YELP Recommendation System Based on Collaborative Filtering

Contacts	Email
Zonghao Li	zl2613@columbia.edu
Tianze Yue	ty2369@columbia.edu
Jiaxi Wu	jw3588@columbia.edu
Minmin Zhu	mz2656@columbia.edu
Yaxin Wang	yw3042@columbia.edu

1. EXECUTIVE SUMMARY

Yelp, the leading publisher of reviews of local businesses in the world, provides users with useful insights about whether the business would satisfy their needs by users' reviews. Yelp's users could engage and interact with the application through searching businesses, writing reviews, rating businesses, connecting with other users, and "checking in" at businesses. Currently, customers have to type in keywords and search in order to get information, which is passive and not intelligent. By building a successful personalized recommendation system, it would significantly improve Yelp's user experience and thus increase revenue.

The paper is going to apply machine learning to build models that will predict the rating given by a user to a business which he/she has never been to before. These ratings will help Yelp in making personalized business recommendations to every user.

A rating which takes into account the feedback of similar users is expected to help people in making the right selection and enhance their experience. With this as motivation, we have utilized Collaborative Filtering models to predict the rating by a user for a business, and have combined that with estimating similarity between users, estimating similarity between businesses and applying conjugate gradient method. Our dataset contains about 5.3 million reviews from 1,326,100 users for 174,567 businesses. After data preprocessing, we structure our final dataset for models to 32,774 reviews from 3,917 users for 420 businesses.

The paper shows that as compared with only using average ratings by a user, or the average ratings for a business, which is viewed as baseline, our method provides more accurate predictions as evidenced by low Root Mean Squared Errors (RMSE) in predicted ratings on test data. The RMSE for each model was provided in the figure below.

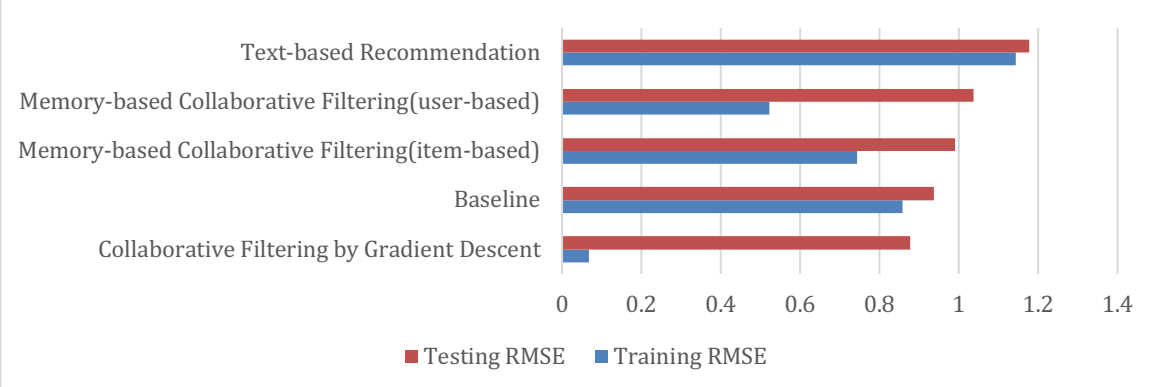

Figure 1 RMSE of recommendation system

For further study, we could take advantage of text review since it contains more information such as users' preference of taste, location, environment and price of restaurants. One approach is to combine latent rating dimensions (such as those of latent-factor recommender systems) with latent review topics (such as those learned by topic models like LDA). This approach may have several advantages. Firstly, highly interpretable textual labels for latent rating dimensions can be obtained, which helps us to justify ratings with text. Secondly, this approach can more

accurately predict product ratings by harnessing the information present in review text; this is especially true for new products and users, who may have too few ratings to model their latent factors, yet may still provide substantial information from the text of even a single review. Thirdly, the discovered topics can be used to facilitate other tasks such as automated genre discovery, and to identify useful and representative reviews.

2. BUSINESS UNDERSTANDING

2.1 Business goal/Current shortfalls

Yelp aims to help people find great local businesses such as restaurants. When users visit Yelp and search for some keywords, they gain useful insight for a list of restaurants potentially ranked by ratings, number of reviews, etc. However, the average rating that a business has, or the top reviews from certain users may not necessarily provide a user with the right perspective that he/she seeks. For example, rating criteria vary from person to person due to their own preferences such as taste, price, location, environment, etc. A high rating by one person might be due to a feature that another person does not appreciate.

Thus, making personalized recommendations to different users based on due to their preference seems a wise choice, and may greatly improve user experience through providing them with businesses that they are most likely to visit and fit their interests well.

2.2 Existing success

Many recommendation systems suggest items to users by utilizing the techniques of collaborative filtering based on historical records of items that the users have viewed, purchased, or rated. Compared with content-based approaches, collaborative filtering has many distinct advantages. It benefits from large user bases, and is flexible to different domains. It produces more serendipitous recommendations and can capture more nuance around items.

For many Internet companies like Amazon, Netflix, Google News, they widely deployed collaborative filtering in their recommend systems. For example, people would find "Suggestions for You" section on the Netflix website, and the suggestions will get refreshed when they rate more movies. The basic recommendation system is called CineMatch, which takes factors into account the films, customer's ratings, combined ratings of all Netflix users, and the algorithm has been proved fairly successful at predicting movies that subscribers would like. The recommendation system keeps updating and improving, in 2006 Netflix launched a contest called the Netflix Prize which promised $1 million to the first person or team to meet the accuracy goals for recommending movies based on users' personal preferences. This could show the great significance of a good recommendation system to a company.

2.3 Proposed success

Our model would predict a user's star rating of a business based on available ratings and reviews by all the users, as shown in Figure 1.

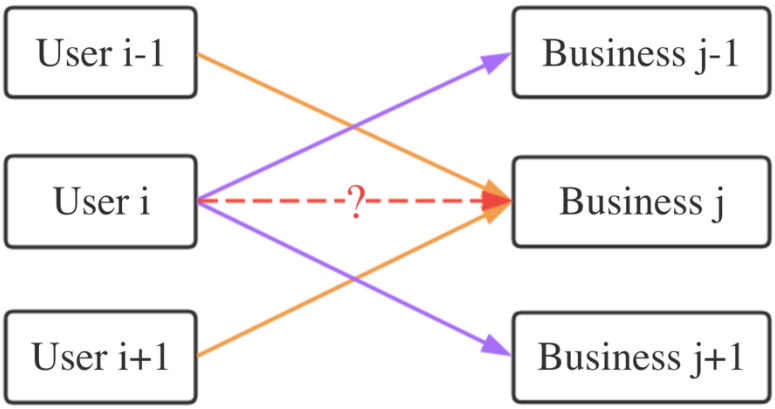

Figure 2 Rating system between user and business

In our recommendation system for Yelp, we use different methods to predict ratings from users to businesses.

The paper considers similarity between users and similarity between businesses. On one hand, we hope to find users with similar preference of businesses, then recommend the businesses they like to each other. For example, user A and his friend have same taste, so it is very likely that A would go to the restaurant his friend recommend to him and likes it. On the other hand, based on a user's rating history, we gain information of those businesses he/she gave high ratings, then recommend businesses similar to those prefered businesses.

Also, enhanced method is applied to decompose user-business matrix from features of users and features of businesses at the same time. We expect that using matrix operations would make the recommendation easier computational realized, more scalable and efficient.

3. DATA UNDERSTANDING

3.1 Data collection

We use dataset of Yelp Dataset Challenge, which contains about 5.3 million reviews from 1,326,100 users for 174,567 businesses. Within the dataset, we used three datasets for business, user and review. We can find basic information of the businesses (opening hours, address, categories, average rating, parking, alcohol, etc.), users (name, review counts, average rating, etc.), and reviews (which user, wrote to which restaurant, the content, and the corresponding rating).

3.2 Data quality assessment

This dataset includes information about businesses on 10 cities all across the world. We've picked the city of Las Vegas, which has the most businesses and users among the whole dataset. Some of the users only have reviews for several businesses, these users' data are less meaningful than users with large quantities of reviews as we view them as core users. Also, in order to make our results more consistent and speed up the computation, we decide to only use the restaurants and users with more than 500 reviews associated with them. As a result, there are 3,917 users and 420 restaurants left, with a total of 32,774 reviews.

3.3 Data subset & clustering

The structure of raw dataset as table 1

Business Dataset		User Dataset	Review Dataset
type:business	latitude	type:user	type
business_id	longitude	user_id	business_id
name	stars:rating	name	user_id
neighbourhoods	review_count	review_count	stars:rating
full_address	categories	average_stars	text
city	open;True/False	votes	date
state			votes:useful,funny,cool

Table 1 Variables in Raw dataset

We focus on user's rating and text review data to business. Our model would structure these data in a matrix with each row representing a user and each column representing a business, and value for the ith row and jth column of the matrix is user i's rating for business j. For those blank space in matrix which means no rating, we view them as 0. For those blank in matrix which means no rating, we view them as 0. The user –business matrix structures as table 2.

	Business 001	Business 002	Business 003	Business 004	Business 005	Business 006
User 001	4	0	0	5	0	4.5
User 002	4.5	3.5	4.5	0	2	4
User 003	0	2.5	0	0	1	0
User 004	0	4	0	4	0	4.5
User 005	0	0	2	0	3	0
User 006	3.5	3	0	4.5	0	0

Table 2 User-Business Matrix

For selection of training dataset and test dataset, 24.46% of the data is randomly chosen to be the testing set. Training set contains 24580 reviews/ratings and test set contains 7959 reviews/ratings. Training set are formed into the matrix mentioned above, while test set remains to be a data frame containing three columns: user ID, business ID and corresponding review.

3.4 Data preparation

In order to obtain the data matrix for modeling, we process the raw data in four steps.

Firstly, we filter the business with three conditions: its city label is "Las Vegas", its category contains several key words such as "restaurant", "food" and its review count is larger than 500. In this way, the list of Las Vegas restaurant which has no less than 500 reviews is obtained (business: city = "Las Vegas", category = "restaurant", review count>500).

Secondly, we extract the review data of these business and filter the user who gave these reviews with the requirement that the user should give more than 500 reviews in total (user: review count>500).

Thirdly, we divide the review data set into 25% test set and 75% training set in an initial attempt. However, this method would lead to a consequence that several user's data only exists in the test set. And all of our model cannot make prediction in this situation. So only users have record in training set are reserved in test set. The final proportion of test set is 24.46%.

Finally, we construct a training data matrix with the business ID list as column name and user ID list as row names. It contains 420 columns and 3730 rows each cell of which corresponding to the rating of a specific business given by a user. This matrix has a lot of missing value because a user usually only gives rating to few restaurants of our 420 business. 0 is given to the missing values, and the training matrix is turned into a sparse matrix with sparsity of 98%.

4. DATA MODELING

4.1 Critical model selection

To predict the rating a user will give to a restaurant, we build the models based on the methods below, which are baseline recommendation system, Memory-based Collaborative Filtering (User-based and Item-based), and Collaborative Filtering by Gradient Descent. All of these methods are usually employed in building recommendation systems.

4.2 Model design and implementation

4.2.1 Baseline recommendation system

For the baseline recommendation system, we first compute the average rating of each user and each restaurant. For each user, $\bar{r}_u^U$ represents the average rating he/she has given. For each restaurant, $\bar{r}_b^B$ represents the average rating the restaurants has received. Here, Super script U denotes something related to a user, and superscript B denotes something related to the restaurants. Therefore, to predicting the rating of user *u* to the restaurant *b*, the prediction is calculated by an average over the average rating of *u* and *b*, which is $\hat{r}_{ub} = (\bar{r}_u^U + \bar{r}_b^B)/2$

4.2.2 Memory-based Collaborative Filtering

For Memory-based Collaborative Filtering, we first construct a user-item matrix $X \in \mathbb{R}^{U \times B}$, where U is the total number of users and B is the total number of restaurants. Each element in the matrix is the rating user u gave to the restaurant b.

We name each row of X to be the user character vector of user, denoted by X_u^U for each user. Each column in X to be the item characteristic vector X_b^B, for the restaurant b. With these characteristic vectors, we can compute the similarities between two users or two restaurants. Here, we use S_{ij}^U denote the cosine similarity for user i and j .

$$S_{ij}^U = \text{sim}(X_i^U, X_j^U) = \frac{(X_i^U)^T X_j^U}{\|X_i^U\|_2 \|X_j^U\|_2}$$

By computing the similarities for all the user-user and restaurant-restaurant pairs, we can construct the similarity matrix for users:$S^U \in \mathbb{R}^{U \times U}$, and for restaurants:$S^B \in \mathbb{R}^{U \times U}$.In this paper, we use two different type of Memory Based Collaborative Filtering algorithms, namely User Based and Item Based, depending on which dimension of the user-item rating matrix is used to find similarities. Each one of these two strategies has pros and cons. User Based algorithms will search for "like minded individuals" following the assumption that similar users like similar items. Item Based algorithms will search for "item rated similarly by various user" following the assumption that if many users rated two items similarly, they will likely be similar items and worthy of recommendation; again, similar items are generally likely by similar users.

1) User-based prediction

The Rating prediction can take into consideration the rating normalizations. When using the mean centering approach, equations become:

$$\hat{r}_{ub} = \bar{r}_u + \frac{\sum_i S_{ui}^U (x_{ib} - \bar{r}_i^U)}{\sum_i |S_{ui}^U|}$$

To predict the rating the user will give to the restaurant, we can first compute the weighted-average of the ratings from all the users to the restaurants b, with the similarities between user u and other users as the weights, here we're subtracting the mean ratings of each user to eliminate the bias that some user tend to always give higher rating.

2) Item-based prediction

In order to produce an estimation of a rating for an unrated item, a weighted average of all available ratings is done, using as weights the correlation values computed with the similarity functions. The equation to compute a rating estimate in an item-based system is given as follows:

$$\hat{r}_{ub} = \frac{\sum_i S_{bi}^B X_{ui}}{\sum_i |S_{bi}^B|}$$

This method is also termed Item-Item recommendation. The aim of Item Based approach is to fix what is believed to be an important drawback of the User Based algorithm in real practice: it does not scale well.

4.2.3 Model based Collaborative Filtering by Conjugate Gradient

Consider the model-based Collaborative Filtering based on matrix factorization, which is the process of transforming the user-item matrix X into the multiplication of two lower-rank matrices P and Q, which are latent feature matrix for users and latent feature matrix for businesses respectively. In this way, we can factorize X by conjugate gradient method. Then the objective becomes transforming X into the product of $P \in \mathbb{R}^{U\times L}$ and $Q \in \mathbb{R}^{L\times B}$ where L is the number of latent features we want to extract. The loss function therefore could be defined as:

$$L = \sum_u \sum_b \mathrm{I}(X_{ub} \neq 0)(X_{ub} - P_u^{\mathrm{T}} Q_b)^2 + \lambda(\|P\|_2^2 + \|Q\|_2^2)$$

Here the indicator function $\mathrm{I}(X_{ub} \neq 0)$ means that the loss is only taken for the known values in the matrix X. If the product P and Q can approximate the known values in X well, then the other values computed by PQ can be used as estimations for the missing values in X. Meanwhile, the gradient for the loss function can be obtained as: $\frac{\partial L}{\partial P} = 2((PQ - X)Q^T + \lambda P)$ and $\frac{\partial L}{\partial Q} = 2((PQ - X)^T P + \lambda Q)$.

Before performing the algorithm, we first normalize the data by subtracting the average over different businesses. Secondly define the loss function and its derivatives on training set. Then randomly initialize P, Q and set the tuning parameters. In the following, we can use minimize function to solve for P and Q by conjugate gradient method. Therefore we can get the prediction of entire rating matrix $\hat{X}$ by $PQ + \bar{X}$. Note that to make the predictions reasonable, we clip the resulting values to [1, 5]. Finally, the RMSE can be calculated by extracting the predictions for test set from X^ and compare it with the test set ratings stored before. Note that we also tune empirically on the training sample by cross-validation for the regularization parameter λ, number of latent features L and maximal iterations n.

For individual prediction algorithms, we choose complexity parameters using cross-validation within the training sample:

• We divide the training data randomly into five folds.
• We loop through each fold; for a given fold, we fit the algorithm for every tuning parameter value on all other folds and form predictions on the given fold

• We obtain one prediction per tuning parameter value for every unit in the training sample; we average the prediction loss over the full training sample for each tuning parameter value.
• Based on these loss estimates, we choose the tuning parameter (typically by picking the one that minimizes estimated out-of-sample loss).
• We then fit the algorithm with the chosen tuning parameter on the full training sample.
• Eventually, we evaluate the function fitted on the full training sample on the hold-out sample; at that point, the tuning parameter is fixed.

Note that in the model-based collaborative filtering by matrix factorization, there are multiple tuning parameters, such as the regularization parameter λ and number of latent features L. So we choose these parameter s jointly; that is, we run the above procedure over a grid of multi-dimensional parameter values. In particular, these parameters will usually interact with each other, and a less complex choice in one dimension may allow for a more complex choice in another.

4.2.4 Text-based Recommendation

Besides only using the numerical information from ratings, we also try to use the text reviews to make recommendation. From all the reviews written by a specific user, we could get the important information about his preference instead of just a rating. We could also get how often these features were mentioned from the reviews for a specific business. By comparing the frequency of these words, we could match up customers and businesses by similarities.

For the model, we first concatenate all of the reviews written by a specific user and all the reviews for a specific business. Then we do feature extraction by converting the two documents into a matrix of TF-IDF (term frequency–inverse document frequency) features. We calculate the cosine similarity which indicates how close the business and the customer is and we could use this to make recommendation.

It is obvious that the ratings for each review is highly correlated to the similarity we just get. So for each customer, we fit a simple linear regression of similarity on all available ratings and fill all missing ratings by making prediction based on cosine similarity.

4.3 Model implementation

For the rating prediction described in our paper, we want to estimate the performance of recommendation system algorithms, including collaborative filtering and text-based model. In generating this output, we have followed the following workflow:

a. Select target business and user. Divide the data into a training set and a test set. Transform training data set from sparse storage organization to full storage organization.

b. Run each of the prediction algorithms on the training data:

(1) For baseline and memory-based Collaborative Filtering methods, we simply calculate the average ratings and similarity matrix. Then we construct full prediction matrix by formula mentioned in previous part.

(2) For model-based Collaborative Filtering based on matrix factorization that involves a complexity choice (such as regularization), we tune empirically on the training sample by cross-

validation; once we have chosen the tuning parameters, we rerun the algorithm with this choice of parameters on the full training sample.

(3) For text-based recommendation, we concatenate all reviews by each customers and all reviews for each business. Then we build a matrix of term frequency–inverse document frequency and calculated the cosine similarity which would be used in the model.

c. After we have stored a prediction matrix, we obtain each predicted rating on the test set and produce the statistics we are interested in. For example, we report in Table 1 the training and test root mean squared error (RMSE).

4.4 Model output internal strength

Generally, the output of a recommendation system should be a list products, in this case, the restaurants, however, it should be hard to validate the performance if the output is simply a list of items. Thus, in the paper, we try to build models that takes a restaurant and a user as input, and to output the prediction of the rating that the user might give to the restaurant. For the sake of recommendation, we can just recommend the restaurants with the highest ratings.

Generally, the output of a recommendation system should be a list products, in this case, the restaurants, however, it should be hard to validate the performance if the output is simply a list of items. Thus, in the paper, we try to build models that takes a restaurant and a user as input, and to output the prediction of the rating that the user might give to the restaurant. For the sake of recommendation, we can just recommend the restaurants with the highest ratings.
To evaluate the accuracy of the predicted ratings, we choose to use one of the most popular metrics for the recommendation system, the root mean squared error(RMSE):

$$RMSE = \sqrt{\frac{1}{N}\sum(r_{ub} - \hat{r}_{ub})^2}$$

Where r_{ub} is the predicted rating of user u to the restaurant b, and N is the total number of rating in the testing set.

4.4.1 Baseline recommendation system

By comparing the prediction of test set to the real value, we can compute the RMSE. The RMSE of baseline is 0.9374, which is only higher than collaborative filtering by conjugate gradient method. The reason is that people usually follow same rating principle, for example, for a customer who tend to give high ratings on average, he people may will give 5 stars to the restaurants they like and give 4 stars to restaurants they do not like. So the baseline model would give a good estimate but making recommendation based on this would lack personalized preference.

4.4.2 Memory-based Collaborative Filtering

The memory-based collaborative filtering can be divided into two methods, User-based and Item-based.

1)User-based:

The scalability drawback of a User Based approach is not always the central issue at the time of choosing one algorithm over the other one. In fact, the decision has much more to do with the nature of the domain. For example, in the context of news, the item dimension changes much faster than the user base, and so the Recommender System should favor a User Based approach.

2)Item-based:

Item Based algorithm can be helpful in offering an explanation as to why an item was recommended. Arguing that similar users have purchased an item is less compelling than arguing that a given item is recommended because it is similar to other items purchased in the past.

4.4.3 Collaborative Filtering by Conjugate gradient

The result turns out to be the best since this method has the lowest RMSE on test set. This matrix factorization approach is one of the latent factor models, which is similar to SVD and PCA. They compress user-item matrix into a lower-dimensional representation in terms of latent factors. One advantage of using this approach is that instead of having a high dimensional matrix containing abundant number of missing values we will be dealing with a much smaller matrix in lower-dimensional space. A reduced presentation could be utilized for either user-based or item-based neighborhood algorithms that are presented in the previous section. There are several advantages with this paradigm. It handles the sparsity of the original matrix better than memory based ones. Also comparing similarity on the resulting matrix is much more scalable especially in dealing with large sparse datasets.

In addition, in the model-based collaborative filtering by matrix factorization, we also tune empirically on the training sample by cross-validation for the regularization parameter λ, number of latent features L and maximal iterations n. In this way, we can avoid overfitting and make the model robust to the data noise efficiently, especially when dealing with high-dimensional data.

4.4.4 Text-based Recommendation

The result from text-based model is not as good as other models because of the sparsity of the review data. For each customer, although we have no missing values for the similarity with each business, we only have several available ratings as dependent variable which makes the linear regression not robust. Also, since the word vector we extracted does not have attitude information, we may pair a customer with some restaurant by some feature that he dislikes a lot. So we could do some further analysis on finding better model than linear regression and also consider the attitude for each feature to calculate the prediction.

5. PERFORMANCE AND EVALUATION

5.1 Results

RMSE	Training RMSE	Testing RMSE	Accuracy rate
Collaborative Filtering by Gradient Descent	0.068	0.877	0.7348
Baseline	0.8585	0.9374	0.6435
Memory-based Collaborative Filtering(item-based)	0.7442	0.9904	0.6226
Memory-based Collaborative Filtering(user-based)	0.5227	1.0369	0.5966
Text-based Recommendation	1.1429	1.1771	0.5756

Table3 Training and Testing RMSE of models

24.46% of data is randomly selected from the dataset to be the test dataset. Using all the models we built from the training dataset, we compute the test error, which is shown in the table below. Comparing all the models we built, we can find that the model with the best performance is the collaborative filtering by conjugate gradient, which has a lowest test error 0.877. The collaborative filtering by conjugate gradient is the only collaborative method that have a better performance than baseline. Baseline also perform well in predicting the rating a user give to a restaurant since people always follow same rating rules. For example, users will usually give 5 stars to the restaurants they like and give 4 stars to restaurant they do not like. Seldom do users give 3 stars or less to a restaurant. Thus, the average rating a user give to restaurants is similar to the average rating the restaurant receives from all users. Thus, baseline is a good method to predict the rating a user will give to the restaurants.

In addition to root mean squared error (RMSE), we report the accuracy rate in predicting rating more than 4 stars. This means we want to measure how much percent 4 or 5 stars restaurant we predicted is actually more than 4 stars. Since our final goal is to recommend restaurant to users which they most likely to go. Prediction accuracy in higher rating is more important, because we would only recommend restaurants with higher than 4 predicted rating to target user. This Table extends the previous predicting results by accuracy rate in predicting rating more than 4 stars for all prediction methods. In all the models we built, the best performance still comes from collaborative filtering by conjugate gradient model, which has the highest successful prediction rate: 73.48%. Also, all the other methods have the lower accuracy rate than base-line model, which is also consistent with the result of test error.

Taking two criteria into account, we need to use conjugate gradient model to do the prediction. It has nearly 75% prediction accuracy, which means nearly 75% recommended restaurant users are interested in. It is quite an efficient recommendation system that can increase the users' consumption willing by without annoying them.

5.2 Improvement

Although the models have shown a fairly good performance, we are still seeking more advanced models which can predict in a more accurate way. For example, we are thinking about combining the existing system into a hybrid model which could take advantage of all the models we built. As for the data, if we could get the real time location information of the user then employing location information in making prediction would make the recommendation system much more intelligent.

5.3. Conclusion

According to the analysis of the models, the best model we recommend for predicting the rating and making recommendation is the collaborative filtering by conjugate gradient model. This model provides significantly lower training and testing RMSE, and it appears to be the most promising algorithm for all datasets when optimizing for performance and accuracy.

6.TECHNICAL APPENDIX

6.1 File preparation

Save all the code files in a same file.
Download raw data set from Yelp Dataset Challenge (https://www.kaggle.com/yelp-dataset/yelp-dataset). This dataset contains seven CSV files. In our project, we use three of them: "yelp_business.csv", "yelp_user.csv", "yelp_review.csv". All of them should be saved in a paralell file named "yelp-dataset" with code file.

6.2 Code Usage

a. data_preparation.py convert raw data into the format can be put into further use.
Several parameters are defined to extract target business and users.
city_name defines the city we're currently dealing with.
business_filter and customer_filter are used to define reviews requirment.
train_size defines the proportion of training set. It generates training matrix and test set and origin training set.

b. cf_method.py defines the baseline, memory-based and item-based collaborate filtering algorithms. Each algorithm is programmed into a function. Each function takes train data matrix, train data set and test data set as input. And the output contains RMSE and prediction error for rating larger than 4.

c. main.py is the main function of the first three method. When want to duplicate the result of first three method, we just need to run this file and it will automatically invoke the above two programs. And the outcome will be stored.

d. ConjugateGradient.py runs the algorithm for collaborative filtering by conjugate gradient.
First read in the rating matrix X, then split the data into training and test sets randomly.
test_values_locations record the locations for test sample, with row_ids and col_ids storing the indices.
Loss function and its derivatives are defined in the function cost.
After setting the parameters, train the model by function minimize.
Predictions are stored in the variable predictions.
Training, test and baseline RMSE are stored in train_rmse, test_rmse and base_rmse respectively.

e. textBased.py is the file for text based model. Simply run this file would run data_preparation.py first and then build the text based model and make prediction. The file would save the predicted values as ‘and’ and also save the training and test error as ‘RMSETrain’ and ‘RMSETest’.