Prediction of Yelp Review Star Rating using Sentiment Analysis

Chen Li (Stanford EE) & Jin Zhang (Stanford CEE)

1 Introduction

Yelp aims to help people find great local businesses, e.g. restaurants. Automated software is currently used to recommend the most helpful and reliable reviews for the Yelp community, based on various measures of quality, reliability, and activity.

However, this is not tailored to each customer. Our goal in this project is to apply machine learning to predict a customer's star rating of a restaurant based on his/her reviews, as well as other customers' reviews/ratings, to recommend other restaurants to the customer, as shown in Figure 1.

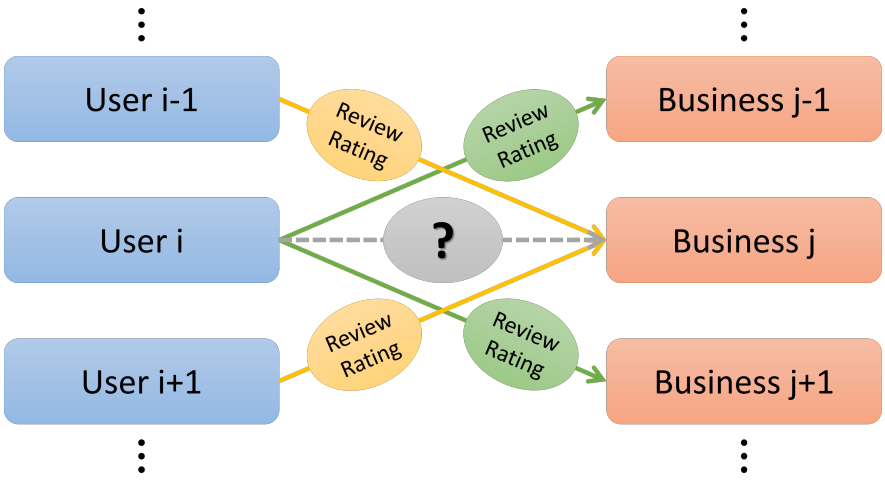

Figure 1: User & Business Connections

The project has two tiers. First, using a customer's review on a business to predict the star rating given by the customer. Second, using many customer's reviews on a business and on different to predict this customer's likely ratings of different businesses.

2 Data Source

For our experiments, we used a deep dataset of Yelp Dataset Challenge which is available online (http://www.yelp.com/dataset_challenge), which includes the data of 42,153 businesses, 252,898 users and 1,125,458 reviews. These data provide useful information such as business profile, review text, user profile, friends and votes. All the data are in json format. For reviews, the data looks like

```
{
   'type': 'review',
```

```
    'business_id': (encrypted business id),
    'user_id': (encrypted user id),
    'stars': (star rating, rounded to half-stars),
    'text': (review text),
    'date': (date, formatted like '2012-03-14'),
    'votes': {(vote type): (count)},
}
```

3 Features and Preprocessing

At first, we need to train our model based on review texts and star rankings from the same user. So, we selected a user who has written 1,137 reviews for experimental purposes. A file containing all the reviews and ratings given by this user was generated.

Then, review texts were cleaned, by removing format, punctuation and extra whitespace. All characters from the dataset are lowercase, so there is no need to preprocess uppercase letters. Word stemming was achieved using Porter stemming algorithm, which erased word suffixes to retrieve the root or stem. Stopwords, that is, words with no information value but appear too common in a language, were also removed according to a list from nltk.corpus.

For feature extraction, we need to count the frequency of every word that appears in the users review pool, remove the ones with frequency lower than a certain bound (we chose 1e-4), so as to reduce the sparsity of our training matrix, and also The frequencies of the rest 1,252 words compose our list of training features. The whole process is show in Figure 2.

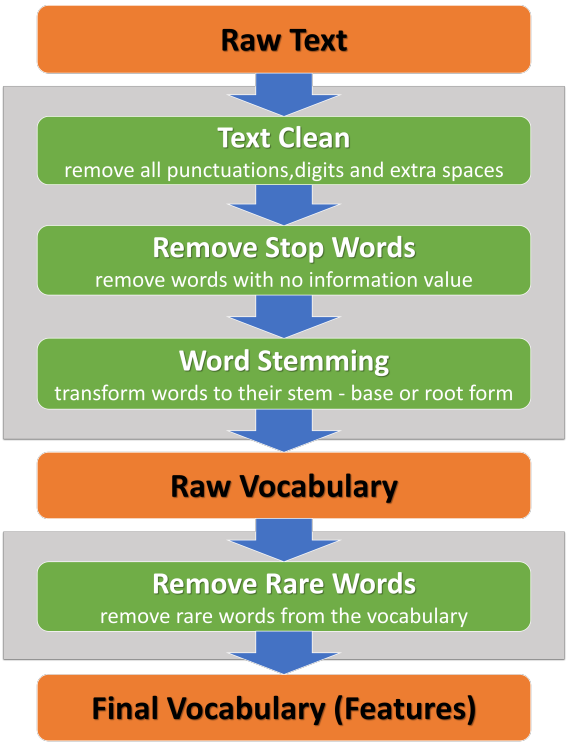

Figure 2: Data Preprocess Flow

Finally, we extracted a sparse training matrix, where the i-th row represents the i-th review, and the j-th column represents the j-th token. The (i, j)-entry of this matrix represents the number of occurrences of the j-th token in the i-th review. The entire data preprocessing is implemented in Python.

For the second tier, we extracted and concatenated all the other users review texts for every business this user has written a review for. The resulted 1,137 long reviews for these businesses are considered as what this user might write down for these businesses before visit, and the actual review this user has written as what this user actually write down after visit. Then, we used the users actual reviews to train the predicting model, while used the concatenated reviews of other users to generate sparse testing matrix.

4 Models

Our first tier of work is a sentiment analysis and classification problem. There has been a lot of existing work on similar topics of various scopes. Some previous work focused on the binary distinction of positive vs. negative[1]. Later work generalized to finer-grained scales such as numerical ratings[3, 4].

First we categorize the star rating with positive and negative (greater/smaller than 3 stars), so the problem is a sentiment polarity analysis. We used Logistic Regression (LR), Naive Bayes (NB) and Support Vector Machine (SVM). The previous two were implemented in Python, and SVM is implemented in MATLAB leveraging the LIBLINEAR package.

Then we generalized to 5-star rating scale classification using Multinomial Logistic Regression (MNLR), Naive Bayes (NB) and Support Vector Regression (SVR)[2]. Naive Bayes was still implemented in Python, and SVR was also implemented in MATLAB leveraging the LIBLINEAR package.

For the second tier, we decided to choose the SVR approach to train the predicting model, since the results from the first tier show that SVR approach performs the best.

5 Results and Discussion

5.1 Rating Prediction

For the first tier, we start with cross validation method. Training set includes 795 samples, which is about 70% of the data, and the test set has 342 remaining samples. The learning results are shown in Tables 1 - 2.

Model	Training MSE	Test MSE	Iters
LR	0.0025	0.1023	6
NB	0.3270	0.5029	/
SVM	0.0013	0.0000	24

Table 1: Polarity Classification

Model	Training MSE	Test MSE	Iters
MNLR	0.0025	1.0760	13
NB	0.4038	1.0585	/
SVR	0.3748	0.2043	19

Table 2: 5-Star Classification

From the above two tables, we can conclude that NB doesnt predict very well for both polarity and 5-star classification, which indicates that the assumption of NB method might not be valid in this case.

Both LR and MNLR has very low training error, but their test error is also very high. This means that the regression tends to overfitting the data.

The SVM and SVR predict the best for polarity and 5-star classification respectively. So we decided to use SVM for our recommendation model

In conclusion, all the methods could give very good results on polarity analysis. However, the 5-star classification error is still very large. So we decided to vary the training set size and plot the learning curves, as shown in Figure 3.

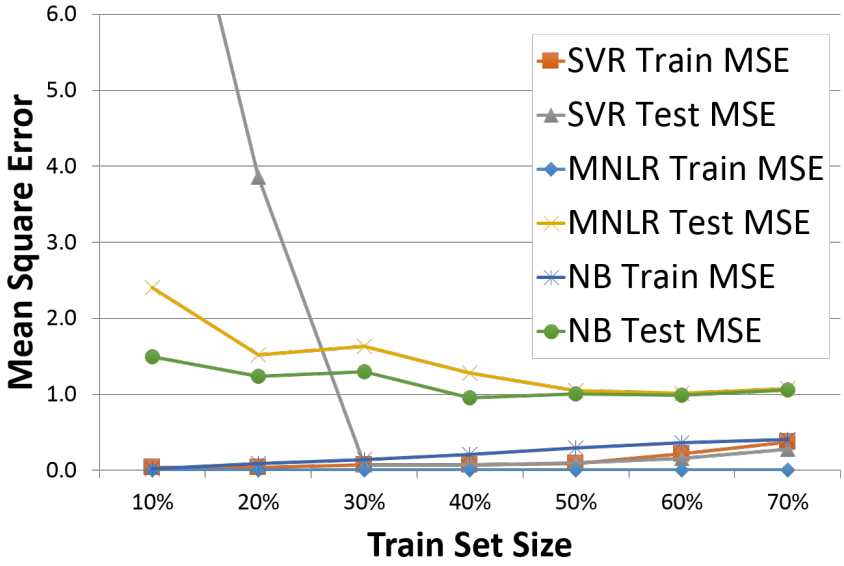

Figure 3: Learning Curve for 5-star Classification.

As we can see, test error and train error approach each other at first, but start to increase or remain unchanged as the training set size gets larger than 30%. This indicates a high bias problem, and larger dataset will not help. A possible solution is to add more features.

In order to get more insight into the problem, we also carried out an Ablative Analysis. We first run the model with no text pretreatment at all, and then add the next treatment step in the following table, and repeat running and adding the next step until the last step "Remove Rare Words", as shown in Figure 4.

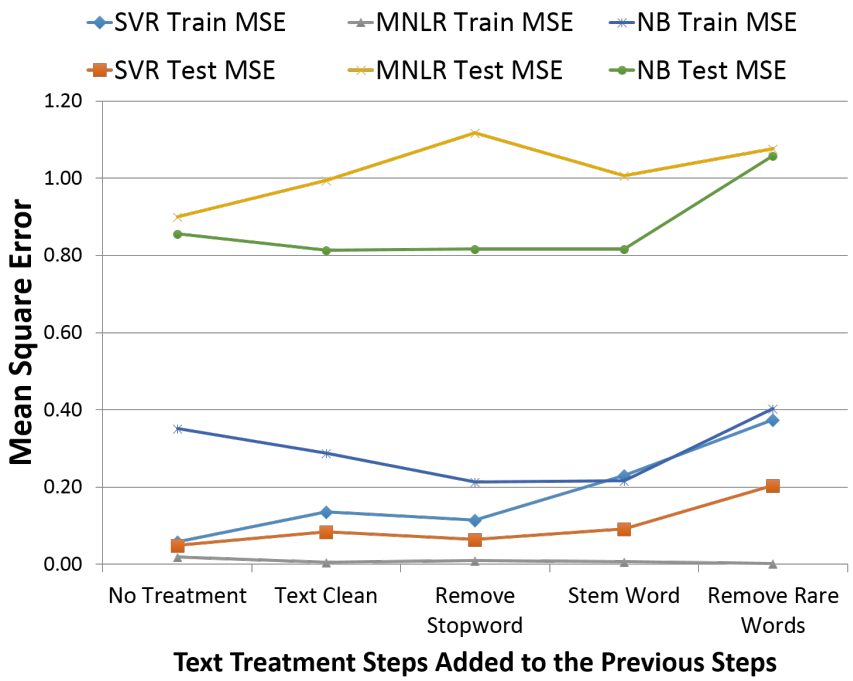

Figure 4: Ablative Analysis for 5-star Classification.

As we can see, removing features may lead to higher mean square error, which supported our hypothesis that the resulted model has high bias and needs more features.

5.2 Recommendation Model

For the second tier, we used all the data samples for tier 1 to train the SVR model, while used concatenated reviews of other users to test the SVR model. The Training Set includes 1,137 samples, and the Test Set includes 642 samples. The results are shown in Table 3.

Model	Training MSE	Test MSE	Iters
Remove Features, 5 star	0.8746	0.7168	19
Remove Stop Word, 5 star	0.1708	0.1079	15
Remove Stop Word, polarity	0.0178	0.0179	21

Table 3: 5-Star Recommendation Model

The first row in the table corresponds to a 5-star prediction model with full text pretreatment, both training and test MSE are really high. According to the ablative analysis, we decided to try a 5-star prediction model with text pretreatment up to removing stop words. The test mean square error (10.79%) does become much more acceptable.

By using a polarity prediction model with text pretreatment up to removing stop words, the results turn out to be highly accurate, indicating that the general recommendation approach, which predicts star rating using other users reviews, is acceptable. The reason why 5-star prediction has much higher error is that its very difficult to distinguish between ratings varying by only 1 star.

6 Conclusions

We have implemented multiple layers of feature extractors and experimented with several classification algorithms to predict star rating from review text, which gives a good result. We carried out learning curve and ablative analysis, and experimented with difference feature extractors. We have also predicted the customer's star ratings for restaurants using all the past reviews given by other customers and this customers predicting model.

7 Future

There is also further work that can be done in experimenting with the recommendation system. One option is to try more feature extraction approach such as bigram, trigram or word chunks. This could effectively increase the information that can be obtained from one review text, and add more features to overcome the high bias problem.

Another option is to improve the recommendation mechanism. If we want to get the rating of a user u on a business b, i.e. $x^{(ub)}$, we consider both business side rating (objective rating) and user side rating (subjective rating):

$$x^{(ub)} = y^{(ub)} + z^{(ub)} + \varepsilon^{(ub)}$$

where $y^{(ub)}$ is the business side rating , $z^{(ub)}$ is the user side rating and $\varepsilon^{(ub)}$ is Gaussian noise.

Assume both ratings as linear combinations of review text features, and get estimated rating with training data of only review text and not the actual rating:

$$y^{(ub)} = {v^{(b)}}^T \phi(r^{(ub)}) + \varepsilon_{\mathrm{B}}^{(ub)}$$
$$z^{(ub)} = {w^{(u)}}^T \phi(r^{(ub)}) + \varepsilon_{\mathrm{U}}^{(ub)}$$

where $r^{(ub)}$ is the review text of user u on business b, $\phi(r)$ is a feature extraction function, and $v^{(b)}$ and $w^{(u)}$ is weight vector for business side rating and user side rating respectively.

Therefore,

$$x^{(ub)} | r^{(ub)}; v^{(b)}, w^{(u)} \sim \mathcal{N}((v^{(b)} + w^{(u)})^T \phi(r^{(ub)}), \sigma)$$

and the log-likelihood is

$$\begin{aligned} & l\left(v^{(b)}, w^{(u)}\right) \\ =& \sum_{u,b} \log p\left(x^{(ub)} | r^{(ub)}; v^{(b)}, w^{(u)}\right) \\ =& \sum_{u,b} \log \frac{1}{\sqrt{2\pi}\sigma} - \frac{\left(x^{(ub)} - (v^{(b)} + w^{(u)})^T \phi(r^{(ub)})\right)^2}{2\sigma^2} \end{aligned}$$

Hence, we can use gradient ascent to obtain the weight vectors of each business ($v^{(b)}$) and each user ($w^{(u)}$) with all the reviews.

In this way, it is likely that we could get a better prediction of an review, since the user side part and business side part are evaluated separately. Also, this could help with a better recommendation system by considering more the user side part.

Reference

[1] Bo Pang and Lillian Lee. A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts. 2004. Proceedings of the ACL.

[2] AlexJ.Smola andBernhardScho lkopf. A Tutorial on Support Vector Regression. 2003.

[3] Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. 2005. Proceedings of the ACL.

[4] Daisuke Okanohara and Junichi Tsujii. Assigning Polarity Scores to Reviews Using Machine Learning Techniques. 2005. In IJCNLP.