Programming assignment 3C : Text classification on restaurant reviews

Sarvesh Meenowa
University of Gothenburg
CY Tech (Exchange student)
gusmeesa@student.gu.se

Neha Devi Shakya
University of Gothenburg
CY Tech (Exchange student)
gusshane@student.gu.se

Khushi Chitra Uday
University of Gothenburg
CY Tech (Exchange student)
guschikh@student.gu.se

Abstract

This project aims to create a text classifier that can classify restaurant reviews as positive or negative reviews. Bernoulli Naïve Bayes, Multinomial Naïve Bayes and Linear Support Vector Classification (Linear SVC) algorithms were selected to develop the text classifier. Our data was collected from review websites such as Yelp through crowdsourcing. Our best performing LinearSVC model achieved a precision-recall trade-off of 0.97-0.97. In the end, we have provided potential ways to improve the performance of our model.

1 Introduction

For years, restaurants have operated under the assumption that providing good food and service is the best way to attract new customers. However, technological advancement, particularly the data generated by online platforms, has pointed to fresh discoveries and opened new avenues. Nowadays, most consumers rate and publish evaluations online, and many trust internet reviews. Customers and restaurant owners can interact through review services such as Yelp and TripAdvisor.

This project aims to create a text classifier that can classify restaurant reviews as positive or negative reviews. The task is formulated as a supervised learning problem for which the model was trained and tested using the data crowd sourced and annotated by the students.

Our classification method is based on the machine learning algorithms Bernoulli Naïve Bayes, Multinomial Naïve Bayes and Linear Support Vector Classification (Linear SVC). We aim to find which of these algorithms gives the best results.

2 Methods

2.1 Data collection and annotation

In order to collect data, over 100 students enrolled in the course of DAT340/DIT867 crowdsourced the data from various restaurant review websites such as Yelp, and TripAdvisor, among others. Each student collected 50 positive and 50 negative restaurant reviews to create a balanced dataset, bringing the total number of reviews collected to 8769. For the first round of annotation, positive reviews were annotated as 1, and negative reviews were annotated as 0. In the second round of annotation, each student received back 100 reviews to annotate; positive and negative reviews were annotated similarly to the first round, except this time, if the student is unsure, the review is annotated as -1. In order to estimate the inter-annotator agreement, we calculated the Cohen's Kappa and obtained 0.89, which means that as a rule of thumb, the agreement is "really good".

2.2 Data cleaning and pre-processing

2.2.1 Review Cleaning

1. The content of all the reviews was converted to lower case.
2. Contractions were removed from all words not to have issues when we remove punctuations. E.g., "wasn't" was changed to "was not".
3. All emojis were removed from the reviews.
4. Single spaces replaced all multiple spaces.
5. All URLs, punctuation marks and @ symbols were removed.
6. All numbers were removed, so we only had text data.

7. All words left in the reviews were lemmatized. Lemmatization was used over stemming as it considers the context of the word in the sentences. Lemmatization also consistently gives the dictionary meaning word while converting into root form and is recommended when the meaning is vital for analysis. For example: for "change", stemming would give "chang", whereas lemmatization would give "change" as the root form.

2.2.2 Annotation cleaning pre-processing

Annotations were in the form of 'Annotation from second round (Annotator 2) / Annotation from first-round (Annotator 1)', i.e. '0/1'. We split the annotation into two columns: one for annotator 1 and the second for annotator 2. There were few wrongly formatted annotations such as : '2/1', '2/0', '1/', '9/1'. Since there were only six wrongly formatted annotations, we manually re-annotated them. Unsure annotations refer to annotations with -1 or disagreements such as '0/1' and '1/0'. There were 265 disagreements which we decided to drop. For annotations with '-1's, to have a third opinion, we used the Stanza library to perform sentiment analysis on the reviews and kept the reviews that agreed with the first annotator and dropped the rest.

2.3 Data representation

In order to use machine learning algorithms on texts, the unstructured data must be transformed into numerical vectors, each vector representing a review. Each vector's entries represent the presence of a feature(word) in that review. We used the Term Frequency-Inverse Document frequency (TF-IDF) vectorizer. The TF-IDF score helps balance the weight of the most frequent words against less commonly used words, resulting in a sentence term matrix with one row per review and single unique terms represented in the columns. Instead of representing a count, the cells make up a weighting used to determine the importance of a word in a given text. From Table 1, we show an example of how few features (aback, abdul, ability) are represented for six randomly chosen reviews.

	aback	abdul	ability
1017	0.050	0	0
1349	0	0	0.137
1509	0	0.398	0
1565	0	0.391	0
3390	0	0	0.183
4712	0.093	0	0

Table 1: Representation of features under TF-IDF vectorizer with the weights for each feature for a particular review.

2.4 Model Training

2.4.1 Selection of model

The dataset was split into 80-20, 80% of the data for training and 20% for testing. In order to select which algorithms to use, we researched commonly used algorithms in the context of text classifications: Multinomial Naive Bayes, Bernoulli Naive Bayes, Linear SVC and deep learning algorithms. However, for this task, we only considered machine learning algorithms. In order to have a baseline model, we used DummyClassifier with the strategy "most frequent", which means that it will predict the most frequent class in the dataset.

Naive Bayes classifier: It is a probabilistic classifier that relies on the Bayes theorem while assuming significant feature independence. This classifier has the advantage of just requiring a small quantity of training data to determine the parameters for prediction. Rather than computing the complete covariance matrix, because of the features' independence, just the feature's variance is computed. The conditional probability for each class(positive, negative), 'c' given a review, 'r' is P($r|c$) and calculated as follows:

$$P(r|c) = \frac{P(c|r)P(c)}{P(r)}$$

Linear SVC(Support Vector Classifier) is a binary linear classifier that is non-probabilistic. Each review is represented as a data point in space using the Linear SVC model. This method is used to examine all vectorized data, and the main idea behind the model training is to locate a hyperplane represented by $\overrightarrow{W}$. The separation of the collection of textual data vectors is considered optimum by a hyperplane only when separated

without error. The distance between the closest points of each class and the hyperplane is the greatest.(Tripathy et al., 2015)

2.4.2 Hyperparameter tuning

One of the essential parts of a machine learning pipeline is hyperparameter tuning. A faulty selection of hyperparameter values may result in erroneous results and a model with poor performance. There are several methods for tuning hyperparameters. In our case, we used GridSearch from the sklearn library in Python.

Grid search is a hyperparameter tuning algorithm that divides the hyperparameter domain into a discrete grid. Then, using cross-validation, it tries every possible combination of values from this grid, calculating some performance metrics. The optimal combination of the hyperparameters is the grid point that maximizes the average value in cross-validation.

We used the "roc_auc" scoring metric to measure the performance of the models. This metric computes the Area Under the Receiver Operating Characteristic Curve (ROC AUC) from prediction scores. We used the default 5-fold cross-validation (cv = 5), used all processors (n_jobs = -1) to carry out the GridSearch.

2.4.3 Evaluation

We chose our primary evaluation metric to be AUC(area under the ROC curve) to evaluate the model. We also used various other metrics such as accuracy, precision and recall. Our dataset is balanced, so just using accuracy would be sufficient; however, we wanted to investigate how the model performs with respect to false positives and false negatives.

2.4.4 Feature importance

After selecting the best model, it is possible to view which features(words) the model views as most important when classifying positive and negative reviews. To do so, we retrieved the best coefficients and feature names from the best performing model presented by GridSearchCV.

3 Results

After performing GridSearchCV for the selected classifiers, we obtained the following results as shown in Table 2.

From Table 2, we can also see that the best performing model was Linear SVC on the training set with an roc_auc score of 0.994.

	Optimal Parameters	AUC
Bernoulli NB	alpha = 0.1	0.959
Multinomial NB	alpha = 0.5	0.991
Linear SVC	C = 1, dual = False	0.994

Table 2: Optimal parameters obtained after performing GridSearchCV evaluated by AUC(Area under ROC curve).

We then evaluated the performance of the trained model using their best parameters on the test set as shown in Table 3.

	AUC	Accuracy	Precision	Recall
Dummy	0.50	0.51	0.00	0.00
Bernoulli NB	0.96	0.86	0.92	0.80
Multinomial NB	0.99	0.95	0.94	0.95
Linear SVC	0.99	0.97	0.97	0.97

Table 3: Performance results on test set using AUC(Area under ROC curve), accuracy, precision and recall.

Linear SVC and Multinomial Naive Bayes had the best AUC score(0.99); however, Linear SVC performed better in other metrics such as accuracy (0.97), precision (0.97) and recall (0.97). Overall, Linear SVC was the best performing model for which we created a confusion matrix(Figure 1) to investigate its errors further.

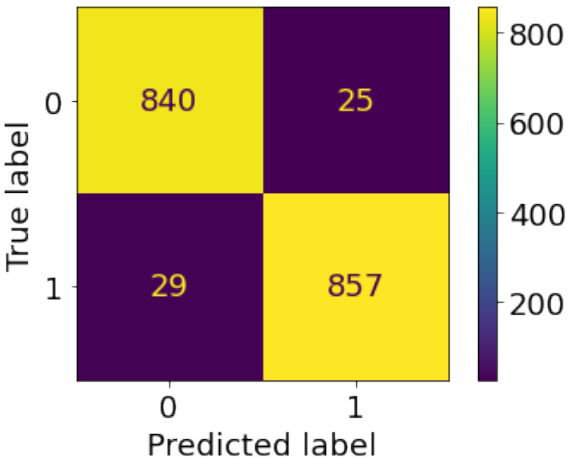

Figure 1: Confusion matrix for Linear SVC model

The model has 857 true positives, 840 true negatives, 29 false negatives and 25 false positives. It means that 29 reviews were misclassified as positive instead of negative, and 25 were misclassified

as negative instead of positive. Hence they had the same recall (0.97) and precision (0.97). We then tried to find an example where a review was misclassified to understand what went wrong. The review : "They don't have a chef and they just warm the foods because they're already cooked it and it comes in packaged when you order they just warm the food and topped with oil to look nice" was misclassified as a positive review.

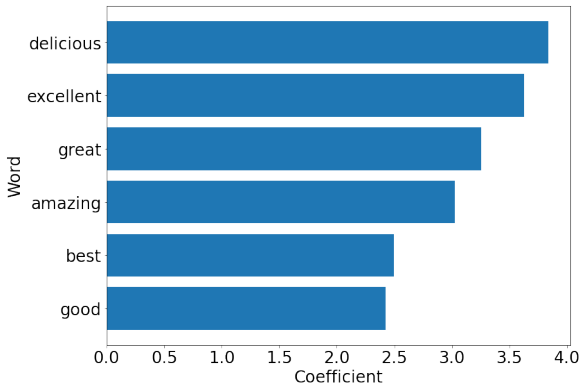

Figure 2: Feature importance for positive class

Regarding the feature importance of the positive class, the higher the coefficients, the more important the features. For the positive class (Figure 2), the six most important features (words) are 'delicious', 'excellent', 'great', 'amazing', 'best', and 'good' respectively.

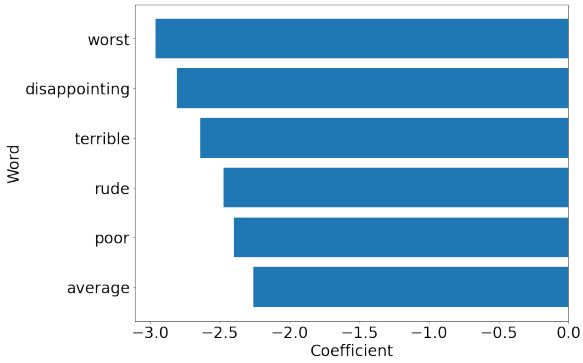

Figure 3: Feature importance for negative class

Opposite to the positive class, the lower the coefficient, the more important the feature for the negative class. For the negative class (Figure 3), the six most important features (words) are 'worst', 'disappointing', 'terrible', 'rude', 'poor', and 'average' respectively.

Overall, we saw words with positive connotations being most important for the positive, and similarly, for the negative class, words with negative connotations were considered most important.

4 Discussion

Linear SVC performed the best and had a very good precision-recall (0.97-0.97) trade-off. When we analyzed the misclassifications, we found that some negative reviews contain words of positive connotation which might mislead the model to wrongly predict the class. Furthermore, we observed that the accuracy depended a fair amount on the pre-processing of data. For example, we decided to drop rows where the annotator 2 did not agree with annotator 1 ('0/1' and '1/0') since it gave a better precision and recall without significantly impacting the distribution of classes compared to when we set the final annotation as the annotation from round 1. We also observed that removing stopwords gave worst results than keeping them, which might be because removing stopwords made the reviews lose their true meaning. For instance, if we consider the review "The food is not good", removing stopwords will remove 'the', 'is' and 'not' therefore, the new review will be "food good", hence losing its true meaning. The performance could be further improved by tuning the TF-IDF vectorizer and performing spell checks to resolve spelling mistakes.

5 Conclusion

We successfully implemented a text classifier using our proposed model, which uses a machine-learning algorithm called Linear Support Vector Classifier (Linear SVC). We achieved a precision-recall trade-off of 0.97-0.97 and an accuracy of 0.97. However, the preprocessing of the reviews and annotations can be improved in the ways discussed above. The Linear SVC model performed relatively better as shown above but one could also argue that a Neural Network algorithm could give better results but it comes with longer training and prediction times.

References

Abinash Tripathy, Ankit Agrawal, and Santanu Kumar Rath. 2015. https://doi.org/https://doi.org/10.1016/j.procs.2015.07.523 Classification of sentimental reviews using machine learning techniques. *Procedia Computer Science*, 57:821–829. 3rd International Conference on Recent Trends in Computing 2015 (ICRTC-2015).

Appendix

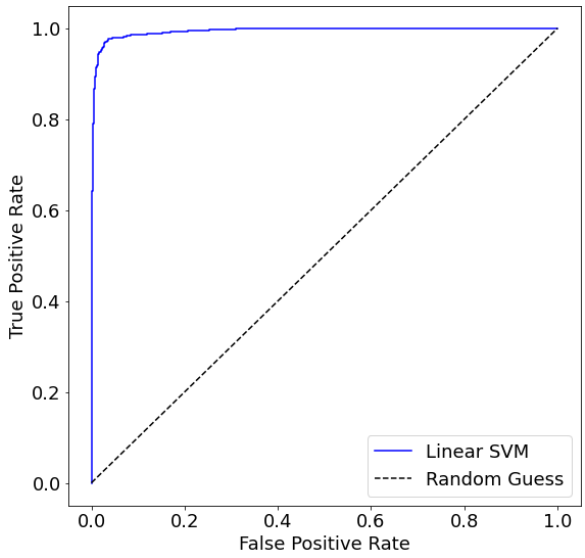

Figure 4: ROC Curve for Linear SVC model