Sentiment Analysis of Yelp‘s Ratings Based on Text Reviews

Yun Xu, Xinhui Wu, Qinxia Wang

Stanford University

I. Introduction

A. Background

Yelp has been one of the most popular sites for users to rate and review local businesses. Businesses organize their own listings while users rate the business from $1-5$ stars and write text reviews. Users can also vote on other helpful or funny reviews written by other users. Using this enormous amount of data that Yelp has collected over the years, it would be meaningful if we could learn to predict ratings based on review‘s text alone, because free-text reviews are difficult for computer systems to understand, analyze and aggregate [1]. The idea can be extended to many other applications where assessment has traditionally been in the format of text and assigning a quick numerical rating is difficult. Examples include predicting movie or book ratings based on news articles or blogs [2], assigning ratings to YouTube videos based on viewers‘comments, and even more general sentiment analysis, sometimes also referred to as opinion mining.

B. Goal and Outline

The goal of our project is to apply existing supervised learning algorithms to predict a review‘s rating on a given numerical scale based on text alone. We look at the Yelp dataset made available by the Yelp Dataset Challenge. We experiment with different machine learning algorithms such as Naive Bayes, Perceptron, and Multiclass SVM [3] and compare our predictions with the actual ratings. We develop our evaluation metric based on precision and recall to quantitatively compare the effectiveness of these different algorithms. At the same time, we explore various feature selection algorithms such as using an existing sentiment dictionary, building our own feature set, removing stop words and stemming. We will also briefly discuss other algorithms that we experimented with and why they are not suitable in this context.

C. Data

The data was downloaded from the Yelp Dataset Challenge website `https://www.yelp.com/dataset_challenge/dataset`.The Yelp dataset has information on reviews, users, businesses, and business check-ins. We specifically focus on reviews data that includes 1, 125, 458 user reviews of businesses from five different cities. We wrote a Python parser to read in the json data files. We only extract text reviews and star ratings and ignore the other information in the dataset for simplicity. We store the raw data into a list of tuples, where an example tuple is of the form: (“text review“, “star rating“), and star ratings are integers in the range from 1 to 5 inclusive. A higher rating implies a more positive emotion from the user towards the business.

We use hold-out cross validation and run our algorithms on a sample size of 100000. We randomly split this sample set into training (70% of the data) and test (the remaining 30%) sets. We assume that the reviews stored in the json files are randomized in business categories, so we could sample our subsets of size N by simply extracting the first N reviews. Possible improvements in sampling could be done by Bernoulli sampling to reduce possible dominance of training set by certain business categories.

II. Results and Discussion

A. Evaluation Metric

We use Precision and Recall as the evaluation metric to measure our rating prediction performance. Our Oracle is the metadata star rating. We compare our prediction with the metadata star rating to determine the correctness of our prediction. Precision and Recall are calculated respectively by the equations below:

$$Precision = \frac{tp}{tp+fp} \tag{1}$$

$$Recall = \frac{tp}{tp+fn} \tag{2}$$

$$\tag{3}$$

where tp, fp, fn are the number of True Positives, False Positives, and False Negatives respectively.

We record our data as shown in Table 1, where the $(i,j)^{th}$ entry represents the number of actual Rating i being predicted to be Rating j.

Rating	1	2	3	4	5
1	79	80	60	90	50
2	79	80	60	90	50
3	79	80	60	90	50
4	79	80	60	90	50
5	79	80	60	90	50

Table 1: *Illustration of precision and recall calculation.*

Thus in our context, precision and recall of Rating i are calculated by the equations below:

$$Precision = \frac{M(i,i)}{\sum_{j=1}^{5} M(i,j)} \quad (4)$$

$$Recall = \frac{M(i,i)}{\sum_{i=1}^{5} M(i,j)} \quad (5)$$

An additional evaluation metric to consider is runtime of our predictor, which becomes particularly important when the dataset is huge and optimization of runtime becomes necessary, which we will discuss further later.

B. Preprocessing

In our data preprocessing, we remove all the punctuations and all the spaces from the review text. We convert all capital letters to lower case to reduce redundancy in subsequent feature selection.

C. Feature Selection

We implement several feature selection algorithms, one using an existing opinion lexicon, the others building the feature dictionary using our training data with some additional variations [4].

Our most basic feature selection algorithm uses Bing Liu Opinion Lexicon available for download publicly from `http://www.cs.uic.edu/~liub/FBS/opinion-lexicon-English.rar`. This Opinion Lexicon is often used in mining and summarizing customer reviews [5], so we consider it appropriate in our sentiment analysis. It consists of 6786 adjectives in total, where 2006 are positive, 4783 negative. We combine both the positive and negative words and define these words to be our features.

The other feature selection algorithms loop over the training set word by word while building a dictionary that maps each word to frequency of occurrence in the training set. In addition, we implement some variations: (1) Appending "not_" to every word between negation and the following punctuation. (2) Removing stop words (i.e. extremely common words) from the feature set using Terrier stop wordlist. (3) Stemming (i.e. reducing a word to its stem/root form) to remove repetitive features using the Porter Algorithm readily implemented in Natural Language Toolkit (NLTK).

The results of the various feature selection algorithms on the test data are shown in Fig 1. Each column corresponds to precision or recall for Ratings 1 through 5, from left to right. We observe that building a dictionary from the dataset followed by removing stop words and stemming gives the highest prediction accuracy.

The advantage of using an existing lexicon is that there is no looping over the dataset. Also, the feature set consists exclusively of adjectives that has sentiment meaning. The disadvantage is that the features that we use are not extracted from the Yelp dataset, so we might include irrelevant features while relevant features are not selected. For example, many words in the text reviews are spelled wrong, but still contain sentiment information. Using such a small feature set causes the problem of high bias.

Building the feature set using training data results in a larger feature set, selects only relevant features from the Yelp dataset itself, and improves both precision and recall significantly. However, looping over the training set to select relevant features can be slow when our training size becomes large. If we loop over a small training set though, the features selected might have high bias and not representative of the entire Yelp dataset.

A large feature set also has the problem of high variance; in other words, while the training error reduces with a larger training set, the test error remains high. This motivates us to remove stop words (i.e. common words with no sentiment meaning) and use stemming to reduce redundancy in the feature set that we built. This further improves our prediction accuracy by a noticeable margin.

Negation handling by appending "not_" was motivated by putting more information of the sentence context into each word. The results however did not improve. This could be caused by overfitting from adding more features. Since we append "not_" to all the words following punctuation, all the nouns following negation were also processed and added, and such manipulation may generate noise on our testing.

D. Perceptron Algorithm

We consider a review not as a single unit of text, but as a set of sentences, each with their own sentiment. With this approach, we can address our sub-problem on the sentiment analysis of one sentence instead of the whole review text. We use perceptron learning algorithm to predict the sentiment of each sentence,

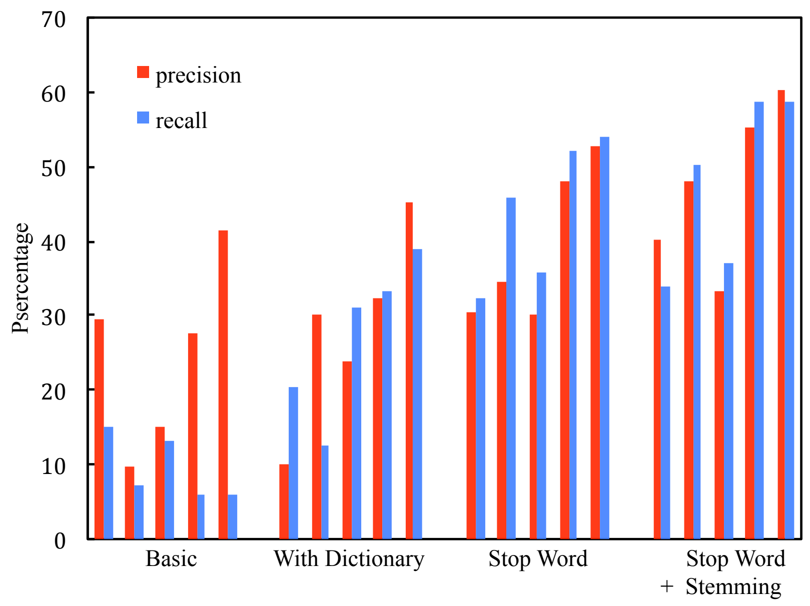

Figure 1: *Comparison of test error for different feature selection algorithms using Naive Bayes.*

where the hypothesis is defined as the following:

$$h_\theta(x) = g(\theta^T x) \tag{6}$$

and g is define to be the threshold function:

$$g(z) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases} \tag{7}$$

We use stochastic gradient descent to minimize the loss function. Each sentence is predicted to be Positive (P) if the hypothesis is computed to be 1 or Negative (N) if the hypothesis is 0. Finally, we compute the star rating for the entire review based on the number of positive and negative sentences in the review:

$$Rating = [\frac{P}{P+N} \times 4] + 1 \tag{8}$$

where P and N are the number of Positive and Negative sentences in the review respectively. The equation above ensures that the rating is scaled in the $[1:5]$ range to be comparable to the metadata rating.

We built the feature set by looping over the training dataset with stop words removed and Porter Stemming and this gives us a total of 39030 weights. The precision and recall for the test set are shown in Table 2.

Rating	Precision (%)	Recall (%)
1	35.6	70.9
2	18.3	18.3
3	20.3	11.5
4	36.2	14.4
5	53.5	76.1

Table 2: *Perceptron algorithm results on test dataset.*

We observe that the precision and recall results are significantly better for Ratings 1 and 5, the two extreme cases. Since we train the features based on only Positive or Negative sentiment (2 categories), it is difficult for our algorithm to predict how positive or how negative the entire sentence is using these features.

Another observation is that the ratings are predicted to be consistently lower than the actual rating. To fix this problem, we scale the predictions to have the same mean and standard deviation as the actual star ratings. However, this did not improve our prediction accuracy. When we trained the weights for the features, we separate the reviews into two groups: 1-3 Star as Positive, 4-5 Stars as Negative. On the other hand, the mean rating is around 3.7. Thus, this manual separation in the training step affects the weights calculated and the rescaling step later might counteract the information that we gained from the training earlier.

E. Naive Bayes

We use the Naive Bayes algorithm in the scikit-learn machine learning library to predict star ratings. Similarly, the features are selected by looping over the training set with stop words removed and Porter Stemming.

Naive Bayes is traditionally used and proved to be the most suitable for text classification. In our Naive Bayes algorithm, we represent a review via a feature vector whose length is equal to the number of words in the dictionary. We use Laplace smoothing to avoid over-fitting. In addition, we implemented a variation of Naive Bayes, i.e. Binarized Naive Bayes using Boolean feature vector. In other words, instead of counting the frequency of occurrence of the words, we use 1 or 0 to denote whether the word occurred or not. This is motivated by the belief that word occurrences may matter more than frequency.

The precision and recall for the training and test set for Binarized Naive Bayes are shown in Table 3 and Table 4.

Rating	Precision (%)	Recall (%)
1	70.1	98.5
2	70.6	95.2
3	83.4	87.9
4	98.9	75.6
5	94.4	88.8

Table 3: *Naive Bayes algorithm results on training dataset.*

Rating	Precision (%)	Recall (%)
1	38.4	32.7
2	45.8	52.2
3	35.3	39.9
4	54.2	57.2
5	58.7	59.1

Table 4: *Naive Bayes algorithm results on test dataset.*

The training error is significantly improved, implying a much lower bias error as compared to the Perceptron Algorithm. Although the precision and recall for the test set are not very high, we observe this is due to the fact that Star 4 and 5 reviews are difficult to be distinguished from each other. same for Star 1, 2, and 3 reviews. For example, more than one third of the Star 4 reviews are predicted to be Star 5 and vice versa. This is expected, because Star 4 and 5 reviews are difficult to be distinguished from each other in the first place. Therefore, if we combine reviews of Star 4 and 5 into one classification category, our prediction accuracy will be significant improved.

F. Other Algorithms

Other algorithms that we have also considered so far are Multi-Class Support Vector Machine (Multi-Class SVM) and Nearest Centroid algorithms. Both were implemented using the scikit-learn machine learning library.

Multi-Class SVM is a generalization of SVM, where the labels are not binary, but are drawn from a finite set of several elements. However, the predictions have extremely low accuracy, even on the training dataset itself. Therefore, we conclude that it is not suitable in the context of sentiment analysis.

The Nearest Centroid algorithm is a classification model that assigns to observations the label of the class of training examples whose mean (centroid) is closest to the observation. In the training step, given labeled training samples $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$ where y_i's are the ratings and x_i's are feature vectors in the high dimensional feature space. The per-class centroids are computed using the formula below:

$$\mu_r = \frac{1}{|C_r|} \sum_{i \in C_r} x_i \tag{9}$$

where C_r is the set of indices of samples that has rating r. In the prediction step, the class assigned to an observation x is computed by the formula below:

$$\hat{y} = arg\, min_{r \in Y} \|\mu_r - x\| \tag{10}$$

However, the precision and recall on the test dataset are found to be low. This is expected, because in many cases, our data are represented in a very high dimensional space with only few components being non-zero. There is little sense of clustering for this model, because when we calculate the average position of the points that are classified as the same group, it might become a point in space that is not close to any of the point in the cluster, but closer to some point that in another group.

We also experimented with the Natural Language Toolkit (NLTK) to tag each word into different group based on parts-of-speech; however, this results in a very low training speed without much improvement on classifying the test data.

G. Comparison of Algorithms

A comparison of precision and recall on the test dataset using different learning algorithms is shown in Fig 2.

Multi-class SVM and Nearest Neighbor both have low precision and recall. Perceptron algorithm has the highest precision and recall for Star 1 and 5 Ratings, but the predictions are poor for Star 2, 3, and 4. It also suffers from high bias on the training dataset. Naive Bayes (binarized) has the best overall performance, but

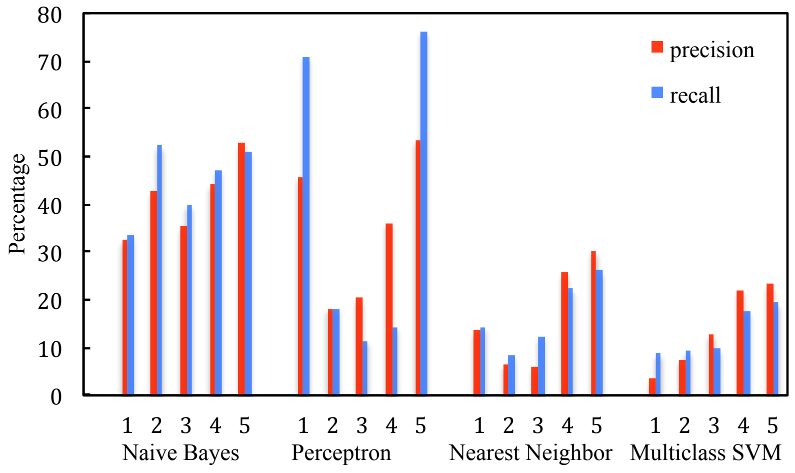

Figure 2: *Comparison of test error for different learning algorithms.*

further error analysis by running the algorithm on different sample sizes shows that it has the problem of high variance. This is evident from the learning curve plotted in Fig 3, where as the sample size increases, the margin between training and test accuracy remains large.

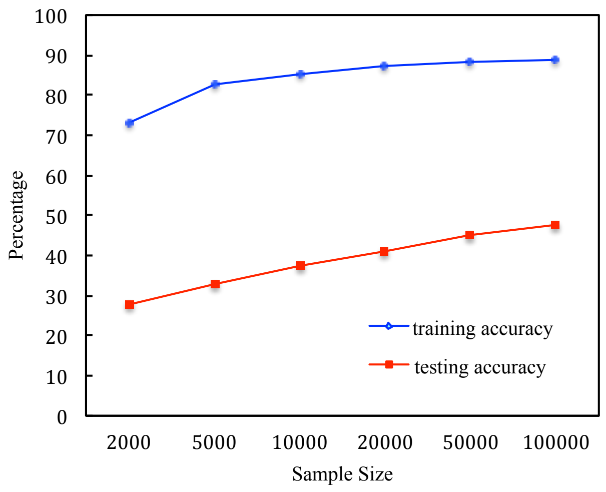

Figure 3: *Learning curve for binarized Naive Bayes algorithm.*

III. Conclusion and Future Work

In conclusion, we have experimented with various feature selection and supervised learning algorithms to predict star ratings of the Yelp dataset using review text alone. We evaluate the effectiveness of different algorithms based on precision and recall measures. We conclude that binarized Naive Bayes combined with feature selection with stop words removed and stemming is the best in our context of sentiment analysis.

Possible improvement could be extracting additional information from the dataset such as Business Categories and use customized feature sets for each Category, because different word features might be more or less relevant in different Business Categories. Runtime of the algorithm could possibly be improved by training and testing within each business category, because of a smaller feature set. We could also try using parts-of-speech in feature selection process to differentiate between the same word features that are used as different parts-of-speech.

References

[1] G. Ganu, N. Elhadad, and A. Marian, "Beyond the Stars: Improving Rating Predictions using Review Text Content.," WebDB, no. WebDB, pp. 1–6, 2009.

[2] N. Godbole, M. Srinivasaiah, and S. Skiena, "Large-Scale Sentiment Analysis for News and Blogs.,"ICWSM, 2007.

[3] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up?: sentiment classification using machine learning techniques," Proceedings of the ACL-02 conference on Empirical methods in NLP, 2002.

[4] K. Dave, S. Lawrence, and D. Pennock, "Mining the peanut gallery: Opinion extraction and semantic classification of product reviews," Proceedings of the 12th international conference on World Wide Web, pp. 519–528, 2003.

[5] M. Hu and B. Liu, "Mining and summarizing customer reviews," Proceedings of the 10th ACM SIGKDD international conference on Knowledge discovery and data mining, 2004.