# COMP3222 - MACHINE LEARNING TECHNOLOGIES COURSEWORK 2023/24

**Mohamed Lafi**
School of Electronics and Computer Science
University of Southampton
433378924
mal1g21@soton.ac.uk

## 1 Introduction

MediaEval 2015 dataset is concerned with classifying fake and real tweets based on Text and Photos uploaded. The specifications of this coursework requires the use of the textual information only. This can prove to be difficult as only relying on the text information does not provide full information for classification. [Oshikawa et al., 2020]

## 2 Data Analysis

The dataset given has a number of different features, namely:

- Tweet ID
- TweetText
- UserId
- imageId(s)
- username
- timestamp

The most useful features to be used would be the TweetText as that's where most of the context resides. Timestamp can be useful as well since the first tweet cannot be a fake tweet.

Tweet ID and username will be dropped as the former would not give any useful information on classification. For the latter, users can change their usernnames on twitter regularly, so marking fake usernames would be a waste of memory. UserId will be used instead as it cannot be changed. However if a Real User shares a tweet that can be identified as True, all their subsequent tweet would be marked as fake as they have been flagged by the system.

Another useful feature is the timestamp. As will be observed later, the first tweet is usually the Real one. Most of the Fake tweets are share after the first instance of a Real Tweet.

The ImageId field provide insightful context into the news topic the tweet is related to. This can be used, for example, if the image and text topic does not match, which mean it is a fake.

### 2.1 Label Classes

As this is a classification problem, the tweets need to be classified as True (Fake) or False (Real). However, within the training set another label (Humor) exists which for simplicity will be classified as Fake.

As can be seen from Figure 1, there is a slight class imbalance between Real and Fake Tweets. This imbalance can be made worse by changing the humor labels into fake labels. To have a more manageable balance between labels within the dataset, the Humor label will not be used.

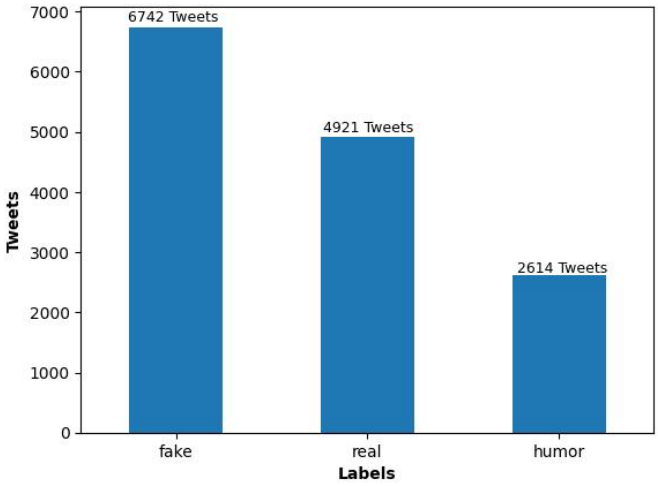


Figure 1: Number of Tweets Based on Label

### 2.2 User IDs

It has been noticed that there are a multiple of duplicate tweets as will. The number of duplicate UserIds in the training set is 1066. The total number of tweets within the set is 11663. That's nearly 10% of the tweets having users tweet a post multiple times. This can be useful for flagging fake accounts. If a certain User ID has been found to post fake tweets, then all subsequent tweets should be predicted as such.

### 2.3 Tweet Languages

The dataset contains multiple languages. An analysis is needed to check the variability among the tweets to mitigate any language imbalances.

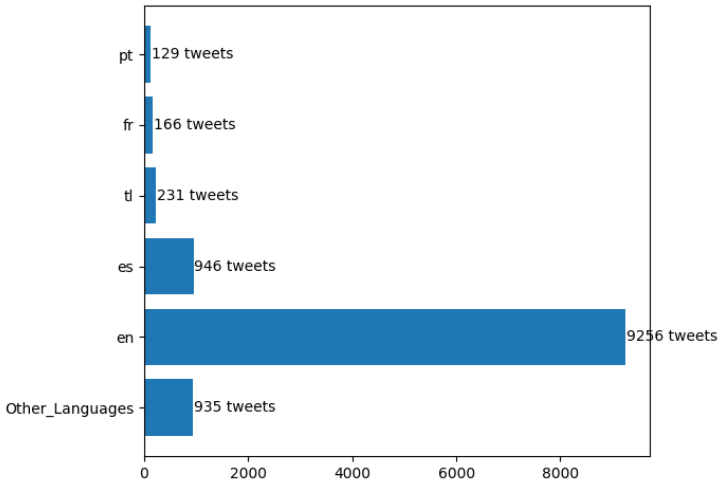


Figure 2: Number of Tweets per Languages

Figure 2 shows a comparison among tweets text languages[1]. As can be seen, the dominant language is English which representing roughly 80% of the total tweets. Languages that has under 100 Tweets has been combined into "Other_Languages". Since the English tweets would be easier to analyse, all other languages will be dropped and tested to check if the F1-Score would be affected.

[1]The langdetect python library is inconsistent with the language detections, so there might be slight variations in the number of tweets

### 2.4 Tweet News Topic

As Each tweet contains an "imagId" that corresponds to the news topic the tweet is concerned with, this has been extracted to analyse the balance between each topic.

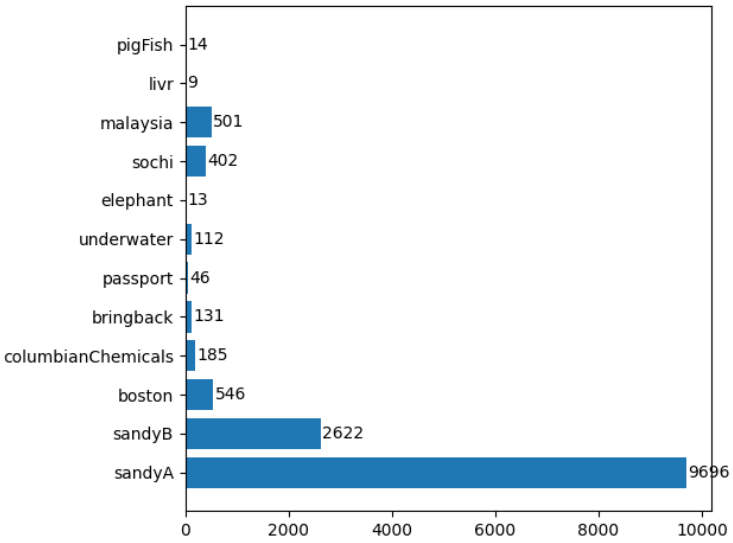


Figure 3: Number of Tweets per News Topic

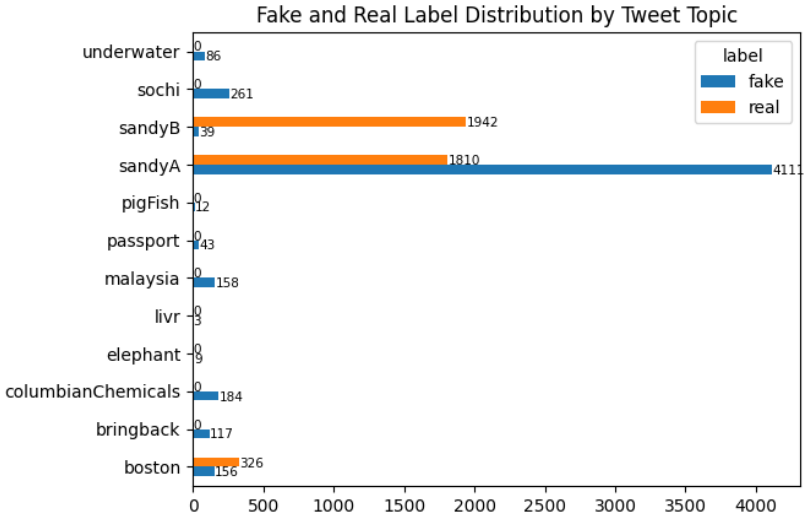


Figure 4: Number of Labels per Topic

Figures 3 and 4 show that the dominant topics is SandyA and SandyB. Only "boston" has a fairly balanced labels distribution. To reduce dimesnionality of the topics, those that have occurrences less than 100 will be dropped.

### 2.5 Linguistic Features Analysis

Taking a few sample tweets show that there are features within the text that can derive useful insights for classification. Previous work like [Antoun et al., 2020] has utilised the number of hashtags within a tweet, whether a link exists, an original tweet or a retweet and the number of emojis within the tweetText.

Table 1 shows a comparison between Fake and Real Labels as function of a tweet containing Emojis, Hashtags or Links. The number of emojis within the fake label exceeds the ones in the real labels by more than a double. For better understanding of those rations, histograms that analyse the difference among the values have been plotted. Analysing figures 5, 6 and 7, it can be noticed that there is a clear distinction between real and fake labels for the use of emojis.

| Name | Emojis | Hashtags | Links | Total |
|---|---|---|---|---|
| Fake | 391 | 3454 | 5150 | 5165 |
| Real | 140 | 3361 | 4078 | 4079 |

Table 1: Comparison Between existence of Emojis, hashtags and links within the labels

Some fake tweets use up to 40 emojis, while the real tweets are only constrained to a maximum of 5. The hashtag relationship shows a near identical pattern, this might add additional confusion for distinguishing between the two labels. Links show a similar pattern to hashtags, although some of the real tweets can have up to 4 links in a single tweet.

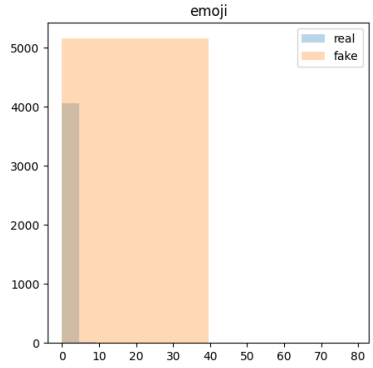


Figure 5: Emoji Distribution

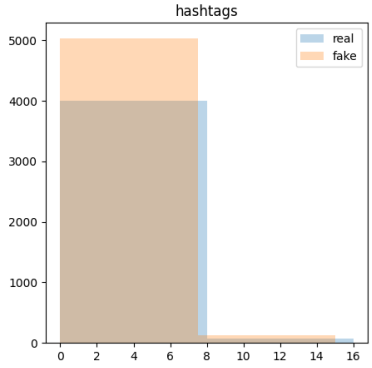


Figure 6: Hashtags Distribution

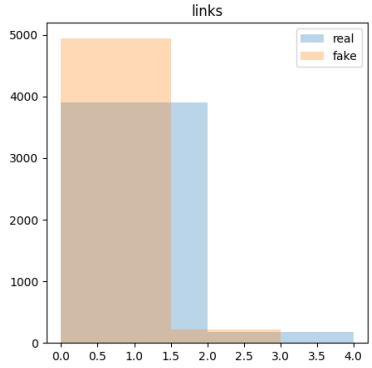


Figure 7: Links Distribution

Other papers has proposed sentiment analysis as an additional distinction between labels. [Oshikawa et al., 2020] surveyed multiple methods and found positve, negative or neutral sentiment to be the best.

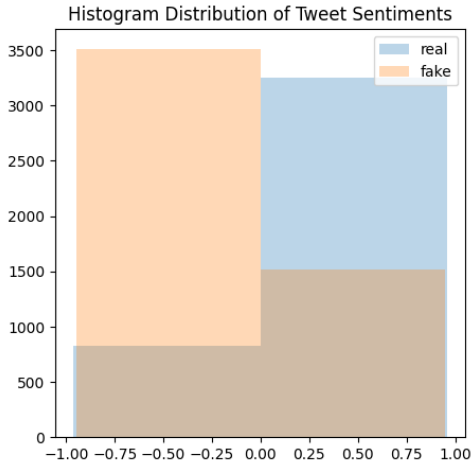


Figure 8: Sentiment Analysis Distribution

Figure 8 shows a distribution of the sentiments as a function of the labels. It is clear that real labels tend to have a more positive tune, as opposed to fake labels which is more clustered towards the negative sentiments.

# 3 Pipeline Design

## 3.1 Pipeline 1: LSTM

### 3.1.1 Feature Selection

As has been concluded from 2, the hashtags and links will not provide any insightful differences for the model. All links, hashtags and random numbers has been cleared from the tweets. There are also user mentions within the tweets which has been cleaned.

Emojis use unicoding which would waste the vocabulary look-up during the tokenisation stage. So, all emojis has been replaced with the "emoji" tag in the sentence. Then, all words have been converted to a lower case to reduce the number of vocabularies. English stop-words and punctuation's has been removed as well. The NLTK Library provides a useful Twitter Tokenizer which tokenizer/splits the string into individual words. This has been applied as a final step to split the string into a list.

### 3.1.2 Model Design

LSTM has been chosen for two reasons. It can process sequential data, as the tweets can be split into a list of strings. And it is known to capture long context dependencies [Oshikawa et al., 2020]. This can be useful in the context of a tweet as it can analyse features accross the tweet. However, as LSTM blocks are complex, the model can quickly overfit if trained for many iterations (epochs) or as the dataset is small and the model will be unable to generalise to unseen data. As has been discussed in 3.1.1, data cleaning has been done to ensure only the essential properties are learned.

As LSTMs accept inputs as a sequence of floats or integers, a text as a list of strings cannot be fed directly into the network. An embedding layer has been used instead. The layer uses a mix of one-hot encoding and dimensionality reduction to decode strings into numbers. It specifically has three different parameters that can be configured:

- **input_dim:** this specifies the number of vocabularies that can be looked-up in a table.
- **output_dim:** this specifies the dimensions of each word. This is where dimensionality is reduced as the less dimensions there are within a word, the shorter the input to the LSTM. This size needs to be set equal to the size of the input for the LSTM.
- **input_length:** the length of the string of input. This has been set to the tweet limit of 140.

Since the input_length is a constant and cannot be optimised, only the input_dim and output_dim values will be tuned.

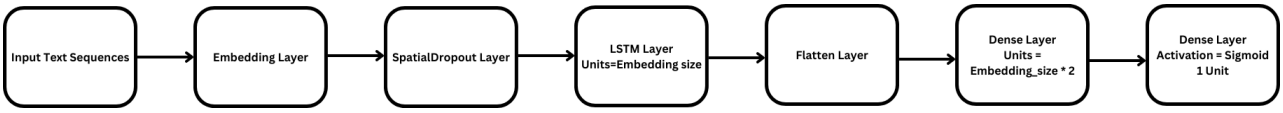


Figure 9: Pipeline Design 1 Model Architecture

Figure 9 shows an overview of the model. A SptialDropout layer has been used as a regularisation to mitigate the model from overfitting to the training data.The Flatten layer has been used after the LSTM Layer to enable the Dense layer to process the hidden vector output from the LSTM in a single dimension. The flatten layer will reduce the dimension and extend the values into a single vector. In order to not lose any of the values within the vector output, the dense layer has double the number of units of the LSTM. Finally, the output layer is a Sigmoid activation dense layer with 1 unit. As the model will deal will binary labels, only 1 unit (Neuron) is needed.

### 3.1.3 Hyper-Parameter Tuning and Evaluation [2]

Table 2 shows the choices of hyper-parameters and the ranges used. Number of embedding has been set to a low value to mitigate overfitting. Vocabularies, on the other hand, has been set to a higher value. Although training has been made only on the English tweets, it is unknown what langauge will be the most dominant in a real case scenario. Hence, the high value for vocabularies in the embedding layer. The threshold value has been set to a lower value as [Pastor-Pellicer et al.] suggest that the best threshold value that maximises the recall and precision would be lower than 0.5.

[2]There might be a variability in the hyper-parameter values and f1-score when verifying the results due to shuffling of the dataset and inconsistencies from some of the libraries used

It can be seen that the best dropout value is nearly the lowest. This is a result of the hyper-parameter maximising the f1-score of the training set. The best embeddings value was 16, this was to be expected due to the size of the training dataset used. During hyper-tuning, the maximum number of epochs has been set to 5 to prevent overfitting.

| HyperParameter | Embeddings | Vocabulary | Dropout | F1-Score Threshold | LR | Dense Activations |
|---|---|---|---|---|---|---|
| Starting values | 4 | 1000 | 0.05 | 0.1 | 1e-2 | ReLU |
| Ending values | 64 | 20000 | 0.7 | 0.4 | 1e-4 | tanh |
| Step | Powers of 2 | 50 | 0.05 | 0.1 | 1e-1 | Softmax |
| Best Values | **16** | **16100** | **0.15** | **0.3** | **1e-1** | **ReLU** |

Table 2: The Best HyperParameters Values

Figures 10 and 11 shows a comparison of the F1 Score, Precision and Recall for the training data. The validation split used was 0.2. The low validation f1-score can be explained due to the topics and labels being imbalanced as proved in 2.4.

By using the best values in 2 and a maximum number of 7 epochs, an **F1-Score of 88%** has been achieved.

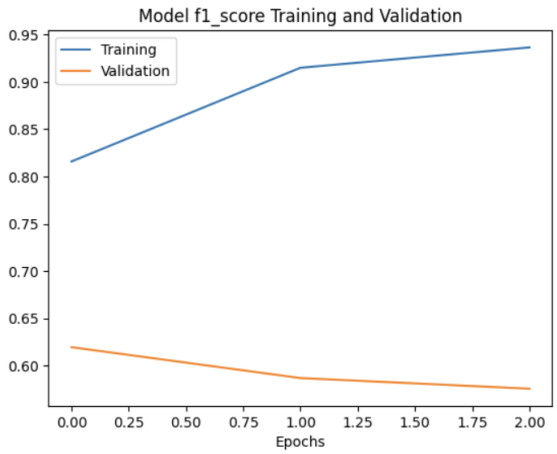


Figure 10: F1 Precision for train and val set

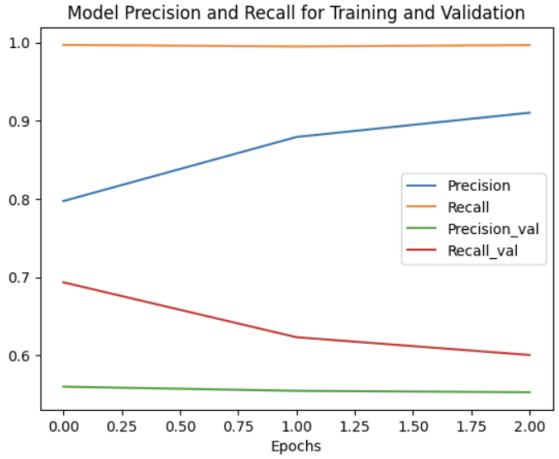


Figure 11: Precision and Recall for train and val set

### 3.1.4 Iterations

At the start of the model design, it was noticed that the model greately overfits to the training data. This was expected as this is a nature of LSTMs. The first regularisation method that has been tested was a Dropout layer with different values. Values higher than 0.8 (Percentage of dropped weights) slows down the learning process. Lower values has been tested with no improvements. [Tompson et al., 2014] encountered the same issue and proposed a different dropout idea. Instead of dropping different values randomly, an entire column or row is dropped. This SpatialDropout has been applied after the embedding layer. This has improved F1-score for the test set, therefore achieving a better generalisation ability. A theoretical explanation would be that since the tokenizer and, hence, the embedding layer is only fitted on the training data, dropping out entire words would allow the LSTM to be more robust to unseen words.

Another design choice that has proved to improve the model is Flattening the hidden vector output of the LSTM to feed it into the Dense Layer. The first iteration of the algorithm used two LSTM layers. One with a number of units equal to the number of embeddings, and the other having a unit of 1. This ensures that the Dense layer can process the last hidden vector. However, this reduced the accuracy of the model as only one feature vector is passed to the Dense layer.

## 3.2 Pipeline 2: AdaBoost

Tree-Based classification models has shown promising results. of Electrical et al. survey has reported great results based on the Kaggle-EXT Reuters Dataset. With Bagging, Adaboost and Decision Tree having an accuracy score of 0.944, 0.949 and 0.858 respectively.

This model uses a DecisionTreeClassifier as the base estimator. Adaboost is known for focusing on training weaker trees. This nature of AdaBoost can be used in this classification problem as the training dataset contains a class imbalance between the labels.

#### 3.2.1 Feature Selection

Only the humor labels has been dropped, and no other class balancing pre-processing has been used as the Adaboost learner improves weak learners. Since a Tree-Based classifier is being used, all values needs to be integers. So the following features will be used:

- Number of Emojis
- Number of Hashtags
- Number of Links
- Positive, Negative or Neutral Sentiment
- Length of the tweet
- News Topic
- Tweet Language
- Readibility of the Tweet

All of this features has been chosen based on the analysis made in 2. The topics, tweet languages has been mapped into a set and the index is applied to the field to indicate each type.

#### 3.2.2 Hyper-Parameter Tuning and Evaluation

For Hyper-Parameter tuning, parameters such as number of estimators and max_depth of the tree has been used. Max depth has a low value due to the training dataset not having variations. Due to computational constraint, Randomised hyper-parameter tuning has been used instead of Grid-search. A low number for the max depth of the decision tree has been chosen to prevent the tree from over-fitting.

| **Hyper-Parameters** | **Number of Estimators** | **Decision Tree Max Depth** |
|---|---|---|
| Starting values | 50 | 1 |
| Ending values | 250 | 25 |
| Step | 1 | 50 |
| Best Values | **64** | **3** |

Table 3: The Best Hyper-Parameters Values

Table 3 shows the best hyper-parameters for the models. The achieved Test F1-Score for this model has been **0.81%**. As compared to results in [Wang et al.], those classifiers achieve a far better accuracy. This is mainly due to the fact that the sentiment and grammar values, from figure 12, are not accurate as the tweets are not translated to English. It can also be seen from figure 12 that the most important features were topic, user-Id and tweet-Length. Note that the language feature has been dropped as its importance was less than 0.001, which This adds a unnecessary complexity to the classifier.

### 3.3 Conclusion and Future Improvements

An LSTM-Based model with embedding has been designed with an F1-Score of 88% on the testset. Pre-processing steps has been taken to ensure the best class balance to reduce bias. Different configurations with all tenable Hyper-Parameters has been tested to gauge the best parameters for the model and mitigate model overfitting.

For future improvements, the full dataset can be used to enable better generalization ability. Translations can be made to other parts of the dataset as the LSTM quickly over-fits to the English Tweets as has been seen in 2. Improvements can be made to the linguistic features as well. Instead of removing emojis, links and hashtags. Those can be tagged within the text to give better context. The sentiment can be added as a tagging process. For example, adding either positive, negative or neutral at the end of the tweet to indicate the tone of the tweet. These can be combined with an

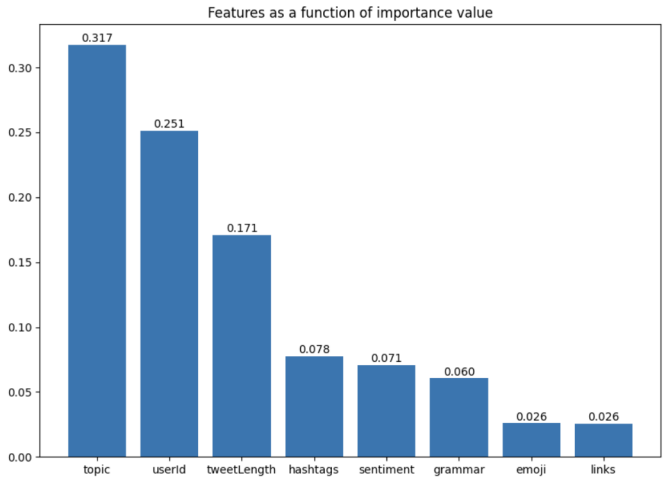


Figure 12: Feature Importance

Attention Layer before or after the LSTM Layer as suggested by [Wang et al.]. The Layer will put embassies on which part of the text contains the most important context, enabling the LSTM to give better predictions as it has more context from the emojis, hashtags, sentiment and links.

Another improvement would be to implement the F1-Score as a custom loss function. As has been seen on 3.1.3, although the Binary Cross-Entropy loss training improves precision, it cannot guarantee the optimisation of the value between precision and recall. [Pastor-Pellicer et al.] Suggests an approximation of the F1-Score to allow differentiation, and hence enabling the calculation of the gradient for back-propagation to adjust the weights. This would highly optimise the recall and precision score to maximise the F1-Score.

Adaboost classifier has been used for the second pipeline by utilising numerical features within the model without the use of embedding or tokenization from the tweet text. Due to the computational limitations, randomised Hyper Parameter tuning has been done to the best configuration of the model. Grid-Search optimisation is guaranteed to achieve better score as it searches through all possible combinations within the tweet.

Another improvement would be to translate the tweets and add a grammar and tone/sentiment dimension to the dataset. As has been concluded from 3.2.2.

In conclusion, Occam's razor principle applies. Both models had a similar F1-Score, although one is much more complex than the other.

## References

Ray Oshikawa, Jing Qian, and William Yang Wang. A survey on natural language processing for fake news detection. pages 11–16, 2020. URL `https://www.snopes.com/fact-check/`.

Wissam Antoun, Fady Baly, Rim Achour, Amir Hussein, and Hazem Hajj. State of the art models for fake news detection tasks. pages 519–524. Institute of Electrical and Electronics Engineers Inc., 2 2020. ISBN 9781728148212. doi:10.1109/ICIoT48696.2020.9089487.

Joan Pastor-Pellicer, Francisco Zamora-Martínez, Salvador España-Boquera, and María José Castro-Bleda. Lncs 7902 - f-measure as the error function to train neural networks.

Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann LeCun, and Christopher Bregler. Efficient object localization using convolutional networks. 11 2014. URL `http://arxiv.org/abs/1411.4280`.

Institute of Electrical, Electronics Engineers, and PPG Institute of Technology. *Proceedings of the 5th International Conference on Communication and Electronics Systems (ICCES 2020) : 10-12, June 2020*. ISBN 9781728153711.

Yequan Wang, Minlie Huang, Li Zhao, and Xiaoyan Zhu. Attention-based lstm for aspect-level sentiment classification.