IEOR 4571
Personalization: Theory and Application
Final Project Report

Tao Li (tl2863)
Naili Ding (nd2588)
Jinglin Chen (jc5059)
Chenchen He (ch3385)
Chaoyue Zheng (cz2529)

12/18/2019

1. Abstract

The team aims to develop a recommendation system for Yelp that can predict ratings of restaurants for different users. Bias model and collaborative filtering model (item-based) were used as baseline model to provide benchmarks. Additional models including collective matrix factorization, content-based model and deep learning models were developed to improve the recommendation system.

2. Objective

The objective of the project is to build a recommendation system for Yelp based on their dataset. The recommendation system is designed to predict the last rating for each active user. Based on the prediction of the ratings, the recommendation system can further recommend restaurants to users more accurately to improve user experience, attract new users and retain current users. In this case, Yelp can be more competitive in the market and make more profits.

The team is trying to optimize RMSE and MAE of models to get as accurate predictions of the ratings as possible. Due to the complexity of models, the team used a subset of the whole dataset to develop the models. The recommendation system is built to serve for users, which can predict ratings of restaurants for different users. In this case, to maximize user satisfaction, it is important to develop a model with the lowest RMSE and MAE. Moreover, to provide users with various recommendations rather than limited number of restaurants, the model needs to have large coverage value.

3. Data

To develop the recommendation system, the team used data from Yelp Dataset Challenge for 2019 (https://www.yelp.com/dataset/challenge). The raw dataset contains various information includes ratings, reviews, business details, and user information. The team further used a subset of the raw dataset to train models.

3.1. Sampling

The raw reviews dataset contains more than 6685900 ratings that is created by 1637138 users across 192609 business between 2004 and 2018. The team considered that the users' preference and business' behaviors might change over time and using the whole dataset seems to be imprudent. Since the raw dataset contains all the ratings for over 15 years and the recommendation system is to predict the last rating for each active user, the team decided to use only recent reviews, three-years data (from 2016 to 2018), and dropped the closed business to develop our models for more accurate prediction.

As the recommendation system is aimed for active users who are also the authentic users, the team further extracted users who have at least one real review which got more than 3 'useful', 'funny' or 'cool' votes, and also have more than 5 reviews in total. Therefore, The subset contains 1151349 reviews that came from 63208 users for 146198 business.

3.2. Data Description

Yelp dataset contains various information for business in the form of business categories, business attributes, business starts. Much of the data contained in Yelp has dependency inheritance. For example, predicting a rating for a specific user and business can be aided by analyzing user preferences from their past reviews and business attributes.

3.2.1 Business Categories

Each business in Yelp dataset is categorized into a set of 1300 categories, including 121 broad-level classes and more than 1000 detailed descriptions. In samples, broad-level classes contain Restaurant, Food, Active Life etc.; find-described classes include Italian, sushi bar, 3D printing etc. According to the figure (Figure 1) below, most business in the dataset are restaurants, and the second is shopping.

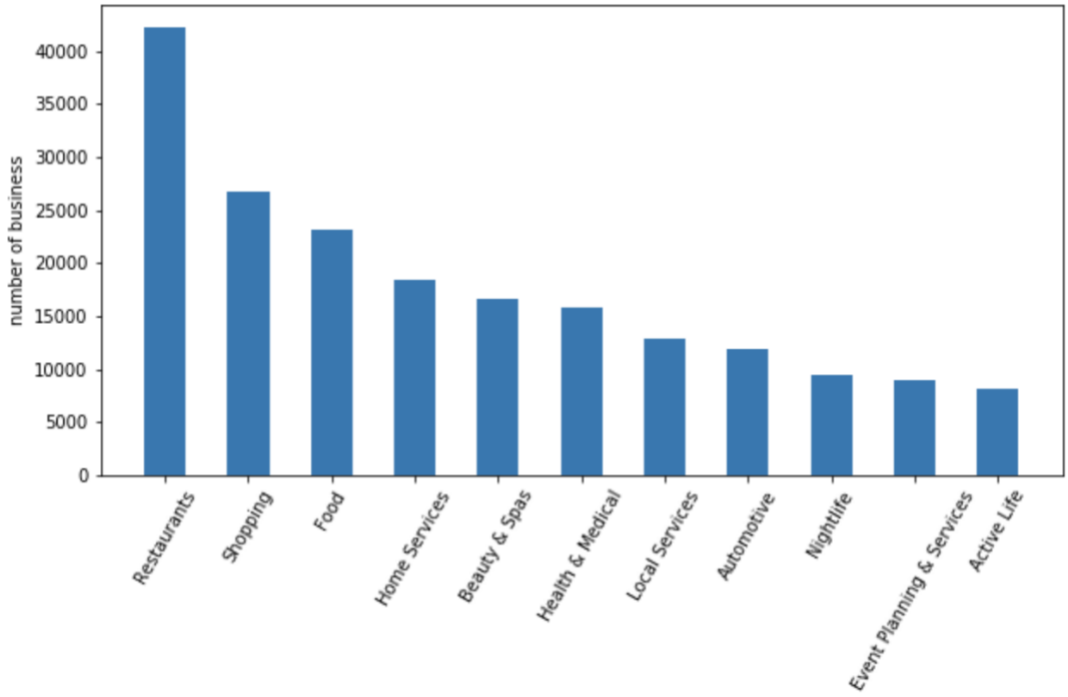

Figure 1: Categories for Restaurants

3.2.2 Business Attributes

Besides categories, Yelp dataset also describes various attributes for each business. Such attributes include noise level, delivers(or not), business parking, etc. The team believes that those attributes can be used to differentiate businesses and they were later used for collective matrix factorization as side information to improve the performance of the recommendation system.

3.2.3 Reviews and Ratings

The ratings given by users are on a 5-scale. In the subset, 43% ratings reach highest number with a rating of 5 and most of the ratings are greater than 4 (as shown in Figure 2).

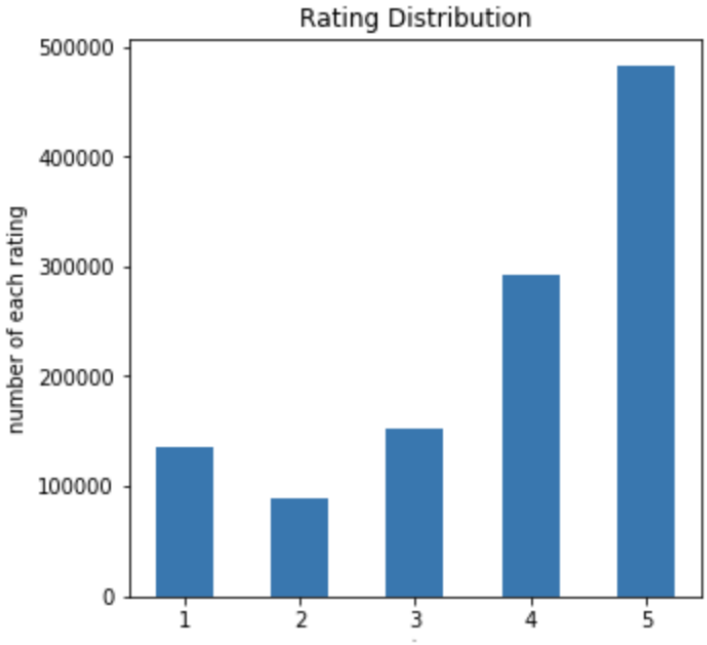

Figure 2: Rating Distribution

Following graph (Figure 3) is the word cloud of the reviews. From the graph, we can see that some of the topics such as place, food occur a lot of time, indicating the aspects that users concern the most when they rate a restaurant and provide a review.

Figure 3: Word Cloud

4. Method

The team first developed two baseline models to obtain benchmarks of the recommendation system. Then more models include collective matrix factorization, content-based model and deep learning model were developed to improve the performance of the system.

4.1. Baseline Model

Two models were used as the baseline models: Bias model and model-based collaborative filtering. The Bias model is defined as follows:

$$\hat{r_{ij}} = \mu + b_i^{user} + b_j^{item}$$

And the bias parameter for user and item are calculated through the following objective function:

$$\text{Mininize } J = \frac{1}{2}\sum_{(i,j)\in S}(r_{ij} - \hat{r_{ij}})^2 + \frac{\lambda}{2}\left(\sum_{u=1}^{m}(b_u^{user})^2 + \sum_{j=1}^{n}(b_j^{item})^2\right)$$

$$\text{where } S = \{(i,j): r_{ij} \text{ is observed}\}$$

If user i is unknown, then the bias b_u^{user} is assumed to be 0. The same applies for item j with b_j^{item}. For the bias model, the RMSE and MAE are 1.357 and 1.060 correspondingly. The team used Stochastic Gradient Descent (SGD) to find the optimal bias parameters with a default regularization term (λ) of 0.02, learning rate of 0.00001 and number of epochs of 20 by doing 3 fold cross validation over the dataset.

Model-based collaborative filtering is trying to transform the ratings into a matrix that sets users as index and restaurants as columns. By solving the matrix factorization problem, the model has the power of approximating the ratings of any restaurants for a given user.

The idea behind matrix factorization is to represent users and items in a lower dimensional latent space to generalize the machine learning problem. There are a number of algorithms to do matrix factorization, for example, SVD (Singular Value Decomposition), ALS (Alternating Least Squares) etc.

For this project, the team is using ALS to solve the matrix factorization. The idea behind this algorithm is it holds user matrix (U) constant when it tries to optimize item matrix (V) and it holds the item matrix (V) constant when it is optimizing the user matrix (U). By using this iterative coordinate-descent optimization technique, the sparsity and the scalability of the large dataset could be more easily dealt with. The RMSE is 1.367 and the MAE is 0.825 for this model.

4.2. Collective Matrix Factorization

Collective matrix factorization is a technique for collaborative filtering with additional information about the users and items, based on low-rank joint factorization of different matrices with shared factors. In its most basic form, low-rank matrix factorization tries to find an approximation of a matrix X given by two lower-rank matrices A and B, which in recommender systems would represent, respectively, a matrix of latent factors for users and items, which are determined by minimizing the squared differences between their product and X.

This basic formula can be improved by adding regularization on the A and B matrices, as well as by centering the matrix X by subtracting its global mean for each entry, adding user and item biases, and considering only the non-missing entries as follows:

$$argmin_{A,\ B,\ U_b,\ I_b} \| (X - \mu - U_b - I_b - AB^T)I_x \|^2 + \lambda(\| A \|^2 + \| B \|^2 + \| U_b \|^2 + \| I_b \|^2)$$

Where X is the ratings matrix. A and B are low-dimensional matrix. Ub is a column matrix of user biases, containing at each row a constant for its respective user. Ib is a row matrix of item biases, containing at each column a constant for its respective item. Mu is the mean of the entries in the X. Ix is an indicator matrix with entries in row {i,j} equal to one when that same entry is present in the matrix X, and equal to zero when the corresponding entry is missing in X. Lambda is a regularization parameter.

Then in collective matrix factorization, the model is further improved by also factorizing matrices of user and/or item side information using the following formula:

$$argmin_{A,\ B,\ C,\ D,\ U_b,\ I_b} \| (X - \mu - U_b - I_b - AB^T)I_x \|^2 + \| U - AC^T \|^2 + \| I - BD^T \|^2 + \lambda(\| A \|^2 + \| B \|^2 + \| C \|^2 + \| D \|^2 + \| U_b \|^2 + \| I_b \|^2)$$

Where, in addition to previous one, U is user side information matrix, I is item side information matrix, C is a matrix of latent factors for user attributes (model parameters) and D is a matrix of latent factors for item attributes (model parameters).

For Yelp dataset, the team dropped user side information and only used business side information in *user.json* dataset, containing columns like fans amount and compliment number could not reflect users own characteristics. On the contrary, business side information contain category and attribute info which show good relationships.

Firstly, the team processed business side information, which reached 158525 in total, each with 1375 tags. Since the matrix was so large that would exceed the RAM, the team only selected tags that are owned by more than 50 businesses. And the team also did multiple correspondence analysis to reduce the matrix dimension but also try to retain the most explanatory power as much as possible. Figure 4 indicates how total inertia changes for different category sizes, and team finally decided to reduce the number of tags to 360.

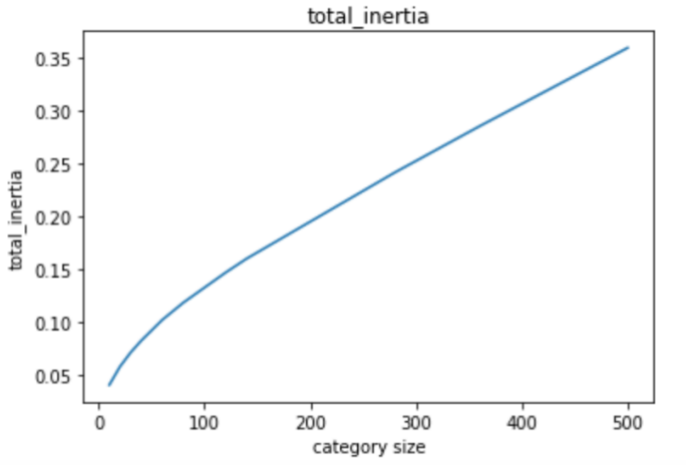

Figure 4: Total Inertia vs. Category Size

Then, the team split sample into train and test subset. Furtherly for test sample, if both user id and business id were existing in train sample, the team put it into test_warm_start category, if business id never appeared in train sample, the team put it into test_new_items category.

The team developed two models in this part. One is basic model without side information and only used warm start test sample to tune the parameters. The other one used all business side information, and evaluated the model using warm start test sample and new items test sample respectively. The results are shown in Figure 5 and Figure 6 below.

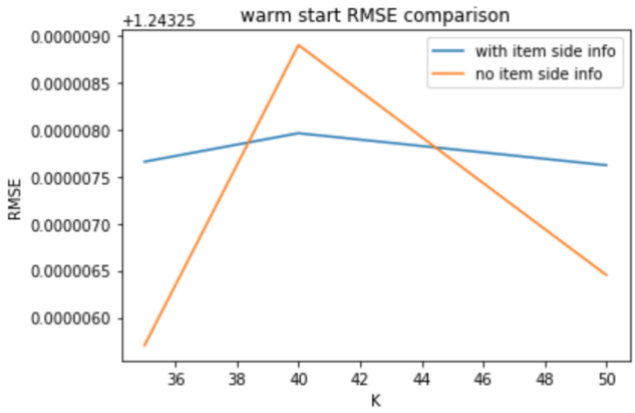

Figure 5: K vs. Warm Start RMSE

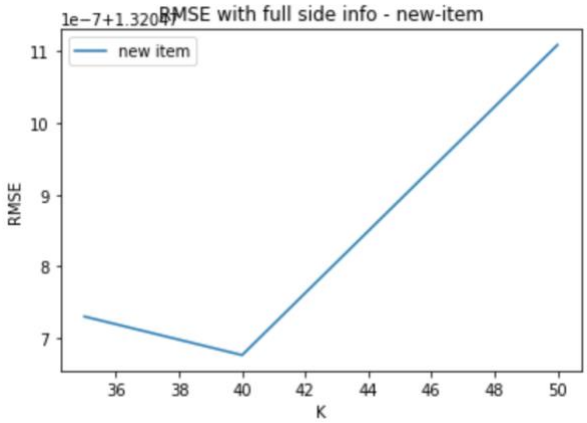

Figure 6: K vs. RMSE for Full Side Information

Based on the performance shown above, the team found that when K is equal to 40, model using full business side info with new item test sample outperforms. For warm start test sample, model using full business side info also improves a lot when K=40 compared with basic model. The team have to mention that RMSE differences among models are slight, which means side information did improve the performance, but not much.

4.3. Content-based Model

Content-based recommendations leverage information about the content of the items that are recommended. There are three main steps to content-based recommendations: preprocessing and feature extraction (content analyzer), building user profiles, and filtering and recommendation (as shown in Figure 7).

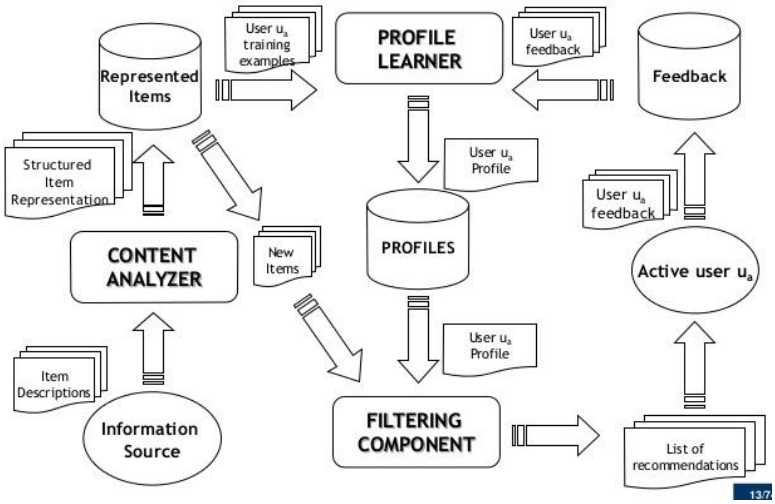

Figure 5: General Structure of Content-based Recommendation

First of all, the team used reviews of the restaurant as the content to extract features and build a vector-space representation for each restaurant. Reviews were treated as a bag of words with the operation of stop-word removal, stemming, lemmatization, and vectorization (tf-idf). To reduce the dimension of the features ,the team applied truncated singular value decomposition (truncated SVD).

Then the cosine similarity between each restaurant was calculated based on those features. According to the cosine similarity, the team was able to find the top k restaurants that a user has rated that are similar to the one to be predicted and to estimate the rating based on the user's rating for those k restaurants.

Besides reviews, the team consider including additional features (*latitude, longitude, review_count, is_open, stars*) into the content-based model as an area of opportunity to improve the performance. In order to integrate those information, the team calculated the Euclidean similarity between restaurants based on those additional features. The feature matrix was first normalized to make sure they have the same scale. Then the Euclidean distance was calculated for each restaurant pair. In order to get the Euclidean similarity, the team performed the following transform to the distance matrix:

$$\text{Euclidean Similarity} = \frac{1}{e^{Euclidean\ Distance}}$$

Based on Euclidean similarity and the cosine similarity, the team was able to calculate a mixed similarity using the following equation:

$$\text{Mixed Similarity} = (1 - \lambda) \times Euclidean\ Similarity + \lambda \times Cosine\ Similarity$$

After tuning lambda, the team found that when lambda equaled to 0.25, the mixed similarity model performed the best (as shown in Figure 8).
The team also tried to use Latent Semantic Indexing (LSI) to find the similarity among restaurants. LSI tries to analyze the relationships between a set of documents and the terms they contain by producing a set of concepts related to the documents and terms. Different from the traditional approach, out of all the preprocessed documents in the sample set, the team pre-selected the texts which the user has already rated and further process the texts into a corpus.

The team then computed the similarity score from the LSI model against the corpus and rank the documents inside the corpus which consists of the texts the user has previously rated. The team made assumptions on the number of topics for the ensemble of documents, which was associated with the set of concepts related to all the documents. Then the optimal number of topics need to be learnt, which was initially set as 100. Figure 9 shows how RMSE changes with different number of topics. Figure 10 and Figure 11 indicate the RMSE and MAE for different content-based models.

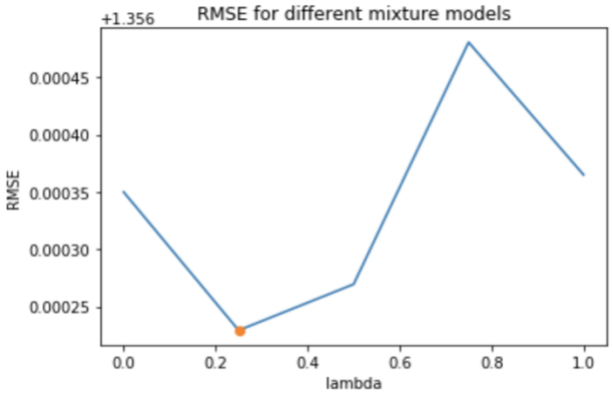

Figure 8: RMSE vs. λ

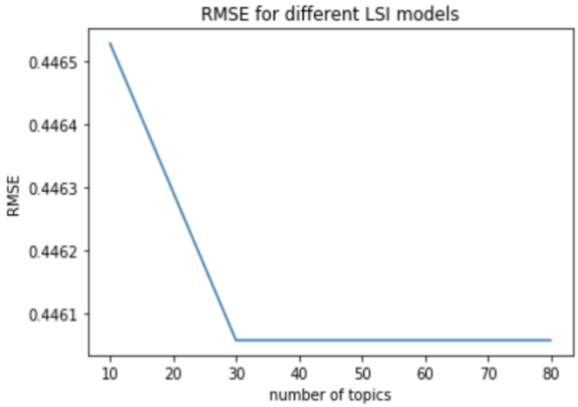

Figure 9: RMSE vs. Number of Topics

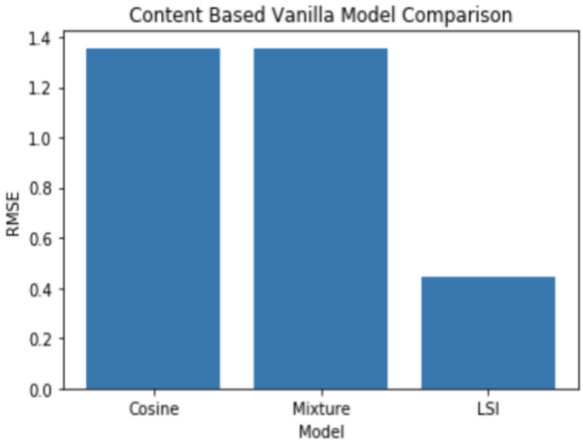

Figure 10: RMSE for Content-based Models

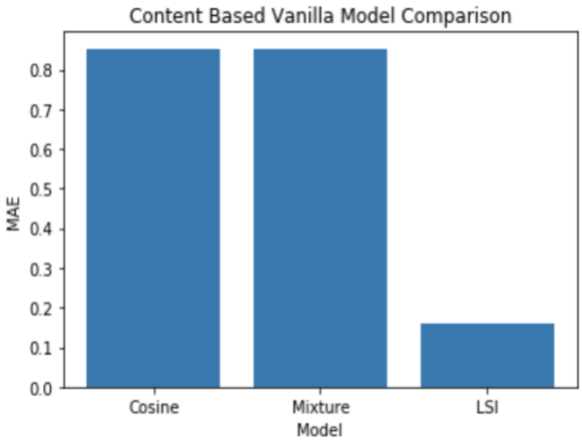

Figure 11: MAE for Content-based Models

For the LSI model as well as the mixture similarity matrix model, the team could tune the hyper-parameters to boost their performance. For example, in the mixture similarity model, the team could try to modify λ to optimize the model. On the other hand, as for the LSI model, the team could change the number of topics to further optimize the LSI model. For example, when λ = 0.2, the mixture similarity matrix would be equivalent to $0.2 \times \text{Cosine Similarity Matrix} + 0.8 \times \text{Euclidean Similarity Matrix}$. In addition, the optimal number of topics for the LSI model is 30.

4.4. Deep Learning Model

Considering both memorization and generalization in one model, the team adopt a combined model, Wide & Deep learning. The model jointly trained a linear model component and a neural network component. The main framework is designed as shown in Figure 12.

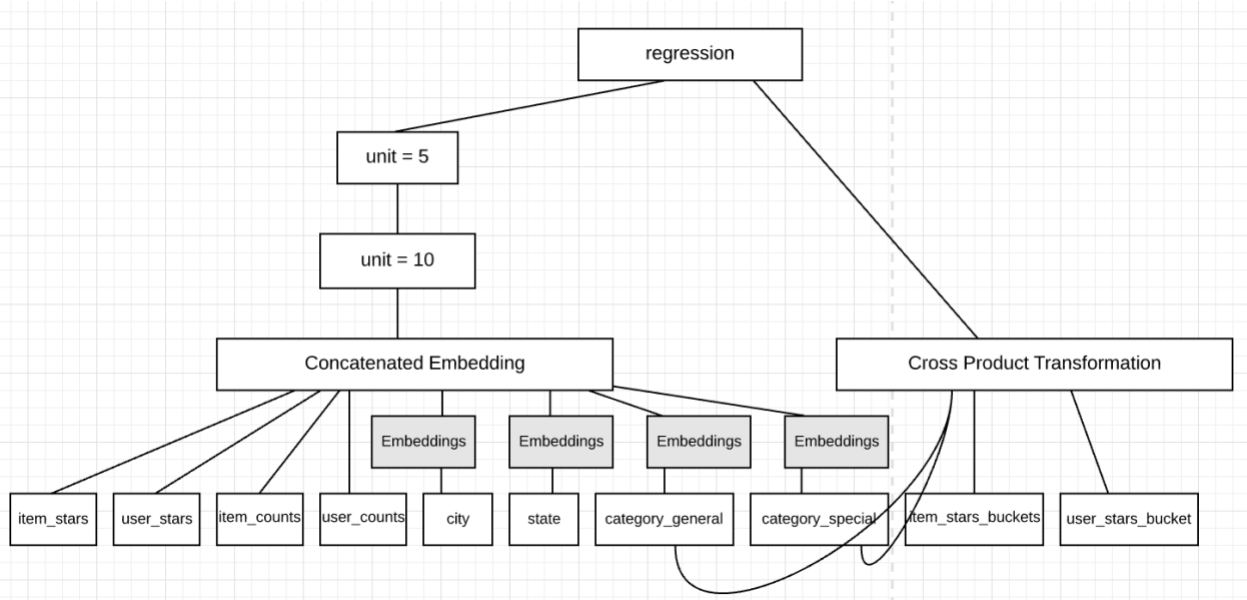

Figure 12: Framework of the Deep Learning Model

In the wide component, the team used the number of a user's reviews and the average rating to describe the user, and uses category (general & special), the number of reviews and the average rating received to describe the business. As the number of cities is quite large and it easily encounters cold start issue, the team attached 'other cities' to all cities' name that have reviews less than 50.

To classify the general category and special category for business, the team calculated the frequency of 1292 categories extracted from all business in train dataset, then chose the most frequent category each business owns as the general category and the least frequent category which occurs more than 1000 times as the special category. In this way, for example, a Chinese restaurant is likely to have a general category 'Restaurant' and a special category 'Chinese', and a barbershop is likely to have a general category 'Beauty & Spa' and a special category 'hair salon'.

Considering the business's average rating and the users' average rating are probably related, and general category is also related to special category, the team set two cross product transformed columns, [item_buckets, user_buckets] and [categories_general, categories_special].

In the deep component, apart from the raw variables in linear regression, the team also added the location (city & state) in the neutral network and deal with all categories variables by embedding.

To improve the model performance, the team tried to tune the learning rate and the number of training steps. The changes of RMSE are as shown in the following Figures. As the scale of the training dataset is quite large, it would take lots of steps to reach the optimal point with the best learning rate 0.09. Thus, taking care of overfitting problem, the team tried to train the model with as many steps as possible.

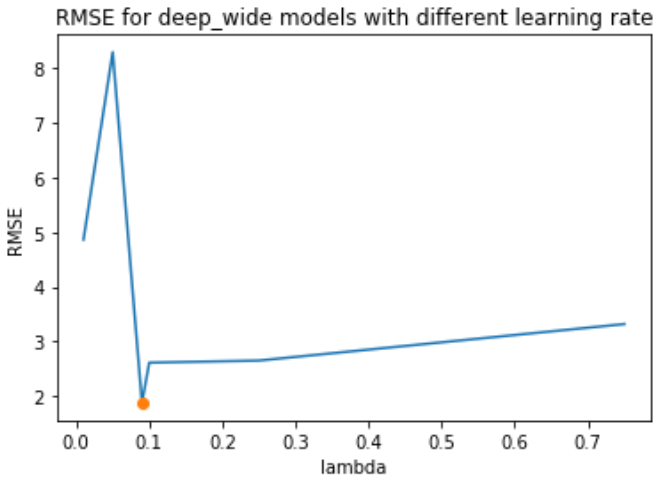

Figure 13: RMSE vs. Learning Rate

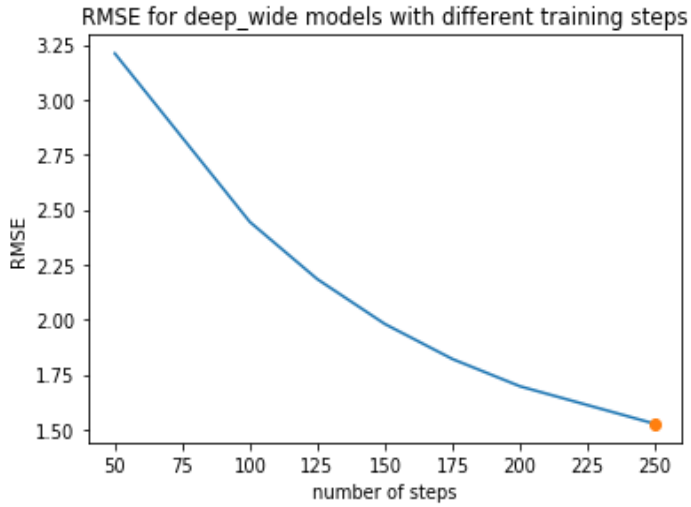

Figure 14: RMSE vs. Training Steps

4.5. Model Comparison

Model	RMSE	MAE	Coverage
Bias Model	1.357	1.060	0.28
Model-based Collaborative Filtering	1.367	0.825	0.78
Collective Matrix Factorization	1.454	1.233	0.55
Content-based Model (Cosine Similarity)	1.356	0.823	0.59
Content-based Model (Mixed Similarity)	1.356	0.852	0.59
Content-based Model (LSI)**	0.446	0.161	-
Deep Learning Model	1.362	1.186	-

The above table indicates metrics for each model. We can see that content-based model using LSI model has the lowest RMSE and MAE and thus, is the best model for the recommendation system. For other models, the RMSE and MAE values are similar to each other except for collective matrix factorization which has the largest RMSE and MAE value. Most of the models have a coverage value that is greater than 0.5, which means those models are able to recommend a variety of restaurants to users as the team wanted to.

For collective matrix factorization, these approaches face a number of disadvantages when applied to complete heterogeneous, multi- relational schema such as that of the Yelp datasets. These approaches are designed for certain types of relations and are restricted to relations of that type. Thus, it is not clear how additional sources of information (such as using partially observed business attributes to improve rating prediction) can be incorporated. Further, by training the model to predict entries of only 1-2 relations, these approaches ignore the dependencies between other relations and entities in the database, such as simultaneously predicting the cuisine of a restaurant, and the users that will like it, from the user reviews of the restaurant. Especially, in this paper, when dealing with a huge sparse matrix for business side information, the team chose MCA method to reduce matrix dimensions, the shrink dimension can only explain nearly 35% of original tags.

According to the table, deep learning model has the performance worse than baseline models. There are several possible reasons that can explain the result. First, deep learning model does not utilize the direct rating data from the pairs of users and business. The most rating-related data based on are the average stars each user gives and the average stars each business receives. The input may be not informative enough for a good prediction.

Then, the approach to classifying the category seems have to be refined further. The team just dropped the category occurring less than 1000 times, however, those detailed categories

probably also provide some information. Meanwhile, the team just split the category into two groups, general category and special category. Perhaps, subdividing the category, like classifying the country of origin for food or restaurants, would make each category features clearer and more meaningful, then improving the model's performance.

Finally, since the team did not limit the location of business when extracting the sample, the business spreads widely in terms of location distribution, which leads to the high variation of the sample. Simply adding columns 'State' and 'City' in the model probably cannot explain well.

4.6. Active vs. Inactive Users

After comparing the performance of different Content Based Models, the team was interested in investigating how each of these models perform with regard to different sample groups. In the first scenario, the users had been divided into active and inactive type. In order to categorize these two types, the team needed to find out the distribution of the number of reviews generated by individual users. After reading some statistics, the team found out 10 reviews to be a good threshold value for user review count.

After subsetting the sample into active user sample and inactive user sample, the team then ran different content based models on each sample group. Figures (Figure 15 and 16) below demonstrate how the model performs on each of these sample groups. Based on the RMSE, the team found on each group, the accuracy on the active user group is generally better than the inactive user sample group.

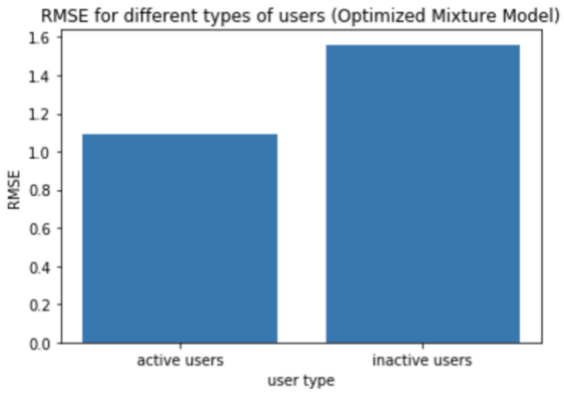

Figure 15: RMSE vs. User Type (Mixture Model)

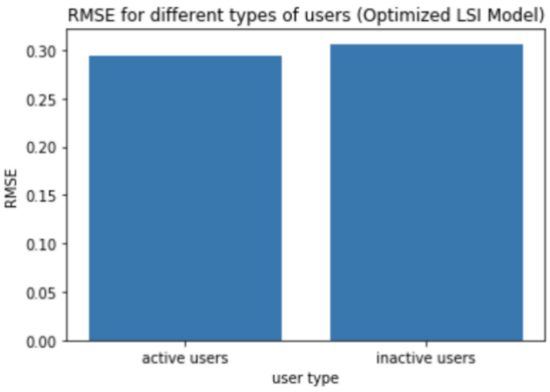

Figure 16: RMSE vs. User Type (LSI Model)

4.7. Popular vs. Unpopular Items

The team mimicked the workflow of the active user vs. inactive user model performance comparison and duplicated the process for different types of businesses, namely, popular and unpopular business establishments. After running the distribution of the review count for each individual business, the team found out 14 reviews per business establishment a fairly reasonable threshold.

The team then ran different content based models on different types of business establishments and found out that, surprisingly, the models perform better on unpopular businesses than the popular businesses with a quite significant margin (as shown in Figure 17 and Figure 18). The potential cause for this phenomenon could be that, some popular businesses might have redundant reviews, which would add more noise to the samples.

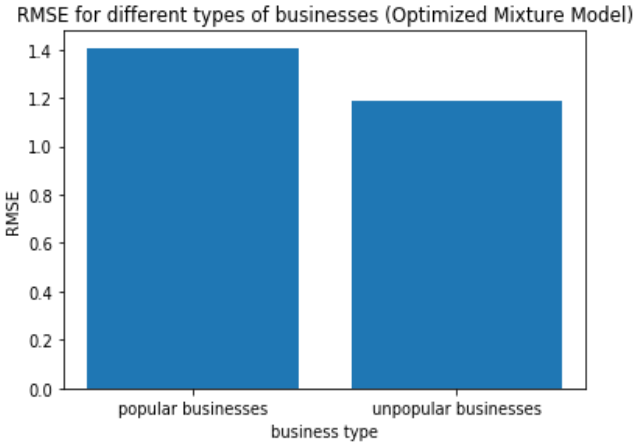

Figure 17: RMSE vs. Business Type (Mixture Model)

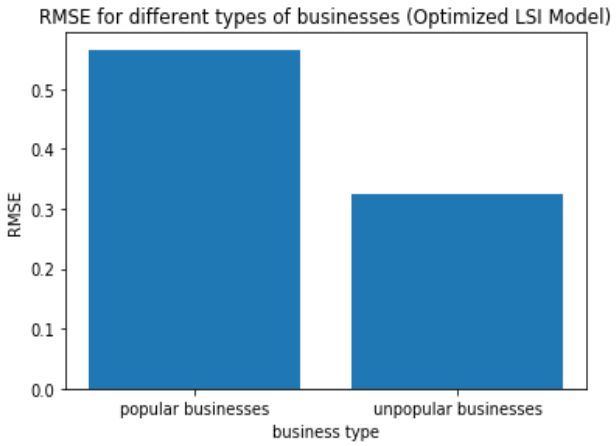

Figure 18: RMSE vs. Business Type (LSI Model)

5. Discussion

Due to computational power, there are some limitations to the models and the team made some assumptions. When tuning parameters to improve the performance of models, the team assumed that the optimized parameters in a small sample would be approximately the same to the optimal parameters for the whole sample. Thus, the team first tuned parameters on a small sample and used the optimized parameters to train models over the whole sample.

When considering the user side information, in *user.json* part, the team could only find users' friends ID, users' elite years, the amount of tags users received. Those information cannot truly reflect users' preference, which means that they are not eligible to become users' side information. Therefore, the team didn't add any users' side information into any of the model. Fortunately, users' cold start problems hasn't appear in train and test dataset. However, while encountering new users that never appears in trainset, the team had no idea or no prior information to predict ratings for new user.

Given the category and attributes chosen by the new business, the collective matrix factorization model could still work based on the business inside information matrix. However, without the reviews, content-based model cannot utilize NLP techniques to extract information from the new added business by reviews, and cannot calculate its similarity with other existing business as well. Therefore, this kind of model has the cold-start issue. As for the deep learning model, the case is quite similar. Since it needs the average stars and the number of reviews the business receives, it is hard for model to predict for the new restaurant. Meanwhile, as there are

some categorical features in the model, the deep learning model also cannot work for those restaurants with new tags. So when choosing the model for recommendation, we should consider the possibility of encountering the cold-start data. If there is a high possibility, we have to adopt Collective Matrix Factorization Model even it has worse performance.