CE807 – Assignment 2 - Final Practical Text Analytics and Report

Student id: 2207938

Abstract

Understanding the impacts and effects of hate speech on an online community is a key step toward curbing the spread. However, for this to get done it is essential to figure out how best these hate speeches or offensive words could be identified. This report portrays the building of 2 classification models that were trained using the OLID dataset that was scraped from Twitter to classify hate speech/offensive tweets from non-hate speech/offensive tweets. Using the basics of performance metrics, these models were proven to be effective and obtained an accuracy of 0.76 and 0.75 respectively.

1 Materials

- Code
- Google Drive Folder
- Presentation

2 Model Selection (Task 1)

The point of the whole project is to be able to create models that'll easily classify if a tweet is offensive or not offensive, From the original data, these labels were presented already, so comparisons of the original labels to the ones generated by the supposed model will determine their level of accuracy, precision, recall, and f1 score. After attempting the building of about 5 models using machine learning like Liner Regression, Decision Tree Classifier, and Support Vector Classifier, The two best models with respect to a shuffle of the OLID Dataset with the seed 2207938 were 0. and —- with relevance to accuracy and other performance metrics.

Now, the first key step that was carried out before model implementation was Data Cleaning/ Preprocessing. Dealing with text data, a function was developed that'll easily clean and pre-process the tweet column in the data frame, considering basic factors like lowercasing all text, removing common non-sensical text, removing non-alphabetic characters, removing numerical values, removing URLs, removing punctuations, tokenizing text, removing stopwords, and lemmatizing the words. Because it's been prepared, all the files were easily passed into it and preprocessed. Theoretically, emojis and uppercase letters are sometimes a representation of feelings that could be described as angry or hurtful but because these words will be converted later on into vectors for model training, they are less important to be considered in this kind of classification, hence they were removed and reduced to lowercase letters respectively to reduce noise and enforce better efficiency of the models to be trained and avoid errors. Thereafter an input of Tokenization. This is the process of breaking down a piece of text into smaller units called tokens. These tokens can be words, phrases, symbols, or even individual characters. The goal of tokenization is to simplify text data and make it easier to analyze. By breaking the text into tokens, it'll perform tasks such as counting the frequency of words, identifying named entities, or building machine learning models to classify or analyze the text.

2.1 Summary of 2 selected Models

The two best models used in this project were Support Vector Machine and Decision Tree Classifier. Pradhan (2012) explored that SVM is a supervised machine learning algorithm, in which a model is built that predicts the category of a new example given a set of training examples, each marked as belonging to one of many categories. He went further to add that The Support Vector Machine (SVM) algorithm is widely considered to be one of the best machine learning algorithms, having been proposed in the 1990s and predominantly used for pattern recognition. SVM has been applied to a wide range of pattern classification problems, such as image recognition, speech recognition, text cate-

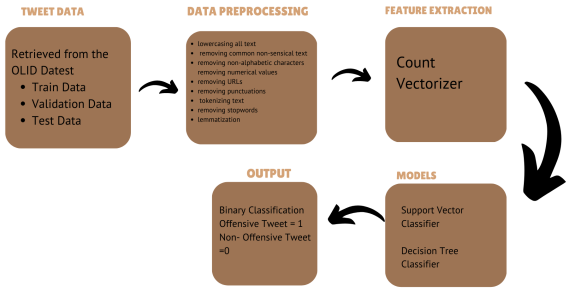

Figure 1: Pipeline Used in Developing Models

gorization, face detection, and faulty card detection, among others. On the other hand, Song Lu (2015) opined that a decision tree is a non-parametric supervised learning algorithm, that is utilized for both classification and regression tasks. It works in a hierarchical, tree structure, which consists of a root node, branches, internal nodes and leaf nodes.

2.2 Critical discussion and justification of model selection

The basis for the selection of the SVM model were as regards to the versatility and robust machine learning method that is well-suited for classification problems like the OLID Dataset. It is insensitive to outliers, making them a good choice for noisy data or datasets with anomalies. SVMs can handle both linear and non-linear classification tasks, making them a versatile tool for a variety of problems. They also have good generalization ability and can generalize from training data to new, unseen data, reducing overfitting by maximizing the margin between the decision boundary and the training data points. However as for the Decision Tree, although it could also be used for regression analysis, it helps to identify important features for a classification task like this one with the OLID Dataset, Ideally these features that appear at the top of the tree are considered more important, making it easy to identify which features are driving the predictions. It also handles both categorical and numerical data, making them versatile for a range of datasets.

- It should contain the Figure 1 of both models

3 Design and implementation of Classifiers (Task 2)

Initially, a defined pipeline was used in calling out each of these models, but the more they were used the more error it span, however when that line of code was taken away, the model took less than 30 minutes for training. Hence these models were just identified and the random state of my student ID was passed into them.

For the first model, the first thing in my design was the parameters. Understanding the hyperparameter tuning is to find the best set of hyperparameters that maximizes the performance of the model on a validation set while avoiding overfitting the training data. This helps to control the behavior of the learning algorithm, affecting the accuracy and speed of training and the accuracy of the resulting model. This was done in each of the models by an automatic use of the grid search technique. It was properly defined in a dictionary of hyperparameters to tune, which included the regularization parameter C and the kernel function. Cross-validation was also performed using the GridSearchCV function with cv=5, which specifies a 5-fold cross-validation.

For the second model, a CountVectorizer is used to convert the text data into numerical feature vectors. and then the target variables are encoded as 0 for "NOT" class and 1 for "OFF" class. A pipeline was created for the decision tree model, and a grid search is performed to find the best hyperparameters for the model. The best model is then saved using joblib.dump. Finally, the performance of the model is evaluated on the validation set using the compute performance function, and the function returns the performance metrics. The performance metrics are also printed to the console along with the path where the model is saved.

- The below are the results from the original set

Dataset	Total	% OFF	% NOT
Train	12313	4092	8221
Valid	927	308	619
Test	860	240	620

Table 1: Dataset Details

Model	F1 Score
Model 1	0.603
Model 2	0.452

Table 2: Model Performance

4 Data Size Effect (Task 3)

For the datasplit, the train data is divided into four subsets of 25, 50, 75, and 100 percentage using a seed of my student id. The purpose of this is to create subsets of the data that maintain the same class distribution (i.e., offensive vs. non-offensive tweets) as the original dataset.The code begins by defining the target variable as 'label', which is the column that contains the class labels (offensive vs. non-offensive). Then, it splits the data into four subsets using the train test split function from the scikit learn library. The function is called three times with different test sizes, each time using the remaining data from the previous split. After creating the subsets, the code concatenates all the different subsets in increasing value of 25 percent, and then it checked if the subsets maintained the same class distribution as the original dataset. It calculates the percentage of offensive and non-offensive tweets in each subset using the group by function from pandas.The results were printed, showing the percentage of offensive and non-offensive tweets in each subset. Then, the subsets are saved as CSV files using the to csv function from pandas, and their file paths are printed to confirm their storage path.

Data %	Total	% OFF	% NOT
25%	3078	1023	2055
50%	6156	2046	4109
75%	9234	3069	6165
100%	12313	4093	8220

Table 3: Train Dataset Statistics of Different Size

The figures shows the full performance metrics, more of the plots of the confusion matrix can be found within the code..

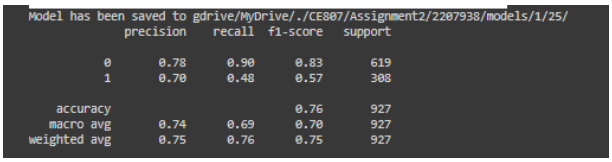

Figure 2: Model 1 of subset 25

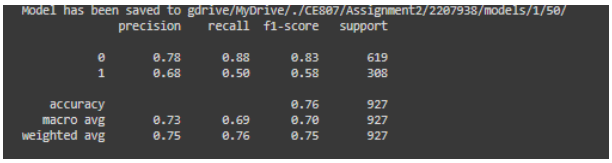

Figure 3: Model 1 of subset 50

5 Summary (Task 4)

In a nutshell, the two models implemented, with hyperparameter tuning while using the grid search technique performed avereagely and could be recommended for future classification of hatespeech. Their performances using the first few rows of the sets they were evaluated and tested showed they match up with state of the art concepts. Although more can be done in improving their accuracy by regular tuning, however this may take more time as the models take quite some minutes to run and get trained each time the functions are called.

6 References

Pradhan, Ashis. (2012). Support vector machine-A survey. IJETAE. 2.

Song, Yan-Yan Lu, Ying. (2015). Decision tree methods: applications for classification and prediction. Shanghai archives of psychiatry. 27. 130-5. 10.11919/j.issn.1002-0829.215044.

id of Output	GT	M1(100%)	M2(100%)
Row 1	NOT	NOT	NOT
Row 4	OFF	OFF	OFF
Row 13	NOT	NOT	NOT
Row 27	NOT	NOT	NOT
Row 44	OFF	OFF	OFF

Table 4: Comparing two Model's using 100% data: Sample Examples and model output using Model 1 & 2. GT (Ground Truth) is provided in the test.csv file.

Rows	GT	M1(25%)	M1(50%)	M1(75%)	M1(100%)
Row 1	OFF	OFF	OFF	OFF	OFF
Row 2	OFF	OFF	OFF	OFF	OFF
Row 3	OFF	OFF	OFF	OFF	OFF
Row 4	NOT	NOT	NOT	NOT	NOT
Row 5	NOT	NOT	NOT	NOT	NOT

Table 5: Comparing Model Size: Sample Examples and model output using Model 1 with different Data Size

Rows	GT	M2(25%)	M2(50%)	M2(75%)	M2(100%)
Row 1	OFF	OFF	OFF	OFF	OFF
Row 2	OFF	OFF	OFF	OFF	OFF
Row 3	OFF	OFF	OFF	OFF	OFF
Row 4	NOT	NOT	NOT	NOT	NOT
Row 5	NOT	NOT	NOT	NOT	NOT

Table 6: Comparing Model Size: Sample Examples and model output using Model 2 with different Data Size

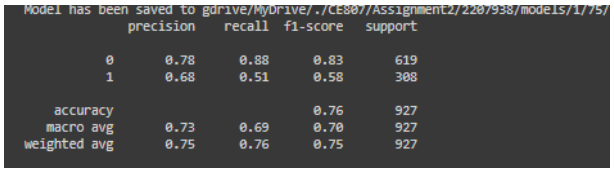

Figure 4: Model 1 of subset 75

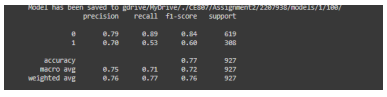

Figure 5: Model 1 of subset 100

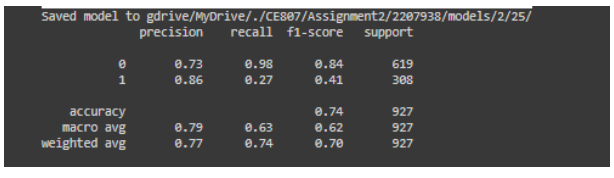

Figure 6: Model 2 of subset 25

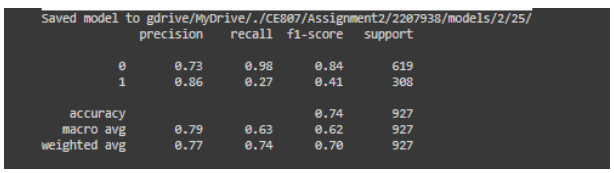

Figure 7: Model 2 of subset 50

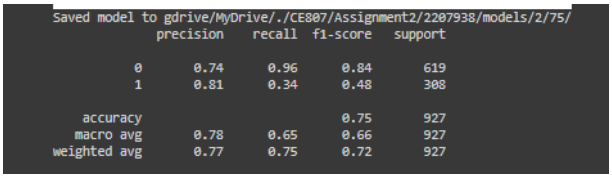

Figure 8: Model 2 of subset 75

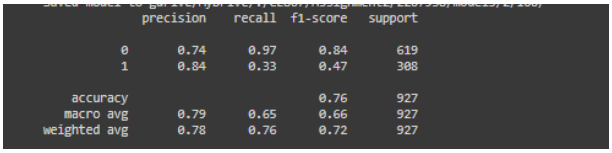

Figure 9: Model 2 of subset 100