Rumour/Fake news detection

Ruhma Mehek Khan
2018362
IIIT - Delhi
ruhma18362@iiitd.ac.in

Ria Gupta
2018405
IIIT - Delhi
ria18405@iiitd.ac.in

Prakriti Garg
2019439
IIIT - Delhi
prakriti19439@iiitd.ac.in

Abstract

With the advent of technology and the development of social media platforms like Twitter and Facebook, it has become very easy for anyone to share news updates with a large number of people. Additionally, with easy access to features such as retweet, share, forward, etc. these news spread like wildfire reaching lakhs of people within minutes. This makes the identification of fake news extremely crucial. Fact-checking any news can allow people to stay away from reacting and taking action on fake news. Such tools can also be extremely useful for news houses to fact-check their news before they share it with the masses. This project aims to develop and propose a fake news classifier. In this project, we tried out various approaches and models to detect fake news. Our best performing model was BERT which was able to correctly differentiate fake news from genuine news 98% of the time.

1 Introduction

Fake news has existed since ages, but with the advent of the internet, the amount of people it can reach within a short span of time has increased tremendously. The effects of fake news can be catastrophic, it can cause riots, sway public opinion, create ideological barriers, etc. To stop its spread we need to identify it first and do that swiftly. Recent studies have shown that fake news spreads faster as they seem more enticing to users. This is leveraged by advertisers and can be used by anyone to serve their purpose, which makes it even more dangerous.

Hence, we formulate our problem statement as follows: Building and deploying a web app that can detect fake news efficiently. We have deployed our model for detecting fake news on https://fake-news-detection-nlp.herokuapp.com/

The pipeline of our model is summarised in Figure 1. We first take our input data, which comprises of title, author and text of the news article along with its label- fake/real. This dataset is cleaned using the preprocessing steps. The clean dataset is then vectorised using supervised and unsupervised learning algorithms. These embeddings are then given as inputs to Machine Learning and Deep Learning algorithms which perform feature learning and then classify the input dataset as Real/Fake news.

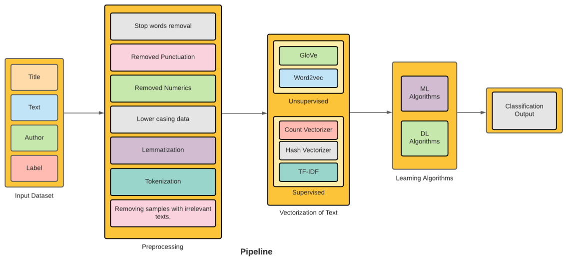

Figure 1: Flow diagram of the project

In the further sections, we give details of our approach.

2 Methodology

2.1 Dataset

We downloaded the publicly available dataset of Fake News detection from kaggle[1]. This dataset had 5 columns, the id of news articles, the title of news articles, news texts, the author names (who reported that article), and the label of whether it is classified as fake news or not. (Here, 1 would mean the news is unreliable, also called fake news, and 0 would mean a reliable piece of information) Our dataset has 20.8k samples.

2.2 Preprocessing

2.2.1 Stopwords

We found and removed all occurrences of stopwords present in the nltk stopwords corpora. According to the power law, the most frequent words

[1] https://www.kaggle.com/c/fake-news/data

in any text are commonly used words like “a, the, is, my etc”. However these words add little helpful information, hence we removed these words during the preprocessing steps.

2.2.2 Numerics and special characters

We have removed all occurrences of numerics (integers, float etc) and special characters such as punctuation marks from the dataset using a regex that only retains alphabet characters.

2.2.3 Lemmatization

Lemmatization uses vocabulary and morphological analysis of words to reduce them into lemmas. It ensures that all forms of the same word are grouped together so that they are understood by our model in a similar way. WordNetLemmatizer module in NLTK is used to perform Lemmatization

2.2.4 Lower case text

Lowering the case of all texts reduces the unique number of words in the corpus, which helps in making our dataset uniform and clean.

2.2.5 Tokenization

We have used NLTK word tokenizer to split a sentence into different tokens.

2.2.6 Padding

The sentences were padded to ensure all input sentences have the same length.

2.2.7 Removing irrelevant texts

We scanned our dataset and removed all those texts that are of length 0, 1 and 2.

2.3 EDA

- The raw dataset has a total of 20800 entries, out of which 10413 fake news and 10387 true news.
- After preprocessing, we got 10334 entries of fake news and 10387 entries of true news.
- Word Clouds are pictorial representations of words and greater importance is given to higher frequency words. The wordcloud of all tokenized words present in the entire dataset can be seen in Figure 3
- The wordcloud of all tokenized words present in the fake news dataset can be seen in Figure 4

Figure 2: Pie chart of fake and true news

Figure 3: Wordcloud of all tokenized words

- The wordcloud of all tokenized words present in the true news dataset can be seen in Figure 5
- The density of number of characters representing the length of characters present in real and fake news can be seen in Figure 6
- The density of number of words representing the length of text present in real and fake news dataset can be seen in Figure 7
- The top bigrams present in fake news dataset can be seen in Figure 8
- The top 10 bigrams present in true news dataset can be seen in Figure 9

2.4 Vectorization

We used various vectorization techniques to convert plain text to machine-interpretable vectors.

2.4.1 OneHot Encoding

One Hot Encoding is a common way of preprocessing categorical features for machine learning models. We used TensorFlow’s one_hot method that takes in an input text, and vocabulary size, and outputs the integer encoding of each word in a given input text. All sentences were converted

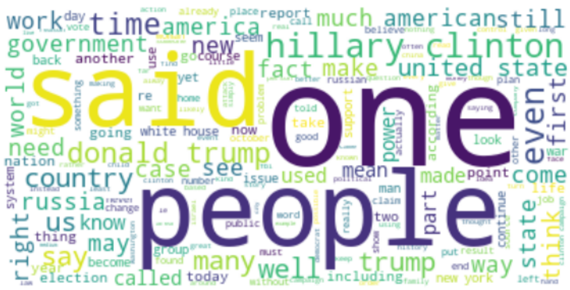

Figure 4: wordcloud of all tokenized words in fake news

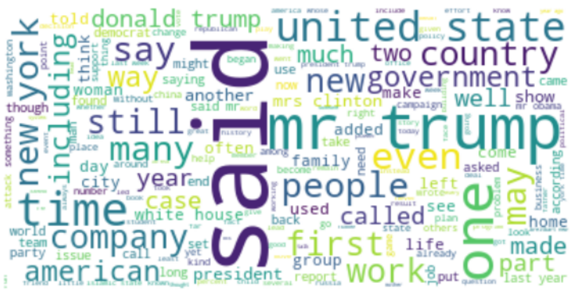

Figure 5: wordcloud of all tokenized words in real news

to one-hot vectors and padded by zero vectors to bring uniformity in the generated vectors.

2.4.2 GloVe Embedding

GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Amongst the pretrained glove models, we used glove.6B.300d.txt, i.e 6B tokens, 400K vocab, uncased, & 300d vectors.

2.4.3 Word2Vec Embedding

Word2Vec is an embedding style that creates word vectors while also retaining context between similar words. The shape of word2Vec embedding is (300,)

2.4.4 Count Vectoriser

Countvectoriser converts a collection of text sentences into a matrix of token counts

2.4.5 TF-IDF

Term Frequency-Inverse Document Frequency vectorizer is a vectorization approach that considers term frequency into account while calculating word vectors. TFIDF value increases proportionally to the number of times that word occurs in the corpus.

2.4.6 Hashing-Vectorizer

Hashing vectorizer is a vectorization approach that uses the hashing trick to find the token string to feature integer index mapping. It converts a collection of text sentences into a matrix of token occurrences.

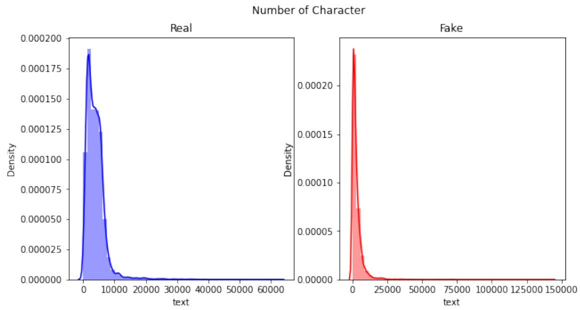

Figure 6: character length plot

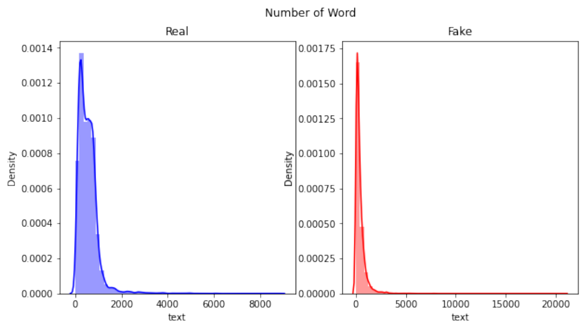

Figure 7: Word length plot

2.5 BERT

Till now, all the vectorisation techniques vectorised a text sequence from left to right, or in a combination of sequential traversal forms. However, BERT applies bidirectional training of the attention model, to compute word embeddings. It has much deeper sense of language context, than any other existing models.

2.6 Learning Algorithms

After vectorising the data, we ran multiple models to classify fake news.

2.6.1 Machine Learning Algorithms

- **Naive Bayes (MultinomialNB):** This algorithm assumes that each word in the sentence is independent of others. Naive Bayes uses the probability of presence of a word given the type of article and the a priori probability of each word to calculate the label of the article.

- **DecionTree:** Decision Tree is a supervised learning algorithm, which uses a tree-like structure to make a prediction. This algorithm models the input features as a tree with the leaf node giving the label of the input.

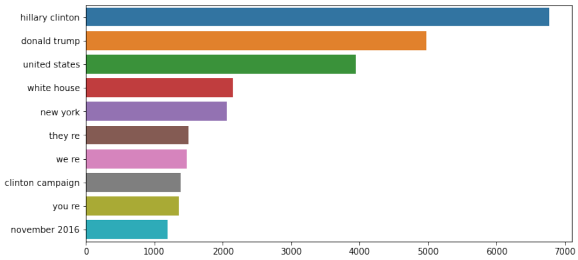

Figure 8: The top 10 bigrams present in fake news

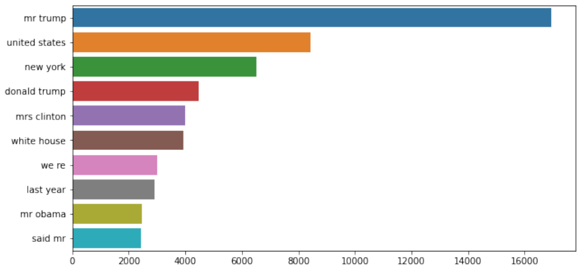

Figure 9: The top bigrams present in true news

- **AdaBoost Classification:** ADABoost uses an ensemble of decision trees (usually stumps of depth 2) to predict the label of the input. Multiple trees helps reduce the error significantly and reduces the chances of overfitting. Additionally, this algorithm learns from the mistakes made by initial trees, and adds more trees to the forest to compensate for them.

- **Logistic Regression** Logistic Regression is a statistical model often used for binary classification problems.

- **Passive Aggressive** It is an online learning algorithm that uses predictions on each sample for training. It reacts passively to correct predictions, that is continues training and aggressively to incorrect predictions (that is updates the model).

2.6.2 Deep Learning Algorithms

- **Multilayer Perceptron** MLP is a feedforward artificial neutral network. It uses back propagation for learning the weights. The last layer consists of an activation function that gives the probability of class labels.

- **LSTM** Long Short Term Memory networks – usually just called "LSTMs" – are a special kind of RNN, capable of learning long-term dependencies. (lst)

- **BERT** BERT provides pretraining of deep Bidirectional Transformers for language understanding (Devlin et al., 2018). BertForSequenceClassification model is a Bert transformer model with a sequence classification/regression head on top i.e a linear layer on top of the pooled output.

2.7 Parameter Tuning

We used GridSearchCV to perform hyperparameter tuning, i.e to get optimal parameters to enhance model performance.

2.8 Combining different features

We used the best model obtained when we used only text as a feature, and trained the same model on combined data which includes concatenated text and authorname, concatenated text and title, concatenated all features (text,author and title name).

2.9 Webserver build using flask, deployed using Heroku

The web server has the functionality to detect fake news given a news text. We have made the fields of title and author name optional, this provides users with some flexibility to detect fake news even if they dont have all the input fields. It has 4 models at the backend, depending on the input method (i.e if only text is entered by the user, if text and author name are entered by user, if text and title are entered by the user, and if all 3 fields are entered by the user.

The webserver has the functionality to load sample input and clear the entered input for making it easier for a first-time user to navigate the web app. In addition to telling the user whether the input news is fake news or not, we also provide the probability values of the respective prediction.

The web app is built using Flask (Grinberg, 2018), which uses a python backend. The app is deployed publicly using Heroku [2]; The web server can be accessed here[3]

3 Related work

With the advent of artificial intelligence, detecting fake news detection has become quick. A tabular representation of related work has been shown in Figure 11. It is observed that more research

[2] https://dashboard.heroku.com/

[3] https://fake-news-detection-nlp.herokuapp.com/

Figure 10: Screenshot of the Web App

has been done on Naive Bayes and SVM classifiers using Tf-IDF, n-gram features, models like LSTM, PassiveAggressiveClassifier, MLP, and features like Hashing vectorizer are less explored. Different categories of approaches used for detecting fake news are language approach, topic agnostic, machine learning, and knowledge-based approach. The language approach focuses on the linguistics of data, how the words are structured, what is syntax and grammar. An example of a language approach is bags of words, in which every word present in any paragraph is assumed as an independent entity and is given equal weightage. Topic agnostic approaches don't focus on the content of data rather they focus on topic-agnostic features like eye-catching lengthier headlines, a lot of advertisements, etc(Castelo et al., 2019); (Horne and Adali, 2017). Fact-checking methods are not much successful since in today's era news spread like wildfire. (Ahmed et al., 2017) used n-gram analysis and Term Frequency- Inverse Document Frequency (Tf-IDF) as feature extraction techniques to detect fake news.We also have crowdsourcing platforms like Kiskkit in which group of people can check the sanity of news (Hassan et al., 2017). (Chen et al., 2015) has made a tool to detect fake news on social media, it uses lexical options that appear in headlines and other powerful language structures. Existing techniques focus more on supervised learning which uses hand crafted input data, which is time consuming.

4 Experiments

We tried multiple different featurizers and model architectures to classify fake news, which can be seen in table 1.

Table 1: Comparative analysis of research studies

Authors	**Proposed Approach**	**Model**	**Dataset**	**Features**
Markines *et al.* (2009)	Analyzed distinct six features for detecting social spammers using machine learning.	SVM, AdaBoost	Spam posts, tags.	TagSpam, TagBlur, DomFp, NumAds, Plagiarism, ValidLinks
Benevenuto *et al.* (2009)	A video response crawler is proposed to identify spammers in online video social network.	SVM	Real YouTube user information.	Video attributes, individual characteristics of user behavior, social relation between users via video response interactions.
Qazinian *et al.* (2011)	Identified tweets in which rumor is endorsed.	Naïve Bayes	Tweets	Content-based, network-based, Twitter specific memes.
Chhabra *et al.* (2011)	Using URLs static features, a method is developed to detect malicious websites.	Naïve Bayes, Logistic Regression, DT, SVM-RBF, SVM-Linear, SVM-Sigmoid	Malicious URL dataset from 'Phishtank'	Grammar, Lexical, Vectors and Static.
Gupta *et al.* (2013)	Analysis of Twitter content during Boston Marathon.	Logistic Regression	Tweets and corresponding user information	Topic engagement, Global engagement, Social reputation, Likability, Credibility
Chen *et al.* (2015)	Analyzed coherence relations between deceptive and truthful news.	VSM	News samples from NPR's 'Bluff the Listener'	Discourse
Rubin *et al.* (2015)	A hybrid approach is proposed combining linguistic and network-based behavior data.	Linguistic, Network models	Simple text sentences	Bag of Words, n-gram
Conroy *et al.* (2015)	A satire detection model is developed.	SVM	US and Canadian national newspapers	Absurdity, Humor, Grammar, Negative affect, Punctuation.
Ahmed *et al.* (2017)	Developed n-gram based classifier to differentiate between fake ad real articles.	LinearSVM	News articles	TF-IDF
Caetano *et al.* (2018)	A predictive model was built to predict 4 subtypes of suspicious news; satire, hoaxes, clickbait and propaganda.	Linguistic models	News posts	TF-IDF, Doc2Vec
Proposed system	Using textual data of articles, an efficient multi-level voting model is developed to detect fake articles.	SGD, PA, MultinomialNB, Gradient Boosting, DT,AdaBoost	News articles	TF-IDF, Count-Vectorizer, Hashing-Vectorizer

Figure 11: (Kaur et al., 2020)

5 Results and Analysis

5.1 One hot vectorization:

We performed one-hot encoding as ML algorithms need data to be in a numerical format to be fed to the model. One hot encoding is a common way to feed the text features as vectors to the model. One hot encoding is preferred over categorical/integer encoding as it doesn't assume any hierarchy or ordering in the text features.

One hot encoding of the text gave us an accuracy of 50%. The poor performance can be attributed to the fact that it doesn't add any feature/word information in the encoding. Since we are assuming the presence of each word is independent of another, it makes the approach very naive, making the model not better than a coin toss.

5.2 TFID/CV/HV vectorisation:

Recognizing the limitations of One hot encoding, we then tried other approaches to convert the text into a numerical format. Count Vectorizer and Hash Vectorizer - It is another way of representing text in vector format. Unlike One-hot encoding which only tells whether a word is present or not, CV gives the count of the number of times each word appears in the text, giving slightly more information for the model to learn. Hashing Vectorizer is another way of converting text to vector, however, since it doesn't store the vocabulary it is much

Featurizer	Model Arch	3-fold cv	Testing acc
CV	NB	91.74	92.3
CV	DT	88.78	89.4
CV	LR	95.46	95.6
CV	AdaB	93.07	93.3
CV	MLP	95.79	96.4
CV	GB	93.21	93.7
CV	PA	94.21	95.2
CV	XGB	93.17	93.5
TF-IDF	NB	73.06	77.1
TF-IDF	DT	87.62	88.9
TF-IDF	LR	94.85	95.8
TF-IDF	AdaB	92.99	93.4
TF-IDF	MLP	96.11	96.6
TF-IDF	GB	93.39	82.5
TF-IDF	PA	96.33	96.9
TF-IDF	XGB	93.27	82.0

Table 1: Model performances

more efficient and uses lesser computational power. TF-IDF- short for term frequency, inverse document frequency, this preprocessing technique used the frequency of a word in the document and its informativeness to give weights to each word. Hence, TF-IDF gives a much more meaningful representation of the text.

Replacing one hot encoding with CV/TFIDF improved the model performance by 90-96%.

5.3 GloVe and Word2Vec vectorisation:

Glove (Pennington et al., 2014) stands for GLObal VEctors, it gives dense embeddings of the word, while incorporating the global context as well. These embeddings provide rich information about the words. Such representations of the text are used with Deep Learning models, which are able to learn these features and classify the text. When we tried to calculate the performance of GloVe and Word2Vec (Church, 2017) on Machine Learning approaches like Logistic Regression, Passive Aggressor, AdaBoost etc, we got an accuracy of around 55-65%. The reason for this low performance scores on ML is that unsupervised vectorisation techniques like GloVe and Word2Vec work extremely good with Deep Learning techniques.

LSTM- short for Long Short Term Memory, make use of the sequence of the words in the text. Bi-LSTMs learn the relations in the text in both directions, thereby learning more information and giving a better performance. We obtained an accuracy of 93% by using Bi-LSTMs with GloVe Embeddings.

5.4 BERT

BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2018) is a transformer based learning technique. We encoded the sentences into BERT encoded vectors using the uncased BERT pre-trained BERT model [4], and then trained them using a BertForSequenceClassification [5] model. This is a pretrained BERT model with a single linear classification layer on the top. Running it on 3 epochs, we recieved an accuracy of 0.978, 0.983 and 0.984 each, giving an average accuracy = 0.981. Metrics on the 20% validation data are:

- Accuracy = 0.985
- F1 Score = 0.983
- Precision = 0.991
- Recall = 0.991

Bert has outperformed all other models tried, and we conclude it to be the best model for detecting fake news.

References

Understanding lstm networks.

Hadeer Ahmed, Issa Traore, and Sherif Saad. 2017. Detection of online fake news using n-gram analysis and machine learning techniques. In *International conference on intelligent, secure, and dependable systems in distributed and cloud environments*, pages 127–138. Springer.

Sonia Castelo, Thais Almeida, Anas Elghafari, Aécio Santos, Kien Pham, Eduardo Nakamura, and Juliana Freire. 2019. A topic-agnostic approach for identifying fake news pages. In *Companion proceedings of the 2019 World Wide Web conference*, pages 975–980.

Yimin Chen, Nadia K Conroy, and Victoria L Rubin. 2015. News in an online world: The need for an “automatic crap detector”. *Proceedings of the Association for Information Science and Technology*, 52(1):1–4.

Kenneth Ward Church. 2017. Word2vec. *Natural Language Engineering*, 23(1):155–162.

[4] https://huggingface.co/bert-large-uncased

[5] https://huggingface.co/transformers/model_doc/bert.html#bertforsequenceclassification

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Miguel Grinberg. 2018. *Flask web development: developing web applications with python*. " O'Reilly Media, Inc.".

Naeemul Hassan, Fatma Arslan, Chengkai Li, and Mark Tremayne. 2017. Toward automated fact-checking: Detecting check-worthy factual claims by claimbuster. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1803–1812.

Benjamin Horne and Sibel Adali. 2017. This just in: Fake news packs a lot in title, uses simpler, repetitive content in text body, more similar to satire than real news. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 11.

Sawinder Kaur, Parteek Kumar, and Ponnurangam Kumaraguru. 2020. Automating fake news detection system using multi-level voting model. *Soft Computing*, 24(12):9049–9069.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.