Dustin Casey

CS 167 Write-up

12/11/2016

**Project 5: Text Classification Write-Up**

**The Data Set:**

I decided to use a data set I found searching through Kaggle (found at URL <https://www.kaggle.com/c/sentiment-analysis-on-movie-reviews/data?sampleSubmission.csv>) and I used the data train.tsv. This data is a TSV file that contains phrases and their sentiment, along with sentence Id’s so the coder can track which phrases belong to each sentence. I only used the phrase and the sentiment to train and test my code. The data contains phrases and sentiments of movie reviews from Rotten Tomatoes, and I decided to use this data after trying to use Twitter data of the Presidential Debates. The Twitter data proved to be very tough to use due to hashtags and links that were attached to the tweets, and the Rotten Tomatoes data had more phrases and therefore more data to use for training. The data had around 130,000 different phrases with sentiments ranging between negative, neutral, and positive. Using these phrases, my goal was to find the sentiment of each individual words/phrases and predict the sentiment of words/phrases that are entered into the learning algorithm.

**Steps to format the data to be suitable for doing machine learning:**

After selecting my data, I read in the train.tsv file using a tab delimiter and created my data attribute. I created an all\_data attribute that was only 20,000 of the phrases so the program wouldn’t take hours to run. I created an edited\_data array to place my edited data into. I ran a for loop for the 20,000 data entries to edit the data. I used BeautifulSoup to break up the phrases and then removed all of the characters that were not letters from the data. I changed all of the words to lower case and split all of the phrases into individual words. I then created a non\_stopwords array and ran a for loop to remove stop words and to store the words from the phrases that were not stop words. After removing the stop words, I joined all of the individuals words back together into phrases and appended the phrases to the edited\_data array. I used a count vectorizer to find the best 5000 words from the edited\_data list. After using the vectorizer, I converted the word\_column list into an array to use in a learning algorithm.

**Machine learning algorithms, Naïve Bayes, Multi-Layer Perceptron, and Support Vector Machines (Kernels: ‘Linear’, ‘Poly’, and ‘Rbf’):**

After cleaning and converting my data, and sorting the words using a count vectorizer, I wanted to find which machine learning algorithm would perform the best in predicting the sentiment of the test phrases. I started by splitting my data into train\_data, test\_data, train\_target, and test\_target using cv.train\_test\_split(word\_columns, data[“Sentiment”][0:20000], test\_size = 0.2). I first tested Multinomial Naïve Bayes (MNB) to see how well it would predict sentiment. Using this sample size of data, MNB predicted sentiment at an average of 62% over 10 tests. I next tried using a Multi-Layer Perceptron Classifier (MLPC) and this learning algorithm took the longest to run and provided the second most accurate results with an average accuracy of 63% over 10 tests. I then moved on to testing Support Vector Machines using kernels for linear, poly, and rbf. Linear accuracy was 57%, while poly accuracy was the best with a 67% over 10 tests. Rbf was the worst out of all of the learning algorithms with 56%. For the SVM algorithms, I used parameters of C = .01 and gamma = 10. I came to these parameter numbers based on trial and error and found these to be the best. I also attempted to use a grid search to find the best kernel, C, gamma, and degree, but it took 8 hours to run the grid search on only 1,000 pieces of data and jupyter crashed when I tried to run it with 10,000 pieces of data.

**Conclusion and General Insights:**

Overall, using Multinomial Naïve Bayes proved to be the second most accurate while SVM polynomial kernel proved to be the most accurate. SVM linear and rbf kernels did not perform as well on the data. I gained many insights into sentiment analysis from this project and one the biggest insights I gained is that it is better to have a lot of data for training the learning algorithms. Another important insight I learned is that cleaning the data and removing stop phrases is integral for training the learning algorithms. This is because if I didn’t clean the data, the learning algorithms would be very inaccurate and would be learning sentiment for words and phrases that were not important. I also believe my data didn’t work as well for this project because there was a lot of data and this makes it harder to clean. I tried my best to remove stop words and phrases from the data, but with the vast amount of different phrases in the data, it is nearly impossible to remove all redundant and unimportant words/phrases. I also believe that the computing capabilities of my laptop restricted me to only use about one fifth of my data, and this also was a limitation that my algorithms had when it came to training them.