Who Tweeted That?

Feature Engineering

Feature selection plays a significant role in measuring the predictive power of text classification models. To make features more statistically independent, an array of feature engineering methods have been utilised.

- Preprocessing of Tweets: In order to run the classification model on noise free tweets, non-discriminative features such as hyperlinks, web addresses, handles containing *"@"*, single alphabets, numbers, punctuations, retweets and English stopwords like "*a"*, *"an"*, *"the"* were purged. Contractions such as *"they've"*, *"shouldn't"* were expanded to *"they have"* and *"should not"*. Emoticons were replaced by words that express the emotion. White spaces were stripped. To eliminate ambiguities in casing, all the alphabets were converted to lower case.

- Stemming: The Porter Stemmer algorithm was used to map related words with the same root to a common feature. Thus, making the feature set compact.

- Normalisation: To handle the varying frequencies of words, **term-frequency** and **inverse document frequency technique** (tf-idf) was implemented. Words incurring a higher frequency in the entire collection can create false correlations which adds noise to parameter estimates. Therefore, such words are given a lower weighting. Rarer words are given higher weights. To increase the accuracy of the model, trigrams at character level are employed.

- Word Embeddings: GloVe vector pretrained model is used. The file - glove.42B.300d.zip, covered 75% of the dataset's vocabulary.

- Additional Feature Extraction: A total of 13 features were extracted. Word count, character count, average word count, hashtags count, uppercase character count, mentions count, retweet count, title word count (first character capital), links count, contraction count, emoticon count, number count and alphanumeric character count. To understand the advantages and disadvantages of the selected feature engineering techniques, different combinations of their representations were explored.

Learners

A wide variety of text classification models were researched and implemented.

1. Multinomial Naïve Bayes(MNB): This model was chosen due to its speed and effectiveness for text classification problems. Since the model works on the Maximum A Posteriori rule, it returns the class having the highest probability for a given set of features.

- Feature Set 1: Balanced sample of 30 raw tweets per author, eliminated authors with less than 30. Normalised dataset with *tf-idf* and *ngarm* at character level. Split train and test data with 60-40%
- Reason: Due to bias effect, MNB shrinks the weight of classes with fewer training examples, hence samples below 30 were discarded. To avoid memory issues, samples of 30 were considered.
- Result: Accuracy on test data: 21.9%, Kaggle 15.8%. The execution time was approximately 7minutes.

2. Neural Network: These models are the state of the art for most text classification applications.

- Feature Set: Processed text, GloVe word vectors, train-test-split 80-20%
- Implementation Parameters: Activation functions used were *rectified linear unit* and *softmax*. *Sparse_categorical_crossentropy* loss function is used since the dataset consists of mutually exclusive classes. Adam optimizer was used to minimise error. A batch size of 300 was used in 15 epochs.
- Result: The accuracy on train was 8.9% and test was 1.27%, indicating a significant overfit. Word Embedding layer using GloVe vector representation reduced overfitting, the train accuracy was 3.7% and test accuracy 2.54%. This low accuracy could be attributed to a flaw in the NN layer design. The model took up significant amount of execution time; approximately 6hrs for the entire dataset.

3. Stochastic Gradient Descent (SGD): SGD is a way of training a model using randomly selected samples to calculate the gradients and hence minimise the error. SGD is good for optimizing a large number of parameters.

- Feature Sets: 1. Entire processed text 2. Balanced Sample of 30 Raw tweets per author using normalisation, train-test-split: 60-40%.
- Implementation Parameter: Modified_huber loss function is used, since it is less sensitive to outliers.
- Result: Accuracy on test with feature set 1 was 0.035%, with feature set 2 26.03%. Kaggle accuracy with feature set 2 was 19.9%. It is also resistant to overfitting, which could explain there was a decline in accuracy when predicting the test set as opposed to a train-test split implemented earlier. Execution time was approximately 12 minutes.

4. Support Vector Machines (SVM): This model is known to work with large feature spaces. It fits the best hyperplane between vectors that belong to a class and those that do not belong to it.

- **Feature set**: Balanced Sample of 30 raw tweets per author using normalization, train-test-split:60-40%
- **Implementation Parameter:** Linear SVC with Hinge loss function is used.
- **Result**: Accuracy on test: 27.3%. Accuracy on Kaggle 24.34%. Since SVM fits a hyperplane with the lowest error, we could see a significant increase in accuracy and a drop in overfitting. Execution time was approximately 20 minutes.

5. Logistic Regression: Logistic regression is the basis for any primitive classification model. Hence, multinomial logistic regression was tested. However, as expected, the results were poor with a meagre 4% accuracy on the test set as logistic regression works best as a binary classifier as opposed to multiclass classifier due to its inherent property of squashing the variables to range between 0 and 1.

6. Random Forest Classifier: This model was considered since its known for its out-of-the-box properties. However, given the large number of classes with training examples, a considerable number of trees were required. Due to limited memory storage it failed to run even with sampling.

7. K-Nearest Neighbors: Since the training examples consisted of highly correlated variables, selecting *n_neighbors* parameter was a challenge. With a value of 23, accuracy was 2%, with value of 1, accuracy increased to 8%, but the predicted value was a single class for all data points. Hence, predicting the author based on the similarity score seemed inherently incoherent.

8. XGBoost: This was a natural modelling choice due to its reputation of execution speed and model performance. It runs on the idea of adding new models to correct existing model errors. However, this model was not preferred as the execution time was fairly large compared to other models in use.

Model Selection

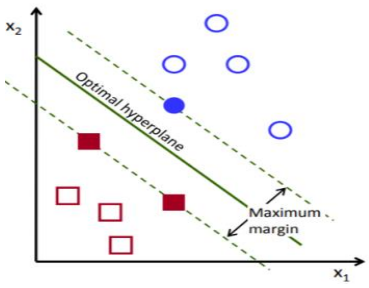

Figure 1: [CSCE 666 Pattern Analysis]

The main criteria for selecting the final model was based on generalisation ability and accuracy. Based on the above results, linear support vector classifier (SVC) minimised overfitting between train and test and produced the best accuracy of 27.3%. Linear SVCs are based on the principle of choosing a hyperplane with maximum marginal separation. Hyperplane furthest away from all vectors will be less sensitive to noise and have more generalising capability.

The hinge loss function measures the errors from probability distributions of the training set and calculates the empirical risk, as the number of samples increases, the empirical risk converges to expected risk. Therefore, Linear SVC naturally avoids overfitting and gives the least chance to misclassify data points and hence works well with large training samples.

Penalty term C is used for misclassification, lower the value of C, higher the margin of a hyperplane. C=1 produced accuracy of 27.5%, but C=5 produced 61% resulting in overfitting, hence C=1 was considered, even though there is no penalty for misclassification, it is less sensitive to noise.

Critical Analysis:

Feature Engineering Analysis

A significant improvement in accuracy was seen across all our models once the samples were considered for authors with a minimum of 30 tweets. This is owing to the good representation of each class in modelling. It is noticeable that around 1K authors were not considered for the analysis. Excluding them from the model decreased our accuracy. However, the final implemented approach focusses on predicting a statistically significant number of authors accurately over poor prediction of all authors.

Feature sets were compared for accuracy levels on the models. '*Clean*' refers to processed tweet.

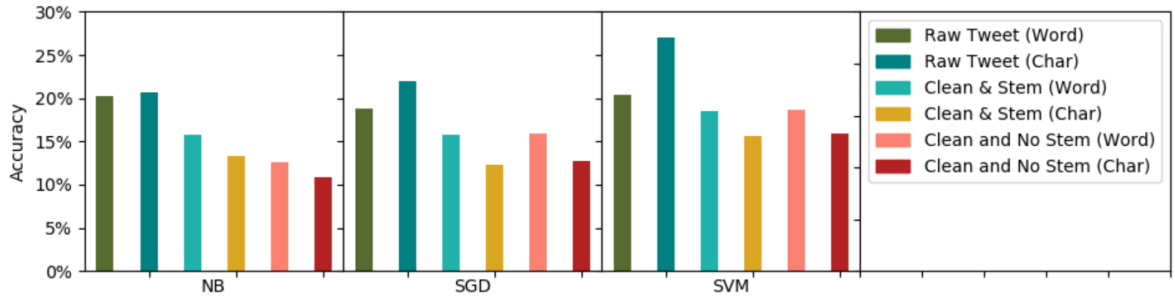

Figure 2: Feature Set Comparison using tf-idf at Word and Character Level

Based on the results, it can be deduced that the normalised raw tweets significantly represent the authors compared to other features. Therefore, it can be concluded that cleaning raw text can eliminate relevant features that can otherwise potentially increase the predictive measure of the model.

Modelling Analysis

The Learning curves for our top 3 models are generated. Score is the accuracy of the Model.

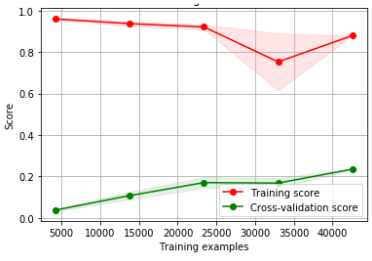

Figure 3: Learning Curve NB

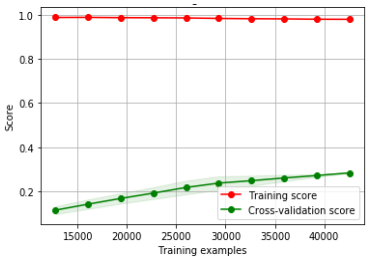

Figure 4: Learning Curve SVC

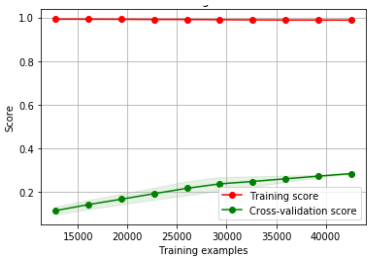

Figure 5: Learning Curve SGD

From figure 3, 4 and 5, it is evident that the models improve with more training examples.

For Naïve Bayes, as the number of samples increases, the shaded portion around the training curve indicates it is more sensitive to error due to bias and the shaded portion of validation curve seems to have a slight increase in variance but not significant enough. Hence, a high bias and low variance indicate possible underfitting.

For SVC, the training curve shows no sign of bias. The validation curve shows a constant increase in variance as the sample increases but not significant enough. Since the training curve is slightly decreased, it can indicate a decrease in overfit.

SGD's graph is quite similar to SVC. Bias remains low, constant increase in variance. However, the training curve does not decrease as the number of examples increase. This can indicate a possible overfit.

Therefore, Linear SVC is chosen as our final model with raw text normalization as the feature set.