A Report for Sentiment Analysis Project

Eric John (IMT2017016)
Kaushal Mittal (IMT2017024)
Srihari (IMT2017045)
International Institute of Information Technology

Under Prof. G Srinivasaraghavan
and Prof. Neelam Sinha

Contents

1 Intro

Analyzing sentiment is an important task and has numerous applications. For example, by analyzing a product review we can understand how a customer perceives a product. Similarly for a movie review, analyzing the comment/ review can help us understand a majority opinion about the movie. Although, previously work has been done on sentiment analysis, we try to build a custom model from scratch with a small dataset. [1]
We show comparative results with the popular python library TextBlob, where our model outperforms it on the available validation set. Our model is built to determine sentiments for texts from diverse domains, for example movie reviews and product review.

1.1 Problem Statement

Given a comment or review or statement, predicts for whether the sentiment of the comment or review or statement is positive or negative.

2 Data Description

The dataset we used was collected by Kotizias et. al[2]. The data comes from three different website/fields: imdb.com, amazon.com and yelp.com. It was extracted from reviews of products, movies, and restaurants.It contains sentences labelled with positive or negative sentiment.

3 Preprocessing

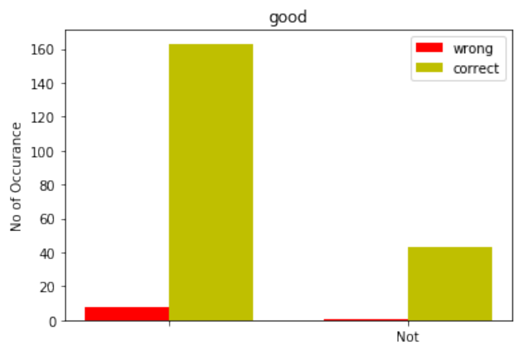
3.1 Data Accumulation

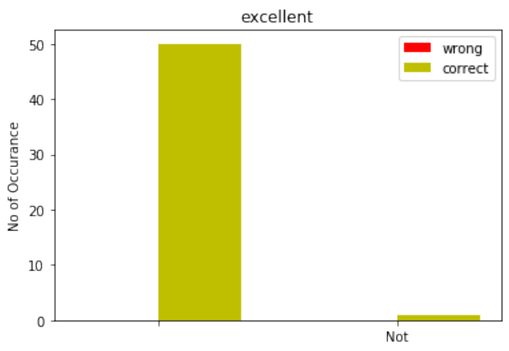
Since we use data from three different sources, we initially concatenate the data into a single array of sentences. We use the delimiters to split the data into an array of sentences. Since the data is in tab separated format (sentence vs label) we separate the sentence and the corresponding labels.

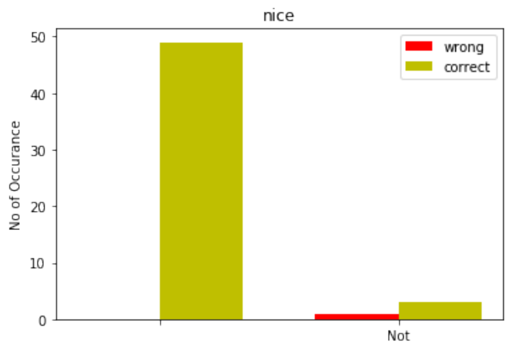
3.2 Data Sanitation

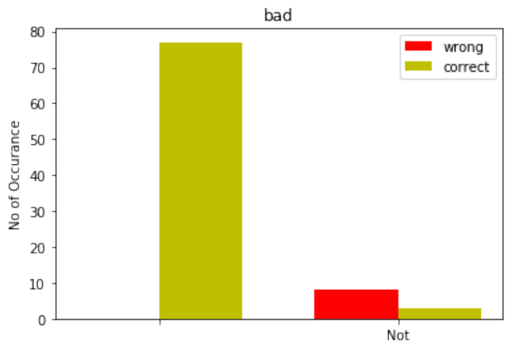
We plan to build a simple model using bag of words approach as our intention is to build a model with small available dataset. We came to this conclusion as we tried building a RNN with LSTM which did not perform well as expected due to small dataset.

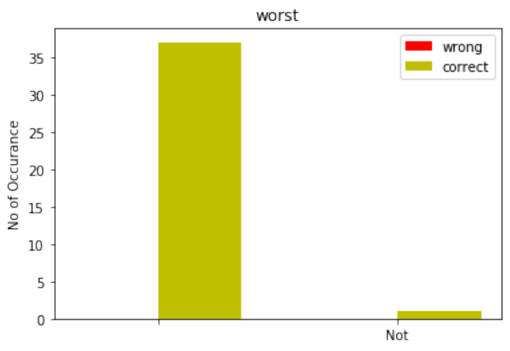
As the first step we clean the text by removing punctuation and irrelevant symbols using TextBlob library[3]


good
wrong
correct
No of Occurance
0
20
40
60
80
100
120
140
160
Not


excellent
wrong
correct
No of Occurance
0
10
20
30
40
50
Not


nice
wrong
correct
No of Occurance
0
10
20
30
40
50
Not


bad
wrong
correct
No of Occurance
0
10
20
30
40
50
60
70
80
Not


worst
wrong
correct
No of Occurance
0
5
10
15
20
25
30
35
Not

Since any NLP problem involves understanding text at a word level, we tokenize the sentences into tokens. Next since the stop words such as, ’a’, ’an’, ’the’, ... carry no explicit meaning (especially when using bag of words where ordering of words does not matter) and also very high frequency we remove such words. For this we use NLTK[4] library.

Also, we use lemmatization to convert each word into its root form. We avoid using stemming as it crudely chops off the end of the words, sometimes resulting in meaningless words.

We split the data into train and test set in the ratio 85:15 ie. 85 percent for training and 15 percent (450 instances) for testing.

3.3 Vectorization

Since we require a fixed length representation of each sentence, we use TF-IDF bag of words approach and create a sparse matrix corresponding to all the sentences. Each word appearing in a particular sentence corresponds to a ’1’ in the corresponding columns of the sparse matrix.

The TF-IDF (Term Frequency - Inverse Document Frequency) vectorizer first creates a vocabulary of words. Gets the terms frequency in the given document and multiplies it with inverse document frequency.

4 Model

4.1 Algorithm description

Naive Bayes is a family of probabilistic algorithms that take advantage of probability theory and Bayes’ Theorem to predict the tag of a text. It assumes that the features are independent to each other ’Naive’.

We use Multinomial-Naive Bayes’ model. We call the built-in model from SciKit-Learn[5] library for python.

5 Evaluation

We compare the predicted labels with the ground truth and give a 0/1 score. Our model score 85.11% accuracy on the test data. We compare our this with the accuracy of TextBlob[3] library which scores 76%.

6 Conclusions

We observe that our model, since it was trained for the specific task, outperforms general purpose library. We show that it is feasible to train a model which is task specific as our model can be trained with small dataset unlike other complex neural network.

References

[1] https://drive.google.com/open?id=1TohSU3ukqzY92ZOyHQEhgYcXuONLDwQX.

[2] Dimitrios Kotzias, Misha Denil, Nando de Freitas, and Padhraic Smyth. From group to individual labels using deep features. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 597–606, New York, NY, USA, 2015. ACM.

[3] https://textblob.readthedocs.io/en/dev.

[4] https://www.nltk.org/.

[5] https://scikit-learn.org/stable/.