**COMP 3222 Machine learning Technologies Coursework**

Tan Wei Shawn

[wst1g18@soton.ac.uk](mailto:wst1g18@soton.ac.uk)

**Introduction and Data Analysis**

In this era of information technology, an unverified news can spread faster in social media than verified news reporting company such as CNN and BBC. Hence, this coursework aims to classify and verify the autenticity of Twitter texts based on the use case data from MediaEval 2015 “verifying multimedia use” task. Twitter text with the label fake and humour are both taken as fake news and definitions of fake news can be found in the coursework sheet [1].

Both the training set and test set text files are first converted into excel file with the steps below to increase accuracy and reduce data loss. First, twitter identification (tweetID) is imported as string because Excel can only register up to the 15th character in an integer and the numbers subsequently will all be replaced with 0s. Besides, there are 15 observations in the training set and 3 obersavtions in the test set that has no seperation in the provided text file. This is corrected during the conversion to Excel file to increase the accuracy of the model. Next, the Twitter text are converted in the format of utp-8 instead of ascii format as there are non-english wordings and also emojis in the string of Twitter text (tweetText).

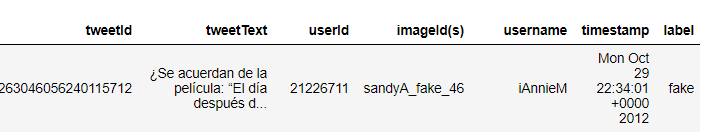


Figure 1: First observation of the training set with seven variables provided.

The initial dimension of the training set and testing set are (14483 x 7) and (3782 x 7) respectively. The 7 imported variables are shown as above. Next, the numbers of fake and true labels are plotted in a bar chart to observe for bias in the training set.

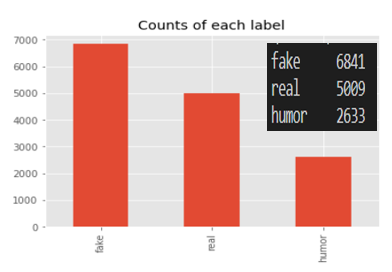


Figure 2: Counts of label.

The count ratio of fake to real is 1.89 : 1, hence, the training set tends to have a bias towards fake news. Next, the labels are encoded with fake and humour as 0 and real as 1.

**Algorithm Design**

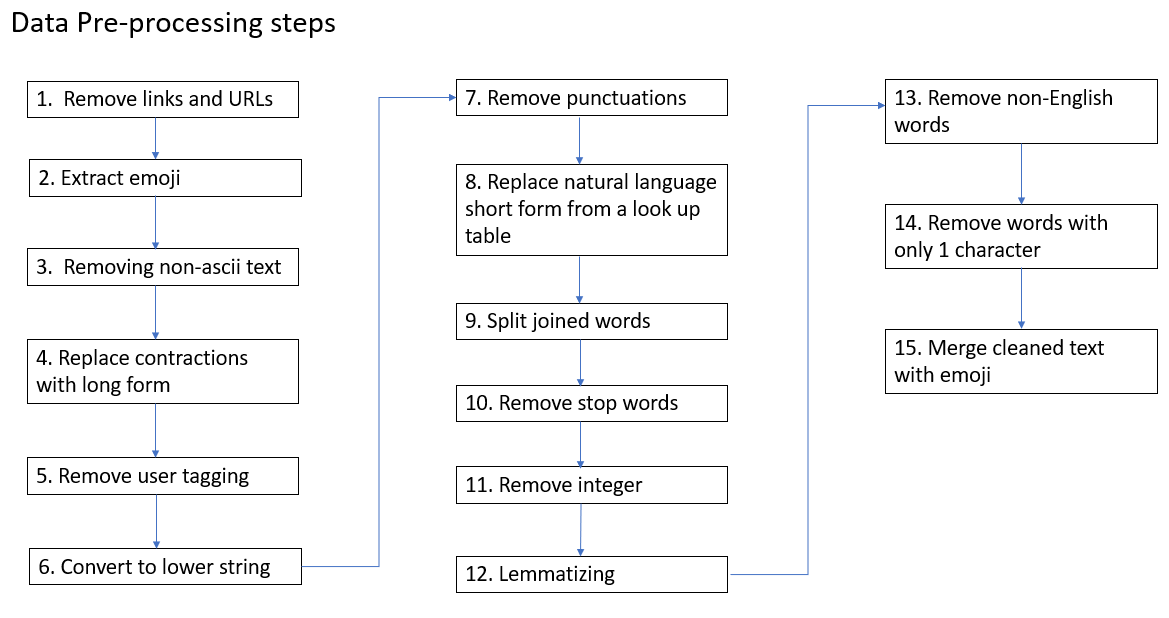


Figure 3: Data pre-processing applied in this exercise.

The final goal of this part is to obtained strings of cleaned text which includes emojis and cleaned hashtag to be used as features. The first step would be to remove all URLs as image processing is not required in this exercise. Next, emojis are extracted from the Twitter text. All functions that are cited online are referenced to give credit to the author.

In step 3, non-ascii text are removed because as shown in figure 4, the top 2 languages of the tweets are in English and Spanish. They both combined to 83.8 percent of the entire dataset. By doing so, the dimension of the features can be reduced.

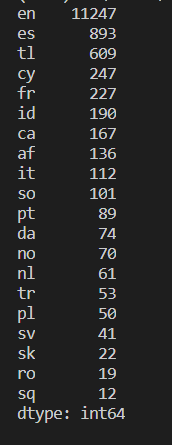
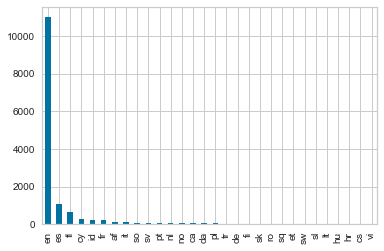


Figure 4: Languages count in the tweet texts.

Next, contractions such as she’s, I’d, it’s and etc are converted to the full form to ensure words that causes noise like ‘she s’, ‘I d’ do not exist after the removal of punctuations in step 7 [2]. In step 8, popular internet acronym such as ‘LOL’, ‘LMAO’, ‘OMG’ and short forms in the hashtag such as ‘NYC’ and ‘USA’ are replaced with the full form based on a lookup table created upon rigorous data analysis.

Step 9 ensures the unseparated hashtag words such as ‘#newyorkcity’, ‘#statueofliberty’ are converted to the correct form and not filtered out in step 13. The hashtags in the Twitter text are not removed nor extracted as it would give a topic to the strings.

In step 12, lemmatization [3] was implemented instead of stemming because lemmatization returns the base form of an English word whereas stemming chops off the suffix of a word. Besides, if stemming was used, the returned non-dictionary word would be removed in step 13.

To clean the data further, non-English words and 1-character word that holds no meaning are removed. Finally, the text in step 14 were merged with the extracted emojis in step 2 to produce a string ready for vectorization.

**Feature Selection**

In the feature selection part of this exercise, the goal is to generate and select the most important attributes from the initial data set [4]. The time stamp, Twitter ID, username, image ID and time are not taken as a feature with the reasons as below.

* Image ID corresponds to unique images with labels but image processing is not used in this exercise and hence, only texts will be considered as a feature.
* Next, Twitter ID and username cannot be used in the classification process because they are mutually exclusive to each other. As shown in figure 5, a same Twitter account might or might not post fake Twitter text and it is not a unique feature to feed into the algorithm.
* Lastly, fake twitter text can still be circulated years later and hence, it is not suitable to be used as a feature.

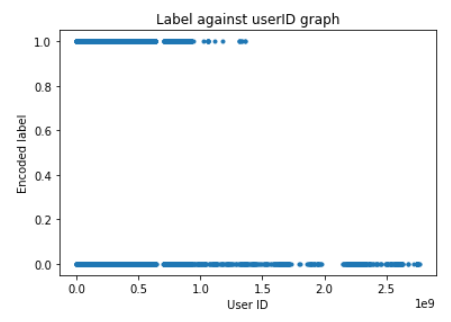


Figure 5: Graph of encoded label against User ID.

**Feature Extraction**

Feature extraction in machine learning is where the selected data inputs from feature selection is converted in to a quantity that can be measured [4]. In this case, the cleaned tweet text string is tokenized and vectorized in to a form recognizable by the computer.

The cleaned text was fitted and transformed using the TF-IDF vectorizer function from sklearn to obtain the sparse matrix of the TF-IDF weight. The parameter of N-gram, max\_df, min\_df were tuned as shown in table 1 to achieve a balance between run time and accuracy.

Table 1: Relationship between n-gram, max\_df, min\_df to the feature dimension

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Index | N-gram | Max\_df | Min\_df | Feature Dimension |
| 1. | Unigram | 1.0 | 1 | 14483 x 3961 |
| 2. | Unigram | 0.5 | 5 | 14483 x 1403 |
| 3. | Bigram | 1.0 | 1 | 14483 x 26561 |
| 4. | Bigram | 0.5 | 5 | 14483 x 3490 |

To visualize the correlation of each feature selected, a heatmap can be plotted as shown in figure 6. Most of the features have no correlation to each other and hence the generalization power of the model increase.

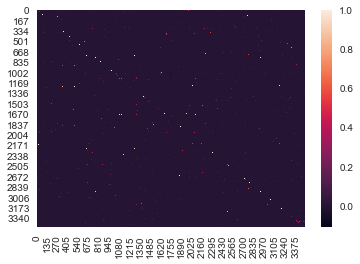


Figure 6: Heatmap of correlation of the features selected with the numbers on the axis represents the word token.

The accuracy and F1-score of each combinations of parameter with different algorithms will be discussed in the algorithm section. Max\_df of 0.5 is the threshold that removes features which appear in more than 50% of the document whereas for min\_df of 5, words that appears less than 5 times are removed. An important observation here is that 64.5% of the features are discarded when parameters is tuned from case 1 to 2. A frequency distribution chart of top 50 tokens is plotted in figure 7. Besides, a word cloud is also created to visualize the most frequent word in figure 8.

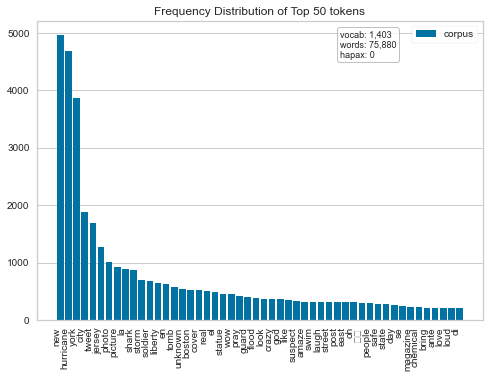


Figure 7: Frequency Distribution of top 50 tokens with N-gram of 1,1, max\_df of 0.5 and min\_df of 5.

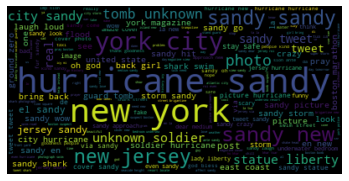


Figure 8: Word cloud obtained from the cleaned training dataset.

From figure 7, the number of words is reduced by 64.5%. Words which are noise are removed from the feature dimension due to the parameter of max\_df and min\_df. Word cloud to visualize the words from the fake label and true label were also plotted but they do not so much difference.

**Dimensionality Reduction**

The first step of dimension reduction is done in the data preprocessing part where the strings are lemmatized and cleaned. Next, To reduce the dimension further, Principle component analysis, PCA is used. Instead of using the n\_components parameter which returns the number of principle components based on the integer provided, the explained variance parameter is more suitable. If a floating number from 0 to 1 was given the ratio of the variance of a principal component to the total variance will be preserved. Table 2 below suggest the reduction in dimension for different explained varaince parameter.

Table 2: Relation of Explained variance and number of components after PCA.

|  |  |
| --- | --- |
| Explained variance | N\_components |
| 95% | 1152 |
| 90% | 1027 |
| 85% | 923 |

As shown in table 4 and table 5 the F1-score of the logistic regression model and decision tree model drop after dimension reduction. Although running time is shorten due to the reduction in features as shown in table 2, some important features are filtered out and this caused the f1-score to drop.

**Algorithm selection**

In the subsections below, the advantages and disadvantages of each algorithms are discussed and tabulated. A final algorithm of my selection is highlighted and all the algorithms tested are based on supervised learning.

Table 3: Comparison between 3 algorithms tested.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Logistic Regression | Decision Tree | Support Vector machine |
| Handling non-linear dataset | No | Yes | Yes |
| Decision boundary for binary classification | Threshold at 0.5 probability | Always parallel to the y-axis and x-axis | Non-linear boundary by transforming data set. |
| Runtime of the program | 1.88s | 69.5s | 205s |
| Performance on semi structural data | Fair | Good | Best |
| Overfitting issue | Yes | Yes | Rarely |

Support vector machine with a Gaussian RBF kernel is the final selected algorithm. The first reason is that it is hard to visualize a text data as they tend to be semi structural and have high dimensions. By assuming that there is no prior knowledge to the distribution of the data, SVM is a better choice because the kernel trick can transform a non-linearly separable data in to another dimension to be solved [5].

Besides, logistic regression and decision tree has a higher tendency to overfit the training data.

SVM is less prone to overfitting as the decision boundary is placed to maximize the margin between closest support vectors of negative and positive element [5]. Logistic regression and Decision tree on the other hand are both probabilistic [6] and the classification process is based on the posterior class probability. To prove this issue, the training set was fed into the decision tree model to be tested and the return F1-score was 0.98 whereas the SVM model tested with training set returns a value of 0.92.

Next, the reason why decision tree is not chosen is because it is more bias towards the dominant class in the training. With a 9:5 ratio of false to true label in the training set, misclassification rate will be higher as the separation of nodes is based on Gini score and information gain. They are both considered to be skew sensitive. Moreover, the decision boundary of the Decision tree is rigid whereby they are always parallel to the axis. This would cause lower classification power for inseparable data.

The downside of a SVM algorithm is that the runtime is relatively longer as compared to logistic regression and decision tree. Based on a training dataset with the same dimension, logistic regression takes 1.88s, decision tree takes 69.5s whereas SVM takes 205s to run.

**Evaluation**

To evaluate the algorithms, accuracy is not a good measurement for assessing classification model. This is because it does not take the unbalanced nature of a class into account. F1-score is a better evaluation method with precision defined as the true positive across a selected elements and recall is the true positive in the relevant elements. The equation to compute precision and recall are shown in equation 1 and 2. The F1-score is calculated using equation 3 instead of using the sklearn metrics as it does not give an accurate value. However, the micro F1-score is still tabulated for marking purposes. Table 4, 5, 6 shows the data obtained upon hyperparameter tuning. The random state of all the algorithms is set to be 42 to act as a control.

[1]

[2]

[3]

Table 4: Tabulation of logistic regression metrics with parameter tuning of N-gram, Max\_df, Min\_df and also dimension reduction using PCA.

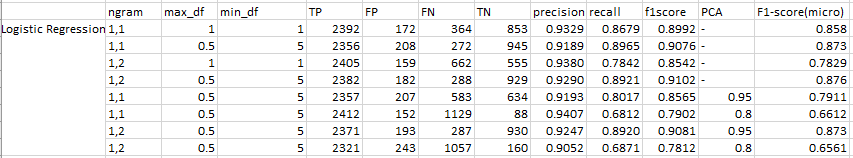


Table 5: Tabulation of decision tree metrics with parameter tuning of N-gram, Max\_df, Min\_df , PCA and class weight.

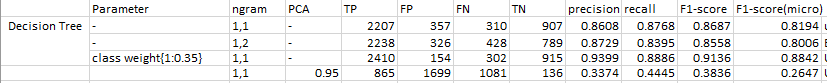
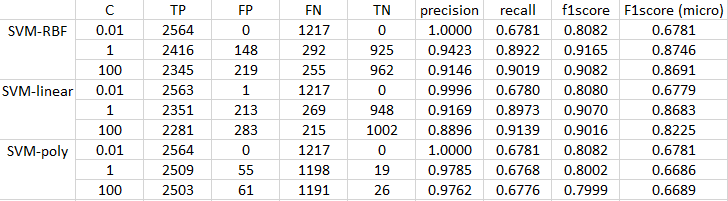


Table 6: Tabulation of SVM metrics with parameter tuning of N-gram, PCA, C and kernel



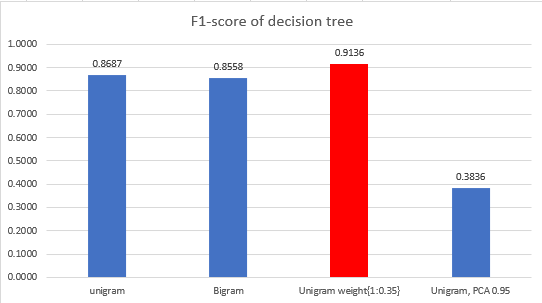


Figure 9: F1-score of decision tree.

In the Decision tree algorithm, when the hyperparameter class\_weight = 1:0.35 was added, F1-score of 0.9136 was achieved. This parameter addresses the unbalanced testing data issue discussed in the algorithm selection part. Besides, unigram gives a better result than bigram and followed by testing set which underwent PCA. From table 5 the true negative value of the last row is relatively low after PCA dimension reduction. We can conclude that important features to classify fake Twitter text are removed due to PCA.

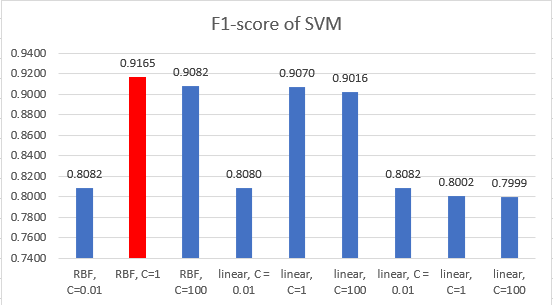


Figure 10: F1-score of SVM model.

RBF kernel has a higher F1-score and true positive count than linear kernel. From this, the intuition of using RBF kernel on an unknown data characteristic training set is correct. Poly kernel SVM shows the worst result in terms of F1-score for C of 1. Hyperparameter C behaves as a regularization term which controls the street size and margin violation. Higher C value gives fewer margin violations but the smaller margin, vice versa to a low C value.

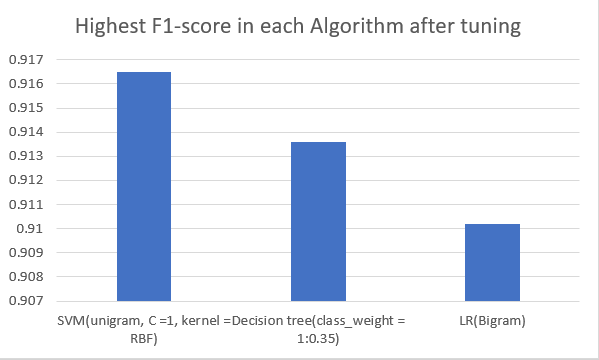


Figure 11: Highest F1-score of each algorithm

SVM with unigram vectorization, C =1, kernel = RBF gives the best F1-score among all three algorithms tested.

**Conclusion**

As a conclusion, data pre-processing plays a huge role in machine learning. If feature selection, feature extraction and dimension reduction are done correctly, algorithm with less parameter tuning would still return a high F1-score. Besides, algorithm must be selected based on data characteristic and in a semi-structural data case like text mining, algorithm with higher generalization power and one that can work both in linear and non-linear region are preferred.

To improve this coursework further, K-fold cross validation can be applied as accuracy paradox tends to occur in an imbalanced dataset. Besides, hyperparameter tuning will have to be done more rigorously. For example, gamma parameter in RBF SVM should also be considered. Lastly, ensemble learning such as random decision forest can be applied to improve the decision tree algorithm further.

**Reference**

|  |  |
| --- | --- |
| [1]  [2]  [3]  [4]  [5]  [6] | COMP3222-assignment-UG-30-09-2020 [Online], Available: <https://secure.ecs.soton.ac.uk/noteswiki/images/COMP3222-assignment-UG-30-09-2020.pdf> <https://stackoverflow.com/questions/43018030/replace-apostrophe-short-words-in-python> <https://stackoverflow.com/questions/43018030/replace-apostrophe-short-words-in-python>  Aurelion Geron, Hands-On Machine Learning with Scikit-Learn and TensorFlow, Mar 2017, chapter 1, pp 26.  Aurelion Geron, Hands-On Machine Learning with Scikit-Learn and TensorFlow, Mar 2017, chapter 5, pp 154-156.  Aurelion Geron, Hands-On Machine Learning with Scikit-Learn and TensorFlow, Mar 2017, chapter 6, pp 173. |