Classification of Online Media posts as Real or Fake using Machine Learning Techniques

Achyut Karnani
33243387
ak19g21@soton.ac.uk

1 Introduction

Social media has become the new source of information for everyone. From sharing personal information to news, everything starts spreading from social media. Hence, it is quite easy to spread a false information. In 2020, false tweets claimed that Covid-19 was artificially created. This was then scientifically proven wrong. In 2020, tweets regarding tampering of US Presidential election flooded. However later it was proven to be false. At times, these fake news can have consequences over human lives. In 2013, Sunil Tripathi a student at Brown university was false accused by "twitter detectives" of Bombings at Boston Marathon. He went missing after the incident. At the time of event, Sunil was missing, it turned out that the boy had committed suicide. His family had to face intense trauma and pain until he was located. Later it was also proved that Sunil had no connection with the incident.

Fake news can spread and hurt both people and society. It can also damage trust in reliable information sources and weaken social and political institutions. Therefore, before posting news online, it's crucial to double-check facts and evaluate sources.

In the following sections we will explore the dataset, design 5 machine learning methods to classify tweets as fake or real and evaluate each methods.

2 Data Analysis

2.1 Corpus Exploration

The dataset selected for the project is taken from the MediaEval2015 website. It consists of social media posts from Twitter. For simplicity we are only considering the textual contents of the posts, excluding other media (like images and videos).

The training dataset consists of a 14277 tweets, of which 47.22%(6742 tweets) of the data is labelled "fake", 34.46%(4921 tweets) is labelled "real" and 18.30%(2614 tweets) is labelled "humour". The aim of this project is to train models to predict these labels.

The definition of each feature used for this project is defined below.

tweetId	Unique id given to each tweets
tweetText	The textual content of the post(includes hashtags and the link for post)
userId	unique Id provided by twitter to each user of the platform
imageId	unique Id provided to each unique image
username	username of the user
timestamp	time at which the tweet was posted/retweeted
label	identifying the tweet as real/fake/humour

Table 1: Contents of the original dataset

The dataset consists of 119 different languages identified using googletrans translator[1]. In this dataset, 79.6% of the tweets are in English, 9.7% are in Spanish and the remaining 10.7% are in other languages.

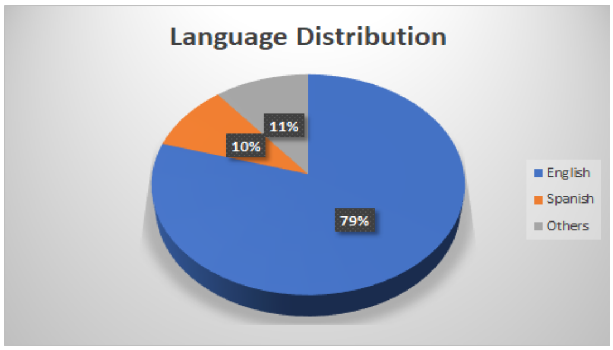

Figure 1: Langugae Distribution in the dataset

2.2 Dataset Extension

Next work involves, extraction of relevant features from the dataset that would be useful to predict the labels. We use the contents of tweets as predicotrs to predict our labels.

More features are extracted from tweet inspired by [2]. We used NLTK package to remove stop words from the tweets. Stop words are redundant commons words like "a","an","the",etc which don't add enough value to our models, hence deleting it seems sensible. NLTK package has stop words in 16 different languages which we used to clean our tweets.

tweetlenght	wordcount
containsExclaimation	containsQuestionmark
countQuestionmarks	countHashtags
countmentions	numNegativewords
numPositivewords	numNeutralwords

Table 2: Additional features used inspired from [Gupta's paper]

We used the NLTK package to do Parts of Speech Tagging to the tweets' content. Then created additional columns with respect to each 36 tags present in the package[3] which recorded whether a particular tag is present in the tweet or not. Using Opinion Lexicon, a list of positive and negative words curated by [4] We count the positive, negative and neutral words present in the tweets.

For the sake of homogeneity, we are considering humor labelled tweets as fake tweets.

3 Classifiers and Algorithms

The following part of the paper will discuss the algorithms designs that can be used to create classifiers.

3.1 Method I: English Tweets using Traditional ML Techniques

In this method, we consider only English data to create a model since it is the most commonly available. Hence, eliminating data from other languages, reducing our training set to 11383 tweets.

The features set used for this model is the extension from the original dataset. i.e the ones mentioned in Table2 and the 36 other added columns(made using POS Tags).

The following are the classifiers used for creating the model with their corresponding hyperparameter. We take an aggregate of the outputs of each of these algorithm and generate an output. Aggregate implies majority voting for all the outputs from the mentioned below classifiers.

3.1.1 Logistic Regression

Logistic regression is a linear classifier which predicts the probability that the output of X is 1. It uses the linear function $f(x) = b_0 + b_1x_1 + b_2x_2 + ... + b_nx_n$ where $b_0, b_1, b_2, ..b_b$ are called as weights or coefficients of the regression equation and X is the dataset. The logistic regression equation p(x) is the function of above linear equation which will output the probability of X equal to 1.

$$p(x) = 1/(1 + exp(-f(x)))$$

There are several methods to estimate the values of weights of the function.One of the most common one is using gradient descent and Maximum Likelihood Estimator.

Once a suitable set of weights are defined for p(x) we can use the function to make predictions for any output x_i. If the value of $p(x_i) > threshold$ then the predicted value is 1 otherwise it is 0. The value of threshold limit is generally kept as 0.5 but can be changed according to the problem statement.

Hyperparameter of logisitic regression:
solver : There are various solvers like newton-cg, lbfgs, liblinear, newton-cholesky, sag which will estimate the weights of the function
penalty : Also known as regularisation which penalise extreme values. Example: l1, l2, elasticnet, none
C: The value of C decides the weightage that is to be given to the dataset for fitting, higher the value of C, higher is overfitting. Example, 1e-5,0.0001, 0.001,0.01,0.1,1,10

3.1.2 Support Vector Machine

Support vector Machine (SVM) is a type of classifier which finds a hyperplane in n dimension to classify the data points. The aim to maximise the distance from both the class in such a way that a future point can be easily classified. The equation of hyperplane:

$$H : w^T(x) + b$$

Here b is the hyperplane intercept. In 2 Dimension, the equation of plane becomes : $Y = mx + c$

The aim is to find the weights of the equation such that it maximises the minimum euclidean distance of the point to the hyperplane . The distance of a point from the hyperplane is given by :

$$d_H(X_i) = \frac{|w^T(X_i) + b|}{||w||_2}$$

Where $||w||_2$ is called euclidean norm which is given by:

$$||w||_2 = \sqrt{w_1^2 + w_2^2 + w_3^2 + ...w_n^2}$$

Once the weight of hyperplane calculated, we can use the equation H to identify if the future point is positive or negative. By substituting the values of X_i in H , if the value is ¿=0, then the predicted value is positive else it is negative.

Hyperparameter of SVM:
kernel : This hyperparameter decides a mathematical function which transforms non-linear data to linear data for easy higher dimension. Like rbf, linear and sigmoid.
C : It decides the level of error model can accept. Values can be like 0.1,1,10,100,1000
gamma : Gamma is used to decide curvature weight of decision boundary. Example 1,0.1,0.5,0.001,0.0001

3.1.3 Decision Tree

Decision Trees is a classification method whose structure looks like a normal tree, consisting of of nodes, branches and leafs. The tree is representation of human like decision making process. It starts from a root node at top and ends at leafs at the bottom. Each node is connected by branches which act as certain characteristic to split the dataset into different nodes. And ultimately getting collection of dataset with similar attributes to make decision.

There are various method in deciding the criteria of split, the commonly used criteria as Information Gain, Gain Ratio and Gini Index. [5]

Hyperparameter that can be tuned for Decision Tree
max_depth : It is the maximum depth of the tree. If not decided, nodes will expand until all nodes are pure. We set the following hyperparameter 5,10,20,30,40,50,75,100,125,150,175,200
min_samples_split : Minimum samples required to split a node. We experimented with 5,10,11,15,20,30,50,100,1000
max_features : Minimum number features to decide for splitting. We experimented with auto, sqrt, log2 The are more hyperparametrs that one can tune. You can get a comprehensive list on sklearn's website.

3.1.4 Random Forest

Random Forest follows a similar structure trees to make a decision, however instead of relying on one tree it is an aggregate of multiple trees.

In random forest, for each tree, the feature and data points are selected at random from the dataset and then the tree is constructed. The predictions made is done by aggregating the decision of each tree(majority voting).

Hyperparameter we experimented with:
max_depth : 5,10,20,30,40,50
min_samples_split : 5,10,20,30,50,100,1000
max_features : auto, sqrt, log2

3.1.5 K Nearest Neighbour

K nearest neighbour is non parametric supervised learning algorithm. It relies on the labels of the input data, and does prediction based on the neighbourhood of the data point in the the space.

Let X_i be a random point for which the label is to be predicted. Then for each data point in the dataset we calculate the distance of the data point to X_i. We select a a number "k" to decide on the number of neighbours. We find the k nearest neighbours of X_i using the distance we calculated. Finally we predict the label for X_i using the aggregate of the labels of the neighbouring data points.(i.e majority)

In Method I, we take an aggregate of all the above mentioned algorithms.

3.2 Method II: All tweets to English and using Traditional ML Techniques

In this method, we use translate the non-English tweets into English, and then feed all the tweets to the ML models. We translate the dataset using the googletrans translator [**?**].

The feature set used for this model is similar to that of method 1. The classifiers used for this algorithm and parameters set against this dataset is same as method 1. Models used are: Logistic Regression, SVM, Decision Tree, Random Forest, K Nearest Neighbour.

3.3 Method III: English Tweets using BERT i.e Neural Network

Inspired from [6], we used the BERT model on our dataset. For this method we apply the pretrained BERT model directly on the tweets.

BERT, stands for Bidirectional Encoder Representation from Transformer is the state of the art model for Natural Language Processing. It makes use of the transformers [7] which is an attention model in it's natural form. Transformers are neural networks which connects each elements of input(words) to each other.

Unlike traditional language models, which used to read inputs in one direction (either left to right or right to left), BERT has the flexibility of reading in both direction. Trained on English Wikipedia Data, the main aim of the model is to predict masked words, in order to make model understand the context.

For this model,we take the subset of the dataset which only includes only english tweets from the dataset, which are identified by the googletrans. We exclude, the links attached in the tweets to avoid any un-recognisable text.

3.4 Method IV: All Tweets using BERT i.e Neural Network

In this method, we apply the same BERT model used in the previous method on the entire dataset. The non-english data is translated to english using googletrans api and then further fed to the BERT algorithm.

Although BERT is trained on 100 languages, there other 19 undetectable languages that can lead to inconsistency while training. Hence we decided to translate all the sentences to English and then apply BERT for classification.

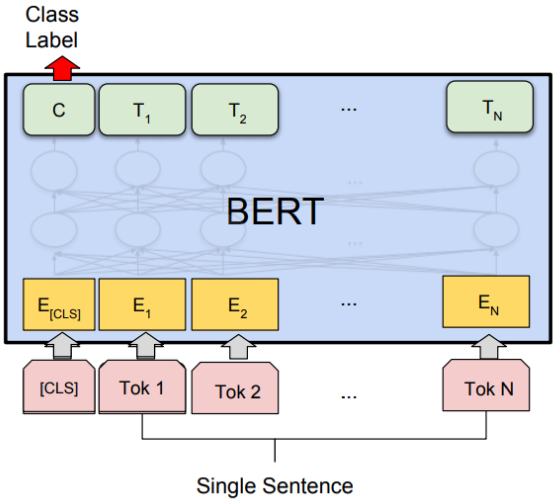

Figure 2: Workflow of Bert for text classification
[6]

3.5 Method V: All Tweets using BERT i.e Neural Network-2.0

In this method, we apply the BERT model of method 3 to the tweets. The dataset fed to this model are the translated tweets but also include the links attached in the tweets. This is done to see if the model could pick any patterns from the links if there is any. The results achieved by adding more information was better than tMethod II and IV , hence another reason to choose this method.

4 Evaluation

The first method involves selection of English data from the corpus and applying classical ML algorithms. This method is an ensemble of the given algorithms hence taking advantages of each of them. The classical approaches are intuitive and easy to explain. Hence making it easier to calculate therby taking less time and memory to train. Although Random Forest and Decision Tree take time to train but on average, this method is relatively faster. Since this method is an ensemble, it reduces the chance of overfitting. This method drops 3000 data points for training dataset because they are not in English. Due to which there's a possibility that some crucial information is lost with it. The methods are easy to implement, hence can be outperformed by complex models like Neural Networks. Some of the algorithms (KNN and Logistic regression) requires pre-processing, like feature scaling before implementation. A small change in dataset can impact the algorithms made by Decision Tree classifier and Random Forest Classifier.

The second method converts the non-English data to English using googletrans and then applies the same algorithms as method1. This way, we could get more data to train and capture some pattern which could be missing. The algorithms used here also inherit the same properties as that of method1 like: the ease of explanation; ease of calculation; less time required; avoiding overfitting. This method relies on the accuracy of googletrans. The errors made by googletrans can affect the overall model. The method requires extra time to translate the data. Like advantages, this method also inherits the disadvantages with method1 like: the algorithms can be outperformed by neural networks; few algorithms requiring pre-processing and the fact that decision tree and random forest get affected by change in datasets.

The third method involves selection of English tweets from the dataset and applying pretrained model BERT. BERT has a capacity to train on larger amount of texts. This method also takes advantage of transfer learning by using the existing weights from the pre-trained BERT model. We have also add a few extra layers i.e., fine-tuned it so that the model identifies more patterns for the specific case of identifying fake and real tweets.BERT has shown incredible results in it's application on text predictions and QnA. Like method 1, we lose 3000 data points as a result we can lose crucial information from the non-English tweets. BERT is slow to train because of its size, as there are a lot of weights to update. Hence this method is relatively slower. This method requires extensive fine tuning. Hence if the results generated are not converging, it is then advised to stop to tune again.

The fourth method uses google trans to translate non-English data to English and then feed

it to the pre-trained BERT model. This way we also don't lose 3000 datapoints that we did in method 3, hence gathering more information for pattern recognition. We also tune the model to our problem statement, hence making the model more accurate. Since BERT uses attention mechanism, while training the context of words are not lost till the end, leading to better results. Reliance on googletrans, can affect the quality of the model. As mentioned above, errors made by googletrans can reflect on the model we train. Also, it is hard to visualise the method through which this method comes up with a result. Similar to method 2, this method also relies on effective hyperparameter tuning.

The final method feeds translated tweets with the links attached to the BERT algorithm. Using this, we gain extra information for classification. This method inherits the properties of method 3 and 4. Hence the advantages and disadvantages of these methods are similar to method 3 and 4.

In order to rank each methods, we will use precision as our metric. The aim of this project is to identify which method that correctly classifies the fake posts.

Repercussions of misclassification a fake news as a real news is higher that classifying a real news as fake news. Since fake news can later be filtered out and then evaluated by moderators, in my opinion we should be concerned with misclassifying fake news. However from the company's point of view, a misclassification of real news as fake news can lead to banning of a genuine account, which can lead to loss of business for the company. In this experiment, we are going to find a model that is concerned with misclassification of fake news as real news. Hence, Precision is going to be my metric of assessment for the model. We will also look at F1 score and accuracy to reduce bias.

The following are scores of the metrics achieved by the algorithm ranked in decreasing order of their score. We applied these methods on a google colab and achieved the following results.:

Rank 1 : Method 5 - This method achieved the test accuracy of 0.678029, precision of 0.678029, and f1 score of 0.808125.

Rank 2: Method 2 -: This method achieved the test accuracy of 0.678029, precision of 0.678029 and f1 score of 0.808126. The individual algorithms achieved relatively higher scores than the aggregate. Which can be found in the table below.

Algorithm Name	**Accuracy**	**Precision**	**f1 Score**
LogisticRegression	0.677763	0.677944	0.807937
DecisionTreeClassifier	0.676698	0.677600	0.807179
RandomForestClassifier	0.678029	0.678029	0.808126
KNNClassifier	0.678029	0.678029	0.808126
SVMClassifier	0.678029	0.678029	0.808126
Aggregate Result	0.678029	0.678029	0.808126

Table 3: Scores achieved by method 2

Rank 3: Method 3- This method achieved the test accuracy of 0.527477, precision of 0.527477, and f1 score of 0.690651.

Rank 4: Method 4- This method achieved the test accuracy of 0.527477, precision of 0.527477, and f1 score of 0.690651.The score achieved is similar to that of method 3.

Rank 5: Method 5: This method achieved the test accuracy of 0.464197, precision of 0.49515 and f1 score of 0.613397. The individual algorithms achieved relatively higher scores than the aggregate. Which can be found in the table below.

Algorithm Name	**Accuracy**	**Precision**	**f1 Score**
LogisticRegression	0.49209	0.514321	0.580468
DecisionTreeClassifier	0.467111	0.496371	0.0.581426
RandomForestClassifier	0.512073	0.0.521719	0.660683
KNNClassifier	0.423813	0.468015	0.552972
SVMClassifier	0.617402	0.612549	0.673303
Aggregate Result	0.464197	0.49515	0.613397

Table 4: Scores achieved by method 1

4.1 Comments

The above results achieved are not high. The scores can however improve if more hyper parameter tuning is done in each methods. In Method I and II, for logistic regression, Recursive feature elimination and use of VIF (Variance inflation Factor) could be implemented to select the best features. VIF is used to removes features(dependant variables) which are co-related to each other. RFE removes features which have low impact on the dependant variable.

For Decision Tree and Random Forest, the results achieved were a cause of overfitting. To avoid it hyperparameter should be tuned by reducing depth of each tree and reducing samples at in each node.

In Method III, IV and V, more appropriate layers could be added to the BERT model for better results. Extensive hyperparameter tuning can improve these results. Experimenting with learning rates, shuffling the datasets and increasing the batch sizes can improve the results. We can observe that Method V game better results when more information was fed to the dataset i.e. when we added links to the model. Hence extracting more features from links for all the above methods can also improve the results.

Pre-trained model like RoBERTa [8] can also be implemented. It is trained for a longer time on a with bigger batch set, it is trained on longer sequences compared to BERT and dynamically changes the masks in the dataset for training.

Data on user, can aid in improving the accuracy of the model. Since, people who spread fake news have a possibility of spamming the information multiple times. Images attached with the posts can also aid in improving the accuracy.

5 Conclusion

In conclusion, we propose five approaches using state-of-the-art algorithms to perform classification to identify the authenticity of the tweets. On preliminary implementation of proposed methods we achieve a score of 0.67, however with implementation of suggestions and more training can solve the problem since it supported by wide literature. The implementaton/codes and the additional datasets used can be found on my github repository: `https://github.com/achyutk?tab=repositories`

References

[1] S. Han, "googletrans: Free Google Translate API for Python. Translates totally free of charge." [Online]. Available: https://github.com/ssut/py-googletrans

[2] A. Gupta, H. Lamba, P. Kumaraguru, and A. Joshi, "Faking Sandy: characterizing and identifying fake images on Twitter during Hurricane Sandy," in *Proceedings of the 22nd International Conference on World Wide Web*, ser. WWW '13 Companion. New York, NY, USA: Association for Computing Machinery, May 2013, pp. 729–736. [Online]. Available: https://doi.org/10.1145/2487788.2488033

[3] "Penn part-of-speech tags." [Online]. Available: https://cs.nyu.edu/~grishman/jet/guide/PennPOS.html

[4] G. Qiu, B. Liu, J. Bu, and C. Chen, "Opinion Word Expansion and Target Extraction through Double Propagation," *Computational Linguistics*, vol. 37, no. 1, pp. 9–27, Mar. 2011. [Online]. Available: https://direct.mit.edu/coli/article/37/1/9-27/2089

[5] D. K, "Top 5 advantages and disadvantages of Decision Tree Algorithm," Dec. 2020. [Online]. Available: https://dhirajkumarblog.medium.com/top-5-advantages-and-disadvantages-of-decision-tree-algorithm-428ebd199d9a

[6] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," May 2019, arXiv:1810.04805 [cs]. [Online]. Available: http://arxiv.org/abs/1810.04805

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," Dec. 2017, arXiv:1706.03762 [cs]. [Online]. Available: http://arxiv.org/abs/1706.03762

[8] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "RoBERTa: A Robustly Optimized BERT Pretraining Approach," Jul. 2019, arXiv:1907.11692 [cs]. [Online]. Available: http://arxiv.org/abs/1907.11692