Assignment 1: Classification of Review Sentiments

Peter Chen
Princeton University
xi@princeton.edu

Austin Williams
Princeton University
aw18@princeton.edu

Abstract

Sentiment analysis of this data has a wide range of applications. We build various sentiment classifiers beginning with multinomial and Bernoulli Naive Bayes classifiers that classify a text as either positive or negative. We evaluate these methods on 2,400 (training) and 600 (testing) labeled sentences, processed into bag-of-words representations of each post. We found that logistic regression had the best performance amongst our models, while other more complex techniques worsened performance.

1 Introduction

In this assignment, we build a sentiment classifier that classifies a text as either positive or negative. Sentiment analysis of this sort has a wide range of applications, such as monitoring social media or tracking customer reviews. We begin by using a Naive Bayes classifier, with both Bernoulli and Multinomial distributions.

To extend the assignment, we will also use logistic regression, bigrams, and GloVe [11] word representations. Lastly, we apply the Bernoulli and Multinomial Naive Bayes classification techniques to the Twitter dataset containing 1.5M tweets with positive/negative labels, randomly sampling the testing/training data sets.

Some inherent difficulties with such a process include short reviews, non-standard English, or ironic and sarcastic language.

2 Related Work

”Feature Selection and Weighting Methods in Sentiment Analysis” by Tim O’Keefe and Irena Koprinska implements similar techniques to solve similar problem of textual sentiment classification [10]. We also use various techniques to process the text data [7] [11] and implement the classifiers via scikit-learn [3][4][2].

3 Methods

3.1 Multinomial Naive Bayes Classifier

We begin with 3,000 text reviews, each given a classifier (1 or 0 indicating postive or negative respectively) and split into 2,400 training and 600 testing datasets. We use the training sample to build our classifier. First, we must process both datasets as thus.

We create a dictionary of tokenized (broken up into pieces without special characters such as punctuation) words from the training data with the default word count threshold of 5; in other words, the dictionary will contain every word that appears at least 5 times in the entire training data. This dictionary will exclude stop words, or commonly used words such as ”the” or ”and” that we choose to ignore because we assume they provide no indication of sentiment, as given by the nltk corpus. Furthermore, each token is lemmatized using the Word Net Lemmatizer, which reduces various versions of a word to a single version, and stemmed using Porter's algorithm, a heuristic for chopping off the ends of words (both performed via nltk [1]). Then, for each review, create a a bag-of-words representation using this dictionary; in this case, each review is represented via a vector where each element represents the number of occurrences of a word in the dictionary. The same bag-of-words representation is also created for the testing dataset using the training dictionary. [7]

Now, we begin building the classifier, starting with Bayes' Theorem:

$$P(C|x) = \frac{P(C)P(x|C)}{P(x)} \propto P(C)\Pi_i P(x_i|C),$$

where C is the class (1 or 0) and x_i is the frequency of word i. To train the classifier, we calculated $P(C = 1)$, $P(C = 0)$, and the conditional probabilities of each word w given class c maximum likelihood estimate (with smoothing) as $P(w|c) = (N_{wc} + 1)/(N_c + V)$. Now we have everything needed to predict our testing data. For any bag-of-words vector (for example [1, 1, 3, 0, 0, 0]) we just use the conditional probabilities of the training data to construct our probability according to Bayes' Theorem (using the same example, $P(C = 1|[1, 1, 3, 0, 0, 0]) \propto P(C = 1)P(w_1|C = 1)P(w_2|C = 1)(P(w_3|C = 1)^3)$. [8] This is implemented using scikit-learn [4].

3.2 Bernoulli Naive Bayes Classifier

We follow the same process outlined above for the Multinomial Naive Bayes Classifier, with the only difference being the bag-of-words representations being binary instead of frequency. In other words, instead of each element in the vector representing the frequency of a word, it's a 1 for if the word appears at all and 0 otherwise.

Now, we are able to build the classifier using the procedure described above. To train the classifier, we calculated $P(C = 1)$, $P(C = 0)$, and the conditional probabilities of each word w given class c maximum likelihood estimate (with smoothing) as $P(w_i|c) = (N_i + 1)/(N_c + 2)$. Now we have everything needed to predict our testing data. For the now binary bag-of-words vector (for example [1, 1, 1, 0, 0, 0]) we just use the conditional probabilities of the training data to construct our probability according to Bayes' Theorem (using the same example, $P(C = 1|[1, 1, 1, 0, 0, 0]) \propto P(C = 1)P(w_1 = 1|C = 1)P(w_2 = 1|C = 1)(P(w_3 = 1|C = 1)P(w_4 = 1|C = 1)P(w_5 = 1|C = 1)(P(w_6 = 1|C = 1))$. [8] This is implemented using scikit-learn [3].

3.3 Logistic Regression Classifier

We follow the same procedure as the Bernoulli Naive Bayes Classifier mentioned above to get the binary vectors. The difference here is that instead of using the same classifier, we pass the vectors into scikit learn's LogisticRegression model [2].

The logistic regression model maximizes the likelihood of observing the sample values we use the train the classifier:

$$P(C = 1|x_1, ..., x_k) = \frac{exp(\beta_0 + \beta_1 x_1 + ... + \beta_k x_k)}{1 + exp(\beta_0 + \beta_1 x_1 + ... + \beta_k x_k)}$$

where the x's represent the bag-of-words from the training the data (the explanatory variable in the regression), C is the corresponding classification, and β's are coefficients derived from maximum likelihood estimators of the model. [5]

3.4 Extending the Dataset

We proceed to apply the Multinomial and Bernoulli Naive Bayes classifiers on the Twitter Sentiment Analysis Training Corpus (from thinknook.com [9]) containing 1,047,568 labeled tweets. We separate testing and training datasets by shuffling the rows and randomly 2400 tweets for training and 600 tweets for testing (we keep the testing and training files the same for all classifications). We then preprocess and train the classifiers in the same manner described above.

3.5 More Interesting Features

We then different features to better predict sentiment. First, we try bigrams. We again use a binary bag-of-words vector, but the ones and zeroes represent occurrences of pairs of words instead of single words. To account for bigrams being less common than each individual word, the threshold for inclusion in the vocabulary is lowered to two. We apply logistic regression to the resulting vectors. Next, we create a binary vector where the vocab consists of both the bigrams and the individual words. Again, the threshold was lowered to two for this model and use logistic regression. Finally, the other type of feature we used was Global Vectors for Word Representation [11]. In this model, the words in each text was translated into its vector representation, and those vectors were summed together to get a vector representation of the text. We then trained a logistic regression classifier on the summed vectors. These models were all implemented using scikit learn.

4 Results

We apply the following formulas to find measures of performance (results for the Multinomial Naive Bayes Classifier are shown):

Accuracy = (Correct classifications)/(Total number of samples) = 0.77
Precision = TP/(TP+FP) = 0.79
Recall = Sensitivity = True Positive Rate = TP/(TP+FN) = 0.7596153846153846
Specificity = True Negative Rate = TN'(FP+TN) = 0.78125
False Positive Rate = FP/(FP+TN) = 1-Specificity = 0.21875
F1 score = 2*precision*recall/(precision + recall) = 0.7745098039215687

These calculations are repeated for the other models. The results are summarized in Tables 1 and 2. ROC curves are generated using scikit learn and matplotlib. The resuliting curves for the classifiers are pictured in Figures 1 through 8.

Table 1: Model Performance

	Accuracy	**Precision**	**Recall**	**Specificity**	**False Positive Rate**	**F1 Score**
Multinomial NB	0.77	0.79	0.76	0.781	0.218	0.775
Bernoulli NB	0.773	0.8	0.759	0.789	0.211	0.779
Logistic Regression	0.81	0.877	0.774	0.858	0.142	0.822
Bigrams	0.567	0.96	0.537	0.813	0.188	0.689
Monograms + Bigrams	0.78	0.823	0.758	0.807	0.193	0.789
GloVe + Logistic Regression	0.785	0.827	0.763	0.811	0.189	0.794
Twitter + Multinomial NB	0.662	0.693	0.624	0.702	0.298	0.657
Twitter + Bernoulli NB	0.68	0.675	0.652	0.706	0.294	0.663

Table 2: Confusion Matrices for the Classifiers

	True Positive	**False Positive**	**False Negative**	**True Negative**
Multinomial NB	237	63	75	225
Bernoulli NB	240	60	76	224
Logistic Regression	263	37	77	223
Bigrams	288	12	248	52
Monograms + Bigrams	247	53	79	221
GloVe + Regression	248	52	77	223
Twitter + Multinomial NB	194	86	117	203
Twitter + Binomial NB	189	91	101	219

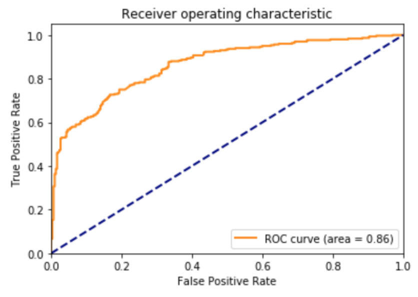

Figure 1: Multinomial NB Classifier ROC

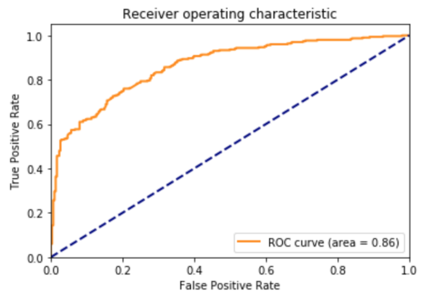

Figure 2: Bernoulli Naive Bayes Classifier ROC

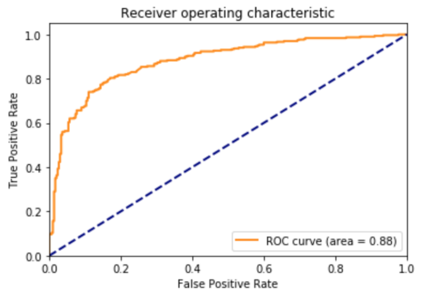

Figure 3: Logistic Regression ROC

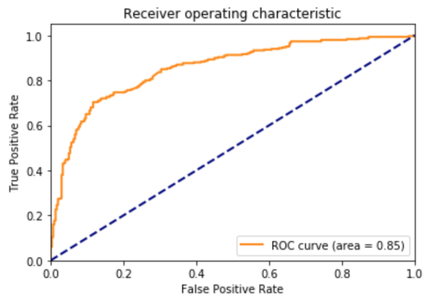

Figure 4: GloVe + Regression ROC

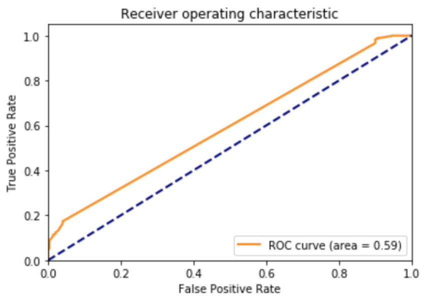

Figure 5: Bigrams ROC

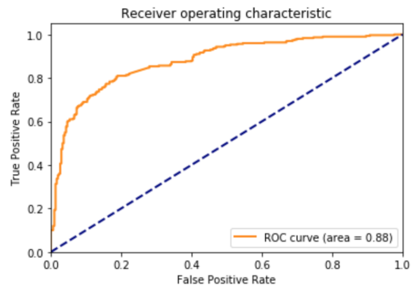

Figure 6: Bigrams + Monograms ROC

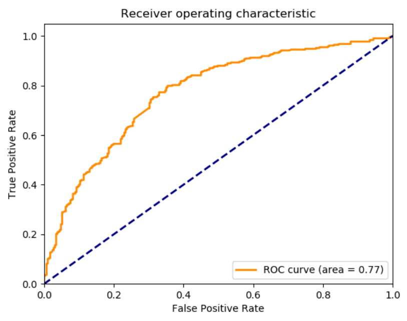

Figure 7: Twitter Multinomial

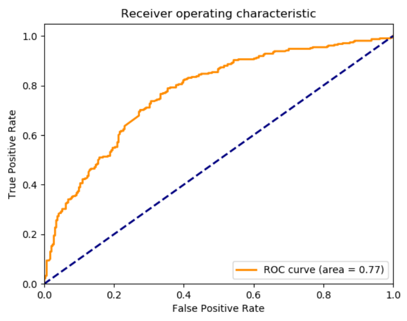

Figure 8: Twitter Binomial

5 Discussion and Conclusion

We look at the following metrics to interpret the effectiveness of our classifiers [6] [10]:

- **Accuracy:** tells us the percentage of correct classifications out of the entire testing data.
- **Confusion Matrix:** displays the number of true positives (1 correctly classified as 1), false positives (0 incorrectly classified as 0), false negatives (1 incorrectly classified as 0), and true negatives (0 correctly classified as 0). Overall, we always seek to minimize false classifications, however under different circumstances, false positives (otherwise known as type 1) may be less desirable than false negatives (type 2) and vice versa. Assuming that we are testing for whether sentiment is positive (the null hypothesis that a review is negative), where it is more potentially more serious to falsely classify a negative review than to falsely classify a positive one, a false positive is less desirable than a false negative, in which case we assume the ”status quo.”
- **Precision:** the amount of correct classifications among all reviews classified as positive, so assuming we are testing for whether sentiment is positive, we especially want a high precision for our classifiers.
- **Recall:** the amount of correct classifications among actually positive reviews.
- **Specificity:** the amount of correct classifications among actually negative reviews.
- **False Positive Rate:** the amount of incorrect classifications among actually negative reviews.
- **F1 score:** a combo-score considering both precision and recall.
- **Receiver Operating Characteristic (ROC) Curve:** plots the false positive rate versus the true positive rate at varying prediction thresholds. We want the curve to approach perfect classification in the top left, whereas the diagonal (y=x) line indicates complete randomness. If the curve skews towards the left, it indicates good early retrieval (and poor if skewed upward). We also want a higher area under curve (AUC), which equals the probability that the given classifier will rank a random positive sample higher than a random negative sample.

Under the assumption that we are testing for positive sentiments (and thus in particular trying to minimize the number of negative sentiments classified as positive), we will focus especially on precision in addition to accuracy. The base logistic regression model is the best of our classifiers, almost across the board. The only metric that it does not have the best performance at is precision, but the only model that beats it essentially guesses a label of 1 every time, and has very low accuracy. This base logistic regression model was also tied for the highest area under the curve in the ROC curves. It seems safe to say that logistic regression was the best overall model out of this group.

Unexpectedly, using more sophisticated techniques, such as adding bigrams or word embeddings, worsened performance in our testing (the monograms + bigrams worked better because the dictionary now contains the most relevant instances of both). Perhaps this added complexity is not beneficial due to the small size of the dataset. Interesting future work would include trying similar analysis on larger and different datasets to see if those results are consistent. Furthermore, cross-validation would provide a more dataset and more reliable metrics.

Acknowledgments

Prof. Barbara Engelhardt, Dr. Xiaoyan Li, who provided the lectures for COS 424

References

[1] “Natural Language Toolkit NLTK 3.2.5 documentation.” [Online]. Available: https://www.nltk.org/

[2] “sklearn.linear_model.LogisticRegression scikit-learn 0.19.1 documentation.” [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

[3] “sklearn.naive_bayes.BernoulliNB scikit-learn 0.19.1 documentation.” [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.BernoulliNB.html

[4] “sklearn.naive_bayes.MultinomialNB scikit-learn 0.19.1 documentation.” [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html

[5] C. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[6] T. Fawce, “Roc graphs: Notes and practical considerations for data mining researchers,” 2003.

[7] E. Loper and S. Bird, “Nltk: The natural language toolkit,” 2002.

[8] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

[9] L. Naji, “Twitter Sentiment Analysis Training Corpus (Dataset).” [Online]. Available: http://thinknook.com/twitter-sentiment-analysis-training-corpus-dataset-2012-09-22/

[10] T. OKeefe and I. Koprinska, “Feature selection and weighting methods in sentiment analysis,” 2009.

[11] J. Pennington, R. Socher, and C. D. Manning, “GloVe: Global Vectors for Word Representation.” [Online]. Available: https://nlp.stanford.edu/projects/glove/