Data Mining Concepts

Elijah Albertson

112065421

Kaggle Report

1. Data Processing

1.1 Data Exploration

In this competition, each person was required to train a model to determine sentimentality where we were given a Twitter data set containing raw data (tweets) to do so. The data zip contained four files– tweet_DM (raw data in JSON format), sample submission data identification and emotion (CSV format). I began by uploading all the files into the Kaggle notebook and began visualizations. I examined the first few rows of all the data sets and ran exploratory analyses. Analysis such as the distribution between all eight emotions (using a bar plot) as well as the ratio between the test and the train data. I then proceeded to clean the data set.

1.2 Data Cleaning

I performed comprehensive cleaning on the tweet_DM raw JSON data set where I removed special characters, (punctuations and symbols), URLs and stop words. Additionally, I applied a cleaning feature known as contractions to the data. This feature tokenizes certain words such as can't and cannot as the same term rather than two different terms. The code also lemmatizes the data so that the model would easily detect root words; by separating root words with the actual word. For example, it lemmatizes the word "angry" as the root word "anger". Moreover, I imported and used a library called emoji, which preserves the emojis in a data set by tokenizing them. The decision to preserve emojis was calculated—as we were performing sentimental analysis and therefore emojis can reflect sentiments. For example, an angry face emoji can indicate that a person is angry and thus may detect and classify that line as "anger".

That being said though, I was unable to utilize the feature due to model training constraints and issues. Upon completion, I saved the cleaned data as a pickle file (cleaned_tweet_df) – as cleaning the data was a tedious task and it would save a lot of time in the future.

2. Merging Data

Subsequently, I merged the cleaned data set (cleaned_tweet_df) with the emotions data set and the data identification data set. I then split the data set into two data frames; training and testing which would later be used for the training model. I assigned the data frame a new name and proceeded to do feature engineering on the date.

3. Feature Engineering.

For this model, I did two primary engineering on the data sets— Count and sentiment analysis algorithms. The count feature simply counts characteristics in the data that can contribute to a better prediction model. This feature would only work with deep learning and pre-trained models as they are constructed to understand the context of sentences and not just words. For example, RoBERTa and BERT would benefit from this feature engineering, however, Word2Vec and TF-IDF would not. The count feature was done on both test and train data frames. The included:

- Word count
- Exclamation mark count (surprise detection)
- Question mark count
- Emoji count
- Punctuation count

Furthermore, I attempted to calculate sentiments using a sentiment analysis algorithm by Text Blob. For example, it assigns positive emotions such as happiness and trust to "1", anger and sadness as "-1", and any other as "Neutral" (which is "0"). Thus, the numbers closest to the integer would be categorized as that sentiment. For example, a sentiment score of 0.55 would be positive while -0.6 would be negative. Lastly, I utilized Roberta Tokenizer to tokenize the text. This tokenizer was used because I initially intended to train the model using RoBERTA.

4. Model Training

4.1 RoBERTa

RoBERTA was the first model that I tried to run. RoBERTa (Robustly Optimized BERT Pretraining) model is a model based on Google's BERT model. It is a pre-trained model optimized for a larger vocabulary than BERT including slang, jargon and shorthand writing commonly used in social texts such as tweets. However, after 40+ hours of training on a low-resource machine, the machine finally gave up and the system crashed.

4.2 Random Forest

I then proceeded to train my data using the Random Forest model. Random forest is a machine learning algorithm which combines the output of multiple decision trees to reach

a single result. I made several attempts at this model however between the heavy resources and the time consumption, I decided to change my model yet again.

4.3 Word2Vec

I then decided to use a less complex model—word2vec. Word2Vec transforms words into vectors which are then distributed in numerical representations of words known as features. This was my first successful model which took approximately one hour to run. That said, the drawback of using a less complex model also means that the efficiency is not going to be as optimal as more advanced models. Thus, making my first official public score 0.304. I modified a few parameters on the model but it either gave similar or lower metrics or quadrupled the running time (which sometimes fails in the end). I also tried incorporating the Linear Regression Classifier however, I encountered the same time-heavy consumption issues.

4.4 Decision Trees.

Like all models before, decision trees predict the value of a target variable by learning simple rules from features. They are a non-parametric supervised learning method used for classification and regression. I decided to go with decision trees as time was a precious commodity that I didn't have. Again, just like word2vec, the accuracy of this model was horrid. Due to an oversight on my part, I submitted the CSV for prediction without knowing the metrics beforehand. As a result, I received my lowest public score in the competition; 0.297.

4.5 TF_IDF

Lastly, I tried a more robust yet simple data training style which was taught in LAB 1. The term Frequency - Inverse Document Frequency (TF-IDF) is a statistical measure that evaluates how relevant a word is to a document in a collection of documents. It was a model that I was very familiar with, and with that, I had my first breakthrough. I landed a validation accuracy of 0,49, which amounted to a public score of 0.404408. To note, I also utilized the Naïve Bayes Classifier to classify the texts inside the data. I decided to continue with this model for two reasons; firstly, it took less than 30 minutes each time to train the data and secondly, with every modification, I received a slightly higher score. Those modifications included the max df, ngram range, min df and the max features. For example, here is one of the parameters I used: max_df=0.99, ngram_range = (1, 8) and min_df=3, max_features = 800000. validation accuracy— 0.5305. Below is a snippet of one of the trainings done:

```
Validation Accuracy: 0.5210863135620876

Classification Report:
               precision    recall  f1-score   support

        anger       0.61      0.16      0.26      7973
anticipation       0.60      0.51      0.55     49787
      disgust       0.41      0.41      0.41     27820
         fear       0.60      0.34      0.43     12800
          joy       0.53      0.77      0.63    103204
      sadness       0.44      0.43      0.44     38687
     surprise       0.62      0.15      0.25      9746
        trust       0.53      0.27      0.36     41096

     accuracy                           0.52    291113
    macro avg       0.54      0.38      0.42    291113
 weighted avg       0.53      0.52      0.50    291113
```

In the end, I was able to achieve a public score of 0.44317 which amounted to a private score of 0.43099 using this training model.

5. Conclusion

I was able to apply every aspect of what I was taught towards the competition. It was an arduous task yet I learnt from my mistakes – and there were several. From inappropriately merging the data to forgetting to include metrics, to what seemed like limitless syntax errors. I would say that time, — rather the lack of time was the most determining factor in the competition. This was also coupled with insufficient computational resources. The irony of it all was that Data processing did not take nearly as much time as model training, however, it did take more critical thinking. That said there were also numerous challenges as one of the famous—Kaggle time-out sessions proved frustrating. In the end, I settled with the TF-IDF model and Naïve Bayes classifier because consumed less time and also required much fewer computational resources.