Rating the Raters: Bias Analysis on Yelp Reviews for Improved Star Rating System

Qiaojing Yan, Jingwei Ji, Haitong Li

I. Introduction

YELP is becoming a vibrant bond connecting people with local business. The user experience and service quality of Yelp are in fact heavily dependent on the users' text reviews and star rating which aid our everyday decisions on local business, especially given tremendous amount of data being generated nowadays. The current Yelp's rating algorithm determines the overall rating without taking into account the reviewers' special characteristics， while different people may have diverse rating tendency. Some people are reluctant to give a five star while some other are the opposite. Even when two people have the same view on a restaurant, one may tend to give it five star while the other follow his habit and give three star. This tendency is hidden underneath the simple star rating system since we don't know what a costumer's real unbiased opinion is. However, review text is good indication of a costumer's real opinion. Rating with star takes less than a second, but writing a review costs much more and text conveys much more information. In this project, we address this interesting problem by using machine learning techniques to bridge the gap between the personalized rating tendency and the overall rating statistics. From a business and practical perspective, this method could potentially help Yelp to improve the fidelity of its rating system by considering the intrinsic variations of the reviewers' personalities.

As a classification problem, it is aimed to predict the 5-star rating results based on multiple features. Specifically, the input of this problem is the extracted from reviewers' comments, plus the User ID of each review. Previous research work has tried star rating prediction using different variations of bag of word features and classification models [1]-[3]. In this project, we use {perceptron, softmax, SVM]} to perform the classification task, where perceptron serves as a baseline to predict the polarity (two-class output). We train the classifier to predict star from review text. Since it is trained on all reviews, this classifier's rating tendency is the average of all reviewers. So, using this classifier, we can assess the rating tendency of a user: i.e., whether he/she intends to give a higher/lower star ratings with a similar review as the others. The assessment is based on the comparison of the predicted and actual star ratings based on reviews of a user. Then, the rating bias of users can be statistically calculated. The project framework is schematically shown in Fig. 1.

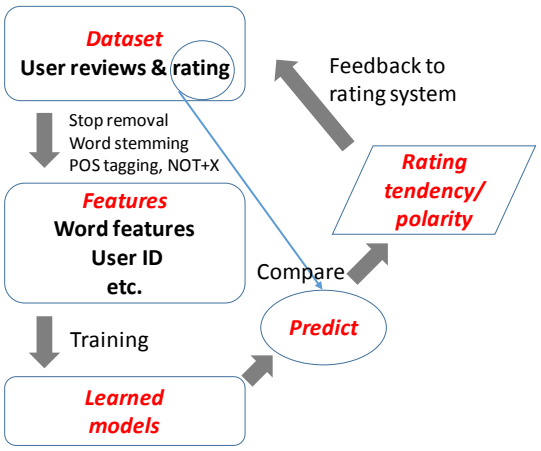

Fig. 1. Project framework: using classification algorithms with NLP word features and user Id feature to reveal the personal rating tendency/polarity, which can be used to statistically quantify the bias of users as the feedback to the rating system in order to deliver more accurate star rating results for a business.

II. Dataset & Features

A. Dataset

In this project, we use the public Yelp Dataset [4]. The whole dataset includes 61,184 Businesses, 366,715 Users, and 1,569,264 Reviews in json format. In the 'review' object type, individual user's review text and star rating from 1 to 5 are given. Besides, there are also many other features of a review such as tags including 'funny', 'useful', 'cool'. Overall, the dataset is large and rich enough for a convincing machine learning task. Another advantage of having a large dataset is that this allows us to experiment with different scale of subsets to generate

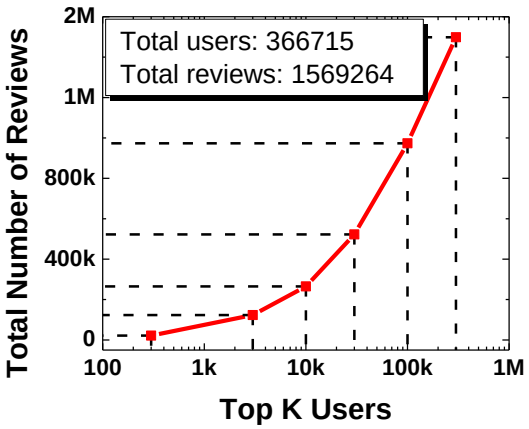

Fig. 2. Total number of reviews (actual dataset size) from sorted top *k* users (a representative 'size'), obtained from original dataset. For each subset, 70% examples are used for training and 30% examples are used for testing.

{'help': 1, 'chicken': 1, 'menu': 1, 'littl': 2, 'tapa': 1, 'black': 2, 'leav': 1, 'plate': 1, 'get': 2, "it'": 1, 'lunch': 3, 'Once': 1, 'tri': 4, 'divers': 1, 'enjoy': 1, 'pork': 1, 'incent': 1, 'grub': 1, 'back': 1, 'realli': 1, 'flavor': 1, 'special': 1, 'concern': 1, 'excit': 2, 'poke': 2, 'definit': 2, 'stuf': 1, 'nacho': 2, 'fella': 1, 'extens': 1, 'reason': 1, 'Everyth': 1, 'sinc': 2, 'wait': 1, 'great': 1, 'would': 1, 'Groupon': 1, 'thing': 1, 'place': 1, 'cod': 2, 'extrem': 1, 'top': 1, "can't": 1, 'miso': 2, 'one': 2, 'open': 1, 'better': 1, 'Monday': 1, 'come': 1, 'fri': 2, 'next': 1, 'avail': 1, 'includ': 1, 'offer': 1, "I'll": 1, 'ma': 1, 'solid': 1, 'us': 1, 'tasti': 1, 'well': 2, 'mind': 1, 'asparagu': 1, 'probabl': 1, 'belli': 1, 'anoth': 1, 'excel': 1, 'fill': 1, 'roll': 2, 'afternoon': 1, 'gill': 1, 'even': 1, 'textur': 1, 'I': 2, 'price': 1, 'moment': 1, 'garlic': 1, 'nasu': 2, 'favorit': 1, 'Saturday': 2, 'causeissu': 1, '808': 1, 'time': 1, 'escap': 1, 'hungri': 1, 'came': 1}

Fig. 3. An example of extracted word feature after NLTK preprocessing (stemming, stop word removal, etc.).

the learning curves on such task. We sort the original dataset based on how many reviews a user has given, and generate the total number of reviews as top *k* reviewers give, as shown in Fig. 2. For the following experiments with {perceptron, softmax, SVM}, we use *k* to represent different data scale, where the actual data size should be on the y-axis of this figure because one active user can generate thousands of reviews. For the full-size experiment, we would have ~1.5M examples as the training (70%) and testing (30%) dataset, which is a computation-intensive task.

B. Features

Our baseline feature is just the bag of tokens in the review text. Based on that, we try to use NLP technics to process the tokens.

1. Stemming and stop word removing:

Word in same meaning may have different form. For example, "argue", "argued" and "arguing". We used Porter stemmer in the Natural Language Toolkit (NLTK) package to do stemming [5]. After stemming, these three words are reduced to the same stem "argu". An example of stemmed word feature is shown in Fig. 3.

2. Part of speech tagging:

Using all the English words as our feature may cause over fitting as there are too many words. We initially assume that words that are adjective and verb carry more information. So we first do part of speech tagging and then used only adjectives (JJ) and adverbs (RB) as our features. The POS tagging is done using the Stanford CoreNLP [6]. However, we found out this technic did not help to reduce error rate.

3. Dealing with negations

We first define a set of words that mean negation: N = {no, not, isn't, wasn't, aren't, weren't}. Then if we find a word n ∈ N, we add a tag "not" to the word after it to form a new feature. For example, "not bad" give us "not-bad". We also concatenate "not" to the second word after it. For example, "doesn't taste good" give us "not-good".

How do we know each user do have a characteristic tendency? Besides the word features on the review comments, we also include the user ID as an additional feature. Error rates when using this additional feature are compared to error rates when using the word features. If the error rate drops, then it verifies that each user do have his own characteristic.

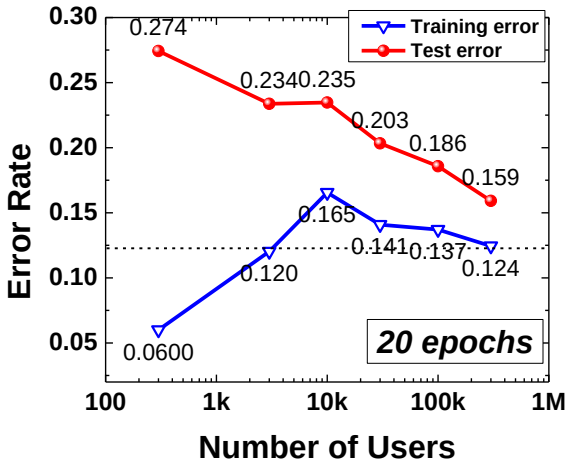

Fig. 4. Training and test error rates using perceptron to predict the polarity of users' rating. Subsets of different scale are assessed varying the number of top *k* users.

III. Results and Discussions

A. Polarity Prediction

We first conduct the polarity prediction task using perceptron algorithm. The boundary for 'polarity' is 3 star, indicating whether a user feels the service is good or not. The reviews from top *k* users are used to train perceptron, where *k* is varied to generate different size of subsets for the training and testing. 70% of the generated subset is used as training set and 30% as test set. The metrics for evaluation is the strict error rate, which simply counts the

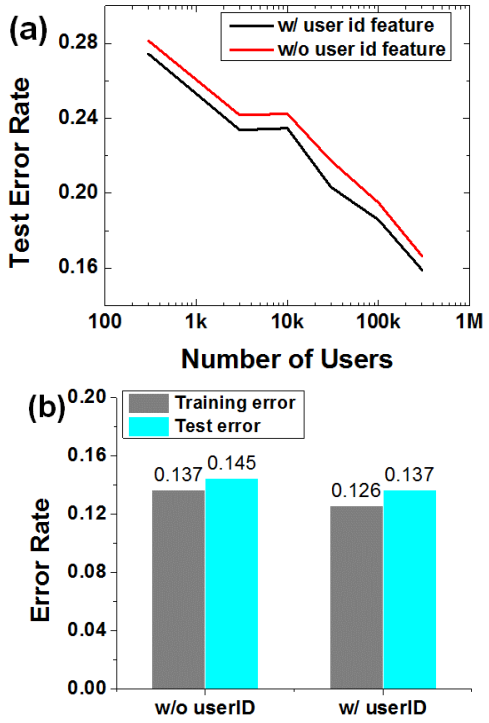

Fig. 5. (a) Test error as a function of including/excluding user ID feature on different data size. (b) Training and test error rates on full-size dataset.

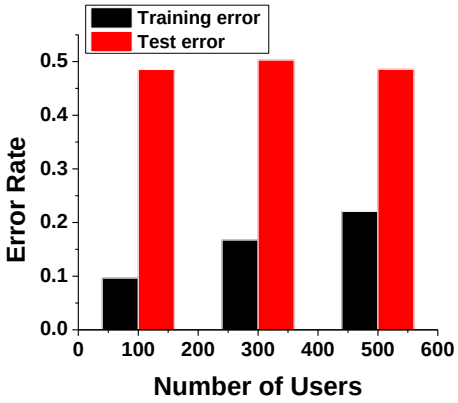

Fig. 6. Softmax prediction results on small-scale subsets (yet still tens of thousands of reviews).

number of misclassified cases for both training and testing phases. The training is done after 20 epochs for each dataset, and the results are shown in Fig. 4 with error rate numbers listed inside. As the user number increases, the actual data size increases even more rapidly as illustrated in Fig. 2. Perceptron is implemented by hand in Python, following lecture notes, and we observe from experiments that the test error drops with data size increasing. For larger data size, the algorithm is able to better separate the above-3-star rating from below-3-star rating. The impact of user ID feature is then evaluated on subsets of various scale and also on the full-size dataset.

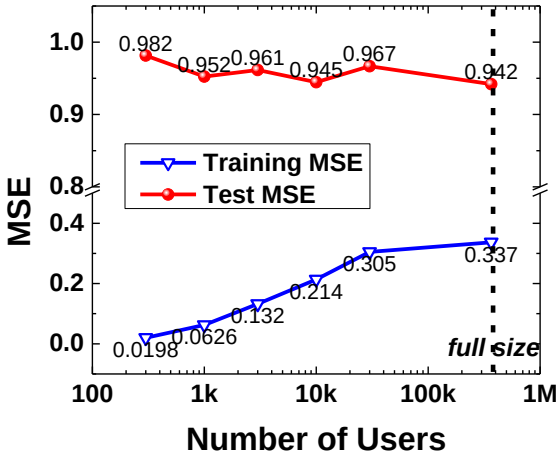

Fig. 7. Mean square error (MSE) of training and test for SVM, with data size enlarging.

As illustrated in Fig. 5(a), the learning trend with data size enlarging is similar between including and excluding user ID feature, which is consistent with the fact that increasing subset size makes reviews richer regardless of the status of user ID feature. Then, based on the assessment on the full-size (~1.5M reviews from 366k users) dataset, the test error drops from 0.145 to 0.137 after incorporating the user ID feature on top of the word feature. This simple user-specified feature has evident impact on the prediction performance, which further helps us to validate our assumption on the users' rating bias hidden beneath the large-scale review stars and texts.

B. Star Rating Prediction

For the 5-class star rating prediction task, we use Softmax and SVM which will be discussed in this section. Softmax is first quickly evaluated before we focus on the comprehensive results of SVM. Softmax algorithm is implemented by hand in Python, following lecture notes, and the test results on small-scale subsets are shown in Fig. 6, where around 49% test error is obtained for 5-class prediction. For large-scale dataset, since SVM is a widely-used standard algorithm, we will then focus on SVM to comprehensively analyze the rating prediction and the underneath bias/tendency. SVM is implemented using the scikit-learn package [7] in Python. We used Support Vector Classification (SVC) model with linear kernel. And here the performance metrics is the mean square error (MSE) for both training and testing. The training and test MSE of SVM on various data size are shown in Fig. 7. The trend of the learning curve is that with data size increasing, overall test MSE will decrease whereas the training MSE increases. For SVM, it is important to make sure that the training examples exceed

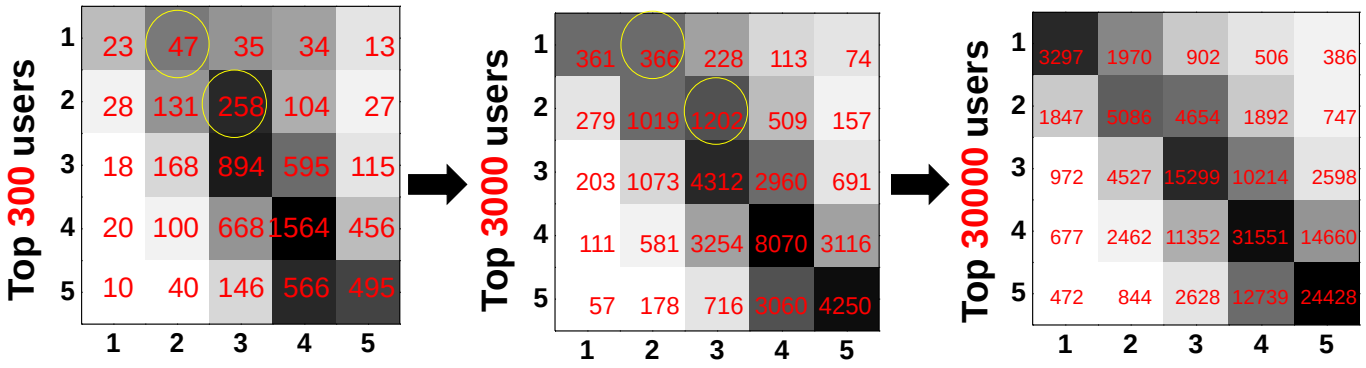

Fig. 8. Classification confusion matrix for (left) top 300, (middle) top 3k, and (right) top 30k users' reviews. The vertical "1" to "5" are the actual star rating while the horizontal "1" to "5" at the bottom indicates the predicted star rating. The gray-scale map is generated based on independently normalizing the numbers in each row. The darker ones indicate dominant prediction stars given by SVM. A "+1" low-star rating pattern can be observed for the misclassification under small data size.

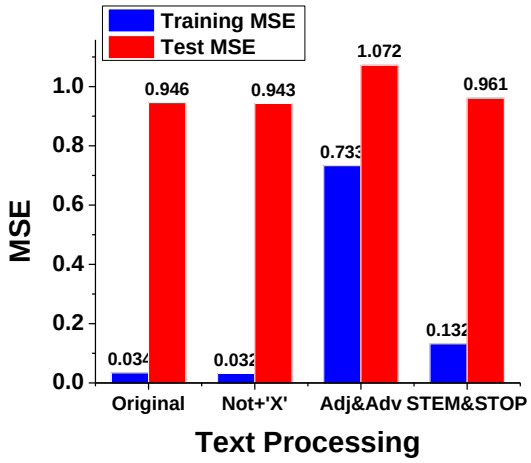

Fig. 9. Training and test MSE after different kinds of text preprocessing methods.

the number of features, otherwise the performance will not be acceptable. On the full-size dataset, 0.942 MSE is achieved, which is a reasonable result given such kind of prediction task and the performance of SVM algorithm.

We further analyze the statistical results of the SVM 5-class classification on various data size, and then generate the confusion matrices as shown in Fig. 8. For relatively small datasets, the misclassification has some structured pattern for low-star-rating region, as illustrated by yellow circles in Fig. 8(a) and (b). The algorithm tends to give "+1" star rating, or more specifically, predicting actual star 1 to star 2, and predicting actual star 2 to star 3. Such pattern is not valid for higher stars like 4 or 5. Then, after further increasing the data size, such "+1" pattern is no longer valid either for low stars or for high stars (a dark diagonal line is shown). Due to the increase in the review dataset, the rating bias of part of users can be neutralized by others, so the algorithm won't give higher stars this time for low-star-rating region, as shown in Fig. 8(right).

Table I
MSE on Full-Size Dataset

	Training MSE	Test MSE
stemming & word removal	0.337	0.942
NOT+'X'	0.330	0.946
NOT+'X' w/ User ID feature	0.188	0.910

Since NLP is involved, we also experiment with different text preprocessing methods to investigate the impact on the 5-star prediction task. Here, we use top 3000 users' reviews and fix the data size. Four different group of features are assessed and compared: {original, Not+'X', Adj & Adv, STEM&STOP}. As can be seen in the result, word stemming or adjective and adverb extraction does not help in reducing the error rate. This is in contrast to our initial expectation. This may mean that the non-stem part of the word, and the words with POS other than JJ and RB are all good indication of sentiment. Also, since the dataset is very big, feature size is allowed to be large, while reducing feature dimension may lead to high bias.

Different feature choices are assessed for the full-size dataset as well. NOT+'X' method performs well on small-size dataset, but for the full-size dataset it can be observed that it helps only with User ID feature included, as illustrated in Table I. Here, including the user ID feature gives a significant improvement in training and test MSE. We will then use this best-performance classifier to perform the statistical analysis on the users' bias.

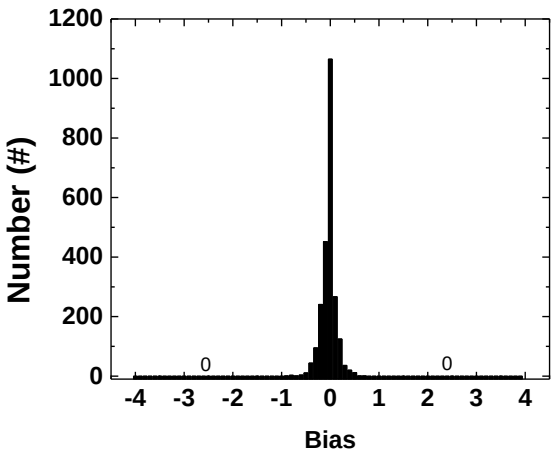

Fig. 10. Statistical distribution of bias of 3000 users. Most of the non-zero counts range between -1 and 1.

C. User Bias Analysis

With SVM classifier trained on full-size data, we could compute each reviewer's bias b_r by:

$$b_r = \frac{\sum_{i=1}^{N_r}(star_i - star_{i,predict})}{N_r}$$

where N_r indicates the total number of reviews the reviewer has written. $Star_i$ represents the reviewer's real star rating of the i-th review, and $Star_{i,predict}$ is the predicted star. In short, b_r is the average of the reviewer's star deviations of every reviews he wrote. With our definition of bias, we can get the statistical distribution of users' biases, as shown in Fig. 10. We only calculate the bias of reviewers with more than 5 reviews, which correspond with experienced Yelp reviewers. From Fig. 10, we can see that most reviewers are neutral, which agrees with natural intuition. Most of the non-zero counts range between -1 and 1, indicating that experienced Yelp reviewers are not likely to be too extreme.

Given the bias of a reviewer, when calculating the star of a business, we should first subtract the bias from the reviewer's original star, and then calculate an arithmetic mean, by which we think the final star would be more objective.

IV. Conclusion

In this project, we have applied machine learning algorithms such as perceptron, softmax, and SVM on large-scale public text dataset to perform classification tasks. The impact of data size and feature selection on the prediction performance is quantitatively studied. The rating bias is statistically investigated based on the learning results. For the future work, we plan to further enhance our feature extractor and incorporate more high-level features such as word dependencies, incorporating word vectors, or do sematic parsing. We don't have time to do this in our project since the dataset is very big and parsing would need a long time. Text understanding is still an active research frontier and we expect to use new technologies to get better results.

References

[1] Linshi, Jack. "Personalizing Yelp Star Ratings: a Semantic Topic Modeling Approach." Yale University, 2014.

[2] A. Maas, R. Daly, P. Pham, D. Huang, A. Ng, and C. Potts, "Learning Word Vectors for Sentiment Analysis," ACL 2011.

[3] Fan, Mingming, and Maryam Khademi. "Predicting a business star in Yelp from its reviews text alone." arXiv preprint arXiv:1401.0864, 2014.

[4] Yelp Dataset Challenge (Data available online): http://www.yelp.com/dataset_challenge, Last Updated: July 30, 2014.

[5] Natural Language Toolkit (NLTK), http://www.nltk.org/, 2015.

[6] Manning, Christopher D., et al. "The Stanford CoreNLP natural language processing toolkit." Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations. 2014.

[7] Fabian P., et al. "Scikit-learn: Machine learning in Python." The Journal of Machine Learning Research 12, pp. 2825-2830, 2011