# Predicting Users' Ratings on Yelp based on Various Recommender System Implementations

Yile Gu
yilegu@umich.edu
Computer Science and Engineering Department

Qingyi Chen
chenqy@umich.edu
Computer Science and Engineering Department

## 1 INTRODUCTION

Recommender System has been a widely researched topic in the field of Data Mining and is particularly applicable in real life services such as commodity recommendation or media recommendation. With the held data of users' ratings on various items, what ratings would new users give to a certain item, and what item should we recommend to a new user are of our interests.

Formally, given a Rating Matrix $R \in \mathbb{R}^{|U|\times|I|}$ where $U$ is the set of users, $I$ is the set of items (in our case, businesses), and $r_{ui}$ is the rating of the user $u$ on item $i$, we would like to predict $\hat{r}_{u'i'}$ for a pair $(u', i')$ that is not present in $U \times I$.

To address the topic, various implementations and algorithms have therefore arisen, such as collaborative filtering, content-based filtering, hybrid recommender system, and etc. Collaborative filtering makes predictions based on how users who are similar to user $u'$ have rated on item $i'$, while content-based filtering makes predictions by using some specific evidence, such as the category of the item and the preference of the user, as the basis. Hybrid recommender systems make use of both to make the predictions.

In this project, we implemented and investigated various recommender systems to predict users' ratings on various businesses on Yelp. The methods we investigated include three collaborative filtering methods: a biased baseline model, a matrix factorization method based on SVD, a matrix factorization method called non-negative matrix factorization (NMF), and also a hybrid model, lightFM.

Among the data mining techniques involved in this project, SVD is the technique we have learned in the course, while the biased baseline model, NMF, and lightFM are not techniques that have been implemented in any of our course projects.

## 2 DATA

### 2.1 Data Overview

The dataset we are using is Yelp dataset [1]. The whole dataset is composed of five sub-datasets storing the data that correspond to the businesses, check-ins, reviews, tips, and the users, each in the form of a `.json` file. For our purpose, we are only using three of the five sub-datasets: the businesses, the reviews, and the users. We are particularly interested in the ratings each user gives to the businesses, and the category of a certain business. Intuitively, these are the useful information that we can use to build the recommender system.

[1]The dataset is published by Yelp and can be downloaded at https://www.yelp.com/dataset

### 2.2 Data Pre-processing

A primary data pre-processing is also applied in the attempt to have a brief understanding of the data statistics, as well as reduce the data size to a reasonable scale.

To have a straight-forward view on the dataset and how we may filter it, we plot three histograms, respectively Figure 1 for the number of businesses per state, Figure 2a for the frequency of every number of reviews that businesses have received, and Figure 2b for the frequency of every number of reviews that users have made.

Based on the statistics, we choose to narrow down the businesses scope to the businesses in Georgia, which contains over 18000 businesses. Also, we filter the businesses and the users with the criteria that only the businesses with $\geq 5$ reviews, and only the users who have made $\geq 10$ reviews on businesses in Georgia, are kept. This reduces the data size to a more reasonable range, and prevents our rating matrix $R$ from being too sparse. This leaves us with 18090 businesses and 18452 users.

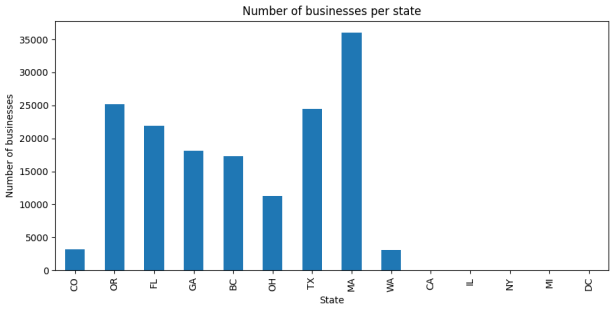
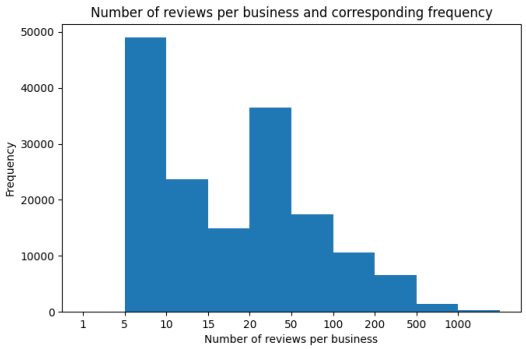



**Figure 1: the number of businesses per state**

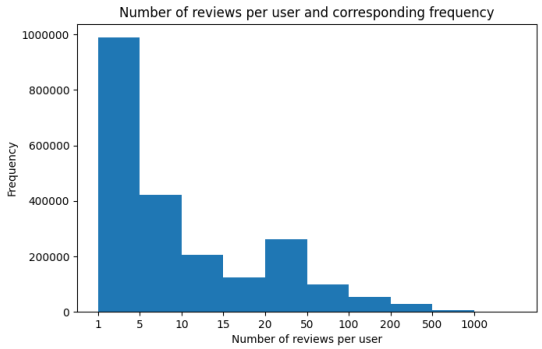
## 3 DATA ANALYSIS

### 3.1 Q: How can the recommender system predict the users' ratings on a certain business and thus make the recommendations?

*3.1.1 Data.* The data we are using are the Yelp dataset as mentioned in Section 2. Specifically, the three sub-data we are using are the `yelp_academic_dataset_business.json` data to extract which state the business is located in and the categories of the business, the `yelp_academic_dataset_review.json` data to extract the reviews and ratings made by the users to the businesses, and the `yelp_academic_dataset_user.json` data to pick out the users who have made $\geq 10$ reviews on businesses in Georgia.




**(a) the frequency of every number of reviews that businesses have received**




**(b) the frequency of every number of reviews that users have made**

#### 3.1.2 *Technique.*

(1) Biased Baseline Model[1]
The simplest model we used to predict users' ratings is the biased baseline model. The main idea is to apply a first-order approximation and compute a user's "bias": how the user rates the businesses over all reviews on average, and a business's "bias": how the businesses are rated over all reviews on average. The rationale here is that each user tends to have a "bias" in rating: some users might be a generous rater, while some others can be strict ones. Similarly, each business will have a "bias" that represents how well they have been rated overall.
Formally, we define our prediction $\hat{r}_{ui}$ as

$$\hat{r}_{ui} = b_{ui} = \mu + b_u + b_i \tag{1}$$

where

$$\begin{aligned} \mu &= \frac{1}{|R|} \sum r_{ui} \\ b_u &= \frac{1}{|U|} \sum_{u \in U} r_{ui} - \mu \\ b_i &= \frac{1}{|I|} \sum_{i \in I} r_{ui} - \mu \end{aligned} \tag{2}$$

and $b_u \leftarrow 0$ if user $u$ is unknown. The same applies to $b_i$ if business $i$ is unknown.
This model is straight-forward in concept, easy to implement, and inexpensive to compute. On the other hand, the performance of this biased baseline model is not satisfying. To explain the relatively bad performance, the model completely neglects the role of users' preferences on their ratings towards a certain business. Even if the user is generally a generous rater and the business is well-rated, it would also be possible that the user does not like a business for its specific category. The model, however, fails to account for that.

(2) SVD-based Matrix Factorization[2]
A matrix factorization method that addresses the question is the SVD algorithm we learnt in this course. Here, the main idea is to decompose the rating matrix $R$ as the product of two matrices $P \in \mathbb{R}^{|U| \times k}$ and $Q^T \in \mathbb{R}^{k \times |I|}$, where $k$ is a hyper-parameter that is chosen by ourselves, such that the reconstructed matrix

$$\hat{R} = PQ^T \tag{3}$$

minimizes the reconstruction error.
Mere decomposition actually fails to account for the bias term introduced in the biased baseline method before, so we improve the prediction by defining our prediction as

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T p_u \tag{4}$$

To acquire the values for these terms, we will train on a subset of the rating matrix $R$, $R_{train}$ and try to minimize the regularized square error

$$\sum_{r_{ui} \in R_{\text{train}}} (r_{ui} - \hat{r}_{ui})^2 + \lambda \left( b_i^2 + b_u^2 + \|q_i\|^2 + \|p_u\|^2 \right) \tag{5}$$

with the stochastic gradient descent (SGD) algorithm we learnt in the class. The update rule will then be

$$\begin{aligned} b_u &\leftarrow b_u + \gamma \left( e_{ui} - \lambda b_u \right) \\ b_i &\leftarrow b_i + \gamma \left( e_{ui} - \lambda b_i \right) \\ p_u &\leftarrow p_u + \gamma \left( e_{ui} \cdot q_i - \lambda p_u \right) \\ q_i &\leftarrow q_i + \gamma \left( e_{ui} \cdot p_u - \lambda q_i \right) \end{aligned} \tag{6}$$

where $e_{ui} = r_{ui} - \hat{r}_{ui}$, $\gamma$ is the learning rate, and $\lambda$ is the regularization term.
The SVD-based matrix factorization method, though computationally harder than the biased baseline model, performs better than the biased baseline model. The rationale is that the SVD-based matrix factorization takes into account the latent factors that influence a rating. For example, users' preference can be one of such latent factors. If we recall from the class, we decompose the rating matrix $R$ such that $R = U\Sigma V^T$ [2], where the matrix $U$ will inherently carry a mapping from user to the latent factors, the matrix $V$ will carry a mapping from the businesses to the latent factors, and the diagonal matrix $\Sigma$'s entries will represent how strong the latent factors are. Note that the hyper-parameter $k$ we chose before have a meaningful interpretation now - the number of latent factors.

[2]for the convention, we use the notation $U$ in this context as a matrix, instead of the general notion in this report that denotes the user set

(3) Non-negative Matrix Factorization (NMF)[3]
Non-negative matrix factorization is a method that is similar to SVD-based matrix factorization, except that it ensures that the user and item factors are kept non-negative. Formally, NMF decomposes the rating matrix $R$ such that

$$\hat{R} = PQ^T \tag{7}$$

where all the entries in $P$ and $Q$ are non-negative.
Similarly, we apply the bias terms, and define our prediction as

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T p_u \tag{8}$$

To ensure the non-negativity, the update rule is given by

$$\begin{aligned} p_{uf} &\leftarrow p_{uf} \cdot \frac{\sum_{i \in I_u} q_{if} \cdot r_{ui}}{\sum_{i \in I_u} q_{if} \cdot \hat{r}_{ui} + \lambda_u |I_u| p_{uf}} \\ q_{if} &\leftarrow q_{if} \cdot \frac{\sum_{u \in U_i} p_{uf} \cdot r_{ui}}{\sum_{u \in U_i} p_{uf} \cdot \hat{r}_{ui} + \lambda_i |U_i| q_{if}} \end{aligned} \tag{9}$$

The improvement of NMF compared to SVD is that it ensures all entries to be non-negative, which, in the practical meaning, makes more sense. In our experiments though, the performance of NMF is quite similar to SVD and even slightly worse. This might be because of the choice of dataset, or choice of hyperparameters.

(4) LightFM[4]
LightFM is a hybrid matrix factorization model that uses both collaborative filtering and content-based filtering. It formulates an embedding that represents the users and the items' latent factors and carries the information of users' preferences in the items. Then how much a user will like a certain item is computed as a score so that the item with a higher score is more likely to be preferred by the user.
Formally, the model will learn a set of use features $F^U$ and a set of item features $F^I$, so that each user $u$ is represented by a set of features $f_u \subset F^U$ and each item is described by a set of features $f_i \subset F^I$. The model is then parameterized in terms of d-dimensional user and item feature embeddings $e_f^U$ and $e_f^I$ for each feature $f$. The representation of user $u$ and the item $i$ is then given by

$$\begin{aligned} q_u &= \sum_{j \in f_u} e_j^U \\ p_i &= \sum_{j \in f_i} e_j^I \end{aligned} \tag{10}$$

and the prediction is defined by

$$\widehat{r}_{ui} = f\left(q_u \cdot p_i + b_u + b_i\right) \tag{11}$$

Notably, however, lightFM is interested in predicting binary data, and the $f$ is chosen to be the sigmoid function $f(x) = \frac{1}{1+e^{-x}}$. A result of this choice is that the $\hat{r}_{ui}$ predicted is not a prediction of the rating like 3.0 or 3.5, but a relative inclination of whether the user will prefer this item over other items. A recommender system based on lightFM can still make reasonable predictions on what items the user will like, but for our analysis, metrics like RMSE do not apply any more (reasons explained in Section 3.1.5). Therefore, we used AUC score to evaluate the performance of lightFM.

#### 3.1.3 *Experimental Setup.*

In terms of biased baseline model, SVD model and NMF model, we first searched for the best set of hyperparameters for each model via RMSE metric. Then we did a final round of training for each model with optimized hyperparamters and obtained its RMSE results and AUC scores.

For biased baseline model, we used grid search method to find the best hyperparameter for learning rate. For SVD model, we used grid search method to find the best set of hyperparameters for latent factors, number of training epochs, learning rate and regularization. It is good to notice that we used one learning rate and one regularization parameter for all the hyperparameter terms in the Equation 6 in order to reduce the training time. For NMF model, we also used grid search for hyperparameters regarding latent factors, number of epochs, and two regularization terms corresponding to $p_u$ and $q_i$ in the Equation 9.

For lightFM model, in addition to the user-item matrix that is commonly used in matrix factorization, we added the category of businesses as a new feature. We used grid search method to find the best hyperparameter for number of hidden components in the model.

#### 3.1.4 *Hyperparameters.*

For biased baseline model, we chose learning rate $\gamma = 0.00005$.

For SVD model, we chose 100 latent factors with 15 epochs to train the model. And we used learning rate $\gamma = 0.02$ and regularization parameter $\lambda = 0.2$.

For NMF model, we chose 10 latent factors with 30 epochs to train the model. And we used regularization parameter $\lambda = 0.08$ for both $p_u$ and $q_i$ in Equation 9.

For lightFM model, we chose 100 hidden components with 10 epochs to train the model.

#### 3.1.5 *Evaluation and Challenges.*

One of the challenges we've met during our work is the discrepancy between the available evaluation methods for lightFM model and the rest of the models.

Biased baseline model, SVD model and NMF model provide explicit prediction on user ratings of the business. Therefore, we are able to use Root Mean Squared Error (RMSE) to evaluate model performance. RMSE measures the difference between our predictions and ground-truth labels and is defined as follows:

$$\text{RMSE} = \sqrt{\frac{1}{|\hat{R}|} \sum_{\hat{r}_{ui} \in \hat{R}} \left(r_{ui} - \hat{r}_{ui}\right)^2} \tag{12}$$

where $\hat{R}$ is a list of predictions, $\hat{r}_{ui}$ is one prediction and $r_{ui}$ is the corresponding ground-truth label.

Nevertheless, lightFM model does not support RMSE metric, because instead of providing a predicted rating for each user on the item, it generates scores that demonstrate the relative ranking of recommendation for users on the items. For example, if we want to predict user $x$'s preference between item $a$ and item $b$, we will compare the scores $s_{xa}$ and $s_{xb}$ returned by the lightFM model. If $s_{xa} > s_{xb}$, we know that user $x$ may rate item $a$ higher than item $b$, and thus recommending item $a$ over item $b$, the similar result holds vice versa.

In fact, lightFM model's reasoning of binarizing user ratings lies in its fundamental assumption regarding user preferences. SVD model treats all missing ratings in the user vector as user having no specific preference for the businesses. However, lightFM model believes that users who do not provide ratings for some businesses show a negative preference for them. For example, the ratings are missing because the user may not like the service the businesses provide.

In order to provide a uniform evaluation metric for all the models, we chose AUC score for the final evaluation. AUC is defined as Area Under the ROC Curve. ROC Curve is the plot of true positive rate versus false positive rate. A popular interpretation for AUC score is that it measures the probability that a random positive example is ranked higher than a random negative example.

For lightFM model, we used the built-in evaluation method that measures AUC score. It will measure the number of correct and incorrect relative rankings returned by the method. For biased baseline, SVD and NMF models, we treat ratings above 3.5 as positive examples and implement AUC score measurement method.

*3.1.6 Observations.*

The results of the RMSE and the AUC scores are recorded in table1.

| Model \ Metric | RMSE | AUC Score |
|---|---|---|
| Biased Baseline Model | 1.2412 | 0.54 |
| SVD-based Matrix Factorization | 1.1045 | 0.67 |
| Non-negative Matrix Factorization | 1.1154 | 0.64 |
| LightFM | - | 0.88 |

**Table 1: models and corresponding performances**

As we can see, the biased baseline model has the highest RMSE and the least AUC score, which implies that it has a performance worse than all the other methods. SVD-based matrix factorization and non-negative matrix factorization have similar performances, as the methods they are using is essentially similar. LightFM, as a hybrid model, turns out to have the best performance as expected. As the category data are also used, it has more evidence to make predictions and thus performs better. Regretfully, RMSE does not apply to lightFM as discussed before.

To verify the result of the SVD matrix factorization model and the lightFM model, we also tried to interpret the model by visualizing the SVD embeddings, as well as finding the top-3 correlated categories for a few sample categories. The visualization[3] is presented as Figure 3 and the table of correlated categories is presented as Table 2.

We could observe some patterns in Figure 3. For example, 'Hertz' car rental shop is closer to similar businesses such as 'Mark Houck Repair' and 'Tips Automotive'. 'Sweet Auburn BBQ' is closer to 'Sweetwater Bar and Grill' than to 'Dr. Green's Salads'. 'LongHorn Steakhouse' is closer to 'R. Thomas Deluxe Grill' as both of them are highclass restaurants. And in general, restaurants stay in the outer circle while everyday businesses are in the inner circle.

[3]To play with an interactive version of the visualization, please visit https://datapane.com/u/ikace/reports/yelp-embedding-1

For Table 2, we found out that lightFM model is able to use added features of category information reasonably. For example, the categories that share the highest similarity in the hidden embedding vectors with 'Japanese' are 'Sushi Bars', 'Asian Fusion' and 'Ramen', where all of them are characteristics of Japanese food.

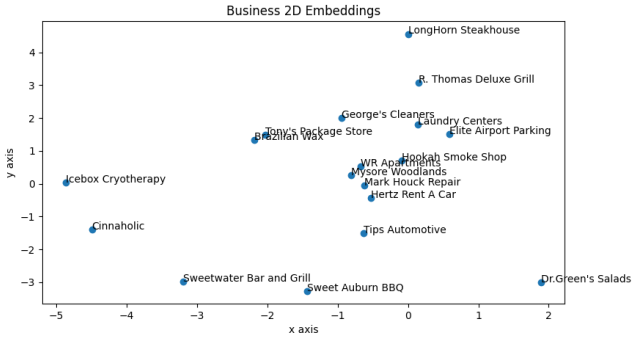



**Figure 3: the example embedding for SVD**

| Category | Related Categories | | |
|---|---|---|---|
| American | Burgers | Barbeque | British |
| Japanese | Sushi Bars | Asian Fusion | Ramen |
| Cafes | Coffee Roasteries | Coffee & Tea | Bar Crawl |

**Table 2: categories and their top-3 correlated categories for lightFM model**

## 4 CONCLUSIONS & DISCUSSION

In this project, we implemented and analyzed various recommender system methods, including the biased baseline model, SVD-based & Non-negative matrix factorization, and lightFM. We defined error metrics and compared the performance of models - the biased baseline model, as the simplest model, performs the worst; matrix factorization models like SVD and NMF perform better than the baseline; the hybrid model lightFM has the best performance. Meanwhile, we tried interpreting the result and the embedding with visualization and sampling category-related key words. The plot and the result are quite reasonable and they verify that these methods are indeed working successfully in recommender systems.

Through this project, we learnt several existing algorithms for a recommender system, how to implement them and analyze their performance. We also learnt how to interpret an embedding and make sense out of the embedding using visualization.

The part that we liked most about our project is that SVD-based matrix factorization model, because we got the chance to apply the knowledge we learnt in class into a real-world problem. We also liked the part where we tried to interpret the result. This not only verified that these algorithms did work but also indicated that the mathematical model did have practical meanings.

Each member contributed to every part of the project, including brainstorming on the project topic, programming of the models and data-preprocessing, and writing of the report.

# REFERENCES


[1] Yehuda Koren. Factor in the neighbors: Scalable and accurate collaborative filtering. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 4(1):1–24, 2010.

[2] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[3] Cédric Févotte and Jérôme Idier. Algorithms for nonnegative matrix factorization with the $\beta$-divergence. *Neural computation*, 23(9):2421–2456, 2011.

[4] Maciej Kula. Metadata embeddings for user and item cold-start recommendations. *arXiv preprint arXiv:1507.08439*, 2015.