CS 412: Intro to Machine Learning
Final Report

How Yelpy are Yelpers?

By

The CS412 Yelp Team
Hongwei Jin
Krutarth Joshi
Aayush Kataria
Natawut Monaikul
Ashwin Sattiraju
Zhan Shi
Dan Zhao

Code: `https://github.com/jinhw1989/CS412Yelp`

Contents

1 Introduction

As of 2015, over 89 million people visit Yelp online per month [12], whether it be to leave a review for a business or look up local restaurants to try. With so many users, there is naturally a massive amount of data being generated each day, so when Yelp Inc. released a dataset containing just about everything collected from their website over the past several years, many machine learning tasks immediately presented themselves. So then, given this vast dataset, what kinds of predictions/inferences can we make?

1.1 Background

Yelp is a website on which users can leave and read reviews about businesses, as well as find general information about those businesses, including hours of operation, menu, location, etc. Businesses on Yelp range from restaurants to salons to department stores, even to universities. Yelp also serves as a social media site, as reviewers can create profiles that display their location, favorite places, past reviews, and so on. Yelp's recently-released dataset includes all of this information and more. The main focus of this project is based on the fact that users can leave a star rating (a whole number between 1 and 5) for a business.

1.2 Problem

As with many domains in which one can leave reviews and ratings for something, a common task is to predict how positively or negatively that something will be rated by its audience (as in [9] for movies, [10] for YouTube videos, and [8] for books). Thus, Yelp lends itself to the machine learning task that we would like to address: can we predict the star rating a reviewer will give a business? Making accurate predictions about how a user will rate a business has applications in detecting spam (fake users leaving reviews that are inconsistent with their star ratings), more accurately aggregating user reviews to determine an overall star rating for a business, and recommending other businesses to Yelp users.

2 Methods

Given the large dataset released by Yelp (retrieved from [13]), we must first determine what subset of the data we will focus on to make our predictions. We explored two directions based on the knowledge source we used to approach the machine learning task: a review-based branch and a profile-based branch.

2.1 Review-Based Branch

When a user rates a business, they can also leave a textual review of the business discussing what they did and did not enjoy about the business to justify their star rating. In the Yelp dataset, each review is listed with the user and the business, the text in the review, the star rating given by the user, and the date of the review. In addition to a full review, a user can also leave a "tip" about the business – a short, simple piece of information a user wants to impart on other users to summarize their thoughts or feelings about a business. These tips can be considered as part of the textual review. In the review-based branch, we claim that all of this information is sufficient to predict the star rating a user will give a business.

2.1.1 Features

We extract several features from users' reviews:

- **Sentiment score.** Given a textual review, we would like to score the positivity or negativity of the text. We calculate this score by summing the positivity of negativity ratings of each individual word. There are many ways to determine these word ratings. One method, as described and employed in [5], is to use a semantic network such as WordNet [7], which stores adjectives as bipolar clusters – a word such as *fast* would have an entry with its synonyms (such as *quick*, *swift*, *prompt*, etc.) and its opposing word (*slow*), along with its synonyms. A set of positive words can then be obtained by starting with a "seed" set of positive words and searching all of the neighbors, where all synonyms would also be considered positive words, while all opposing clusters would be considered negative words. This has also led to the development of SentiWordNet [3], a resource for obtaining positivity and negativity ratings of words.

  However, in our system, we would like to minimize the use of external resources. Therefore, to obtain ratings for words strictly using only our training set, we first calculate relative frequencies of each word in five-star reviews and one-star reviews in the training set. The idea behind this is that positive words will appear frequently in the most positive reviews, while negative words will appear frequently in the most negative reviews. Clearly, many words inherently occur frequently, regardless of its positivity or negativity, so we also remove the 500 most frequent words across all reviews in the training set to filter out common words. The relative frequencies of the remaining words then provide "scores" for each word, where positive words receive a score equal to their relative frequencies, and negative words receive a score equal to the negative value of their relative frequencies. A list of some of the words

extracted using this method are given in Table 1. Using these word lists, the sentiment score of a review is simply the sum of the weights over all words in the text that are found in the lists. Not only did this method provide a way to score words, it also exposed some common aspects of businesses that people tend to focus on, for positive or for negative reasons, using words that would otherwise have no connotation of polarity in a general context. For example, *dry* was found to be a commonly used word in negative reviews; however, in its most general sense, *dry* arguably does not necessarily have a positive or negative connotation, but in the context of, say, food, it is generally a negative word.

Positive Words		**Negative Words**	
Word	**Score**	**Word**	**Score**
thank	0.00145	worst	-0.00378
perfectly	0.00134	horrible	-0.00343
easy	0.00132	rude	-0.00313
yummy	0.00129	terrible	-0.00278
reasonable	0.00129	waited	-0.00254
professional	0.00127	poor	-0.00188
attentive	0.00116	problem	-0.00181
glad	0.00112	bill	-0.00176
wow	0.00112	leave	-0.00166
recommended	0.00105	dry	-0.00143

Table 1: Sample positive and negative words and scores

- **Punctuation count.** When people become emotionally excited, whether it be in a positive or a negative manner, their "online language" tends to reflect their emotions [11]. One of the features of this language is the amount of punctuation used. For example, "Very good!!!!" and "Very bad!!!!" contain several exclamation points and reflect a strong, emphasized, emotional statement. Furthermore, punctuation is commonly used to draw emoticons, which can again reflect the polarity of the text.

- **Cap word count.** Another feature presented in [11] in conjunction with excessive punctuation is the use of upper-case letters. For example, "I LOVE it" and "I HATE it" use a completely capitalized word to emphasize the word, indicating an expression of strong emotion.

- **Emphasized word count.** The final character-based feature we consider (again, as proposed in [11]) is the number of words which contain the intentional repetition of letters in a word, e.g., *mmmmmm* and *ewwwwww*.

- **Word count.** A general feature extracted from a piece of text is the number of words in the text. We do not posit a specific relationship between the number of words and the polarity of a review, but we include the feature in hopes that there might be.

- **Adjective count.** The number of adjectives used in a piece of text can also suggest the polarity/neutrality of it [1]. Therefore, we extract as a feature the number of adjectives used in a review with the help of a part-of-speech tagger (provided by Python's `nltk` package).

- **Adverb count.** It is also shown in [1] that the number of adverbs used in a piece of text reflects the writer's sentiment, so using the same method, we also consider this feature.

- **Weekday/Weekend.** It has been found in a study analyzing product reviews in 2014 that weekday purchases receive more positive feedback than weekend purchases [2]. In light of this, we also use the date of a review to determine whether the review was left on a weekday (Monday-Thursday) or a weekend (Friday-Sunday) and use this as our final feature.

2.1.2 Classifiers

We use a variety of classifiers with these features to predict the star rating of a review (with the help of Python's `scikit` package): Decision Tree, Random Forest (with 10 decision trees), AdaBoost (with decision trees as weak learners), 5-Nearest Neighbors, 10-Nearest Neighbors, and 20-Nearest Neighbors. Each classifier predicts one of five labels (the five whole-valued star ratings) for each review in the test set.

2.2 Profile-Based Branch

Another branch is based on profiles, which include both user and business. When guessing the potential rate a user gives to a particular business, it is obvious that the profile of a user and the profile of the business affect the rate associated. For example, a really yelper will leave a pertinent rate to a restaurant of his or her favorite. In this approach, we will combine the profiles of user and business first. And then analysis most relevant features when rating a business. We will try to apply several classifiers to work on the training data and to see how those works.

For this branch, we posit that information from user and business profiles is sufficient for predicting which star rating a user will give a particular business. We draw data from user profiles, which include information about a user, from number of "fans" (other users who like this user's reviews) to "elite" status (Yelp's method of rewarding and distinguishing users who are more active in the community), as well as business profiles, which include information about a business, from hours of operation to whether or not the business has Wi-Fi.

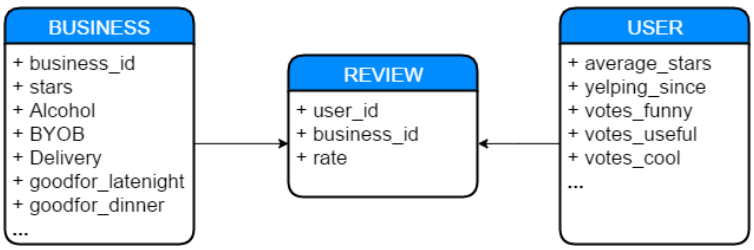

Figure 1

2.2.1 Features

- **Flattening**. Since in the original data is a JSON nested format. In most classifier, we need a flattened data rather than nested format. After expanding the features, the business has 74 features and the user has 20 features, so totally, there are 94 features for each review.

- **Categorizing**. In order to get the data suitable for `scikit-learn pipeline`, we need to modify the data as integers, especially for the data with string format and missing data.

```
{
"business_id": "b9WZJp5L1RZr4",
"categories": [
"Breakfast & Brunch",
"Restaurants"
],
"review_count": 38,
"attributes": {
"Alcohol": "none",
"Ambience": {
"romantic": false,
"casual": true
},
...
}
```

$\Longrightarrow$

```
"b9WZJp5L1RZr4F1nxclOoQ": {
"parking_garage": false,
"attr_upscale": false,
"Accepts Insurance": null,
"Noise Level": "average",
"Takes Reservations": true,
"goodfor_lunch": true,
"Has TV": false,
"Attire": "casual",
"Wheelchair Accessible": null,
"attr_hipster": false,
...
}
```

- **Feature selection**. Since there are so many features and some of them are strong related to the rate, while some of them not. Picking the best features seems more important such that we can reduce

the computation of classifiers. The best top 10 features are: *"average_stars", "stars", "elite", "yelping_since", "review_count", "Has TV", "parking_lot", "Attire", "goodfor_latenight", "BYOB/Corkage".*

2.2.2 Classifiers

In the profile-based branch, we train and evaluate a total of four classifiers. Three of the classifiers are SVM, but with three different kernels (Gaussian, linear, and polynomial of degree 2). The fourth classifier is Random Forest with 10 trees.

3 Evaluation

We evaluated the classifiers from each branch separately (and the recommender system in the profile-based branch).

3.1 Review-Based Branch

Due to the large size of the entire dataset (over 1.5 million reviews), we systematically divided the dataset in our evaluation. Moreover, because a part-of-speech tagger significantly increases the running time of extracting all of our features, we measured our classifiers' performances when they contained and did not contain the adjective and adverb count features separately. We first randomly chose 100,000 reviews from the full dataset, from which we randomly selected 70,000 to be the training set and 30,000 to be the test set. The features not including the adjective and adverb counts were then extracted from the training set, and then the classifiers were trained on these features and evaluated on the test set. The process of randomly choosing and splitting 100,000 reviews and training and evaluating the classifiers was repeated 9 more times, and the results were averaged over the 10 trials.

We then randomly chose 10,000 reviews from the full dataset, from which we randomly selected 7,000 to be the training set and 3,000 to be the test set. The full set of features was then extracted from this smaller training set, and the classifiers were again trained and evaluated. As in the larger dataset, we performed a total of 10 trials. The smaller dataset allowed for the features, specifically the adjective and adverb count, to be extracted in a reasonable amount of time. Separating the dataset like this also allowed us to examine the impact the adjective and adverb count had on classification.

To evaluate the performance of the classifiers, we recorded the general accuracy of each classifier (the percentage of correctly-labeled reviews), the accuracy of each classifier when predicting whether or not a

review is five stars, the accuracy of each classifier when predicting whether a review is one star, five stars, or neither, and the "soft" accuracy of each classifier, in which the classifier is said to have made a correct prediction if the predicted label is at most one star off from the actual label. These types of accuracy allow us to look at different aspects of the classifier – how well did it do in general, how well did it perform on predicting polarized reviews, and how often was it close enough to the actual label? This accounts for the possibility that the features were better suited for detecting polarized reviews (one star or five stars) rather than choosing one of the five class labels, as well as the possibility that many reviewers left reviews that were inconsistent with their final star rating from another perspective. These are important to consider, since it has been found that people online tend to report extreme feelings rather than average feelings, and positive feelings rather than negative feelings [4].

3.2 Profile-Based Branch

3.2.1 Classifier Evaluation

- **Random Forest** A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement.

  In random forests, each tree in the ensemble is built from a sample drawn with replacement (i.e., a bootstrap sample) from the training set. In addition, when splitting a node during the construction of the tree, the split that is chosen is no longer the best split among all features. Instead, the split that is picked is the best split among a random subset of the features. As a result of this randomness, the bias of the forest usually slightly increases (with respect to the bias of a single non-random tree) but, due to averaging, its variance also decreases, usually more than compensating for the increase in bias, hence yielding an overall better model. The `scikit-learn` implementation combines classifiers by averaging their probabilistic prediction, instead of letting each classifier vote for a single class.

- **Support Vector Classifier** A support vector machine constructs a hyper-plane or set of hyper-planes in a high or infinite dimensional space, which can be used for classification, regression or other tasks. Intuitively, a good separation is achieved by the hyper-plane that has the largest distance to the nearest training data points of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier.

SVC is one of the main function in the $scikit-learn$ package. The fit time complexity is more than quadratic with the number of samples which makes it hard to scale to dataset with more than a couple of 10000 samples. And it could apply linear, polynomial, rbf, sigmoid and precomputed kernel functions.

3.2.2 Recommender System Evaluation

In our recommender system, we use a *user-based* recommendation algorithm. We select 500,000 reviews and consider the set of users and businesses involved in those reviews. Considering different users may have different "baselines" around which their ratings are distributed, we use Pearson's correlation coefficient:

$$sim(u,v) = \frac{\sum_{i \in C}(r_{u,i} - \bar{r_u})(r_{v,i} - \bar{r_v})}{\sqrt{\sum_{i \in C}(r_{u,i} - \bar{r_u})^2}\sqrt{\sum_{i \in C}(r_{v,i} - \bar{r_v})^2}}$$

to estimate the similarity between two users, where C is the set of items that are co-rated by users u and v (i.e., items that have been rated by both of them), $r_{u,i}$ and $r_{v,i}$ are the ratings given to item i by the target user u and a possible neighbor v respectively, and r_u and r_v are the average ratings of u and v respectively. Finally we give an estimated rating by

$$p(u,i) = \bar{r_u} + \frac{\sum_{v \in V} sim(u,v) * (r_{v,i} - \bar{r_v})}{\sum_{v \in V} sim(u,v)}$$

where V is the set of k similar users, $r_{v,i}$ is the rating of user v given to item i, $\bar{r_u}$ and $\bar{r_v}$ are the average ratings of user u and v respectively, and $sim(u,v)$ is the Pearson correlation described above. For efficiency, we assign k as 100, i.e, the first 100 top-related users, and then extract 20% of the ratings from the rating matrix as test data, leaving 80% of the ratings as the base for calculating similarity.

4 Results

4.1 Review-Based Branch

The results for the general accuracy of each classifier is given in Figure 2. We see that the highest accuracy achieved is 43.49% by AdaBoost. We also see that, on average, the classifiers performed better when the features of adjective and adverb count were excluded.

The `scikit` package also provided a way of ranking the features by their importance when using Random Forest – the more useful a feature was in making decisions (from the amount of variability in the feature), the

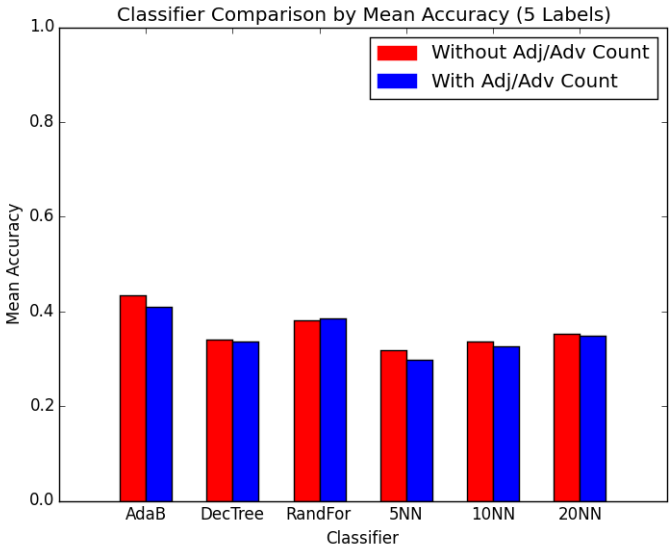

Figure 2

higher the importance. It was found that the most important feature was the sentiment score, while the least important feature was the emphasized word count. Furthermore, the features of adjective and adverb count ranked fourth and fifth, which is unsurprising given the poorer classifier accuracy resulting from including those features.

We then explored the low classifier accuracies, specifically for AdaBoost, to see the distribution of correct and incorrect labels. Figure 3 shows this distribution, plotting the frequencies of predicted-actual label pairs. The larger the point (x, y), the more reviews labeled y were predicted to be of class x. The points in red represent correctly-predicted reviews, as the predicted label matches with the actual label. From this, we can see that the classifier performed very well for five-star reviews (and was very close on four-star reviews), but also that the classifier predicted five stars for a large portion of reviews. This may also reflect an imbalance in the dataset. This imbalance can be seen in the star distribution of one of the datasets in Table 2. Since each dataset was generated using the same process, this distribution is representative of each dataset. Clearly, the majority of the reviews in the dataset are five- and four-star reviews, which sheds light on the classifier performance in Figure 3. This distribution may also be unsurprising given the finding in [4] that people online tend to report positive emotions rather than negative emotions.

Table 2: Star Distribution for One Dataset

Rating	1	2	3	4	5
Percentage	10.29%	8.98%	14.20%	29.44%	37.09%

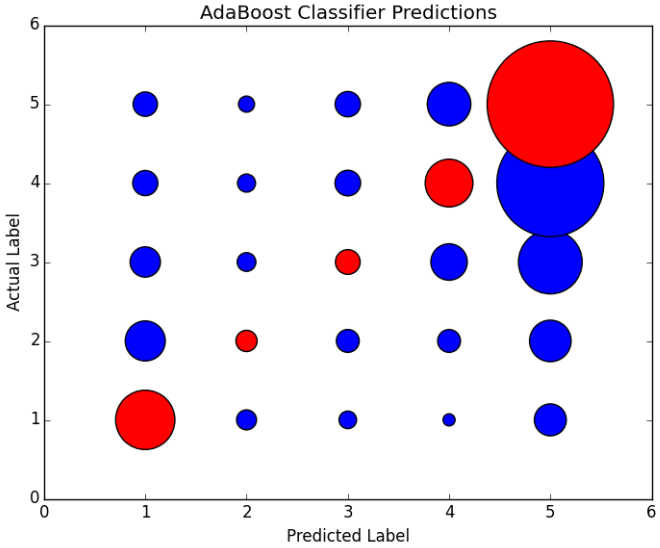

Figure 3

Figure 4 shows the mean performance of the classifiers in determining whether or not the review is polarized (one- and five-star reviews vs. neither) on the left and the mean performance of the classifiers in determining whether or not the review is a five-star review on the right. Again, the inclusion of the adjective and adverb count features makes little difference in classifier performance; however, it did slightly increase the performance of the best classifier in this task – Random Forest, which correctly predicted whether or not a review received five stars 59.99% of the time. We can also see that the classifiers, on average, performed better at determining whether or not a review received five stars than determining whether or not a review was polarized, which may again be a manifestation of the star distribution in the dataset.

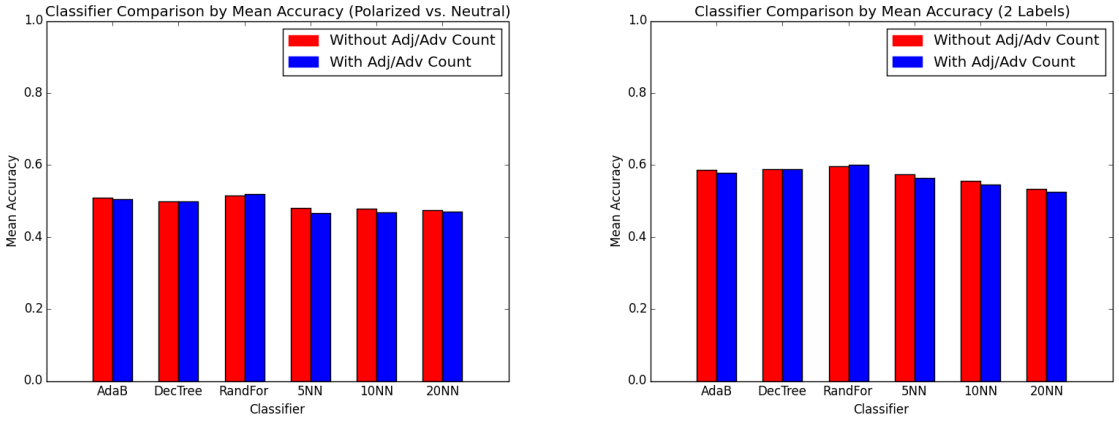

Figure 4

Finally, Figure 5 shows the mean "soft" accuracy of each classifier – how often the classifier was at most one star away from the actual star rating of a review. AdaBoost performed the best again, correctly (and almost correctly) predicting the labels of 78.61% of the reviews in the test set. Of course, we expect a higher accuracy in this task than in the original task of strictly correct label predictions, but we note that the performance is still better than the expected performance with random label assignments. If each review has a 20% chance of being in one of the five classes, and if each review is randomly assigned a label as a prediction (each of which has a 20% chance), then the expected "soft" accuracy is 52%, so our classifiers still outperformed random assignment.

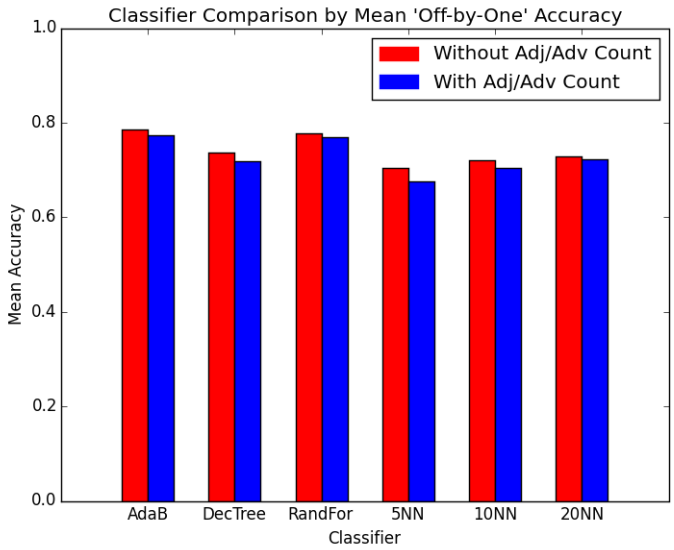

Figure 5

4.2 Profile-Based Branch

- RandomForest (all reviews)
  - number of trees in the forest: 10
  - minimum number of samples required to split an internal node: 1
  - number of jobs is set to the number of cores: 4
  - result: average cross validation score: 0.385 (3-fold)
- SVC (support vector classifier) (10,000 reviews) [1]

[1] Due to the SVC function limitation in scikit-learn

– Gaussian kernel: gamma (0.1), c(1)

– Gaussian, 45

– Linear c=1, 43

– Polynomial degree 2, 44

– result: average cross validation score: 0.44 (3-fold)

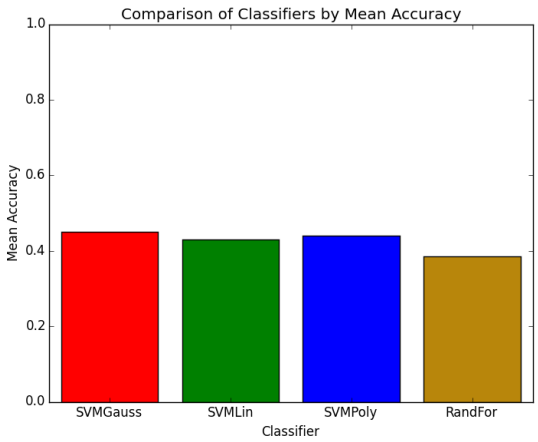

Figure 6

5 Conclusion

5.1 Summary

The results from both branches show that using the features described in Section 2, our classifiers exhibited a mediocre but promising performance. The "soft" accuracy reported in the review-based branch and the root mean squared error in the profile-based branch suggest that, while our classifiers were not extremely accurate, they were generally very close to predicting the actual labels, at least above expected from random assignment. Moreover, each of the classifiers used performed with about the same accuracy as the other classifiers in each task.

5.2 Lessons Learned

On a low level, we were able to become much more familiar with packages in Python, specifically `nltk`, `scikit`, and `crab`, that greatly increased our productivity in this project. We were also able to gain

experience in cleaning and dividing data appropriately for our tasks. We learned that NLP techniques do not necessarily increase classifier performance and that more features do not necessarily imply greater accuracy. Furthermore, the results from our experiments have shown us that in classifying reviews, positive and negative words are very important, and if we are to continue in another direction, this feature must be considered at the very least.

5.3 Future Work

In future work, we would like to find ways of improving the base accuracy of our classifiers. One possible direction is to consider combining the best and most important features from both branches to have a better feature vector for the classifiers. On the review-based side, we may also want to consider using a topic modeling scheme, such as LDA, to have different quantitative measures on each word, as we found that word frequency in positive and negative reviews played a large role in classification. This direction has shown promise in [6], authored by a grand prize winner of the Yelp Dataset Challenge – the motivation for this project. In our approach, we only considered frequent words in one-star and five-star reviews, but there may be some information in word frequencies in more neutral reviews that may help the classifiers perform better on non-polarized reviews. We also noted an imbalance in the dataset, which is something we may account for in the future by choosing a more proper dataset.

References

[1] Benamara, F., Cesarano, C., Picariello, A., Reforgiato, D., and Subrahmanian, V. Sentiment analysis: Adjectives and adverbs are better than adjectives alone. In *International Conference on Weblogs and Social Media - ICWSM* (2007).

[2] Digital Strategy Consulting Ltd. Online review habits: Weekday purchases get better feedback. `http://www.netimperative.com/2015/03/online-review-habits-weekday-purchases-get-better-feedback/`, 2015.

[3] Esuli, A., and Sebastiani, F. SENTIWORDNET: A publicly available lexical resource for opinion mining. In *Proceedings of the 5th Conference on Language Resources and Evaluation (LREC06)* (Genova, Italy, 2006), pp. 417–424.

[4] Guerra, P. C., Jr., W. M., and Cardie, C. Sentiment analysis on evolving social streams: How self-report imbalances can help. In *Proceedings of the 7th ACM WSDM Conference* (2014).

[5] Hu, M., and Liu, B. Mining and summarizing customer reviews. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD-2004)* (Seattle, Washington, USA, 2004).

[6] Linshi, J. Personalizing yelp star ratings: A semantic topic modeling approach. In *Yelp Dataset Challenge*. 2013.

[7] Miller, G. A. WordNet: A lexical database for English. *Communications of the ACM 38*, 11 (1995), 39–41.

[8] Mooney, R. J., and Roy, L. Content-based book recommending using learning for text categorization. In *Proceedings of the fifth ACM conference on Digital libraries* (2000), pp. 195–204.

[9] Oghina, A., Breuss, M., Tsagkias, M., and de Rijke, M. Predicting imdb movie ratings using social media. In *Advances in Information Retrieval*, vol. 7224. Springer Berlin Heidelberg, 2012, pp. 503–507.

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

[11] Yassine, M., and Hajj, H. A framework for emotion mining from text in online social networks. In *2010 IEEE International Conference on Data Mining Workshops* (Genova, Italy, 2010), pp. 1136–1142.

[12] Yelp Inc. An Introduction to Yelp Metrics. `http://www.yelp.com/factsheet`, September 2015.

[13] Yelp Inc. Yelp Dataset Challenge. `http://www.yelp.com/dataset_challenge`, 2015.