Prediction of Yelp Ratings Based on Reviewer Comments Segmented by Business Type

Kent Lee[1*] and James Ross[2*]

[1]Stanford University, Department of Structural Biology, Biophysics Program

[2]Stanford University, Department of Computer Science

I. INTRODUCTION

Yelp[1] is a popular website that allows users to submit numerical ratings (integers 1-5, inclusive, 5 being positive sentiment) and text reviews (reviewer's comments) to express their experiences interacting with various businesses. This information helps others determine which businesses best fit their needs. The prevalence and utility of rating and review services provide the opportunity and motivation to study the prediction of ratings based on reviews. While these services are prevalent, many internet forums are dedicated to publishing user reviews of products such as laptops[3] and vehicles[4] without quantitative ranking. Often, forums hosting qualitative reviews such as "laptop stopped working but helpdesk was useless!" and "customer service was horrible" would benefit from an algorithm predicting rating from reviews to summarize user sentiment quantitatively. Furthermore, prediction algorithms allow rating and review websites gather additional rating information by predicting from external enthusiast forums where reviews are more descriptive.

Our goal was to apply machine learning techniques to Yelp's published dataset to accurately predict ratings from reviews. An input of a string of words into our Multinomial Naive Bayes algorithm would output a predicted rating between 1 and 5 describing the customer's sentiment regarding a business. While this has previously been attempted[5][6], most approaches do not focus on training data segmentation. We believed that segmenting training data by business sector (e.g. restaurants) allows more accurate predictions because feature sets become less "diluted" or noisy and words gain meaningful value when given a focused context. For example, an auto mechanic business customer may be "extremely satisfied with the new cold air intake system!" while a restaurant customer may say that "the food took a while and came out cold!" Taken together, if feature size was fixed, the word "system" would offer as little predictive value in ranking restaurants as "food" would for the auto business. Furthermore, "cold" is sentimentally negative in the restaurant review, yet positive in the auto business review. Training data segmentation aims to reduce such uninformative occurrences.

*These authors contributed equally to this work

II. RELATED WORK

Interesting approaches involve looking up the quantified sentiment of each word from SentiWordNet and summing the positive/negative/neutral in a document and fitting to regression rather than a completely probabilistic approach[12][11]. While I think this is the best method, the disadvantages are being limited to characterized words. Groups using non-word features such as part-of-speech is an approach we have not considered[10]. A useful troubleshooting approach is displaying the highest impact features in a prediction[9]. The potential to use latent factors through matrix factorization seemed to have potential, but others who tried it received lower MSE than us[13]

III. DATASET AND FEATURES

A. Dataset

Data was downloaded from Yelp's Dataset Challenge[2] in JSON format containing 1.6 million reviews and 61 thousand businesses. The dataset was trimmed by a custom C# script to yield a dataset containing entries with only metrics of interest: review text on the business and reviewer rating of the business. This formed the dataset allReviews. To create a dataset containing only reviews from one business segment, we identified the top represented business segment in allReviews and created a dataset containing only entries from this segment. The top represented segment was Restaurant (21,892 entries) so all reviews on restaurants made up segReviews. Thus, allReviews and its subset segReviews are the two datasets that were used. Each dataset contained

8,000 entries as training set, 1,000 as validation set, and 1,000 as test set. Set sizes were smaller than available data due to hardware constraints during optimization. Data was converted from the JSON format to the form of
***Reviews_Ratings.txt:

```
5
4
4
...
```

***Reviews_Reviews.txt:

```
this vegas thai place can be tricky to find
    it is off the strip hidden away in a
    strip mall ...
yum yum thai iced coffee fried tofu starter
    and ...
the commercial center mall is not the most
    obvious choice for las vegas best thai
    ...
...
```

B. Preliminary Dataset Processing

allReviews and segReviews were processed as follows. For each review entry, all characters were removed except for whitespaces and English alphabet characters. All characters were converted to lowercase. Each occurrence of a whitespace character was replaced with one space character. All consecutive space characters were replaced with one space character. This preempts potential string parsing and string comparing problems, improves compatibility with potential dictionaries and third party algorithms (e.g. stop word corpus, stemming algorithms), and most importantly, normalizes and combines similar features (e.g. don't vs dont, Thai vs thai, delicious vs delicious!).

C. Dataset Processing During Optimization

Stop words based on Natural Language Toolkit's (NLTK)[7] stop word corpus were removed to reduce common neutral-sentiment features that provide little predictive value. Stemming was introduced by accessing NLTK's implementation of Porter's stemming algorithm, which allows for further normalization and combination of similar features.

D. Feature Selection

After preliminary processing, the baseline feature size was 19,898 for allReviews and 18,102 for segReviews. Manual removal of features was attempted by looking through the feature list. Ngrams were used to increase the feature set to include groups of words that may collectively have sentimental value while sentimentally-neutral individually. SelectKBest using a Chi-squared distribution was used to select features that were found to have a dependence on specific labels. SelectKBest and Ngrams were implemented through scikitlearn[8].

IV. Methods

A. Multinomial Naive Bayes

Like many learning algorithms, Multinomial Naive Bayes first attempts to derive parameters that maximize the joint likelihood of the training data, given by

$$\mathcal{L}(\Phi_{y=0},...,\Phi_{y=p-1},\Phi_{j|y=0},...,\Phi_{j|y=p}) = \prod_{i=1}^{m} p(x^{(i)},y^{(i)}) \tag{1}$$

where $\Phi_{y=q} = P(y=q)$, $\Phi_{j|y=q} = P(x_j = 1|y=q)$, q is a number in the range of classes, j is a number in the range of features, m is the number of training examples, p is the number of classes, and $(x^{(i)}, y^{(i)})$ are the ith training example. Making the "naive" assumption that given any y, all x_i are mutually independent, the solution to the maximization is

$$\Phi_{j|y=q} = \frac{\sum_{i=1}^{m} \mathbb{1}\left\{x_j^{(i)} = 1 \wedge y^{(i)} = 1\right\}}{\sum_{i=1}^{m} \mathbb{1}\left\{y^{(i)} = q\right\}} \tag{2}$$

$$\Phi_{y=q} = \frac{\sum_{i=1}^{m} \mathbb{1}\left\{y^{(i)} = q\right\}}{m} \tag{3}$$

Once the parameters from equations 2 and 3 are fit from the training data, prediction of a given test example is simply calculating $\operatorname{argmax}_q P(y=q|x) = \frac{P(x=1|y=q)P(y=q)}{P(x)}$, where q is the class with the highest posterior probability of all classes and x is the feature vector.

B. Support Vector Classifier

SVC attempts to define a line (or hyperplane in higher dimensions) that separates classes in a plot of the training set that maximizes the distance (also known as the margin) between the line and the closest training example of each class. This line is called the decision boundary. If line is given by $f(x) = (w^T x + b)$, it can be shown that the decision boundary parameters can be set by

$$min_{\gamma,w,b} \frac{1}{2}|w|^2 + C\sum_{i=1}^{m}\xi_i \tag{4}$$

with the constraint that

$$y^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i, i = 1,...,m \tag{5}$$

$$\xi_i \geq 0, i = 1,...,m \tag{6}$$

where $(x^{(i)}, y^{(i)})$ are the ith training example, w and b are parameters of the line, C and ξ are penalty terms that penalize the objective function for each training example on the wrong side of the decision boundary, and m is the number of training examples. After the optimization problem is solved, the decision boundary is known and classification can be made by determining which side the training example lies with respect to the decision boundary. It turns out that the decision boundary depends only on the training examples closest to the decision boundary (known as support vectors), which makes training SVCs computationally efficient. Furthermore, it turns out that the solution can be written as a linear combination of $\langle x^{(i)}, x \rangle$, which is the form of a Kernel, allowing efficient computation for large feature sizes. Thus, features are often transformed into other feature spaces to achieve better data separation. Here, we used a linear kernel. For classification with more than 2 classes, simply classify one class versus the rest of the classes, for each class. We applied scikitlearn's implementation of both algorithms. Immediately before running any algorithm, the reviews were first tokenized with the exception of experiments with SelectKBest, where the features are tokenized, then run through SelectKBest, then run through the machine learning algorithm.

V. Discussion

A. Metrics

The main metric used was accuracy, given by the number of validation or test examples correctly labeled divided by the total examples to be predicted. A higher accuracy suggested that the predictor was often able to correctly classify the reviews. While cross validation was not done on validation and test data, 10-fold cross validation was used in generating learning curves. In the final comparison, mean squared error (MSE) was used to determine the average difference between the predicted label and the actual label: $\frac{1}{n}\sum_{0}^{n-1}(y_{i,pred} - y_{i,actual})^2$.

B. Baselining

To construct an initial accuracy comparison between predictors trained on unsegmented and segmented datasets, baseline prediction accuracies of allReviews and segReviews were each determined by training a Multinomial Naive Bayes (MNB) algorithm on 6,000 training examples and testing on 1,000 test examples, with a vocabulary of the 5,000 highest frequency words. MNB was used because it was one of the simplest and quickest to train classification algorithms available. While we started with MNB, we later attempted SVC after optimizing our features. The number of training examples and feature size initially used were picked arbitrarily to ensure room for optimization in either direction. This prevented wasting time optimizing the model in the beginning when many decisions were still in flux.

Table I presents the data that will be discussed in the rest of the discussion section. Attention drawn to specific rows will be of the form "R#".

The baseline showed that segReviews (R2) had higher accuracies than allReviews (R1) for the test set (0.55 and 0.52 respectively), which supported our hypothesis that segmented training examples yield better predictors than unsegmented. However, lack of difference in training errors (both 0.72) led us to suspect that test set accuracy differences may have been noise.

C. Stop Words and Stemming

Once a baseline was formed, it was more efficient to focus on improving only segReviews accuracy instead of applying each subsequent change to both segReviews and allReviews. Once performance on segReviews was satisfactory, we returned to apply the same optimizations to allReviews and compare accuracies again. Thus from here onwards, optimization was against segReviews only. Stop words and stemming is known to help with feature normalization and reducing the possible total vocabulary by removing words that are either neutral or common in all classes (e.g. "the", "and", "was") and by replacing words stemming from the same root with the root itself (e.g. replacing "purchase", "purchased", "purchasing" with "purchas", collapses 5 features into 1). After removing stop words (R4), training accuracy increased marginally to 0.73 while validation accuracy remained constant at 0.53 as compared to validation baseline (R3). Stemming (R5) decreased training accuracy to 0.72 and validation accuracy decreased to 0.52. A plausible explanation for no significant improvement after stemming and removing stop words was that our feature set was still too large for such optimizations to make a difference; the top features may still contain words that have little sentimental value. It is also possible that the features used may contain few stop words and words affected by stemming.

D. Learning Curve Analysis

We wanted a rational approach to improve our model rather than attempting random optimization pathways so we generated a learning curve with scikitlearn to test for overfitting or underfitting. Plots have points representing averages of 10-fold cross validation, with shaded regions being standard deviation. As seen in Figure 1, training accuracy was significantly higher than cross validation accuracy, indicative of high variance and overfitting. The high number of features allowed the algorithm more degrees of freedom to fit the training data better,

Row	Procedure	Dataset	Type	Validate/Test Accuracy	Train Accuracy	MSE	Algorithm
1	Baseline	allReviews	Test Data	0.515	0.721	0.869	MNB
2	Baseline	segReview	Test Data	0.554	0.724	0.61	MNB
3	Baseline	segReview	Validation	0.534	0.724	0.799	MNB
4	A = Baseline + Stop Words	segReview	Validation	0.533	0.732	0.777	MNB
5	B = A + Stemming	segReview	Validation	0.515	0.722	0.774	MNB
6	C = B + FeatureSize=200	segReview	Validation	0.542	0.544	1.151	MNB
7	D = C + TrainingExamples=8000	segReview	Validation	0.521	0.535	1.12	MNB
8	E = D + Manual Stop Words	segReview	Validation	0.535	0.559	1.009	MNB
9	F = E + skb=500 + FeatureSize = 5000	segReview	Validation	0.589	0.628	0.685	MNB
10	G = F + Bigram + Trigrams (Optimal)	segReview	Validation	0.597	0.635	0.667	MNB
11	G (Optimal)	segReview	Test Data	0.632	0.635	0.522	MNB
12	G (Optimal)	allReviews	Test Data	0.548	0.582	0.833	MNB
13	G (Optimal)	segReviews	Test Data	0.616	0.709	0.563	SVC
14	G (Optimal)	allReviews	Test Data	0.558	0.672	0.864	SVC

TABLE I

The effect of optimization procedure on prediction accuracy and mean squared error based on dataset and algorithm used.

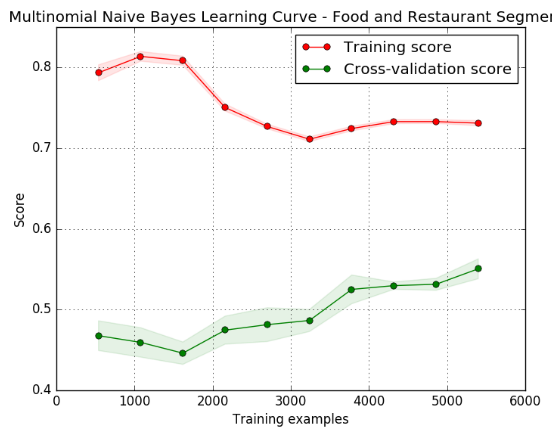


Fig. 1. Learning curve after stop word removal and stemming with 5,000 features shows high variance. Score is prediction accuracy.

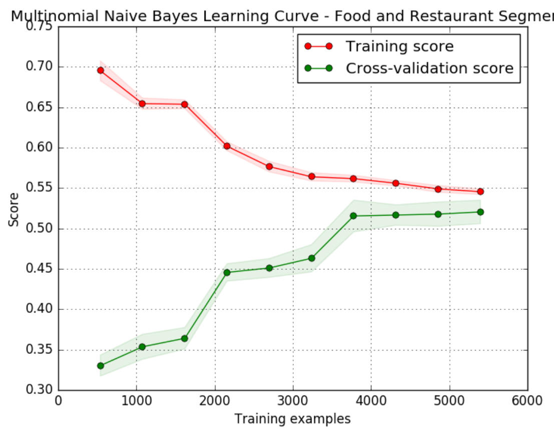


Fig. 2. Learning curve after stop word removal and stemming with 200 features is optimal. Score is prediction accuracy

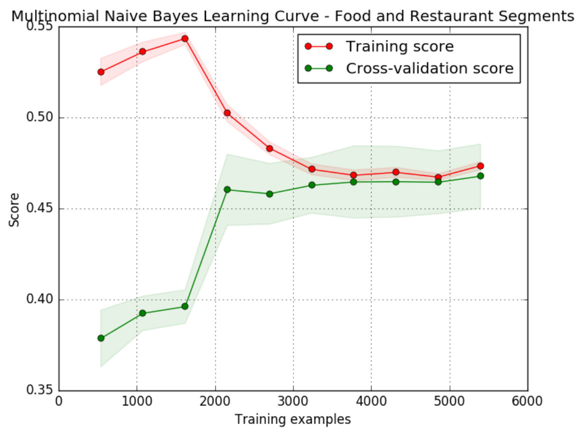


Fig. 3. Learning curve after stop word removal and stemming with 20 features shows high bias. Score is prediction accuracy

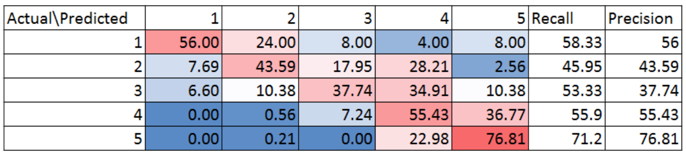
yielding significantly higher training accuracy. To reduce overfitting, we attempt to reduce the number of features. Figure 2 shows the learning curve with 200 features while Figure 3 shows underfitting when feature size decreased to 20, where training accuracy decreased dramatically from a maximum of 0.82 to a maximum of 0.54, indicating high bias. After empirical optimization, stop word removal and stemming with 200 features was optimal (R6), yielding lower training and higher validation accuracy (0.54 for both training and validation accuracy). Figure 2 shows no more overfitting or underfitting. The increase in validation accuracy was encouraging and the decrease in training accuracy was associated with the correction from overfitting. With 6,000 training examples and 200 features, this agrees with the idea that the number of features should be significantly smaller than the number of training examples. The convergence between training and cross validation accuracy implies that the predictor generalizes well to unseen data. Figure 2 shows slight upwards slope in cross validation accuracy, suggesting that increasing training examples may increase validation accuracy further. Increasing training set size from 6,000 to 8,000 revealed clear convergence (plot not shown) of training and cross validation accuracy, but decreased validation accuracy to 0.52 (R7), suggesting that cross validation learning curve has plateaued and no more optimization of features size or training examples is necessary.

E. Feature Optimization

Starting with manual feature optimization, 83 sentiment-neutral words (e.g. food, burger, dish, Thai, chicken, Bellagio) were identified from the features to add to the stop words list, with the next highest frequency words replacing those removed. The goal of this is similar to that of removing stop words and the process (R8) increased test accuracy to 0.54, speaking to the value of customized stop word lists. However, it is nontrivial to manually remove all words we considered sentiment-neutral given the large vocabulary. Furthermore, it is likely that using the highest frequency features may shadow some highly predictive mid-frequency features. This led us to SelectKBest using the Chi-squared, which evaluates the likelihood that a given feature is independent of a label and selects features that are not independent of a label. Chi-squared describes how likely the appearance of a specific feature for a specific label is due to random chance based on the training features and labels. From 5,000 features, SelectKBest selected 500 features whose appearance for a label are unlikely to be due to random chance (and thus are dependent on the label). This significantly improved validation accuracies to 0.59 after some optimization (R9). Bigram and trigram features were used to further increase the features SelectKBest could choose from, increasing validation accuracy to 0.60 (R10). nGram helps because some features have little correlation with labels independently, but more correlation when grouped together, giving SelectKBest more dependent features. Select features identified by SelectKBest are: "pretti damn", "goooood", "lick plate", "flavorless", "oh well", "perfect", "realli tast", "tasteless", "would highli", and "appoint".

F. Confusion Matrix Analysis

Figure 4 shows a confusion matrix for the optimal state (R10). The matrix has percentages that add to 100 across rows. Adjacent misclassifications make up a majority of errors, which is understandable since the difference between 4 and a 5 is slight and subjective. We are pleased to see the majority of classifications in each row are correct (red in the diagonal). Troubling areas are actual = 2, predicted = 4 and actual = 3 and predicted = 1 and 5.



Actual\Predicted	1	2	3	4	5	Recall	Precision
1	56.00	24.00	8.00	4.00	8.00	58.33	56
2	7.69	43.59	17.95	28.21	2.56	45.95	43.59
3	6.60	10.38	37.74	34.91	10.38	53.33	37.74
4	0.00	0.56	7.24	55.43	36.77	55.9	55.43
5	0.00	0.21	0.00	22.98	76.81	71.2	76.81

Fig. 4. Confusion matrix for fully optimized segReviews with MNB in percentages for each row. Total sample size is 999. Calculated by scikitlearn

G. Return to Baseline

Satisfied with the improvements, we compare test data accuracies between allReviews and segReviews after applying our optimizations to both and against baseline. Baseline (before optimization), segReviews (R2) was 0.03 more accurate than allReviews (R1) (0.55 against 0.52, respectively), but after optimization, segReviews (R11) was 0.08 more accurate than allReviews (R12) (0.63 against 0.55, respectively). Additionally, optimization (R11) increased segReviews (R2) accuracy by 0.08 compared to baseline. With SVC, after applying the MNB optimization, segReviews (R13) was 0.06 more accurate than allReviews (R14) (0.62 against 0.56, respectively).

VI. Conclusion

We have shown that training data segmentation has a significant effect on prediction accuracy, especially if incremental upon other dataset optimizations. Training only on Yelp's restaurant data shows that an accuracy improvement of 0.08 over unsegmented training data when given the same optimization procedure with MNB. The highest performing algorithm was MNB with NLTK stop word removal, Proter's stemming, manual stop word removal, 8,000 training examples, bigram and trigrams, and Select 500 best from 5,000 features. This was better than SVM likely because the whole optimization process was done with MNB. At optimal, our MNB MSE was 0.522, so on average, our prediction was off less than one a Yelp rating (0.71).

For future work, implementing forward search, backward search, and mutual information calculations to select better features may yield better selections than SelectKBest. Introducing weights to tokenization through TF-IDF, with TF factor being raw counts, binary, or log normalized (1+log(feature frequency)) may help adjust for frequently appearing, low impact words. Finally, Applying segmented training sets to other business segments would help confirm our finding.

References

[1] Yelp. (n.d.). Retrieved December 7, 2015, from http://www.yelp.com/

[2] Yelp Dataset Challenge. (n.d.). Retrieved December 7, 2015, from http://www.yelp.com/dataset_challenge

[3] NotebookReview. (n.d.). Retrieved December 7, 2015, from http://forum.notebookreview.com/

[4] Values & Prices Paid. (n.d.). Retrieved December 7, 2015, from http://forums.edmunds.com/discussions/tagged/x/values-prices-paid

[5] Li, C., Zhang, J. (n.d.). Prediction of Yelp Review Star Rating using Sentiment Analysis. Retrieved December 7, 2015, from http://cs229.stanford.edu/proj2014/Chen Li, Jin Zhang, Prediction of Yelp Review Star Rating using Sentiment Analysis.pdf

[6] Xu, Y., Wu, X., & Wang, Q. (n.d.). Sentiment Analysis of YelpâĂŸs Ratings Based on Text Reviews. Retrieved December 7, 2015, from http://cs229.stanford.edu/proj2014/Yun Xu, Xinhui Wu, Qinxia Wang, Sentiment Analysis of Yelp's Ratings Based on Text Reviews.pdf

[7] Bird, S., Klein, E., and Loper, E. Natural Language Processing with Python. O'Reilly Media, 2009

[8] Pedregosa et al. Scikit-learn: Machine Learning in Python. JMLR 12, pp. 2825-2830, 2011.

[9] Mooney, R. J., Bennett, P. N., Roy, L. (1998). Book recommending using text categorization with extracted information. In Proc. Recommender Systems Papers from 1998 Workshop, Technical Report WS-98-08.

[10] Gupta, N., Di Fabbrizio, G., Haffner, P. (2010, June). Capturing the stars: predicting ratings for service and product reviews. In Proceedings of the NAACL HLT 2010 Workshop on Semantic Search (pp. 36-43). Association for Computational Linguistics.

[11] Qu, L., Ifrim, G., Weikum, G. (2010, August). The bag-of-opinions method for review rating prediction from sparse text patterns. In Proceedings of the 23rd International Conference on Computational Linguistics (pp. 913-921). Association for Computational Linguistics.

[12] Siersdorfer, S., Chelaru, S., Nejdl, W., San Pedro, J. (2010, April). How useful are your comments?: analyzing and predicting youtube comments and comment ratings. In Proceedings of the 19th international conference on World wide web (pp. 891-900). ACM.

[13] Feng, Y., Zhengli, S. Yelp User Rating Prediction. `http://cs229.stanford.edu/proj2014/Yifei%20Feng,%20Zhengli%20Sun,%20Yelp%20User%20Rating%20Prediction.pdf`