**Author:** Mahesh Gaya

**Project #5** – Final assignment

Social Media Sentiment Analysis

I ran a text classification experiment on social media data retrieved from Kaggle. The data primarily contains sentences retrieved from blogs. The link for the data is <https://inclass.kaggle.com/c/si650winter11>.

There are two files from the website. One is the training data with 7086 sentences and the second one is the test data with 33052 sentences. The training data is labeled with 0 and 1 for each sentence. 0 means a negative sentiment and 1 means a positive sentiment. The test data is unlabeled, which is why I ignored the test data for this experiment. I used only the training data for this experiment. I am going to reference this training data as data for the next few paragraphs.

In this experiment, I am trying to predict what sentences are positive and which ones are negative. First of all, I did some clean up on the data. I had to convert the original .txt file into a .csv file for it to work with the Pandas library. I also added “Result” and “Message” as column names for the data so that it is easier to reference them using Pandas. Next, I also ran the data into BeautifulSoup to make sure that there are no HTML tags in the data. I also used Pickle to tokenize the data and ran the data into word2Vec to convert the tokens into vector data.

The classifier algorithms works best if the data is in some sort of vectors. There are millions of words in the English Language. It is very difficult to come up with an algorithm that can deal with all the words specifically. This is why it is important to convert those sentences into some sort of vectors.

Next, I ran several machine learning algorithms on the data to predict the sentiments of each sentence. I split the data into a training list and a testing list. I used 20 percent of the data for the testing list and I ran each algorithm 20 times to get an average accuracy. From the big spectrum of algorithms, I used the Gaussian Naïve Bayes algorithm, Bernoulli Naïve Bayes algorithm, and Support Vector Machine. I also tried PCA, and GridSearch. But I did not go forward with those as they took too long to process. Below are the results for the three algorithms I mentioned before:

|  |  |  |
| --- | --- | --- |
| Algorithm | Parameters | Average Accuracy |
| Gaussian Naïve Bayes | Default | ~0.79 |
| Bernoulli Naïve Bayes | Default | ~0.84 |
| Support Vector Machine | Kernel=”linear”, degree=3 | ~0.93 |
| Support Vector Machine | Kernel=”poly”, degree=1 | ~0.57 |
| Support Vector Machine | Kernel=”poly”, degree=2 | ~0.57 |
| Support Vector Machine | Kernel=”rbf”, degree=3 | ~0.63 |

Some algorithms performed well while others performed poorly. The Naïve Bayes algorithms did better than most of the kernels in Support Vector Machine. On average they had about 80 percent accuracy. However, the best algorithm is Support Vector Machine with Linear kernel. Its accuracy is 0.93 which is the highest accuracy of all the algorithms I tried. This experiment performed best as linear because there are only two outcomes: 0s and 1s. This is why Support Vector Machine with Linear kernel performed the best.