bias b_u and the factors p_u are assumed to be zero. The same applies for item i with b_i and q_i.
- SVDpp: Is a extension of SVD taking the implicit ratings into account and the prediction is set as $\hat{r}_{ij} = \mu + b_u + b_i + q_i^T(p_u + |I_u|^{-1/2} \sum_{j \in I_u} y_j)$, where y_j terms are a new set of item factors that capture implicit ratings.

To solve the cold start problem for new users, this algorithm considers the bias factor b_u, and p_u factor as zero. The same logic is applied to new business ids.

Besides, we tried an ensemble among several SVD and SVDpp ratings. We have run each method independently several times and used as a prediction the average value predicted. This approach gave us the best result.

4 Experiments

We had the access to user ratings just in the training set, so in order to evaluate the model adjusted, we kept 10% of the training set separate to use as evaluation before make the prediction on test set. This 10% were select randomly and not used to adjust the model. The evaluation was made using root-mean-square error (RMSE):

$$rmse = \sqrt{\frac{1}{n}\sum(\hat{r}_{ui} - r_{ui})^2} \tag{3}$$

Different approaches and their RMSE on 10% of training set is summarized in the below Table 2. Considering the approaches we tested, the ensemble between SVD and SVDpp outputs had the best results(score about 1.30 on Kaggle), in second place the SVD(score about 1.31-1.32 on Kaggle).

Table 2: Summary RMSE on 10% training set

Approach	CF User-based	CF Item-based	SVD	SVDpp	SVD+Date
RMSE	1.57	1.44	1.27	1.27	1.28

Each SVD model can be different, because depends on the seeds used inside the algorithm and also depends on which part of 10% was removed from the training set to use in the evaluation phase. By using an ensemble, we decrease this variation and dependencies, and the algorithm have more accurate predictions. In our final output, we use the average of 20 SVDs and had a score of 1.304 (mean 3.80 and standard deviation 0.579). We also tried an ensemble with 360 SVDs, but the evaluation on testing set was 1.307.

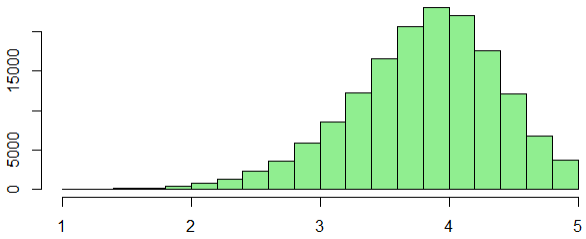

Figure 3: Rating distribution on testing set

5 Conclusion

The goal of this project was constructing a recommender system in order to predict ratings in 2017 based on ratings between 2004 and 2016. We tried to work just with recent ratings, but our results indicate that working with more data is better. After several tests and results, our experiments indicate that the best approach for this problem was use an emsemble of SVD and SVDpp. This result makes sense, once Collaborative Filtering tries to use user-user or business-business similarity to make predictions, and there are many users that just rated 2 items at most. Besides, Collaborative Filtering has more issues with cold start for users and business ids, matrix factorization can predict reasonably well where we have no prior rating by leveraging the underlying interaction between user's and business .

Individual Contributions

- Raquel Aoki(35%): Data Analysis, Latent Factor Models, Report, Ensemble
- Biswarup Ghosh(35%): Data Analysis, Collaborative Filtering, Report
- Mohsen Katebi(30%): Report, Temporal relation

References

N. Hug. Surprise, a Python library for recommender systems. `http://surpriselib.com`, 2017.

Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8), 2009.

J. Leskovec, A. Rajaraman, and J. D. Ullman. *Mining of massive datasets*. Cambridge university press, 2014.

B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM, 2001.