***[Tai Tran, 5162-4453]***

**CIS4930 Individual Coding Assignment**

**Spring 2023**

1. **Problem Statement**

Reading a person’s intent and underlying meaning behind what they say with text can be a difficult thing to figure and understand fully. Many online posts and articles can be written with the intent of irony or sarcasm, but the outside perspective can unintentionally misconstrue it and make the author look bad or take a whole different viewpoint. This problem can be solved by training a model that can help determine the intention behind a message and rate it on a scale of positive to negative meaning and intent. This is to add context to a piece of text to help the reader better understand the author’s intent. I will train and test a model using hundreds of thousands of tweets from different users as data and help figure out whether it’s a positive or negative meaning tweet.

1. **Data Preparation**

First, I made sure to remove all rows with empty values for both the training and testing dataset. I set all the letters to lowercase, remove any sort of numbers or special characters to make it easier to extract data features for the training dataset. I also removed the “http” part of a string if it had any since it was indicative of a website. This is to remove any sort of websites features that were left after removing special characters. I also decided to remove any sort of stop words since those don’t contain much emotional or sentimental value. I had a function that would tokenize the words as well as perform all these preprocessing tasks. Since the normal dataset was too much, I decided to take a small sample of 4000 values randomly. For the language feature extraction, I decided to use BOW (Bag of words) and TFIDF since those already were in a library. I could not figure out how to work the word2vec and fit its parameters or constraints so I could not implement it like I wanted to.

1. **Model Development**
   * Model Training

For the training phase, I had two separate functions that would BOW and TFIDF separately. Since I had already tokenized them while preprocessing them, all I had to do was to fit and transform the tokenized training dataset (x) into the respective models with the sentiment as the y-variable. There was no need to split the training and testing dataset since I was now provided with two separate files, one for testing and one for training. The testing data was just transformed into the models so that they can be compared with the performance of the models later. I decided to stick pretty close to what I did for assignment one, which was to test if the model prediction was right and the likelihood of a model prediction.

* + Model Evaluation

\*Refer to the GitHub for the results\*

1. **Discussion**

For bag of words, the results show that the Logistic Regression model had the highest precision when identifying if a tweet was positive with a low recall and it’s the complete opposite for identifying if a tweet was negative. The SVM model has the highest recall for identifying negative tweets but a mediocre precision. It was once again the complete opposite when identifying positive tweets. The Naïve Bayes and Random Forest models performed about the same with their precision, recall, and f1-scores being relatively the same with the exception of the precision scores for positive tweets between the two. Overall, it looks like the models had better performance identifying negative tweets than positive due to the high recall score. This is indicative of a class imbalance issue. Bag of words fixes the problem well since bag of words would identify whether certain words would be used more frequently on either a positive or negative tweet. It was able to tell certain emotions behind each tweet well. I think I screwed up by not balancing the class out well.

For TFIDF, all the models did not perform particularly well with the dataset. On average, the LR model had precision of 0.72 and a recall of 0.54. SVM had a precision of 0.76 and recall of 0.54. Naïve Bayes had a precision of 0.55 and recall of 0.53, and random forest had a precision of 0.66 and recall of 0.57. For accuracy, all the models got similar results, ranging from 0.53 to 0.57. The F1-scores were generally low, ranging from 0.18 to 0.68, meaning that it could not predict positive and negative results well. This was different from bag of words since it was just low all around and had no highs. I think TF-IDF performed poorly since it does not handle synonyms or words with similar meanings. With internet slang, the model might have been tripped on certain terminology for the tweets and such.

Some of the challenges I met were the huge amount of data I had to work with. I had never worked with a dataset that huge before. It was tough at first but then I realized I could just randomly take out different values and rows and use them for my dataset. I decided to take about 4000 pieces of data since it seemed reasonable for my computer to handle as well as being a passable sample size for model training. Thinking on it now, this might have been an issue that caused the class imbalance for my bag of words and relatively poor performance for my TFIDF. I could have balanced the data out, but I couldn’t figure it out. I was never able to figure out how to get word2vec to work with my dataset however.

This assignment was fun. It helped me work on how to deal with large datasets like this. I could not brute force my way through this like I have done before. It was interesting to understand the new models that were introduced in this assignment, and it went a long way in helping my group understand what we must do for our final project.

1. **Appendix**
   * *https://github.com/taiphlosion/Individual-Coding-Assignment-02-Language-Modality-.git*