Analysis of Fake News Classification Methods

Akshit Nanda

May 2, 2021

Contents

1 Abstract

The advent of global connectivity with the World Wide Web and the massive adoption of social media platforms like Facebook, Twitter and Reddit has paved the way for misinformation dissemination at an unprecedented scale. While the current usage of these platforms enables the sharing of ideas and content between users, some of the content online is misleading with no relevance to real world facts. Even an expert from a domain has to employ multiple aspects before giving a verdict on the truthfulness of an article. Automated classification of a given text article as misinformation / fake news is a challenging task. This work explores the various techniques used by researchers for the automated classification of news articles and tries to draw a comprehensive comparison between them. We have also experimentally evaluated different methodologies on 2 real world datasets for Fake News Classification.

2 Introduction

According to digital global report 2020[6], the number of users for digital media reached 4.75 billion, and the social media users reached 301 million in 2020. This literally brings the users within one click of accessing any information they need. This transformation has also brought along some challenges, and the detection of misinformation / Fake News on the network is an uphill task for today's researchers.

Social media can have two sides for news consumption, i.e., can be used to keep the community updated about current events and happenings, on the other hand, can be a source of spreading false news and misinformation. However, social media is a low cost, quick access, and fast distribution of news and information and to know what is happening worldwide. Zhang et al. [17] summarize the basic characteristics of fake news as velocity, volume, and variety, and hypothesise a definition of fake news: "fake news refers to all kinds of false stories or news that are mainly published and distributed on the Internet, in order to purposely mislead, befool or lure readers for financial, political or other gains." Fake news has become a focal point of discussion in the media over the past three years due to its impact on the 2016 US Presidential election.[2] Reports have estimated that the success of deception detection for humans is restricted to 54%[5] Therefore, there is a need for automated methods for detecting misinformation and being able to distinguish between Fake and Real News.[14] There is sufficient evidence that people have reacted absurdly to news that later proved to be fake[12] [15]. One recent example is the spread of novel corona virus, where fake reports spread over the Internet about the origin, nature, and behavior of the virus[9]. The situation worsened as more people were exposed to the fake content and mass hysteria took over. This work aims at analysing the computation techniques used for Fake News classification and compare the effectiveness (accuracy) of these methods. The next section describes briefly the various approaches used by different researchers, which form the basis of our analysis.

3 Related Work

Some of the early research around Fake News analysis aimed at the classification and understanding of misinformation in text. Rubin et al.[13] did an in-depth analysis to overview the requirements for a Fake News corporus for predictive modelling and suggest the pros / cons of different types of Fake News like Serious Fabrication, Large Scale Hoaxes or Humorous Fakes. The authors argued that the advancement in natural language processing (NLP) and deception detection techniques could be helpful in detecting Fake News. However, the lack of available corpora for classification modeling was an important limiting factor in designing effective models to detect misinformation. We will restrict the scope of research to papers which have performed computational analysis over fake news corpora for providing a solution to the classification problem.

Ahmed et Al.[1] presented a new ISOT Dataset which was entirely collected from real world sources. They collected news articles from Reuters.com (News website) for real news articles. Fake news were collected from a fake news dataset on kaggle.com. The data set contained fake news items from unreliable web sites that Politifact (a fact checking organization in the USA) has been working to stamp out. They had a total dataset of about 45k news articles in total (evenly distributed among real and fake news) but they only worked with a small subset of this data catering specifically to political domain texts. Using a combination of data pre-processing and linguistic feature extraction, the authors compared linear and non-linear classifiers From the results in their experiments, Linear-based classifiers (Linear SVM, and Logistic regression) achieved better results than nonlinear ones. However, nonlinear classifiers achieved good results too with the Decision Tree achieving 89% accuracy. The highest accuracy was achieved using Linear SVM at 92%.

One of the other studies on fake news detection and automatic fact-checking with more than a thousand samples was done by using Wang[16]. The LIAR dataset contains 12.8K human-labeled short statements from POLITIFACT.COM. The statements were labeled in six different categories, such as pants fire, false, barely true, half true, mostly true, and true. The study used several classifiers such as logistic regression (LR), support vector machine (SVM), a bidirectional long short-term memory (Bi-LSTM) networks model), and a convolutional neural network (CNN) model for predicting fake news. For LR and SVM, the study used the LIBSHORTTEXT toolkit and significant performance on short text classification problems was observed. The study compared various techniques using text features only and achieved an accuracy of 0.204 and 0.208 on the validation and test sets. Due to overfitting, the Bi-LSTMs did not perform as well on this model. However, the CNN outperformed all models, resulting in an accuracy of 0.270 on the holdout data splitting.

Similarly, another study compared three datasets such as LIAR datasets[16], fake or real news dataset from Kaggle[1], and another dataset generated by collecting fake news and real news from Internet. The study made a comparison among various conventional machine learning models such as SVM, LR, decision tree (DT), Naive Bayes (NB), and K nearest neighbor (KNN), respectively, using lexical, unigram, and bigram techniques with term frequency and inverse document frequency (TF-IDF). Furthermore, many CNN models such as NN, CNN, LSTM, Bi-LSTM used with Glove embedding and character embedding to train the model. They found that the performance of the LSTM model highly depends upon the size of the dataset and the result showed that NB, with n-gram (bigram TF-IDF) features produced the best outcome of approximately 0.94 accuracy with the combined corpus dataset.

Conversely, the LIAR study indicated that the CNN model outperformed the LIAR dataset. However, the later work showed that the CNN model is the second-best for all the datasets. The NB model showed the best performance for the LIAR dataset with 0.60 accuracy and 0.59 F1-score.LSTM-based models showed the best outcome on the combined corpus dataset, where both Bi-LSTM and C-LSTM produced an accuracy of 0.95 and F1-score of 0.95.

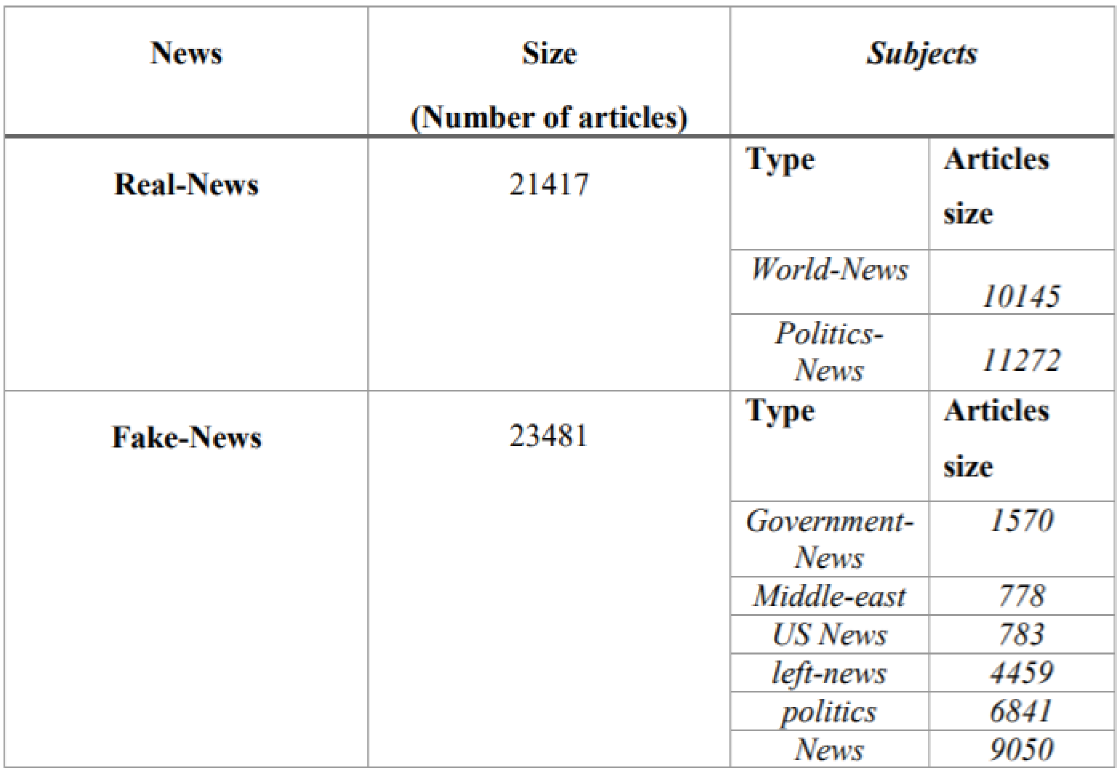
In another study DeepFake[10], the authors proposed a fake news prediction model focused on media news content, social media content, and the combination of both. The implementation is carried out on the BuzzFeed dataset and the PolitiFact dataset, where the input is fed to the 3-mode sensor, and the matrix-tensor factorization is used to extract the features. Furthermore, the extracted features are used for classification using XGBoost and DNN. The experimental results show that the proposed model achieves 85.86% accuracy on BuzzFeed dataset and 88.64% accuracy on PolitiFact dataset. The authors, however, did not demonstrate the feature extraction using content and context.

One of the more recent works by Goldani et al.[7] introduces a neural network technique for the identification of fake news. The primary aim of this research is to identify fake news considering the length of the text and to apply different features, with non-static embedding being used during the learning phase. The research is carried out on two datasets, namely ISOT and LIAR datasets. The findings show that the proposed model achieves 99.1% accuracy for ISOT training dataset and 99.8% accuracy for ISOT testing dataset, 40.09% accuracy for liar training dataset and 39.5% accuracy for liar testing dataset.

Another study [3] used a hybrid aproach by combining deep learning and NLP semantics on the ISOT and LIAR datasets. They compared some classic ML models like MNB, SGB, LR, DT and SVM to deep learning models like CNN, LSTM and CapsNet. CapsNet (Capsule Networks) outperformed the other models with an accuracy of 0.649 on the LIAR dataset, primarily improved because of the integration of Named Entity Recognition (NER) sentiments.

4 Datasets

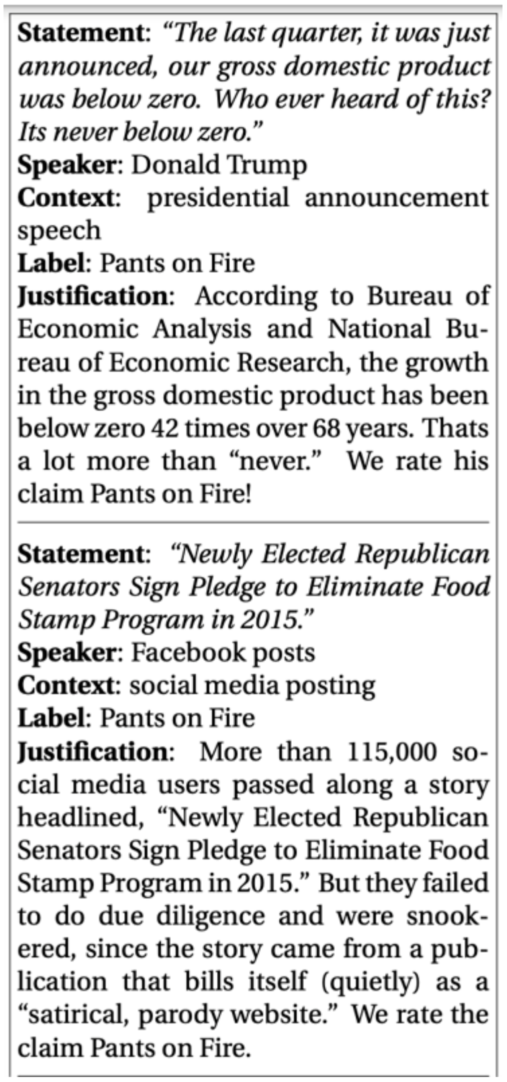
The first dataset is called the “ISOT Fake News Dataset”[1] which contains both true and fake articles extracted from the World Wide Web. The true articles are extracted from Reuters.com which is a renowned news website, while the fake articles were extracted from multiple sources, mostly websites which are flagged by Politifact.com. The dataset contains a total of 44,898 articles, out of which 21,417 are truthful articles and 23,481 fake articles. The total corpora contain articles from different domains, but most prominently target political news. The dataset contains the title of the article and the full text along with date and author metadata details.



News	**Size (Number of articles)**	***Subjects***	
		Type	**Articles size**
Real-News	21417	*World-News*	*10145*
		Politics-News	*11272*
Fake-News	23481	**Type**	**Articles size**
		Government-News	*1570*
		Middle-east	*778*
		US News	*783*
		left-news	*4459*
		politics	*6841*
		News	*9050*

Description of the data in the ISOT Fake News Dataset

The second dataset for our analysis is the LIAR[16] dataset which includes 12,836 real-world short statements, and each statement is manually-labeled with six-grade truthfulness (’pants-fire’ ’false’ ’half-true’ ’mostly-true’ barely-true’ ’ ’true’). The information about the subjects, party, context, and speakers are also included in this dataset.

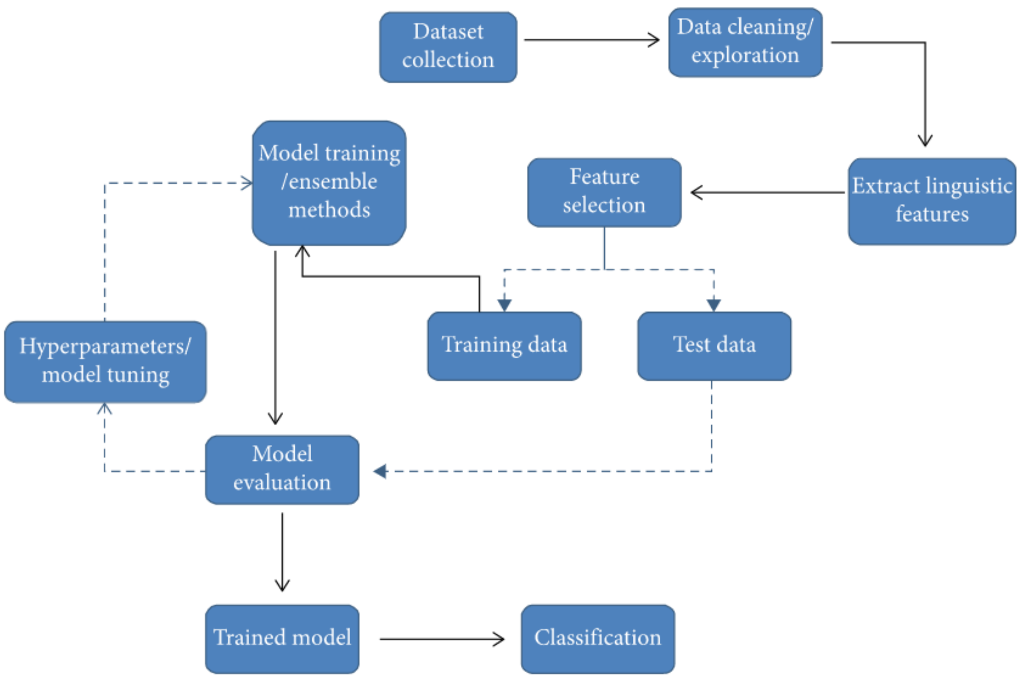


Statement: *"The last quarter, it was just announced, our gross domestic product was below zero. Who ever heard of this? Its never below zero."*
Speaker: Donald Trump
Context: presidential announcement speech
Label: Pants on Fire
Justification: According to Bureau of Economic Analysis and National Bureau of Economic Research, the growth in the gross domestic product has been below zero 42 times over 68 years. Thats a lot more than "never." We rate his claim Pants on Fire!

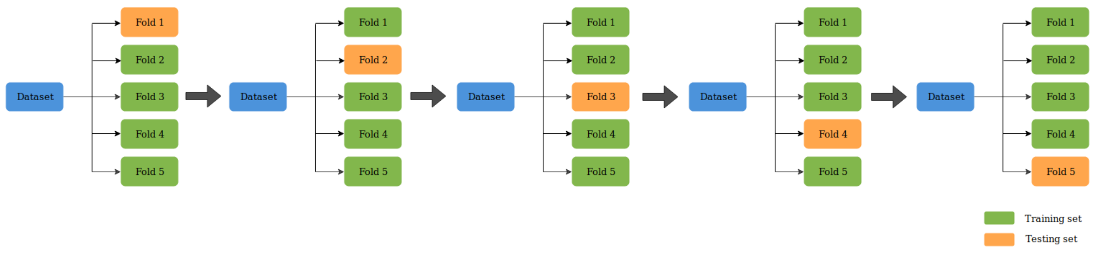
Statement: *"Newly Elected Republican Senators Sign Pledge to Eliminate Food Stamp Program in 2015."*
Speaker: Facebook posts
Context: social media posting
Label: Pants on Fire
Justification: More than 115,000 social media users passed along a story headlined, "Newly Elected Republican Senators Sign Pledge to Eliminate Food Stamp Program in 2015." But they failed to do due diligence and were snookered, since the story came from a publication that bills itself (quietly) as a "satirical, parody website." We rate the claim Pants on Fire.

Examples of the data in the LIAR Fake News Dataset

5 Proposed Experimental Pipeline


Dataset collection
Data cleaning/ exploration
Extract linguistic features
Feature selection
Model training /ensemble methods
Training data
Test data
Hyperparameters/ model tuning
Model evaluation
Trained model
Classification

The above figure describes the basic steps in a classification pipeline. Starting with the dataset in question, some pre-processing and cleaning techniques are applied to the text in order to remove punctuations, stop words, stemming of words and lemmatization. Once the data is sanitized (and cleared of null values if any), the next step is feature extraction/selection. Two of the most commonly used feature extraction methods are CountVectorizer and TfIDF, which basically translate words into vectors based on their occurrence / frequency in the document. Once this data is split into Train-Test subsets, these transalted vectors can be used as inputs to our predictive models for training. Alternatively, for comparison we also included pre-trained word embeddings like Word2Vec and GoogleNews-pre-trained word embeddings, which provide a better representation of words rather than just relying on the count/frequency in the restrictred scope of the input.
Hyperparameter tuning can be optionally performed to extract the best permutation of parameters which result in the highest accuracy. This trained model can then be used for evaluation of our test set to get accuracy / precision scores. This is the same steps we have used in our experimental analysis of commonly used modelling techniques. Additionally we also performed 5-fold cross validation on the training set to get an estimate range on the outcome of our classifier to validate our results.



5.1 Data Processing

- Tokenization
  Tokenization was the first step in our pre-processing pipeline. This is an approach to break the given sentence / document into smaller chunks or units called *tokens*, while removing certain unnecessary characters like special characters and punctuations. This is a basic step to compute a set of vocabulary for the given text/document to be used.

- Count Vectorization
  Following the Bag-of-Words approach for text processing , Count Vectorization is an approach to create sparse matrix for our text data , to calculate the frequency of occurrence of our tokens / words in the vocabulary, across multiple texts or documents. This involves the creation of vectors that have the dimensionality of the size of our vocabulary and mapping the number of times a word appears in a given text/document. Stop words, which are common to any language, can be efficiently removed using this technique to focus on the words specific to the given domain of the problem.

- Tf - Idf
  Only the frequency of occurrence of a word in a given document is not enough, and the frequency of the word appearing across documents can be a useful metric for powerful prediction. Term Frequency is a technique to achieve the frequency of the word appearing in a document . Inverse document frequency refers to the metric of how rare or common a word is across multiple documents.

  $$TF(word) = \frac{\#\, of\, times\, word\, appears\, in\, document}{Total\, \#\, of\, words\, in\, the\, document}$$

  $$IDF(word) = \log\left(\frac{Total\, \#\, of\, documents}{\#\, of\, documents\, which\, contain\, word}\right)$$

  The product of the TF and IDF results is known as the TF-IDF score. If this score is high, then the term is a rare term and can be used for distinction between different documents.

6 Methods

6.1 Logistic Regression

We we are classifying text on the basis of a wide feature set, with a binary output (true/-false or true article/fake article), logistic regression model is used, since it provides the intuitive equation to classify problems into binary or multiple classes.[11] We performed hyperparameters tuning to get the best result for both individual datasets, while multiple parameters are tested before acquiring the maximum accuracies from LR model. Mathematically, the logistic regression hypothesis function can be defined as follows:

$$h_\theta(X) = \frac{1}{1 + e^{-(\beta_0+\beta_1 X)}}$$

Logistic regression uses a sigmoid function to transform the output to a probability value; where the objective is to minimize the cost function to achieve a desired probability. The cost function is calculated as

$$Cost(h_\theta(x), y) = \begin{cases} \{log(h_\theta(x)) & y = 1 \\ -log(1 - h_\theta(x)) & y = 0 \end{cases}$$

6.2 Multinomial NB

Multinomial Naive Bayes algorithm is a probabilistic learning method that is based on the Bayes theorem and predicts the tag of a text. It calculates the probability of each tag for a given sample and then gives the tag with the highest probability as output. Based on the conditional probability of events A and B

$$P(A|B) = P(A) * P(B|A)/P(B)$$

we can calculate the posterior probability of all conditions and predict the outcome based on the highest probability.

6.3 Support Vector Machine

Support vector machine (SVM) is another model for binary classification problem and is available in various kernels functions [4]. The objective of an SVM model is to estimate a decision boundary (or hyperplane) on the basis of feature set to classify data points. The dimension of hyperplane varies according to the number of features being used for classification. As there could be multiple possibilities for a hyperplane to exist in an N-dimensional space, the task is to identify the plane that separates the data points of two classes with maximum margin. A mathematical representation of the cost function for the Linear Kernel SVM model is defined as

$$J(\theta) = \frac{1}{2}\sum_{j=1}^{n}\theta_j^2$$

such that

$$\theta^T x^i \geq 1, y^i = 1,$$
$$\theta^T x^i \leq -1, y^i = 0$$

6.4 K Nearest Neighbours (KNN)

KNN is an unsupervised machine learning model where a dependent variable is not required to predict the outcome on a specific data. Enough data is provided to the model to train and decide which neighbourhood to assign to a data point. KNN model estimates the distance of a new data point to its nearest neighbors, and the value of K estimates the majority of its neighbors' votes. For K=1, the nearest class is assigned to the data point. The mathematical formulae to estimate the distance between two points for our study is the Minkowski Distance defined by the following formula

$$\Big(\sum_{i=1}^{k} |x_i - y_i|^q\Big)^{\frac{1}{q}}$$

6.5 Decision Trees

Decision Tree is a supervised machine learning model which can be used for classification using a process known as binary recursive partitioning. This is an iterative process of splitting the data into partitions, and then splitting it up further on each of the branches. The entropy of a data point is defined as

$$-\sum_{i=1}^{c} P(x_i) log_b P(x_i)$$

Using the decision algorithm, we start at the tree root and split the data on the feature that results in the largest information gain (IG) which is defined as

$$IGT(T, A) = Entropy(T) - \sum_{v \in A} \frac{|T_v|}{T}.Entropy(T_v)$$

where T is the target column, A is the variable column and v is the value for the decision column.

6.6 Random Forest

Random Forest is another supervised machine learning model which improves over decision trees. RF consists of a large number of decision trees working independently to predict the outcome of a class and the final prediction is based on the majority vote from these components. The error rate in RF classifiers is low because of the low correlation among separate trees. For this classification approach, we have used Gini Index as the cost function to estimate the data split. Gini Index (G_{ind}) is defined as

$$G_{ind} = 1 - \sum_{i=1}^{c} (P_i)^2$$

6.7 LSTM

The Recurrent Neural Network (RNN) are a type of feed-forward neural network which perform exceptionally on different tasks but suffer from the problem of gradient vanishing gradient. In order to overcome these issues Long Short-term Memory network was developed [8]. The network performs better because of the three memory and cell memory states defined as

$$X = \begin{cases} h_{t-1} \\ x_t \end{cases}$$

$$f_t = \sigma(W_f.X_{bf})$$

$$t_t = \sigma(W_i.X + b_i)$$

$$o_t = \sigma(W_o.X + b_o)$$

$$c_t = f_{tt-1} + i_t(W_c.X + b_c)$$

$$h_t = o_t(c_t)$$

where W_i, W_f, $W_o \in R^{dX2d}$ are the weighted matrices and $b_i, b_f, b_o \in R^d$ are biases of LSTM to be learned during training, parametrizing the input, forget and output gates. $\bigodot$ is the element-wise multiplication and x_t is the input to the LSTM unit for the word embedding w_t.

7 Quantitative Validation Method

Since we are dealing with a classification problem the key performance parameters are the following :

7.1 Accuracy

Accuracy is often the most used metric representing the percentage of correctly predicted observations, either true or false. To calculate the accuracy of a model performance, the following equation can be used:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Generally, we can attribute high accuracy as a stable metric of a good model, but this is not the only metric for analysing a model. For a classification problem False Positives and True Negatives coupled with accuracy give us a clearer picture of the performance of a model.

7.2 Precision

Precision is a measure of how precise or accurate are the model predictions. It is defined as the ratio of true positives to all events predicted as true.

$$Precision = \frac{TP}{TP + FP}$$

7.3 Recall

This is a metric to figure out how many of the actual positives our model captures by labeling them as positives.This is a model metric used when the cost of false negatives is high.

$$Recall = \frac{TP}{TP + FN}$$

7.4 F1 Score

This is a function of recall and precision which seeks to create a balance between the two, and is used as a truer measure of performance, rather than relying on Accuracy only.

$$F1\,Score = 2 * \frac{Recall * Precision}{Recall + Precision}$$

8 Results

8.1 ISOT Dataset

Most of our linear models performed exceptionally well for the ISOT dataset achieving not only high accuracy but incredible precision and recall scores. In addition to linear models, LSTM was also employed against this dataset using Word2Vec and GoogleNews-pre-trained word embeddings. The results from our evalutaion are summarized below

Model Used	Accuracy	Precison	Recall	F1 Score
CountVectorizer + Logistic Regression	99.60%	0.996	0.997	99.66%
TfIDF + Logistic Regression	99.47%	0.993	0.997	99.47%
CountVectorizer + MultinomialNB	97.85%	0.977	0.977	97.75%
TfIDF + MultinomialNB	96.88%	0.969	0.968	96.73%
CountVectorizer + KNN	71.68%	0.828	0.515	63.55%
TfIDF + KNN	93.30%	0.847	0.933	88.82%
CountVectorizer + LSVM	99.67%	0.996	0.997	99.66%
TfIDF + LSVM	99.47%	0.994	0.996	99.45%
CountVectorizer + Decision Tree	99.69%	0.996	0.995	99.68%
TfIDF + Decision Tree	99.46%	0.996	0.992	99.44%
CountVectorizer + Random Forest	95.54%	0.972	0.933	95.25%
TfIDF + Random Forest	95.56%	0.973	0.933	95.27%
Word2Vec + LSTM	99.70%	0.998	0.995	99.69%
GoogleNew pre trained + LSSTM	99.48%	0.997	0.992	99.46%

8.2 Binary Classification on LIAR Dataset

We analysed the performance of different models along the same metris and parameters for the LIAR Dataset. We merged the 6-labels into groups of 3 to have True and False labels in the modified input data. This was performed to compare the results achieved from the last experiment on the ISOT dataset. The linear models could not achieve comparable performance on this dataset, which provided some interesting insights. The results are summarised in the table below

Model Used	Accuracy	Precison	Recall	F1 Score
CountVectorizer + Logistic Regression	55.12%	0.446	0.390	41.65%
TfIDF + Logistic Regression	55.47%	0.443	0.339	38.43%
CountVectorizer + MultinomialNB	56.41%	0.409	0.143	21.27%
TfIDF + MultinomialNB	59.12%	0.555	0.014	2.80%
CountVectorizer + KNN	54.29%	0.447	0.488	46.70%
TfIDF + KNN	54.53%	0.434	0.364	39.68%
CountVectorizer + LSVM	55.12%	0.446	0.396	42.01%
TfIDF + LSVM	56.77%	0.447	0.232	30.62%
CountVectorizer + Decision Tree	54.06%	0.442	0.465	45.37%
TfIDF + Decision Tree	53.35%	0.431	0.433	43.26%
CountVectorizer + Random Forest	58.65%	0.480	0.103	17.02%
TfIDF + Random Forest	57.24%	0.454	0.215	29.23%
Word2Vec Pretrained + LSTM	59.12%	0.571	0.011	2.25%
GoogleNewsPretrained + LSTM	53.82%	0.427	0.373	39.87%

8.3 Multilabel Classification on LIAR Dataset

Conidering the 6-label problem in the original LIAR dataset, we compared the performance of linear and bagging ensemble models . As expected, linear models provided very low accuracy scores for multilabel classification whereas the non-linear models performed slightly better. The results for our experiment are summarized in the table below

Model Used	Accuracy	Precison	Recall	F1 Score
CountVectorizer + Logistic Regression	19.67%	0.183	0.179	18.01%
TfIDF + Logistic Regression	20.85%	0.199	0.187	18.88%
CountVectorizer + MultinomialNB	24.14%	0.202	0.199	17.5%
TfIDF + MultinomialNB	24.02%	0.201	0.192	13.37%
CountVectorizer + KNN	20.61%	0.210	0.178	11.74%
TfIDF + KNN	21.67%	0.192	0.192	18.73%
CountVectorizer + LSVM	55.12%	0.446	0.396	42.01%
TfIDF + LSVM	21.67%	0.168	0.181	16.80%
CountVectorizer + Decision Tree	17.90%	0.164	0.163	16.29%
TfIDF + Decision Tree	18.13%	0.167	0.166	16.66%
CountVectorizer + Random Forest	22.02%	0.180	0.188	18.18%
TfIDF + Random Forest	21.90%	0.179	0.188	18.20%

9 Discussion

Through the results of our experiments we were able to conclude that multi-label classification for Fake News is a challenging task and current models cannot be aptly trusted for accuracy. Binary classification is able to see better performance, but is restricted to the quality of the dataset and the comprehensive nature of labels. Even though we ran experiments for binary classification for both ISOT and LIAR dataset, the results from the two models were very different.
After observing the discrepancy between the performance of linear models for the ISOT dataset and the LIAR dataset, we inspected the ISOT dataset for possible information

leaks , which might be responsible for such high accuracies, even with simple experiments. Information leak is when training and/or validation data contains information about the target which would not realistically be available at the time of prediction.
Upon quick lookup, we noticed that all the real news articles were collected from Reuters and had the text ”Reuters” in almost all the entries. Using just a quick string match for ”Reuters” we were able to achieve 99.6% accuracy since only 170 (0.38%) of true news documents do not contain the text (Reuters), and only 9 (0.02%) fake news documents do contain the text (Reuters). We removed the substring from the text and the results were slightly diminished from before, but not too much. We saw a decrease of about 1-2% in the prediction accuracy with Multinomial NB (accuracy 96.5%) and LR (98.8%). Although the models still perform exceptionally, the decrease in performance confirms our reservations about the predictability of some key elements in the dataset. Some other issues int the ISOT dataset include URLs in date columns for fake dataset and presence of additional / unnecessary labels in the subject column.

10 Future Directions

From the results we can see that LSTM paired with pre-trained word embeddings can achieve better results than feature extraction backed-linear methods. We think that word embeddings play a key role in determining the performance of text based classification tasks and the use of advanced word vector techniques like transformers (BERT, RoBERTa models) might be able to take the analysis of fake news forward.
Use of hybrid deep neural networks is also a direction for the future with specialised neural networks like GAN and CapsNet. Although the feature extraction from text is limited by an upper bound, use of semantic features and metadata associated with articles can also be leveraged towards a more comprehensive and complete modelling technique.

References

[1] Hadeer Ahmed, Issa Traore, and Sherif Saad. “Detection of Online Fake News Using N-Gram Analysis and Machine Learning Techniques”. In: Oct. 2017, pp. 127–138. ISBN: 978-3-319-69154-1. DOI: 10.1007/978-3-319-69155-8_9.

[2] Hunt Allcott and Matthew Gentzkow. “Social media and fake news in the 2016 election”. In: *Journal of economic perspectives* 31.2 (2017), pp. 211–36.

[3] Adrian MP Braşoveanu and Răzvan Andonie. “Integrating Machine Learning Techniques in Semantic Fake News Detection”. In: *Neural Processing Letters* (2020), pp. 1–18.

[4] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods.* Cambridge University Press, 2000. DOI: 10.1017/CBO9780511801389.

[5] Sherry Girgis, Eslam Amer, and Mahmoud Gadallah. “Deep learning algorithms for detecting fake news in online text”. In: *2018 13th International Conference on Computer Engineering and Systems (ICCES).* IEEE. 2018, pp. 93–97.

[6] *Global Social Media Stats - DataReportal – Global Digital Insights*. URL: https://datareportal.com/social-media-users.

[7] Mohammad Goldani and Saeedeh Momtazi. “Detecting fake news with capsule neural networks”. In: *Applied Soft Computing* 101 (Mar. 2021), p. 106991. DOI: 10.1016/j.asoc.2020.106991.

[8] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory”. In: *Neural Comput.* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. URL: https://doi.org/10.1162/neco.1997.9.8.1735.

[9] Jinling Hua and Rajib Shaw. “Corona virus (Covid-19)“infodemic” and emerging issues through a data lens: The case of china”. In: *International journal of environmental research and public health* 17.7 (2020), p. 2309.

[10] Rohit Kaliyar, Anurag Goswami, and Pratik Narang. “DeepFakE: improving fake news detection using tensor decomposition-based deep neural network”. In: *The Journal of Supercomputing* 77 (Feb. 2021). DOI: 10.1007/s11227-020-03294-y.

[11] Tom Michael Mitchell. *The discipline of machine learning*. Vol. 9. 2006.

[12] Amanda Robb. “Anatomy of a fake news scandal”. In: *Rolling Stone* 1301 (2017), pp. 28–33.

[13] Victoria L. Rubin, Yimin Chen, and Nadia K. Conroy. “Deception detection for news: Three types of fakes”. In: *Proceedings of the Association for Information Science and Technology* 52.1 (2015), pp. 1–4. DOI: 10.1002/pra2.2015.145052010083.

[14] Kai Shu et al. “Fake News Detection on Social Media: A Data Mining Perspective”. In: *CoRR* abs/1708.01967 (2017). arXiv: 1708.01967. URL: http://arxiv.org/abs/1708.01967.

[15] Jacob Soll. “The long and brutal history of fake news”. In: *Politico Magazine* 18.12 (2016), p. 2016.

[16] William Wang. “”Liar, Liar Pants on Fire”: A New Benchmark Dataset for Fake News Detection”. In: (May 2017).

[17] Xichen Zhang and Ali A. Ghorbani. “An overview of online fake news: Characterization, detection, and discussion”. In: *Information Processing Management* 57.2 (2020), p. 102025. ISSN: 0306-4573. DOI: https://doi.org/10.1016/j.ipm.2019.03.004. URL: https://www.sciencedirect.com/science/article/pii/S0306457318306794.