MediaEval 2015 Tweet Classification Abiram Panchalingam: 33167354

Introduction and Data Analysis

A lot of events in real life trigger social media posts over many networks. Twitter is the popular one for sharing breaking news. However, a lot of fake news is also shared using engineered images and captions trying to describe the image as real. It is important to be able to classify such tweets correctly as 'fake' as not only will it misinform individuals but also news companies who will sometimes make news reports from tweets. This in turn will then negatively impact a larger group of people i.e., the general public.

Problem Characterization

The presented problem is a classification type problem. A set of tweets would be taken as input (test set) and each tweet should be classified as being either "real" or "fake". Ground truth labels are provided in the dataset to help our algorithms learn. A Machine Learning algorithm would likely be used to achieve our goal, especially supervised models since they will be able to make use of the training set of tweets given with labels to train and using generalisation, we can aim to classify tweets correctly on the test set and more importantly on a wider range of tweets in the real world. For example, in a real time system to classify tweets as they are posted which could have a genuine positive impact on social media by preventing fake news from spreading.

Five algorithms should be chosen and evaluated against each other based on their pros and cons and their suitability with the given problem and dataset. Wider literature will also support me in comparing the algorithms and how they have been successful in different applications.

The problem defined identifies posts as fake when they are: **1.** media from the past reposted as being a current event, **2.** digitally engineered and **3.** synthetic such as realistic artwork which is then presented as a real image.

We have not been given the media aspects of the tweets such as images and videos; only the tweet text and metadata has been given. So will have to use these to classify whether the tweet is 'fake' or 'real'. The text/caption of the tweet should be the most useful as natural language processing methods can be used to extract useful features which the machine learning algorithm can then learn. Features must then be extracted in a numerical formal so it can be read as input data into the algorithm. This can be done using various vectorisation methods such as Bag of words or TF-IDF [1].

Data Characterization

The "Mediaeval 2015" dataset that is given has 14483 tweets in the training set whilst the test set has 3781 tweets. This means that the training set has a satisfactory number of tweets to work with as a lot of behaviours and patterns can be learnt from a high number such as 14483.

The dataset consists of labelled data with each tweet labelled as either 'fake', 'real' or 'humor'. For the purposes of this report 'humor' will be deemed as 'fake'. So, there will be two classes to learn from and the same two classes to classify. However, the 'humor' label could come in useful to learn features from.

Tweets Counts

Overall, in the training dataset there are 9464 'fake' tweets and 5004 'real' tweets. As mentioned previously, the 'fake' tweets include the tweets labeled as 'humor'.

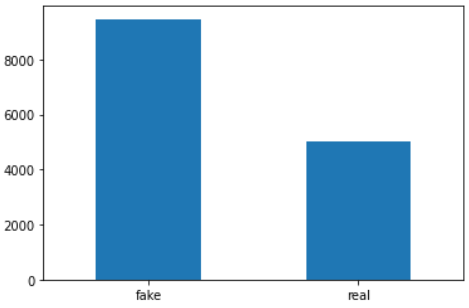

Figure 1: Fake 9464, Real 5004

Around 65% of the tweets are therefore ‘fake’. This is nearly twice as much as the number of tweets that are ‘real’. This could pose a problem as it means the dataset is not balanced so a potential ML algorithm could learn characteristics of the ‘fake’ class far better than the ‘real’ class. This in turn means the algorithm will perform better on test datasets where there are more ground truth ‘fake’ tweets and less so when there are more ground truth ‘real’ tweets. In the first instance we would get an incorrect indication of how good our ML algorithm really is.

We can remedy this bias towards fake tweets by oversampling our data. In essence we will be duplicating ‘real’ tweets so their features can be learnt and not overshadowed by the tweets in the ‘fake’ class [2]. However, since the bias isn’t 1:100 or more then it may not be worth oversampling as it is not a severe skew [2].

Common Keywords

Certain events are represented more frequently in the dataset than others. The word clouds below show which words are the most represented within the training set separated by the two classes. Please note the fake class also contains the humor class as detailed in the specification.

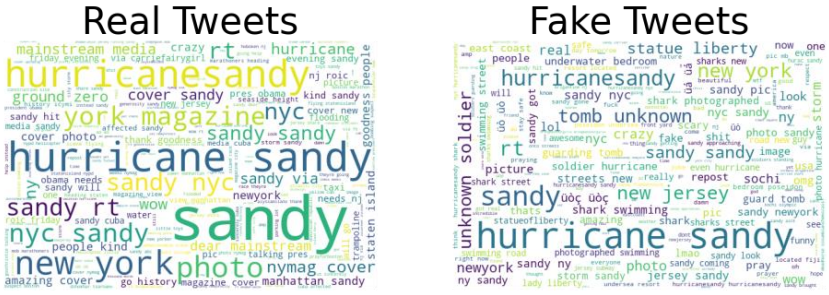

Figure 2: Word clouds for training set tweet texts

As you can see Hurricane Sandy features heavily in both word clouds. Most of the tweets in both sets seem to be related to disasters and mostly Hurricane Sandy in particular. In fact, 12507 out of 14468 tweets are about Hurricane Sandy which is a mammoth 86.4%. Using the “imageId(s)” field it was possible to extract the topic/event of each tweet which could then be counted:

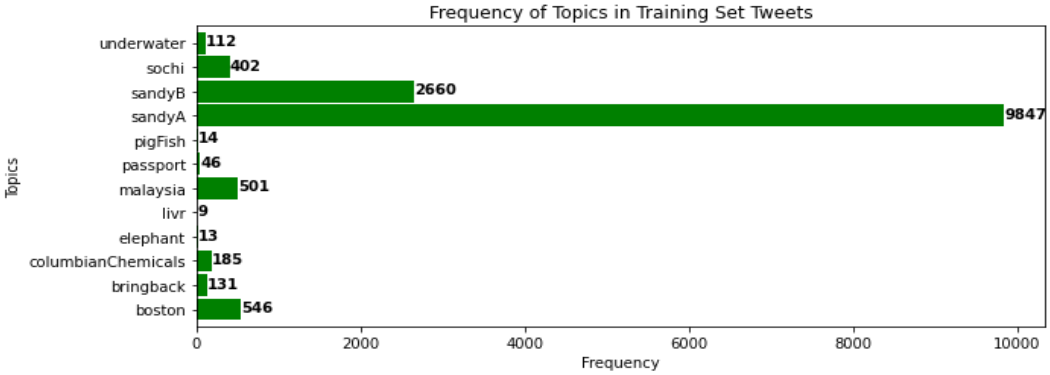

Figure 3: Training set count of events for all tweets

This clearly shows that the dataset has a heavy bias towards tweets related to Hurricane Sandy. This means the Machine Learning algorithm that will be learning from this dataset may not perform well in real world situations to identify fake tweets since they are unlikely to be related to Hurricane Sandy, as it is an event that happened around 10 years ago in 2012. Therefore, it is important for us to choose algorithms that can generalize well; so, algorithms that learn patterns but doesn’t overfit to the training data [3]. Some algorithms support regularization which is a term for various methods that allow us to decrease overfitting and therefore improve our algorithm’s generalization capability.

It is worth mentioning that the given test set does not have a bias towards Hurricane Sandy like the training set does. The test set covers 6 topics a little bit more evenly but with clear biases towards Nepal and Syrian Boy as shown in the table. The fact that Hurricane Sandy does not feature may mean accuracy scores such as F1, Precision and Recall may be affected for different classes and the overall set.

Topic	Frequency of Tweets
Eclipse	277
Garissa	78
Nepal	1359
Samurai	218
Syrianboy	1784
Varoufakis	61

Table 1: Test set count of events for all tweets

Tweet Lengths

The lengths of the tweets vary with the majority having more than 50 and less than 100 characters. Tweet lengths matters as some classification algorithms work better with shorter pieces of texts whilst others work better with longer.

```
Number of rows < 50: 5387
Number of rows < 100  and >= 50: 6694
Number of rows < 150  and >= 100: 2307
Number of rows >= 150 : 95
```

Figure 4: Training set tweet lengths count

Missing Tweets

There are 15 missing tweets in the dataset as shown in Figure 5. They are the same 15 tweets for each field so still 15 overall. They will have to be removed as they cannot be used in the training process without the label as we do not know whether the tweet is “fake” or “real”.

```
tweetId        0
tweetText      0
userId        15
imageId(s)    15
username      15
timestamp     15
label         15
dtype: int64
```

Figure 5: Missing Values in dataset

Algorithm Design

Five different algorithms will be chosen for this tweet classification task with varying pre-processing, feature selection, dimensionality reduction and machine learning methods. Algorithms may share certain aspects such as the feature selection method but will be different in some keyway which will be discussed. These differences will then enable me to critically compare them. Below is a brief discussion for each aspect of the overall algorithm.

Pre-processing

The decision was made to translate non-English tweets into English. Even though some important features can be lost this way such as local grammar and dialect, this decision was taken because most tweets were in English, so it makes sense to do this as the machine learning algorithm is likely to perform better when similar patterns are available. Translating non-English Tweets also keeps the door open for POS/NER tagging which may become useful. So, the steps taken were to use the langdetect module to detect language of each tweet text and then translate non-English tweets only using Google translator.

Then non-useful noise such as symbols, links, various punctuation and stop words were removed as they are not useful information. Removing such features will also lessen the size of the training set before use, thus lessening training/execution times.

Feature Extraction/Selection

The machine learning algorithms will all require inputs to be submitted in a numerical format; they cannot directly read text. Therefore, the tweet text and the various features we want to represent must be converted to some sort of numerical vector format, which a machine learning algorithm will then be able to interpret. A method can be used to assign weights [1] [4] to various words in the tweets which could allow for a higher accuracy in terms of classification. Feature selection methods include Bag of Words, Bag of N Words, TF-IDF, One-Hot Encoding and Word2vec and many more. All have different strengths and

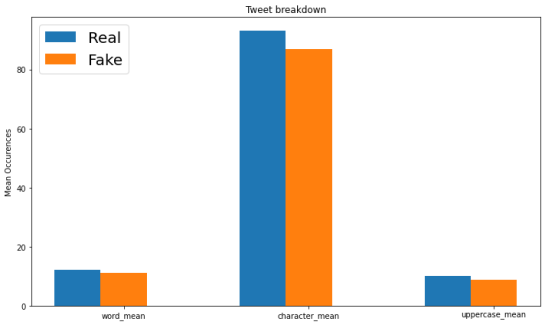

Figure 6: Disparity of count for various features between real/fake

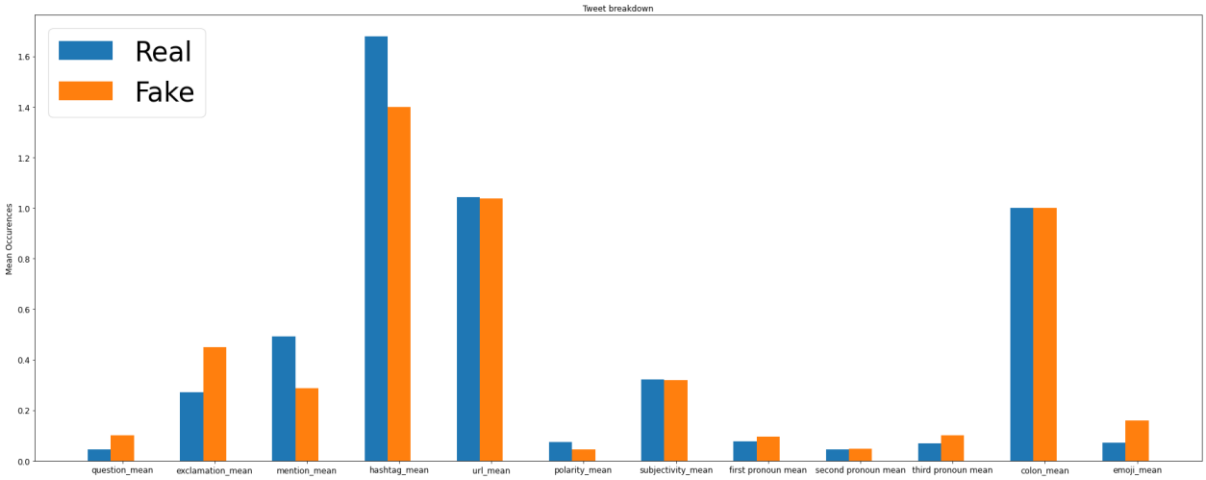

Figure 7: More disparity between features

weaknesses but fundamentally they do the same job which is to convert the text into a numerical format in some way.

In addition to the vectorization methods detailed above; we can extract our own features from the dataset. Firstly, the number of words, characters and uppercase characters were analyzed for every single tweet in the training set (Figure 6). The mean word count, character count and uppercase character count was calculated separately for “real” and “fake” tweets. It showed that perhaps “real” tweets had more characters, but this information had less variation than expected so is not as useful as the following features.

The features in Figure 7 have enough variation between “real” and “fake” tweets to warrant pursuing further. Especially the number of question marks which correlates well between the two classes. These features have been used with varying good results by [7]. In addition, sentiment analysis techniques such as text polarity and subjectivity were also extracted from the dataset as they vary slightly between the two classes.

Dimensionality Reduction

There will be many words in a normal text document, let alone a dataset containing 14,468 tweets. So, the whole classification process can be delayed by high time and memory complexity [1]. Dimensionality reduction can help manage this by reducing input features through various methods including Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), and non-negative matrix factorization (NMF) [1].

However, because the given dataset does not have technically a high dimension of attributes, we will not require dimensionality reduction. E.g., we do not have high-dimensional data [5]. Typically, high-dimensional data have more than 100,000 features so we can confidently say we

will not need such techniques. Therefore, Dimensionality reduction techniques will no longer be considered in the overall algorithm.

Machine Learning Algorithms

There are many classification algorithms available for use in this type of application. Deep learning techniques such as RNN, CNN and normal multi-layer perceptrons can all be adapted for use with text classification, tree-based algorithms such as Random Forest can also be used. One of the simplest classification algorithms is logistic regression [1]. All have different pros and cons which will be evaluated in the context of our problem and dataset.

Deep learning techniques have also been used with great results in the text classification space. CNNs are renowned for being able to extract spatial features with RNNs capable of extracting temporal features [17].

Whilst deep learning models do demonstrate great results, it wouldn't make sense to build hybrid deep learning models such as in [16]. This is because we are not required to classify large documents, only short tweets. So, we would not need to utilize the processing powers of both CNN and RNN architecture as in [16].

Below are the 5 different algorithm designs that have been chosen to be compared based off wider literature

Algorithm 1 – Naïve Bayes classifier

Pre-Processing: Translate tweets, clean text by removing URLS, symbols, erroneous words. Will also need to remove stop words and use Stemming/Lemmatization techniques to get more similar words which will be useful in the next step.

Feature Extraction: Bag of words. So, each word will become a new column essentially and their appearances in each tweet will be counted. Stemming/Lemmatization will be key as similar words could be shortened to become the same word (Stemming) or words can be simplified to a different more basic version (so we can achieve a common base) Both of which means the size of the resulting vector from Bag of Words will be smaller. Having less data will mean faster execution times and more links between tweets so in a way the quality of the data is increased.

ML Algorithm: Naive Bayes (and its variations) has been used quite often for text classification problems due to its computational efficiency and relatively good performance [6]. It works well in conjunction with the Bag of Words vectorisation method [8]. Naïve Bayes in fake news classification would work by calculating the probability of each word being in a "fake" or "real" tweet given that is "fake" or "real" [8]. This derives a formula which is in turn derived from Bayes' theorem.

Algorithm 2 – SVM Classifier

Pre-Processing: Same steps as for the previous algorithm

Feature Extraction: TF-IDF method will be used to extract numerical features from the dataset as it seems to work well an SVM classifier for text classification purposes [9]. TF–IDF is a step ahead of bag of words as it calculates the rarity of a word appearing in the dataset as well as it just appearing in a tweet. So, a term that is important for classifying fake tweets and is rare will be picked up by this vectorization method. However, semantics and the general context is still not picked up using this method.

$$tf\,idf\,(t, d, D) = tf\,(t, d)\ .\ idf\,(t, D)$$

Figure 8: TF-IDF Formula [10]

ML Algorithm: SVM with linear kernel as in [18] can be used for text classification purposes. SVM works by setting a large boundary between two classes to limit the effects of outliers in a dataset. This means for our problem the patterns of unusual cases that lead to fake tweets, and it's affects can be limited. For example, the tweets with the "humor" label may have unique patterns that are not related to most other fake tweets. Such patterns won't affect SVMs as much as other ML algorithms.

Algorithm 3 – Convolutional Neural Network (CNN)

Pre-Processing: Same steps as for previous algorithm.

Feature extraction: Word embedding techniques can be used such as Word2vec. Such techniques help us learn the semantic information in tweets. Also, the size of the resulting vector is small compared to Bag of Words/TF-IDF but the context/semantics in the text will still be preserved.

Word2vec itself is a 2-layer neural network [11] that works by vectorizing words and phrases. It's vector output format works well with deep neural networks such as CNNs. Word2vec establishes a word's meaning and its relation to other words by using its appearances in past text. In our case a word's appearance in one tweet will help Word2vec make assumptions when it is re-encountered in a different tweet/phrase. For example, the word "Sandy" is very prominent in our given dataset, Word2vec would be able to determine which words accompany "Sandy" in fake tweets since we are given the ground truth label. Thus, this very important pattern could be learnt using this feature extraction method but not the others previously mentioned.

ML Algorithm: Convolutional Neural Networks (CNNs) were primarily built for detecting images using their Convolutional filters which is built in as part of their architecture. The architecture of CNNs usually consists of convolutional layers, pooling layers, a fully connected layer and an output layer. The convolutional layers extract features from images (usually) and the pooling layers reduce the spatial size. It has been proven to work well with word embedding models [12]; but must deal with the difference in feature dimension size from images to text.

Algorithm 4 – Recurrent Neural Network (RNN) with LSTM

Pre-processing: No changes from previous algorithms

Feature Extraction: As this is also a deep learning model; It is likely to work well with similar word-embedding techniques as used in the previous algorithm. GlOve is a popular word embedding technique that has been proven to work well with RNNs.

ML Algorithm: Recurrent Neural Networks (RNN) are used normally with time-series data as its strengths lies in remembering previously learnt information from previous inputs.

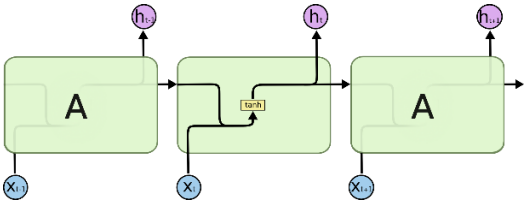

Figure 9: RNN Architecture [13]

A RNN thinks about the data of previous nodes in a very complex manner which allows for better semantic analysis of a dataset's structure. In our case a sentence can be evaluated more accurately using an RNN as it considers its semantics and syntax (e.g., how the word is used in the sentence). RNN primarily works using LSTM for text classification. LSTMs help tackle the vanishing gradient problem [13] that RNNs suffer from.

Algorithm 5 – Random Forest Classifier

Pre-processing: No changes to previous algorithms

Feature Extraction: TF-IDF as it is good at extracting the patterns from text that a technique such as Bag of Words may miss. Namely the effects of rare words in the dataset that might have influence of which class the tweet should belong to. Also, TF-IDF is proven to work well with Random Forest classifiers in similar use cases [14]. However Random Forests are insensitive to monotone transformations of features so using TF-IDF may not present any improvements over Bag of Words. This could be a disadvantage with our dataset as there are frequent words such as "Sandy" that won't be removed as a stop word but also doesn't have much of an impact on classification.

ML Algorithm: Random Forest works by generating random decision trees from the give training set.

Evaluation

The advantages and disadvantages of each algorithm is listed below; these are used to critically compare them against each other. The deep learning algorithms (CNN/RNN) are slower at training than the non-deep learning algorithms, but they learn deep patterns and are faster at predictions than Random Forest.

Advantages

- **Naïve Bayes**
  - Naïve Bayes is a quick and scalable algorithm – can make real time predictions
  - NB is suited for categorical input values which applies to our dataset when vectorized
  - Can perform better than a lot of other models with less data, especially deep learning models which require a lot of data
  - Can perform well when independence of features holds; this is true with our feature extraction step
- **SVM**
  - Effective when there are a high number of features -> this applies well in our case after vectorization
  - Relatively memory efficient when compared to other techniques, especially neural networks
  - SVMs have simple process patterns which is the reason it can predict quickly
- **CNN**
  - Can learn complex patterns in the data
  - Proven results in an online learning system; can handle new data seamlessly as it comes
  - Less training time needed than RNNs as no relationship built between timesteps
- **RNN**
  - Can detect patterns from temporal features; so, patterns from previous words in a sentence can be learnt

    - Complex functions can be modelled due to recurrent nature
    - Many different hyperparameters to tune, i.e. has scope to improve performance greatly
- **Random Forest Classifier**
    - Fast to train for text data
    - Less likely to overfit than other tree algorithms: important due to the bias in our dataset in terms of more fake tweets, and tweets being biased towards the Hurricane Sandy event
    - Robust to outliers; important as algorithm shouldn't learn irrelevant patterns

Disadvantages

- **Naïve Bayes**
    - Zero frequency phenomenon: if a training set doesn't account for a certain class, then it is given 0% probability and won't be classified at all.
    - Always assumes features are independent
    - Poor with co-related features; unlikely to be the case in our case
- **SVM**
    - SVM does not react well when characteristics from both sets overlap
    - SVM can underperform when the number of features exceed the number of samples; however, this does not affect our specific training set which would have 12238 distinct words (therefore features) and 14483 samples.
    - Training speed can be slow
- **CNN**
    - Requires a large amount of training data to improve accuracy
    - Can be expensive computationally to train models
    - Hard to fine-tune models without prior experience; also, sometimes hard to find the optimal setup even if after taking all necessary steps
- **RNN**
    - Needs a large amount of data to perform well
    - Computationally expensive due to recurrent nature
    - Can still suffer from the vanishing gradient problem even with LSTM
    - Could be prone to overfitting, need to be careful with this as there are clear biases in our dataset that do not reflect tweets in real life
- **Random Forest Classifier**
    - Slow at predictions after training: can manage by reducing features
    - Because Random Forest creates a lot of trees; it is computationally expensive. For our dataset this is important as we have a lot of tweets
    - Unlike deep learning models, it could struggle to learn low level representations in the data

Using these advantages and disadvantages we can score each algorithm based on three important criteria: Speed, Usability, Computational efficiency:

Naïve Bayes

Speed: 10 – Suitable for when real time predictions are required. Also quick at training as only probabilities are mathematically calculated.
Usability: 9 – Better for shorter documents (like tweets), however since the defined problem is for social media in general, we cannot give a perfect score as other social media platforms such as

Facebook allow longer posts. Also requires parameter optimization.
Computational efficiency: 9 - Low space complexity, even on large datasets [15]

SVMs

Speed: 9 – Generally fast at training and predictions, perhaps less so than NB. (more features make the SVM slower at learning)
Usability: 8 – Suitable for a variety of text lengths. Also requires parameter optimization
Computational efficiency: 7 – Computational cost varies with size of dataset

CNNs

Speed: 8 – Requires much more data than the non-DL techniques therefore training will take longer also backpropagation and the nature of how NNs train also adds to the training times
Usability: 10 – Strong at learning low-level features, dedicate methods available to optimize hyperparameters
Computational cost: 6 – Expensive to run as is a Neural Network; gradient descent, backpropagation all makes the algorithm more expensive to run CPU/GPU wise than non-DL methods

RNNs

Speed: 7 – In addition to the same reasons as CNN, the recurrent nature can make RNNs even slower
Usability: 8 - Prone to overfitting compared to CNN
Computational cost: 6 – Similar to CNN

Random Forest

Speed: 5 – Slow at predictions due to the nature of the architecture
Usability: 6 – Also prone to overfitting easily, can be hard to visually interpret
Computational cost: 7 – Needs to build a lot of trees considering the dataset

Based off these criteria, the ranking becomes: **(1)** - Naïve Bayes (**2) -** CNN **(3) -** SVM **(4)** - RNN **(5) -** Random Forest

Conclusion

In conclusion Naïve Bayes has been selected as the best model for this Fake Tweets classification problem. It seems to be the best all-rounder due to its speed, usability, computational efficiency and all-round suitability to this task. It can learn fast due to it only haven't to execute mathematical probability-based calculations. It doesn't require as much training data as the deep learning methods and requires less computational resources. However, if low level features need to be learnt, i.e., if the features that determine whether a tweet is "real" or "fake" is complex then a CNN is highly recommended. So, if the use case is to apply the algorithm in real-time systems and to detect fake posts over multiple social media networks then CNN would have to be placed above Naïve Bayes.

In the future we hope real-time systems can be produced to block fake tweets as fast as possible. Future improvements could include to use a less biased dataset in terms of class and contents of the tweets. If a more variety of data is used on a higher variety of events; then when a new event occurs the algorithm in question would have generalized enough to detect fake news with no problems. This means that there will be less people misled due to fake news as a result. Many of the chosen algorithms have very high F1 scores on the datasets they were tested on, however in the future we hope to see these algorithms implemented on a large scale over all the major social media platforms. This should mean fewer fake posts get to less people therefore

improving the knowledge of the public. This could have economic/safety benefits to a lot of countries where fake news badly affects the public's well-being.

References

[1] Kowsari, J. Meimandi, Heidarysafa, Mendu, Barnes, and Brown, 'Text classification algorithms: A survey', *Information (Basel)*, vol. 10, no. 4, p. 150, 2019.

[2] J. Brownlee, 'Random oversampling and undersampling for imbalanced classification', *Machine Learning Mastery*, 14-Jan-2020. [Online]. Available: https://machinelearningmastery.com/random-oversampling-and-undersampling-for-imbalanced-classification/. [Accessed: 29-Dec-2021].

[3] A. Stöffelbauer, 'Generalization, regularization, overfitting, bias and variance in machine learning', *Towards Data Science*, 26-Dec-2019. [Online]. Available: https://towardsdatascience.com/generalization-regularization-overfitting-bias-and-variance-in-machine-learning-aa942886b870. [Accessed: 02-Jan-2022].

[4] M. Siddharth, 'Feature extraction and embeddings in Natural Language processing', *Analytics Vidhya*, 20-Jul-2021. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/07/feature-extraction-and-embeddings-in-nlp-a-beginners-guide-to-understand-natural-language-processing/. [Accessed: 03-Jan-2022].

[5] K. El Bouchefry and R. S. de Souza, 'Learning in big data: Introduction to machine learning', in *Knowledge Discovery in Big Data from Astronomy and Earth Observation*, Elsevier, 2020, pp. 225–249.

[6] E. Frank and R. R. Bouckaert, 'Naive Bayes for text classification with unbalanced classes', in *Lecture Notes in Computer Science*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 503–510.

[7] 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 - WWW '13 Companion*, 2013.

[8] M. Granik and V. Mesyura, 'Fake news detection using naive Bayes classifier', in *2017 IEEE First Ukraine Conference on Electrical and Computer Engineering (UKRCON)*, 2017.

[9] M. G. Hussain, M. Rashidul Hasan, M. Rahman, J. Protim, and S. Al Hasan, 'Detection of Bangla fake news using MNB and SVM classifier', in *2020 International Conference on Computing, Electronics & Communications Engineering (iCCECE)*, 2020, pp. 81–85.

[10] A. Simha, Senior Software Engineer, and Kai Chatbot Team, 'Understanding TF-IDF for machine learning', *Capital One*. [Online]. Available: https://www.capitalone.com/tech/machine-learning/understanding-tf-idf/. [Accessed: 08-Jan-2022].

[11] T. Mikolov, K. Chen, G. Corrado, and J. Dean, 'Efficient estimation of word representations in vector space', *arXiv [cs.CL]*, 2013.

[12] F. Qian, C. Gong, K. Sharma, and Y. Liu, 'Neural User Response Generator: Fake news detection with collective user intelligence', in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, 2018.

[13] 'Understanding LSTM networks', *Github.io*. [Online]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs/. [Accessed: 09-Jan-2022].

[14] S. T. Selvi, P. Karthikeyan, A. Vincent, V. Abinaya, G. Neeraja, and R. Deepika, 'Text categorization using Rocchio algorithm and random forest algorithm', in *2016 Eighth International Conference on Advanced Computing (ICoAC)*, 2017, pp. 7–12.

[15] V. Jayaswal, 'Understanding Naïve Bayes algorithm', *Towards Data Science*, 08-Nov-2020. [Online]. Available: https://towardsdatascience.com/understanding-na%C3%AFve-bayes-algorithm-f9816f6f74c0. [Accessed: 11-Jan-2022].

[16] C. Zhou, C. Sun, Z. Liu, and F. C. M. Lau, 'A C-LSTM Neural Network for Text Classification', arXiv [cs.CL], 2015.

[17] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, 'Gradient-based learning applied to document recognition', Proc. IEEE Inst. Electr. Electron. Eng., vol. 86, no. 11, pp. 2278–2324, 1998.

[18] M. Goudjil, M. Koudil, M. Bedda, and N. Ghoggali, 'A novel active learning method using SVM for text classification', Int. J. Autom. Comput., vol. 15, no. 3, pp. 290–298, 2018.