Text Classification using

Multinomial

Naive Bayes Classifier

William Lee

CSC 447 Machine Learning

Professor Erik K. Grimmelmann

Table of Contents

Introduction

Text classification is using concepts from Natural Language Processing (NLP) to classify text into categories based on the content of the text. A common text classification is the Naive Bayes classifier that classifies text based on the probabilities of the categories it is being classified as. Text classification has a wide range of applications such as spam detection for emails, filtering news through categories using artificial intelligence, vertical search engines such as Google and Bing that bring up the most relevant search results, and much more. In this project, text classification would be applied to classify whether the text is a positive or negative review.

Preparing Dataset

Here we will be using a dataset from Kaggle of TripAdvisor hotel reviews that were categorized ranging from five stars down to one star. The dataset's star rating is replaced with a positive and negative rating because reviews are subjective and there might not be much of a difference between five-star and four-star reviews as well as one- and two-star reviews. The reviews are classified as good if they had a five to three stars rating, and the rest are classified as bad. Before the dataset can be used to train the text classification model, it needs to first be cleaned.

```
import re
import string

def clean_text(text):
    text = text.lower()
    text = re.sub(r'[^A-Za-z0-9 ]+', '', text) # removes any non alphanumeric values but keeps whitespace
    text = re.sub('\[.*?]', '', text) #remove symbols
    text = re.sub('[%s]' % re.escape(string.punctuation), '', text) # remove puntuation
    text = re.sub('\w*\d\w*', '', text) #remove numbers
    text = re.sub('['''""...]', '', text) #further removes symbols
    return text
```

Figure 1. Function used to clean the text in the dataset, code is from https://github.com/adashofdata/nlp-in-python-tutorial/blob/master/1-Data-Cleaning.ipynb

The code shown in figure one was used to clean the reviews within the dataset to remove any unnecessary text that does not contribute to the weight of the classification. First, the text is all converted to lower case so that the classification would not be case sensitive. Then it uses the regular expression library to look for various patterns in the text that would be removed. The text had a lot of letters with an accent and random symbols so non-alphanumerical values were removed to ensure that only English text is used to classify. Follow that is a series of additional cleaning to ensure that punctuations, numbers, and symbols were further removed from the reviews such that only words remain.

```
#Splittitng the data into training and testing set
from sklearn.feature_extraction.text import CountVectorizer

sentences = data_clean['Review'].values
y = df['Rating'].values

sentences_train, sentences_test, y_train, y_test = train_test_split(sentences, y, test_size = 0.20)
```

Figure 2. Used sklearn to split dataset into training and testing data for the Classification model

After cleaning the dataset, it is split into training and testing datasets. It was decided that twenty percent of the dataset was used for testing the model as that seemed to yield the best results. Now that the dataset is cleaned and split it is ready to be tokenized and normalized before it can be used in the Naive Bayes classification model.

Tokenizing & Normalizing Dataset

With the dataset prepped and cleaned we can now apply feature extraction via the CountVectorizer function from sklearn library. CountVectorizer creates a bag of words which creates a list of vocabulary of all unique words that appear in the dataset or also known as corpus. The parameter stop word was applied to the function to remove stop words such as “the, so, and, etc.”. Stop words are commonly used in a language but do not have much meaning hence, they are removed from the corpus so that the tokenization is performed on words that

provide information towards categorization. In the process of creating a bag of words, the unique words from the corpus are tokenized. Each word is given a unique index and that index holds the term frequency (number of occurrences) of that word in each review.

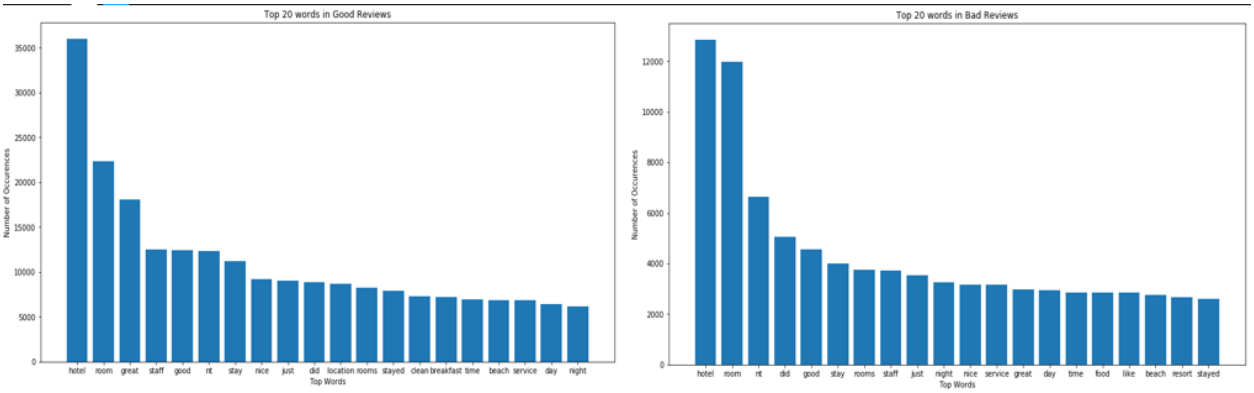

Figure 3. Graph of Top 20 words from good and bad reviews

Upon analysis of the top words from reviews of both categories as shown in figure two, hotel is the top word on both lists and both lists contain the word nt as well. From this sample of the top words, we can see that there are a lot of words that do not contribute much in terms of determining whether the review is good or bad, but they hold a high term frequency. This will cause the model to be skewed as a considerable amount of weight is given to these high term frequency words however, this can be solved by normalizing the data.

To normalize the data the term frequency-inverse document frequency (TF-IDF) library was used. The TF-IDF value provides weight to each word in the corpus which signifies the importance of the word in each document or in this case a review. TF-IDF is computed using the formula by multiplying the term frequency by the inverse document frequency (TF-IDF = Term Frequency (TF) * Inverse Document Frequency (IDF)). Term frequency is defined as the measure of the frequency of a word in a document. Term frequency can depend on the size of the document, a common term may have a higher probability of appearing in a larger document than a smaller document, but it does not necessarily mean that the larger document holds more

importance. Therefore, to normalize the frequency we need to divide it by the total number of words in the document as shown in the equation below.

$$TF\ (term, document) = \frac{count\ of\ term\ in\ document}{number\ of\ words\ in\ document}$$

With the term frequency value computed the data still has some problems. As examined previously on the top words, frequently reoccurring words will be given a lot of weight in classifying which will lead to producing bad results thus, normalizing the weight will provide a better representation of the corpus' classification. Document frequency is the measure of importance of a document in the whole corpus by counting the number of documents in which a term exists (df(term) = occurrence of a term in documents). The inverse document frequency (IDF) measures the informativeness of a term. For a word that appears frequently such as stop words, the IDF value will be low as given in the formula.

$$IDF\ (term) = log(\frac{N}{df+1})$$

Here N is the number of documents in the corpus and we take the log to reduce the value if the value N is large which in this dataset is over twenty thousand reviews. The document frequency is summed by one to smooth out the value and to prevent dividing by zero.

$$TF\text{-}IDF\ (term, document) = \frac{count\ of\ term\ in\ document}{number\ of\ words\ in\ document} * log(\frac{N}{df+1})$$

Finally, we multiply term frequency (TF) with inverse document frequency (IDF) to get the TF-IDF value.

Naive Bayes Classifier

Naive Bayes is a probabilistic classifier that applies the Bayes theorem to compute the probability of each class with various attributes in order to predict a classification. Bayes Theorem states that the probability of an event A happening given another event B is equal to the

probability of event B to occur given event A multiplied by the probability of event A happening all divided by the probability of event B happening as shown in the equation below.

$$P\ (A \mid B) = \frac{P(A \cap B)}{P(B)}\ \&\ P\ (B \mid A) = \frac{P(A \cap B)}{P(B)}\ \textit{therefore,}\ P(A \mid B) = \frac{P(B|A)\ P(A)}{P(B)}$$

Here the Naive Bayes classification model will use Bayes Theorem to compute two probabilities to make its prediction. The model will apply Bayes Theorem to compute the probability of a good review given a review and compare it to the probability of a bad review given the same review. The output with the higher probability will be the classification the model uses to predict for that review, and it repeats the process for the rest of the corpus.

Multinomial Naive Bayes is a form of Naive Bayes, where the features are generated from a multinomial distribution. A multinomial distribution is a generalization of a binomial distribution where it computes the probability of a given set of events to occur.

$$P(X) = \frac{X!}{X1!*X2!*\ldots*Xk!} * p_1^{x1} * p_2^{x2} * \ldots * p_k^{xk}$$

$$\textit{Given that } x1 + x2 + \ldots + xk = x\ \&\ p_1 + p_2 + \ldots + p_k = 1$$

The multinomial equation shown above allows the Bayes theorem to consider a feature vector where a term represents the number of times it appears hence, it is good for using it in text classification. In the multinomial equation, k is the number of events that occur, the X's represents each event, X is the total number of occurrences of all the events, and the p's represent the probabilities of each event to occur. In terms of text classification, the events will be the vocabulary of words for a given review. For example, lets compute the probability of a good review given a sentence with the application of the multinomial equation on top of Bayes Theorem. Let a sample review be "The hotel stay was good." called A, then Bayes Theorem states that P (good review | A) = P (A | good review) * P (good review) / P (A). Here the probability of a good review and probability of review A are constant but multinomial

distribution is applied to the probability of review A given that it's a good review. This probability is computed by broking review A up to each word. In reference to the multinomial equation, each word in the review is the X's and the P's is the probability of that word appearing in a good review which considers the IF-IDF values as explained prior. By applying the multinomial equation to compute the probability in the Bayes Theorem it provides a better representation of the features as opposed to another form of Naive Bayes such as Gaussian which assumes that the features follow a normal distribution.

Results

After training the model with sklearn's built in Multinomial Naive Bayes classifier the test dataset is applied for the model to predict the reviews to be good or bad. The accuracy of the model came out to be around seventy-six percent accurate which is quite good.

Accuracy: 0.7611612588436204

	precision	recall	f1-score	support
bad	1.00	0.04	0.08	1024
good	0.76	1.00	0.86	3075
accuracy			0.76	4099
macro avg	0.88	0.52	0.47	4099
weighted avg	0.82	0.76	0.67	4099

Confusion Matrix

	bad	good
bad	45	979
good	0	3075

Figure 4. Results of the Multinomial Naive Bayes Classifier

Figure four shows the accuracy of the model, the metrics classification report, and the confusion matrix. In the metrics classification report, precision is the number of correct results divided by number of total returned results. Recall is the number of correct results divided by the number of correct results that should have been returned. F1 Score decribes the balance between the precision and the recall value via the formula, $F1\text{-}Score = \frac{2*precision*recall}{precision + recall}$, where the best value is one and the worst value is zero. Support is the number of occurences of each class in the

testing dataset. As shown in figure four, the recall value for bad reviews is very low but when you look at the good review the model was very accurate in predicting good reviews. Upon further analysis of the dataset used to train and test the model, it turned out that the dataset was overfitting. Reviews that were rated five stars comprised of roughly fourty-four percent of the entire dataset which resulted good reviews to be around seventy-three percent and bad reivews being only twenty-six percent of the dataset. This caused the dataset to be overfitting towards good reviews and the results agree. From the confusion matrix we can see that the model rarely predicts the reviews to be bad but rather most of the time it predicts a given review to be good. Part of this way be due to the data not being cleaned enough as things like reviewer's typos were not accounted for but it is evident that the dataset was overfitting for the model that was trained.

After concluding that the dataset was overfitting, the dataset was reclassified to make it so that only five star reviews were good reviews and any review with four stars and below was a bad review. This provided a more balanced dataset where good reviews comprised of fourty-four percent of the dataset and bad reviews sumed up to fifty-five percent.

Accuracy: 0.687972676262503

	precision	recall	f1-score	support
bad	0.65	0.94	0.77	2248
good	0.84	0.38	0.52	1851
accuracy			0.69	4099
macro avg	0.74	0.66	0.65	4099
weighted avg	0.74	0.69	0.66	4099

Confusion Matrix

	bad	good
bad	2115	133
good	1146	705

Figure 5. Results of the Multinomial Naive Bayes Classifier with the altered dataset

Ensuing the training of the new model, figure five displays the results as shown above. Although the accuracy of the model dropped to sixty-eight percent accuracy, the model is providing better predictions as it is no longer classifying the majority of reviews under one category. The confusion matrix shows that it is providing a better distribution of predictions than the earlier

model in spite of the new model predicting reviews to be bad more than it is good. This may be due to bad reviews includes four and three-star reviews which are not bad ratings so they may share common words used in five-star reviews.

Conclusion

Text classification with Naive Bayes classifier had several benefits and drawbacks. Naive Bayes algorithm is a commonly used text classifier due to its simplicity, low computation cost, and works well with high dimensional data such as text to determine whether the review was good or bad. However, the algorithm becomes less accurate when the independent assumption of the feature does not hold as well as when very small probabilities occur in the calculations.

The Kaggle dataset of hotel reviews also contributed tremendously to how the text classification model behaved. The majority of the reviews in the dataset were positive causing the model to predict mainly reviews to be good. This is shown in the results through the metrics and confusion matrix and even though changing the classification to be less fitting it then runs into the problem of not having enough difference between good and bad reviews.

In order to improve the text classification model, there needs to be a few changes to be implemented. First, the dataset will have to contain more negative reviews (one- and two-star reviews) to make the dataset less overfitting. With an influx of more negative reviews, the model's training will be exposed to more bad reviews and use that to more accurately classify reviews between positive and negative. Next is to address typos in reviews during the cleaning process. If the dataset is not clean then during the tokenization stage, the bag of words will contain words that do not resemble an English word. This was seen as top words in both good reviews and bad reviews had the word, "nt" occur a lot in the reviews but it does not mean anything and most likely was due to a typo but its weight is skewing the calculation in Naive

Bayes algorithm. With a cleaner dataset, the model will have more precise weights as it will ideally be classifying with words that contribute to its category (good versus bad) hence, providing possibly higher accuracy.

With the discrepancy from the dataset and cleaning process, more work needs to be done for the model to be not biased. As a result, it is unlikely with the current dataset that the model can effectively predict with great accuracy, whether a given review has a positive or negative sentiment.

References

- Brownlee, Jason. “Classification Accuracy Is Not Enough: More Performance Measures You Can Use.” *Machine Learning Mastery*, Machine Learning Process , 21 Mar. 2014, machinelearningmastery.com/classification-accuracy-is-not-enough-more-performance-measures-you-can-use/#:~:text=F1%20Score,0%2B0)%20or%200.
- Paulo, Sao. “Trip Advisor Hotel Reviews.” *Kaggle*, 30 Sept. 2020, www.kaggle.com/andrewmvd/trip-advisor-hotel-reviews.
- Scott, William. “TF-IDF for Document Ranking from Scratch in Python on Real World Dataset.” *Medium*, Towards Data Science, 15 Feb. 2019, towardsdatascience.com/tf-idf-for-document-ranking-from-scratch-in-python-on-real-world-dataset-796d339a4089.
- Solanki, Gautam. “Multinomial Naive Bayes Classifier Algorithm.” *GreatLearning*, 20 Sept. 2020, www.mygreatlearning.com/blog/multinomial-naive-bayes-explained/.
- Zhao, Alice. “NLP In Python Tutorial.” *GitHub*, 26 July 2018, github.com/adashofdata/nlp-in-python-tutorial/blob/master/1-Data-Cleaning.ipynb.