# NLP Coursework 1

## Overview:

In this project we aim to build a model that classifies news articles into either “fake” or “real” using natural language processing. The report below outlines the process and steps followed to reach this goal.

Firstly we create two variables label and text. Label here, is the converted multiclass label that we acquire from the convert\_label function. Hence instead of having 6 different labels we only have 2 that a news article can be. Text here, is the actual text from the news article that we further pre-process. The output of the parse\_data\_line is a tuple hence we return it in the form (x, y). we can also convert is using the tuple () method.

Secondly, we implement tokenisation steps in the pre\_process function. Tokenisation is the essential process by which we convert long sentences into easily processible chunks called tokens. These tokens can be either words, pairs of words characters or sub words. We have taken the tokens to be single words. To carry out the process of tokenisation I have used regular expressions.

The first expression replaces any spaces or special characters at the start of the word and the second does the same for the end of the word. Once completed we use the .split() method to split the sentence into tokens. To normalise all the tokens, we use the .lower() method

We work to create a dictionary containing the tokens as the keys and the weight of the token for the values. The method I have used is I have created a blank dictionary called fn\_dictionary. Once created I check if each token is new or already exists in the dictionary. If it does exist then the value 1 is added to its existing weight else, the value 1 is assigned to it. Using this method, I am able to build a dictionary representing the bag of words model as the values will contain the number of occurrences for each token in the dataset. The bag of words model allows us to interpret the dataset of tokens in an extremely simple way and provides an easily adjustable method by which to represent the tokens.

The split\_and\_preprocess\_data function is used to create the training and test split in the data set, here we are using an 80:20 ratio of training to test. As we observe the raw\_data is 10241 and of which we are using 8192 training samples. Using the to\_feature\_vector we obtain 13560 features. We divide the data into 10 random groups i.e., folds. This is done in the cross\_validate function, once the data folds are created, we train the classifier on the same. Using “pred” we get the predicted labels, and we store the accuracy, recall, precision and f1 scores in lists. We store the data of one-fold which we use later to obtain the confusion matrix. The cross\_validate function returns a list with mean values for the accuracy, recall, precision and f1 scores.

From the confusion matrix, we can infer that our model is able to classify true positives (actual real news) the best, however falls short when classifying true negatives (actual fake news). The model also has a fairly high number of false negatives and false positives. This poor performance is further highlighted by the particularly low accuracy of 56%.

If we print the false negatives and false positives, we are able to pinpoint where the errors are occurring and how many data points are being mis classified. According to the confusion matrix we obtain:

* True positive: 300
* True negative: 168
* False positive: 180
* False negative: 172

To improve the poor accuracy acquired in the previous step, I have tried to implement 3 methods to improve our data pre-processing:

1. **Count parameter in Linear SVC**: The linear SVC considers a default value of 1 as its count parameter however as the value is reduced tending to zero the parameter tells the SVC to have a lower tolerance for misclassification
2. **Lemmatization**: This is the process by which we classify words with the same root together, for example dies and die will both be considered as die. Hence this reduces the number of training features and allows us to reduce any noise in the dataset.
3. **Stop word removal**: this method allows us to remove words such as (and any, are, aren't, as, at, be, because, been, before, etc.) which have no bearing on the context of the news, this method adds to the same effect as lemmatization and further helps reduce noise.

With the first two methods I achieved a very small improvement in accuracy from 0.568 to 0.573, however using stopword removal results in a significant drop in accuracy and number of features. Upon using stopword removal the confusion matrix the model has 0 false negatives and 0 true negatives.

Here we make use of the additional columns present in the dataset and see what effect it has on the accuracy. However, the accuracy of the model fails to improve with any of the different columns of data.

## Conclusion:

To conclude the model that we developed has mediocre performance and only detects the correct classification of news 59.9% of the time. To further improve the model in the future we can implement more pre-processing techniques and expand the dataset to allow the model to have a wider range of training data.