CSCI 5622 Project Report

Recommender System Based on Yelp Dataset

Xu Han
Department of Computer Science
University of Colorado Boulder
Boulder, CO, 80309
xuha2442@colorado.edu

Juan Lin
Department of Computer Science
University of Colorado Boulder
Boulder, CO, 80309
juli4722@colorado.edu

Yan Li
Department of Computer Science
University of Colorado Boulder
Boulder, CO, 80309
yali2241*@colorado.edu*

Yichen Wang
Department of Computer Science
University of Colorado Boulder
Boulder, CO, 80309
yiwa6864@colorado.edu

Abstract

With the explosion of the web information, users are being presented with tremendous ranges of choices, at the same time, sellers are being faced with the challenge of personalizing their advertising efforts. In this case, recommendation system(RS) provides a good solution. In this project, we build a recommendation system based on Yelp dataset and use a user interface to visualize our recommendation result. Collaborative filtering(CF) algorithm is used to create baseline and machine learning methods to train RS. The best performance of our framework has achieved an accuracy of 0.896(binary classification) and 0.828(5-class classification). What's more, we also generate feature importance results, which is inspiring for future feature engineering work.

1 Introduction

In this project, we used the dataset provided by Yelp to predict users' ratings for certain businesses. The prediction of users' ratings is a multi-class classification problem. We firstly did the pre-processing step to convert data format and clean the dataset. Secondly, we did feature engineering and extracted basic features and complex features. Then, we did classification and compared the performances of multiple classifiers, including classic supervised classification algorithms as well as a blend model. We also evaluated and ranked the features we used and generated the importance of each feature.

2 Data and Data Processing

2.1 Dataset

The data is Yelp's public dataset which contains 5.2 million reviews, 174 thousand businesses, and over 1.2 million business attributes[3]. Below is the data structure looks like,

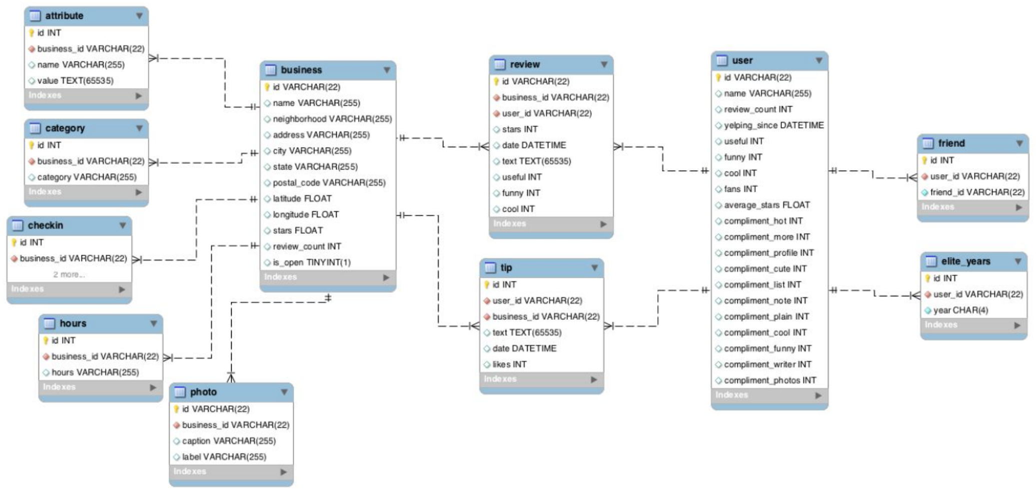

Figure 1. the data structure of Yelp dataset

Due to the large size of the whole dataset, we finally extracted 50000 users' information, with all their reviews they wrote and the businesses they reviewed. There were around 800000 reviews which contained all the information we needed.

The MySQL server was built and the SQL file was loaded through the server. The whole dataset is shared with the other team members so everyone could extract the features from the server.

2.2 Data Preprocessing

2.2.1 Data Cleaning

The data cannot be used directly. There are some empty reviews in review table. We ignored them when we extracted data.

For text data, sentences always contain meaningless words. Sometimes, the whole sentence is even meaningless. We usually tend to ignore those commonly used but meaningless words because it would influence the results when we do text data analysis. Stop words are these kinds of words. They are commonly used words such as 'the', 'a' and they don't have accurate meaning. When we extracted review features, the stop words were ignored.

2.2.2 Unbalanced data

According to our statistics, the number of reviews whose ratings are above or equal 4 (above-4 reviews) is far more than the reviews whose ratings are below 4, which is very unbalanced for the classification.

Therefore, we applied two methods here to deal with the unbalancing problem: 1) Undersampling. The basic idea is to remove some reviews whose ratings are above 4 from the training data to make the training data more balanced. Therefore, we conducted random sampling for above-4 reviews to make the ratio of above-4 reviews to below-4 reviews be around 2.5:1. Our results shows this ratio works good for most classifiers. 2) Threshold moving. The basic idea is instead of dealing with training set, we just assign a weight to the prediction result, and the weight is the original ratio of non-repeat buyers to repeat buyers. The second method will be used directly in classifiers by setting the “class_weight" to “balanced".

2.2.3 Feature Scaling

Since the range of values in raw data varies widely, some machine learning algorithms will not work properly. Some features may govern the classification. Therefore, we need to scale the features to make them contribute approximately proportionately.

3 Feature Engineering

Feature engineering is a way to get multiple features from dataset and do algorithms on them. Personalized Recommender System is based on feature engineering. The choice and design of features affects the accuracy of results significantly. In our project, we would like to find and extract different features as many as possible. Then, we use these features to train our data. We would like to choose some of these features depending on how high the accuracy is. The last but not the least, we do multiple algorithms on these features and then, we give them a rank or combination rank. Our system gets the highest accuracy with targeted features and running in the best algorithm we can find.

Yelp dataset contains 11 tables such as business, category, checkin, etc. We use four of them to extract features we want. They are business table, category table, review table and user table. At last, we extract 22 features. There are mainly 4 groups of features, user-related features(7), business-related features(3), user-category features(5) and review-related features(7).

3.1 User-related Features

A user’s basic profile can uncover some information about his/her review style. A review can receive “useful”, “funny”, “cool” remarks from others and a user can receive compliments for his/her personal information from others. We extracted those features for each user, from which we can infer is a user and the reviews he/she wrote are popular. Number of reviews a user wrote and the average rating a user gave to other business can reveal the review style, so we also extracted them.

3.2 Business-related Features

A business’s features can tell us if it is popular. We used a business’s average rating, which is a direct indication of a business’s reputation. We also extracted the average rating of business in a certain neighborhood and zip code area. These business-location features can help to understand the overall reputation of a specific area’s businesses.

3.3 User-category Features

Intuitively, a user may have his/her own flavor for a type of business. For example, one may like Italian foods but dislike mexican foods. Based on this, we extracted user’s preference on different businesses’ categories. More specifically, we extracted a user’s number of reviews, average ratings and the useful, funny, cool remarks received for all the categories visited.

.

3.4 Review-related features

3.4.1 Review profile features

For each review, the basic profile can reveal its popularity and reliability, so we extracted the number of useful, funny and cool remarks it received.

3.4.2 Polarity and Subjective

Although we have some features like businesses' average rating which can be used to analyze a business, they are hard to get the actual sentiment of users. So we evaluate the polarity and subjectivity of users. Polarity means the degree of positive and negative in reviews. Subjectivity shows the degree of objective or subjective reviews.

3.4.3 Meaningful words count

The number of meaningful words in a review would influence the recommendation results. The meaningful words means they are words and they are not stop words. We use data clean method to remove punctuation, work on word segmentation. For example, if a word is IDF. It would be separate as three words. We use word segmentation to count it as one word.

3.4.4 CountVectorizer and TF-IDF

We give each review a vector. If they are similar reviews, the distance of these vectors are small. It is one of the best way to analyze the important information or words in a sentence. TF-IDF is an improvement of it. TF-IDF considers about the importance of words and the frequency they are in different documents.

4 Methodology

A lot of research has been done on recommendation system, and most popular approaches are based on low-dimensional factor models. The RS techniques are broadly divided into two types:

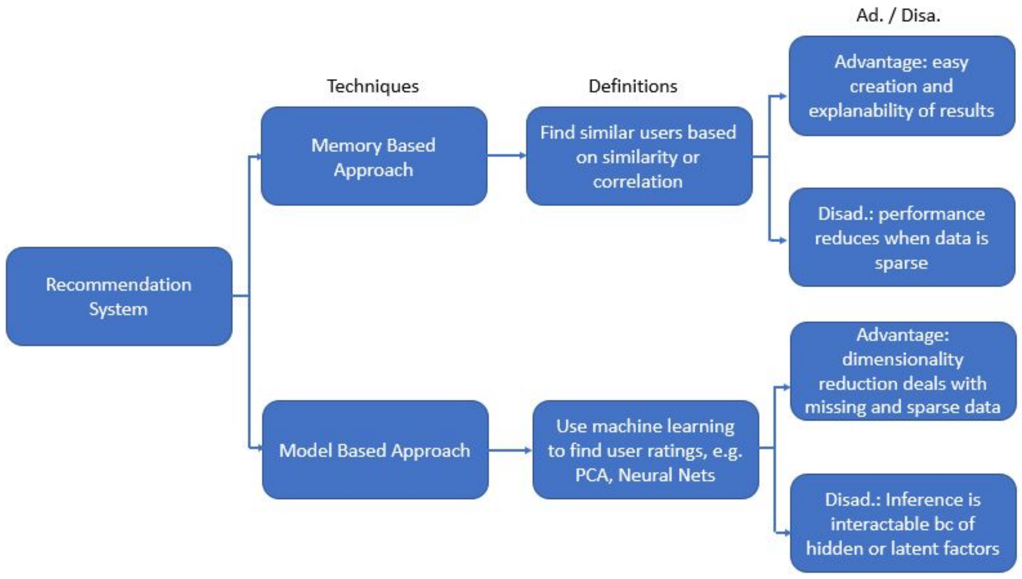
Ad. / Disa.
Techniques
Definitions
Advantage: easy creation and explanability of results
Memory Based Approach
Find similar users based on similarity or correlation
Disad.: performance reduces when data is sparse
Recommendation System
Advantage: dimensionality reduction deals with missing and sparse data
Model Based Approach
Use machine learning to find user ratings, e.g. PCA, Neural Nets
Disad.: Inference is interactable bc of hidden or latent factors

Figure 2. Recommendation system Techniques

4.1 Collaborative Filtering

Memory-based collaborative filtering approaches can be divided into two main sections: user-item filtering and item-item filtering. A user-item filtering takes a particular user, find users that are similar to that user based on similarity of ratings, and recommend items that those similar users liked. In contrast, item-item filtering will take an item, find users who liked that item, and find

other items that those users or similar users also liked. It takes items and outputs other items as recommendations.
In our case, maintaining a item-item similarity matrix is more computationally expensive than an user-user similarity matrix, so we choose to use the user-user based due to the computation simplicity.
Similarity measures include Jaccard similarity, Cosine similarity and Pearson correlation similarity. We choose to use Pearson correlation similarity because our data has explicit ratings, and it also removes effects of mean and variance[1].
We want to use collaborative filtering as a reference for comparison with ML methods applied above, at the same time, also because user-based collaborative filtering is an well-maintained technique and few Python libraries are available, we choose to use GraphLab after few rounds of experiments[2].
we built accuracy and precision/recall to examine the accuracy instead of using the built-in function from GraphLab. This is mainly because the built-in user-based model predicts non-binary rating, while our system recommends items based on a binary rating (like or dislike), so we rewrote the predicted rating to binary ratings, and then calculate accuracy, precision/recall from there. Here we set up the threshold of average_ratings is 4, the rating equal to and above 4 would be considered as like and the rating smaller than 4 would be considered as dislike.

4.2 Machine Learning Methods

For model based approach we used machine learning to predict user's rating. We conducted experiment on the following classifiers: Logistic Regression[9], Support Vector Machine[5], Random Forest[6], AdaBoost[7], Xgboost[4] and neural network[8]. We first did binary classification and compared results with collaborative filtering. We set label 1 to above-4 reviews and 0 to below-4 reviews. There are 2 parts in binary classification: 1) only user-related features were used. 2) user-related, business-related and user-category features were used. Then we used all the 22 features and did 5-class classification. For each classifier,, we did grid search to find the best parameters.

We also used a blend model which combined all of the above classifiers together and assigned different weights to each classifier's results:

$$p = \sum_{i=1}^{k} w_i * p_i.$$

where *p* is the probability that a user will give the corresponding rating for a business, *pi* is the probability predicted by the *i*th classifier, *wi* is the weight assigned to the ith classifier and *k* is the number of our models.

In addition, we used Xgboost to rank the features and generate each feature's importance. The importance is based on f score, which is a metric that simply sums up how many times each feature is split on.

5 Conclusion

5.1 Collaborative Filtering

We combined the user and business table and joined them together with the joining part of user_id. So there are 80,000 datasets in total and the average user rating is from 1 to 5. In our case, GraphLab is used to calculate the user-based similarity. A virtual dev. environment is used here to utilize GraphLab because it only works with Python 2.7.

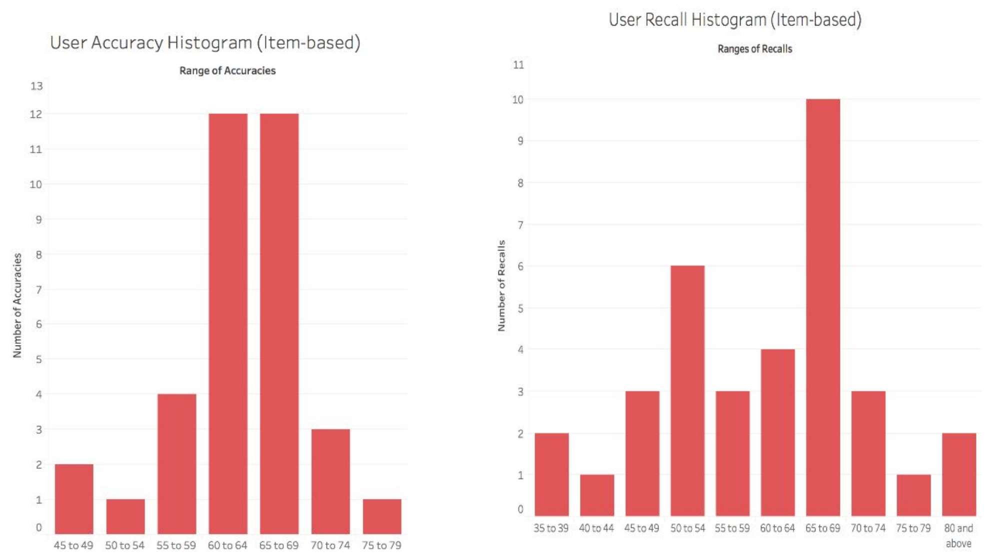

Figure 3. User Accuracy (Left) and User Recall (Right) Histogram

5.2 Machine Learning

After grid search and model training, we get the result of each classifier we deploy. For binary classification, we compare our classifiers' performances with CF algorithm. We first only use user-related feature groups, which is totally same as the features used in CF, to train our models, then all the feature groups. The result is shown in Table 1 and Fig 5. The highest AUC scores have been highlighted.

Table 1. AUC score for each classifier with different feature groups

	Logistic Regression	Support Vector Machine	Random Forest	Neural Networks	Xgboost	Adaboost	Blended Model
Only user-related feature group	AUC= 0.769	AUC= 0.747	AUC= 0.772	AUC= 0.770	AUC= 0.780	AUC= 0.769	AUC= 0.783
All feature groups	AUC= 0.952	AUC= 0.866	AUC= 0.964	AUC= 0.964	AUC= 0.966	AUC= 0.965	AUC= 0.967

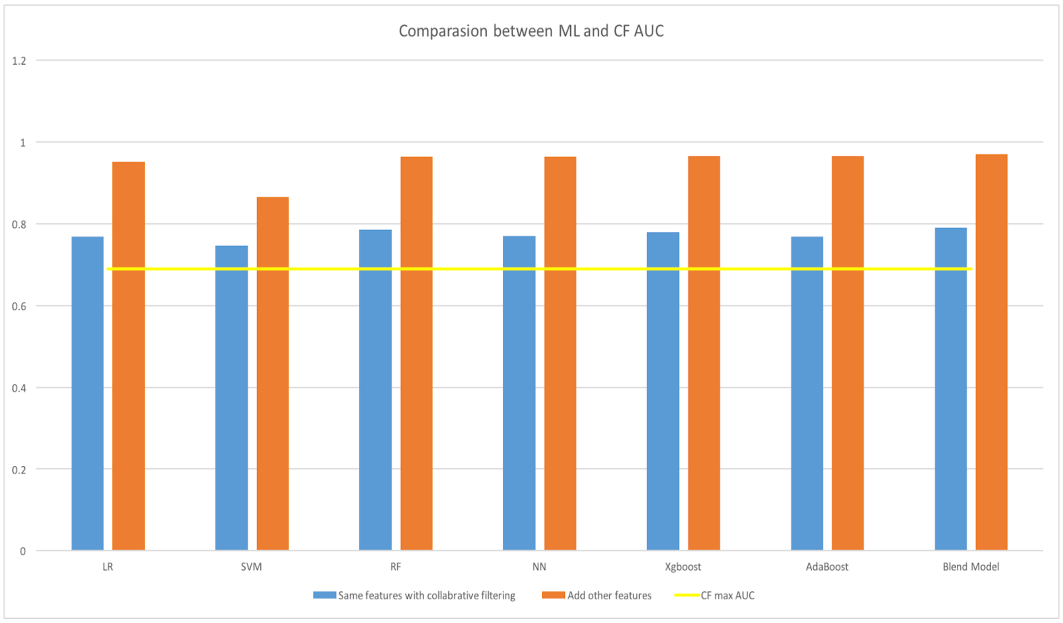

Figure 4. Comparison between CF and ML's AUC

For 5-class classification, the accuracy of each model is shown in Table 2. The highest score also comes from our self-created blended model.

Table 2. Accuracy for each classifier under 5-class classification

	Logistic Regression	Support Vector Machine	Random Forest	Neural Networks	Xgboost	Adaboost	Blend Model
Accuracy	0.654	0.725	0.793	0.797	0,817	0.805	0.828

From Table 1 and Fig 4 we can know that machine learning methods have better performance than CF method. From Table 1, Fig 4 and Table 2, we can know that our self-created blended model has the best performance over all the classifiers and Xgboost has the best performance over other non-blended models.

5.3 Feature Ranking

We measure the ranking of all 22 features based on the importance score retrieved by Xgboost[4]. Xgboost's feature importance method calculates F score, which indicates how many times the feature split on. Higher the F score is, more important the feature is. The results are shown in Fig 6.

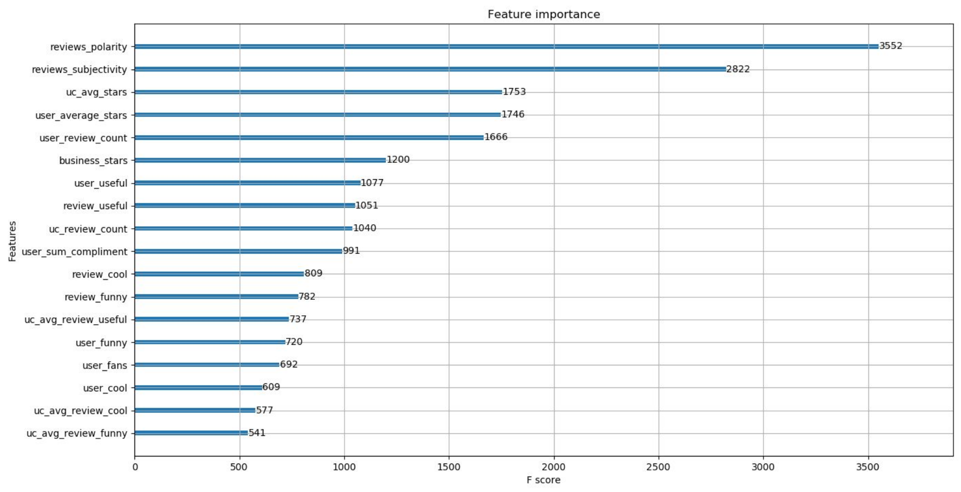

Figure 5. Feature ranking results

From Fig 5. we know that the top three features are reviews polarity, reviews subjectivity and user-category average stars.

We also generate excluded accuracy for each feature group to see the importance and we find that user-category feature group contribute most to the overall model training. The result is shown in Fig 6.

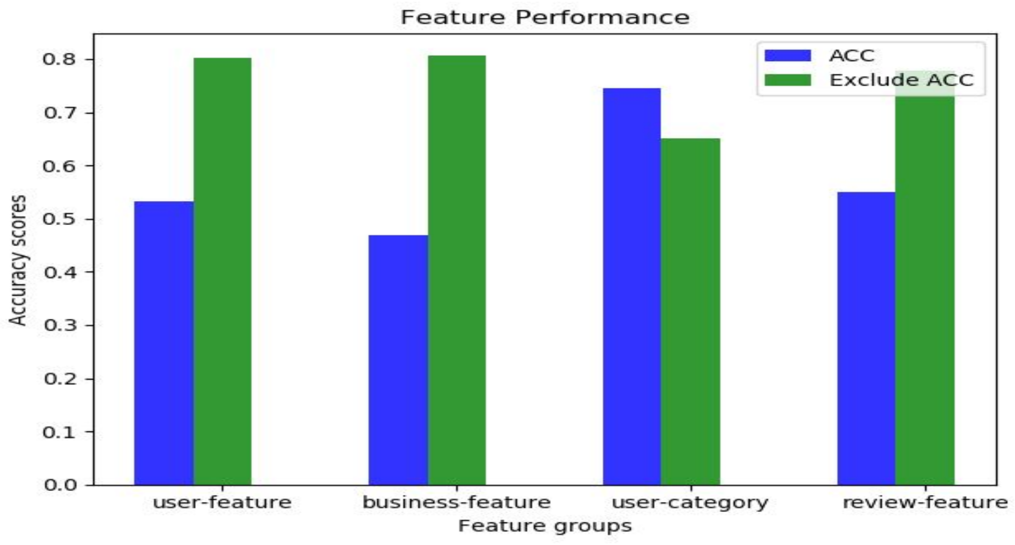

Figure 6. Different feature groups' performances and their excluded ACC

5.4 User Interface Implementation

The longitude and latitude information of the business table was used here to generate the real-time map to locate the recommended restaurants. The Plotly Python Library is used to visualize the data. In order to use and access the Plotly map function, you need an access token which could be signed up here. Once the recommended items are generated from the different modeling methods mentioned above, we could use the business_id to find out the associated latitude and longitude for each restaurant, then we could display the business on the map. The user could zoon or move the cursor to find out the detail of their recommended restaurants, for instance, the geographical location, business name and the associated ratings.

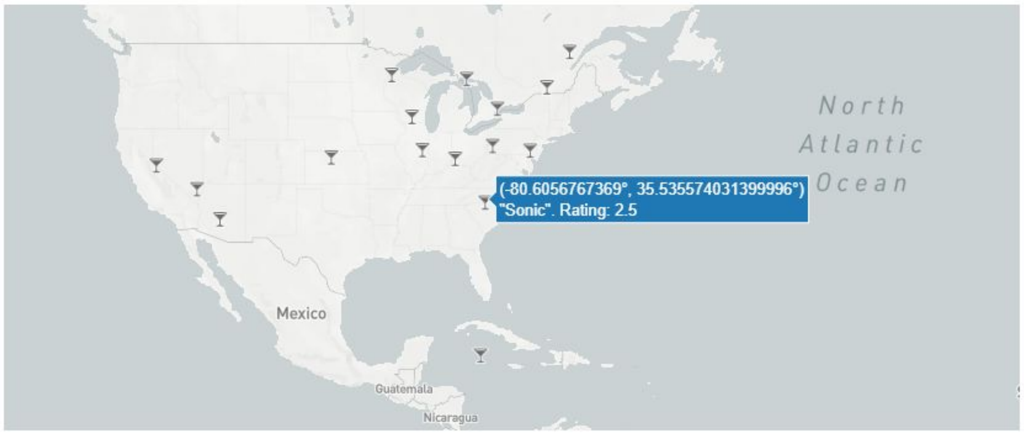

Figure 7. Recommendation System User Interface

5.5 Discussion

The key difference of memory-based approach from the model-based techniques is that we are not learning any parameter using gradient descent or other optimization algorithm. The user-based collaborative filtering approaches are generated by only using cosine similarity or Pearson correlation coefficients, which is only based on arithmetic operations. Therefore, it could explain that why ML methods generally performed better than CF method.

6 Work Distribution

Xu Han:

- Data preprocessing
- Machine learning model training (AdaBoost, Xgboost, blend model, including grid search)
- Feature ranking and analysis
- Model evaluation and analysis

Juan Lin:

- Build MySQL server and process the data for the use of collaborative filtering method
- Explore and analyze the raw data, plot the top 20 restaurants
- Research and implementation of collaborative filtering methods
- Compare and analyze the result of ML methods behave better than CF method
- Create UI for displaying the recommended items

Yan Li:

- Cleaned data with stop words library, regression expression and extracting the stem of each words
- Extracted review text with review ID
- Built sentiment analysis tool for review sentences
- Created multiple text vector metrics and use two of them as features

- Tired to build a web application with PrimeNG in Angular 5 for future work

Yichen Wang:

- User-related, user-category, review profile, business-related feature extraction and union feature
- Machine learning model training(LogReg, SVM, Random Forest, Neural Network with graid search)
- Model evaluation and analysis

7 Work Repository

Please find our project on Github repository.

Acknowledgments

Thanks for the contribution of all the team members. Also appreciation for Dr. Chris Ketelsen and TA.

References

[1] I. Portugal, P. Alencar, D. Cowan (2015) The Use of Machine Learning Algorithms in Recommender Systems : A Systematic Review. *IECS*

[2] https://turi.com/products/create/docs/graphlab.toolkits.html

[3] https://www.yelp.com/dataset/challenge

[4] Chen, Tianqi, and Carlos Guestrin. "Xgboost: A scalable tree boosting system." *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. ACM, 2016.

[5] Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: a library for support vector machines. ACM transactions on intelligent systems and technology (TIST) 2, 3 (2011), 27.

[6] Leo Breiman. 2001. Random forests. Machine learning 45, 1 (2001), 5–32.

[7] Yoav Freund and Robert E Schapire. 1995. A desicion-theoretic generalization of on-line learning and an application to boosting. In European conference on computational learning theory. Springer, 23–37.

[8] Yoshua Bengio et al. 2009. Learning deep architectures for AI. Foundations and trends® in Machine Learning 2, 1 (2009), 1–127.

[9] John Neter, Michael H Kutner, Christopher J Nachtsheim, and William Wasserman. 1996. Applied linear statistical models. Vol. 4. Irwin Chicago.