COMP3222 Coursework : Tweet Classification using Machine Learning Methods

Student ID : 31874614
Name : Oluwatosin Ogunribido
Email : oo2g20@soton.ac.uk

January 2023

1 Overview

The datasets used in this project were derived from the MediaEval Challenge 2015 dataset. There are 14,484 datapoint in the training dataset and 3,782 in the test set. Both datasets contained information of the tweets in relation to events mentioned, as well if the tweet was real or fake. The features of the tweet included:

Feature	**Description**
tweetId	These are unique IDs for each tweet.
tweetText	These are the content of the tweet. In this column, there are also URLs for images. However, this is not used for image processing.
imageId	These are IDs given to images contained in the tweet. They are unique to the image and will be different for each images in a tweet containing multiple tweets.
Username	unique individual IDs linked to a unique account.
Timestamp	Time and date that the tweet was posted.
Label	This can be real, fake or humour. In this dataset, fake tweets are tweets that contained false events or tweets that have manipulated events such that they are false. Additionally, humour labels are converted to fake labels.

Table 1: Dataset features

They also contained modified tweets and retweets, which also need to be handled in specific ways.

2 Data Analysis

To get a better understanding of tweets and differentiate characteristics of fake and real tweets, the data was analysed based on the features provided.

There are two datasets provided through specifications. However, only the training datasets can be used for most data analysis. This is because using the test dataset may lead to overfitting, where the performance of the test is significantly better than that of training.

One of the major features provided are the content of the tweets and their labels. There are three labels provided; real, fake, and humour. However, for the sake of this project, humour is considered as fake. Although, when the data is analysed, the label humour is connected to other features related to both the real and fake tweets. It can be observed that there are significantly more fake tweets than real tweets, i.e. 65% of the tweets are fake. This is taken into consideration because it may affect the predictions of real tweets due to data availability.

Another feature that can be analysed, is the length of tweet texts and its relationship with the labels and images. From the graphs plotted, we could observe that there was a proportional distribution of real and fake tweets between different image IDs and text lengths. This is helpful to observe as it shapes the features we select for data when implementing the model for predictions.

However, when comparing models and username, the graphs show tweets from the same username were more likely to have the same label. As a result, it might be helpful to include the username as a feature for the model, as it may improve performance. However, this might include bias to our model on certain usernames compared to others.

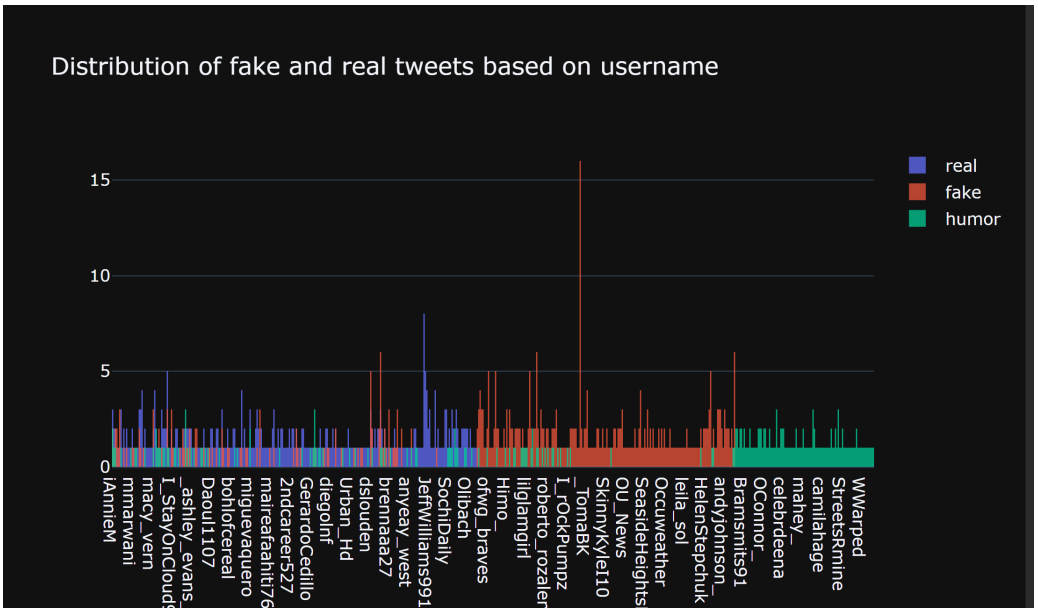

Figure 1: Graph of username against tweet labels

Another important feature, looked at during data analysis, is the frequency of individual words in the dataset. To understand the dataset a word cloud was used. The word cloud shows the text with their font size related to their frequency. This showed that the context from most tweets in the training set are around New York City and hurricane sandy. This feature helps to characterise the data and how the model performed.

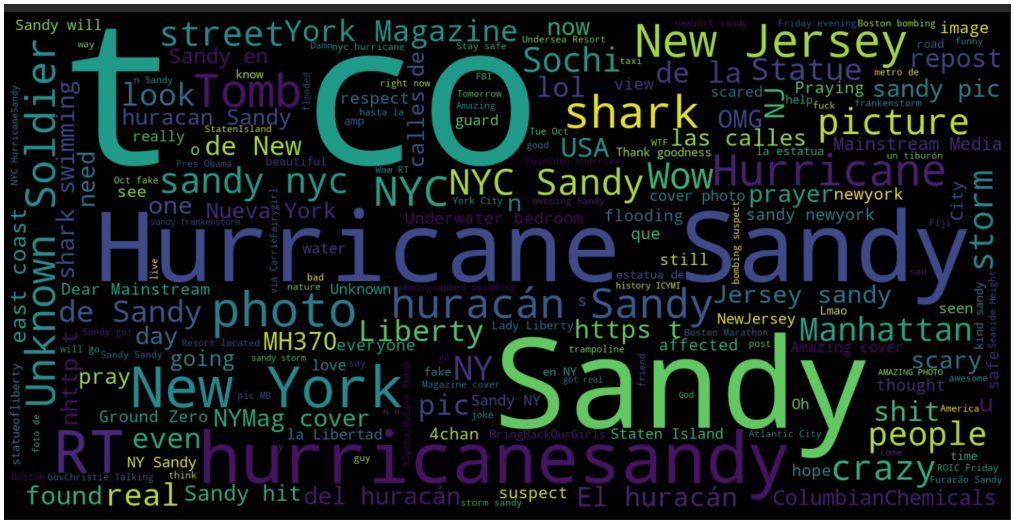

Figure 2: Word Cloud of Text Frequency

3 Data Preprocessing

Given that no image processing is done, the image IDs are dropped. Additionally, the timestamp and tweetId are removed as they are not used for processing and increase the computational cost of searching through the dataframe during model evaluation.

3.1 Online Specific Features

When preprocessing text for machine learning models and natural language processing, punctuations and tags from tweets can become noise when in excess and reduce the performance of the model. Given, the nature of the data and the importance of these tags and punctuations as well as research conducted, punctuations and tags were removed from the data.

The data provided may be linked to people in the real world as result, it is ethical to avoid using personal or specific data linked to people. This is because it might imply bias in the model, as the mentions may be linked to real or fake labels. However, this is not to say that it is not helpful, but it is preferable to not use it unless explicit consent has been given. Therefore, for the aim of this project, mentions were removed. Consequently, emojis were also removed as the model does not understand them, as a result they become noise to the model rather than helpful information.

3.2 Translation

Machine Learning models work in the computer programming language it is coded in. This is then translated for the computer to understand and process. As a result, by default, machine learning models should be able to perform similarly for different human languages.

However, the dataset contains multiple languages which will need additional processing to allow the model to develop relationships between words from different languages and the

tweets. To establish this, there are several ways to deal with the problem.

One of the best ways to do this, would be clustering the data based on the language it was tweeted [9]. However, not enough data is provided in the dataset to cluster it and model it with a good enough performance.

The next way to improve the performance, would be to translate to one language, enabling the machine language to establish better relationships between tweets. To do this, all the data in the dataset were translated to English, as this was the most frequent language detected in the model[9]. Additionally, the English language is also the best language to translate to when compared with other languages given the type of software package used, i.e. googletrans in our project to translate the data.

3.3 Lemmatisation

In Natural Language Processing, lemmatisation is the process of converting words to the base form for which it is grouped by. While stemming is the process of reducing a word to its stem that affixes the word [7]. These methods are primarily used to standardise text, making it easier for the model to relate words together.

Both of this method are helpful, however it is not useful to add both methods to the preprocessing stage as they produce similar outputs and need to be implemented at the same stage of the process. As a result, only one method is required to improve the model. On the basis of performance of the model through background research, lemmatisation works better than stemming on our datasets and model [3]. Therefore, for this model, lemmatisation was used after tokenisation.

4 Evaluation of preprocessing method

All these methods mentioned before were implemented separately to understand their effects on the performance of the model. To ensure that the model was only affected by these changes, a base model was used. This model was made up of two stages. The first extracted features using a character TF-IDF vectoriser which had a frequency threshold of 0.8 and an n_gram range of (4,8). After the features were extracted, they were passed on to a multinomial naive Bayes model.

As a result of this, it was found that the individual preprocessing methods impacted the accuracy of the model. However, when the preprocessing methods were combined, the accuracy improved and was similar to models without preprocessing methods, with a better prediction for both positive and negative labels. Moreover, without the noise caused by emojis and punctuations.

Method	Performance $f1\_score$
Without Preprocessing	0.91
Without Emoji	0.88
With Lemmatisation	0.86
Without Links and Mentions	0.89
Converted all to lowercase	0.88
Without punctuations	0.90
Without tags	0.88
Translated to English	0.88
When Combined	0.91

Table 2: F1_score for preprocessing Methods using Multinomial NB and a character vectoriser where n_gram = (4,8), max_df=0.8

5 Feature Extraction

There are multiple ways of extracting features from text for classification. Some of the most popular methods include bag of word, Term Frequency-Inverse Document Frequency Vectoriser, Count Vectoriser. However, bag of words are used primarily for deep learning models due to its intensive computation. Additionally, for characterisation of words using bag of words, the dataset needs to be large enough with a large variance of words. Therefore, the two primarily methods implemented were the TF-IDF vectoriser and Count Vectoriser. These models were implemented using a Naive Bayes model as its machine learning model.

The count vectoriser counts the frequency of words or characters in an entire text [6]. This is very helpful during classification as it shows the relation of a word to a specific label. However, it does not show how strong a word might correlate to a specific label when compared to other words or in a separate text. Similarly to count vectoriser, the TF-IDF vectoriser also calculates the frequency of word or characters, however rather than relying on one text it aims to compare the frequency from multiple documents [1]. The TF-IDF vectoriser can be mathematically defined as :

Inverse Frequency [1]

$$idf(w,d) = log(\frac{N}{f(w,d)}) \tag{1}$$

Term Frequency [1]

$$tf(w,d) = log(1 + f(w,d)) \tag{2}$$

TF-IDF: [1]

$$tfidf(w,d) = tf(w,d) * idf(w,d) \tag{3}$$

By calculating its relation to a document compared to other document, this method takes into account the importance of words and phrases. However, it is limited as it does not take into account the word's position in the document or its meaning. Additionally, the performance was relatively the same for both character TF-IDF vectoriser and word TF-

IDF vectoriser. This was mostly due to the best optimal word n-gram being a minimum and maximum of 1 and the character n-gram being the average size of a word. However, due to the range of the maximum and minimum values being larger than that of the word vectoriser, the character was used.

By comparing both vectorisers during implementation and looking at other research [2], it was observed that the TF-IDF vectoriser worked better with a better performance where it increased from 48% to 86%. This was mainly due to it taking into account the importance of different words and characters.

6 Data Evaluation

There are many machine learning models and deep learning models used for text classification. However, the focus of this project were the machine learning models which included but are not limited to Linear Support Vector Machine, Random Forest, Multinomial Naive Bayes and Logistic Regression.

By using default parameter for a char TF-IDF vectoriser, the performance of each model were calculated based on f1 score. During this process, the random forest classifier had a higher computational cost, with similar performance. As a result, the implementation was done using Multinomial Naive Bayes model, Logistic Regression and Linear Support Vector Machine.

6.1 Logistic Regression

Logistic regression is a classification model used for binary classification. However, it can also be used for multi-class classification. It is commonly used for text classification due to its efficiency. Logistic regression models the probability of an event taking place, i.e. the probability of a label given a regression coefficient $p(y|X)$. In binary classification, logistic regression can be mathematically defined as :

$$p(y|X) = \frac{1}{1+\epsilon^{-x}} = w_0 + w_1x_1 + w_2x_2 + w_3x_3...w_nx_n \tag{4}$$

To reduce the loss in logistic regression, Maximum Likelihood Estimation is used. Maximum Likelihood estimation aims to reduce the error by calculating the gradient of the function and adjusting the weights based of it. The loss function which is used for optimising this model is Cross Entropy Loss. Cross Entropy Loss calculates the error by checking if the prediction of point is right or wrong when compared to its true label.

$$L = (y\log(p) + (1-y)\log(1-p))[4] \tag{5}$$

However, this only represent the loss function of one data point. Therefore, the cost function can be defined as :

$$C = \frac{\sum_i L_i}{N}[10] \tag{6}$$

where N is the number of datapoints and L is the loss function at x_i.

This method was implemented using Scikit-learn with default parameters. However, hyperparameters can be tuned, including penalty.

6.2 Multinomial Naive Bayes

Multinomial Naive Bayes Model is a generative model which predicts the label of an input using the Bayes theorem [5] . In text classification, multinomial naive Bayes model, calculates this by predicting the label of each tag or token assigned to input.

In Bayes theorem, the likelihood of occurrence based on knowledge of events happening previously, i.e. conditional probability. This is mathematically defined as:

$$P(A|B) = \frac{P(B|A) * P(A)}{A}[5] \tag{7}$$

Additionally, this classification method also assumes that are these tokens are independent of each other. The Multinomial Naive Bayes model implements this by calculating the probability of every category C_k given a feature vector W. This can be defined in relation to the posterior, likelihood, and evidence. Mathematically:

$$Posterior(C_k|W) = \frac{Likelihood * Prior}{Evidence} = \frac{P(C_k) * P(W|C_k)}{P(W)}[5] \tag{8}$$

Once all classes have been calculated, the label of the feature W becomes the class with the maximum probability. However, to be able to achieve the best results for this model, there needs to be a bag of words or token with a large size where the tokens are independent.

Although this can be implemented from scratch, Scikit-learn provides a model which can be integrated into python programs. As a result, that was used during the implementation of this project.

6.3 Linear Support Vector Machine

A support vector machine is a discriminative model which works by creating a hyperplane which classes of data. It was implemented to solve the problems of logistic and linear regression where there are many lines which are able to classify the data, as a result it becomes hard to find the global optimum. The support vector machine does this by implementing a margin on both sides, where the margin may be hard i.e. does not allow points in the margin during training or soft i.e some points are allowed. In Linear SVM the hyperplane is defined by :

$$y = wX + b[8] \tag{9}$$

where: w = weights, b = bias, x = input,

This hyperplane is optimised by calculating the loss using the hinge loss function. This can be defined as :

$$L = ||m||^2 + C \sum max(0, 1 - \hat{y} f(^n))[8] \tag{10}$$

. where only misclassification inside the margin contribute to the loss and C is the hyper-parameter of regularisation. When using support vector machines, other hyperplanes can be used, including polynomial and logistic hyperplanes. However, for this model, only the linear support vector machine was implemented.

6.4 Implementation

In addition to implementing all three models using Scikit-learn, a grid search was used to find the optimal n_gram for each model's vectoriser with a maximum frequency threshold of 0.8 where the optimal was based on their f1_score. Additionally, During this implementation, labels were encoded into numbers to improve efficiency. The ratio of true negatives (real) were also taken into account in relation to the ratio compared to true positive and its actual distribution.

It was observed that the difference between maximum frequency did not really improve the score if it was greater than 0.5. However, the n_gram range makes a difference in performance with respect to the model. This is illustrated in the table below.

Model	(1,1)			(4,8)			(10,10)		
	f1_score	precision	recall	f1_score	precision	recall	f1_score	precision	recall
Linear SVC	0.815	0.964	0.706	0.932	0.968	0.900	0.934	0.971	0.900
Multinomial NB	0.817	1.0	0.690	0.875	0.994	0.781	0.897	0.994	0.817
Logistic Regression	0.815	0.956	0.710	0.925	0.985	0.872	0.920	0.989	0.860

Table 3: Comparing n_grams for evaluation models

7 Model performance

These models mentioned previously were compared during implementation using the f1_score, recall and precision where the n_gram range of the TF-IDF were optimal for when the maximum number of character or words was set to 10. From this, their performance was observed, and it showed that all three model performed well with text classification with similar results. However, the Linear SVC was ultimately the best, with an f1_score of 0.94 for the validation test. However, it does not do well with the test set, with an f1_score of averagely 70%. Therefore, the performance of all the model were compared for the test set. From this, it was observed that the Linear SVM and Logistic Regression were underfitting the model such that the average percentage drop was between 10-15% however the Multinomial Naive Bayes was overfitting the model by 1-3%. As a result, it can be concluded the multinomial

Model	Optimal n_gram	validation f1_score	validation precision	validation recall
Linear SVC	(4,9)	0.93	0.97	0.9
Logistic Regression	(3,7)	0.93	0.98	0.87
Multinomial NB	(8,8)	0.91	0.99	0.81

Table 4: **Model performance for validation set**

Model	optimal n_gram	test f1_score	test precision	test recall
Linear SVC	(4,9)	0.80	0.87	0.74
Logistic Regression	(3,7)	0.82	0.86	0.78
Multinomial NB	(8,8)	0.92	0.99	0.88

Table 5: **Model performance for test set**

naive Bayes worked the best. The performance of these models have also been illustrated in table 4 for the validation set and 5 for the test set.

8 Conclusion

In conclusion, the design of the optimal model involved preprocessing data to remove noise and tokenise the data, feature extraction using TF-IDF and modelling using a Multinomial Naive Bayes. Additionally, it was observed that even though f1_score was relatively high, the specificity was lower than the precision for all the machine learning models. This is shown in the table below where the true negative's f1_score is 0.72.

This could be due to the inability of the model to semantically analyse the data to look for more context. This can be improved using further natural language processes during the preprocessing stage. Additionally, this problem could also be due to the distribution of fake and real tweet in the training set.

The evaluation of the model is able to predict the labels of the dataset in 10 seconds. However, due to the translation in the preprocessing step, the model processes and predicts

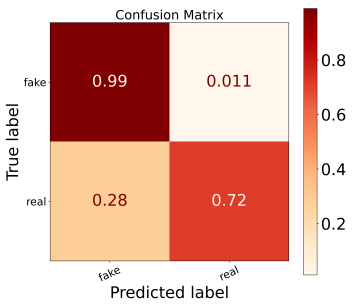

Figure 3: Confusion matrix

the labels of the whole dataset in 30 minutes.

References

[1] M. Borcan. "TF-IDF explained and python sklearn implementation," Medium. (Jun. 8, 2020), [Online]. Available: https://towardsdatascience.com/tf-idf-explained-and-python-sklearn-implementation-b020c5e83275 (visited on 01/09/2023).

[2] O. Rakhmanov, "A comparative study on vectorization and classification techniques in sentiment analysis to classify student-lecturer comments," *Procedia Computer Science*, 9th International Young Scientists Conference in Computational Science, YSC2020, 05-12 September 2020, vol. 178, pp. 194–204, Jan. 1, 2020, ISSN: 1877-0509. DOI: 10.1016/j.procs.2020.11.021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050920323954 (visited on 01/09/2023).

[3] D. L. Yse. "Text normalization for natural language processing (NLP)," Medium. (Mar. 12, 2021), [Online]. Available: https://towardsdatascience.com/text-normalization-for-natural-language-processing-nlp-70a314bfa646 (visited on 01/09/2023).

[4] *Logistic regression*, in *Wikipedia*, Page Version ID: 1129858286, Dec. 27, 2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Logistic_regression&oldid=1129858286 (visited on 01/07/2023).

[5] A. V. Ratz. "Multinomial nave bayes' for documents classification and natural language processing (NLP)," Medium. (Apr. 8, 2022), [Online]. Available: https://towardsdatascience.com/multinomial-na%C3%AFve-bayes-for-documents-classification-and-natural-language-processing-nlp-e08cc848ce6 (visited on 01/07/2023).

[6] S. Singh. "CountVectorizer vs TfidfVectorizer," Medium. (Jan. 31, 2022), [Online]. Available: https://medium.com/@shandeep92/countvectorizer-vs-tfidfvectorizer-cf62d0a54fa4 (visited on 01/06/2023).

[7] *Stemming*, in *Wikipedia*, Page Version ID: 1108161550, Sep. 2, 2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Stemming&oldid=1108161550 (visited on 01/09/2023).

[8] *Support vector machine*, in *Wikipedia*, Page Version ID: 1129233078, Dec. 24, 2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Support_vector_machine&oldid=1129233078 (visited on 01/09/2023).

[9] C. Wolfe. "Many languages, one deep learning model," Medium. (Nov. 1, 2022), [Online]. Available: https://towardsdatascience.com/many-languages-one-deep-learning-model-69201d02dee1 (visited on 01/09/2023).

[10] "Cost function — types of cost function machine learning." (), [Online]. Available: https://www.analyticsvidhya.com/blog/2021/02/cost-function-is-no-rocket-science/ (visited on 01/09/2023).