Shahjalal University of Science and Technology
Department of Computer Science and Engineering

Fake News Detection in Bangla Language Using Machine Learning Algorithms

S.M MAHAMUDUL HASAN
Reg. No.: 2016331032
4^{th} year, 2^{nd} Semester

NIRJAS MOHAMMAD JAKILIM
Reg. No.: 2016331101
4^{th} year, 2^{nd} Semester

Department of Computer Science and Engineering

Supervisor

Enamul Hassan

Assistant Professor

Department of Computer Science and Engineering

7^{th} July, 2021

Fake News Detection in Bangla Language Using Machine Learning Algorithms

This thesis was submitted to the Department of Computer Science and Engineering at Shahjalal University of Science and Technology in partial completion of the requirements for the Bachelor of Science in Computer Science and Engineering.

By

S.M Mahamudul Hasan
Reg. No.: 2016331032
4^{th} year, 2^{nd} Semester

Nirjas Mohammad Jakilim
Reg. No.: 2016331101
4^{th} year, 2^{nd} Semester

Department of Computer Science and Engineering

Supervisor
ENAMUL HASSAN
Assistant Professor
Department of Computer Science and Engineering

7^{th} July, 2021

Recommendation Letter from Thesis Supervisor

The thesis entitled *Fake News Detection in Bangla Language Using Machine Learning Algorithms* submitted by the students

1. S.M Mahamudul Hasan
2. Nirjas Mohammad Jakilim

is a record of research conducted under my supervision, and I thus authorize submission of the report in partial fulfillment of the criteria for the granting of their Bachelor Degrees.

Signature of the Supervisor:

Name of the Supervisor: Enamul Hassan

Date: 7th July, 2021

Certificate of Acceptance of the Thesis

The thesis entitled *Fake News Detection in Bangla Language Using Machine Learning Algorithms* submitted by the students

1. S.M Mahamudul Hasan
2. Nirjas Mohammad Jakilim

on 7th July, 2021 , hereby, accepted as the partial fulfillment of the requirements for the award of their Bachelor Degrees.

Head of the Dept.	Chairman, Exam. Committee	Supervisor
Dr. Muhammad Abdullah Al Mumin	Dr. Muhammad Abdullah Al Mumin	Enamul Hassan
Professor and Head	Professor and Head	Assistant Professor
Department of Computer Science and Engineering	Department of Computer Science and Engineering	Department of Computer Science and Engineering

Abstract

Due to the ease with which information may be accessed and the exponential increase of accessible information on the internet, distinguishing between false and genuine information has become difficult. Fake news may be used to disseminate propaganda against a person, a community, a government, or a political party. This study analyses the Bengali fake news dataset and compares the performance of machine learning and deep learning models to discover which ones are the most successful. The objective is to create a model for a product that, using a supervised machine learning algorithm, can categorise news as true or false. Additionally, the study will assess the model's effectiveness against a range of linguistic features and word vectorizers. This research analyses existing research on fake news detection and compares it to traditional machine learning models in order to develop a product model with a supervised machine learning algorithm capable of classifying fake news as authentic or false, using tools such as tensorflow deep learning. It is discovered that Word2Vec performs the worst in general, while the Count Vectorizer performs somewhat better than the TF-IDF models in the majority of cases. The better result is obtained by combining CV and LSTM, whereas recurrent neural networks (RNN) perform best. The purpose of this study is to assist scholars in developing a comprehensive understanding of the many aspects and characteristics of the bengali dataset. It paves the way for the future. There is space for further study and development of a hybrid model that exploits numerous characteristics and combines different models to get superior outcomes.

Keywords: Fake News Detection, Machine Learning, TF-IDF, Fact-checking, Natural Language Processing, word2vec, RNN, CNN.

Acknowledgements

We would like to express our gratitude to the Department of Computer Science and Engineering at Shahjalal University of Science and Technology in Sylhet, Bangladesh, for their support of this study. We are very appreciative of Md Saiful Islam for his invaluable guidance and assistance.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

"A falsehood may go halfway across the globe before the truth can put on its pants." — William S. Churchill

Fake news is not a novel notion. However, the phrase "Fake News" was almost unknown in the broader context prior to the 2016 US presidential election. However, in recent years, it has developed into a genuine menace that cannot be disregarded.

1.1 What is Fake News

Fake news is information that is incorrect or misleading and is presented as news. It often seeks to harm a person or entity's reputation or to profit from advertising income. The emergence of false news demonstrates how, in the digital era, long-standing institutional safeguards against disinformation are eroding. Global concern exists about the issue. [17] The word "fake news" has evolved to imply a variety of things to various individuals. At its most basic level, we define "fake news" as untrue news reports: the narrative itself is manufactured, without verified facts, sources, or quotations. At times, these tales may be deliberate propaganda intended to mislead the reader, or they may be created as "clickbait" for commercial gain.

According to Wikipedia, *"fake news is a subset of yellow journalism or propaganda that consists of purposeful disinformation or hoaxes propagated via conventional print and broadcast news outlets as well as online social media platforms.* [18]

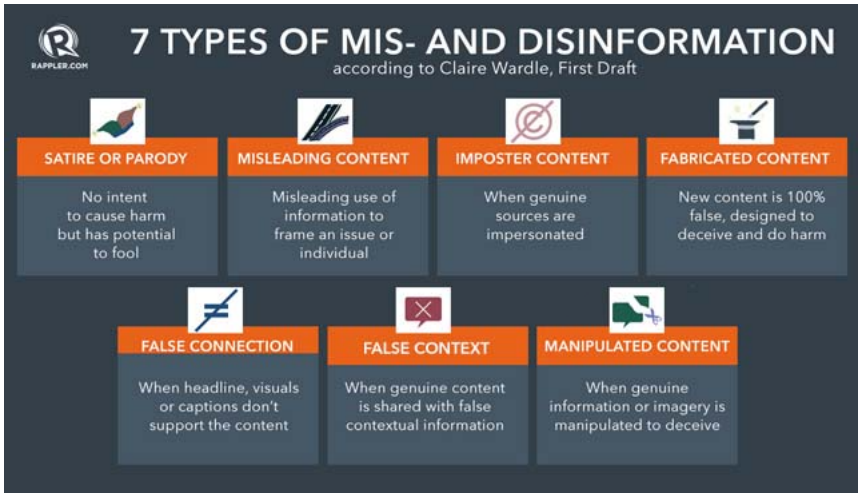


Figure 1.1: Types of Fake News [3]

1.2 Problem Statement

By and large, people are incapable of determining whether or not a piece of news is phony. This is due to three factors. To begin, the majority of individuals think that what they know and get is real. Second, some individuals readily accept ambiguous data. Thirdly, confirmation bias may allow individuals to perceive just what they want. As a result, it is very easy for false news to fool a typical person. Now, the duped individual may post and retweet them on other social media sites, unknowingly contributing to the spread of bogus news. As a consequence, they are sometimes propagated virally. For instance, after the Malaysian Airlines tragedy, a bogus news story headlined "MH370 discovered in the Bermuda Triangle" quickly became very popular on Twitter. As a result, we urgently need an efficient fact-checking system. Indeed, several online apps, such as Snopes.com, FactCheck.org, and PolitiFact, serve as fact-checkers. However, these systems rely on human workers to verify facts manually. While these services give correct information in the majority of cases, they are inefficient in the fight against false news since they are not automated. To resolve the issue, we offer an automated approach based on machine learning and natural language processing.

1.3 Our Approach to the Solution

We are investigating the area of natural language processing in this paper, which is the broad study of how computers and robots can comprehend human-to-human communication and how engines interpret texts based on contextual information. For this report, we will categorize news items as "genuine" or "fake," which will be a binary classification issue requiring us to categorize the samples as positive (containing false news) or negative (not containing false news). We examined the semantic meaning of each word and concluded that the existence of certain terms had an effect on the meaning. This was significant to us, since we thought the text's contextual meaning needed to be maintained and examined for improved categorization. Other studies place a premium on the user and the characteristics associated with them. We divided the task into three phases: pre-processing, text-to-numerical representation conversion using pre-trained techniques, and finally, model evaluation using state-of-the-art machine learning methods. We analyzed the data set, focusing on the text portion, and then translated each text to a numeric representation using pre-training models such as TF-IDF and CV for vector representation. Finally, we classified our numeric conversion data using well-known machine learning methods such as SVM, LR, and classification techniques.

Chapter 2

Background Study

2.1 Word-Embedding

Word embedding simply means vectorizing the words. In our dataset, the data is in textual format. As we know, many machine learning models can't process textual information directly. So, we have to embed the words in vectors. There are various techniques to perform this task. Among them, we have used the Frequency-based Word Embedding methods and

2.1.1 Frequency-based Word Embedding

The frequency of a word in a sentence or a document is the core focus of frequency-based word embedding methods. There are several such methods like **TF-IDF, countvectorizer** etc.

2.1.1.1 TF-IDF

TF-IDF basically analyzes the importance and relevance of a word in a document. Where **TF** stands for "Term Frequency", the **IDF** stands for "Inverse-Terms Frequency". It does not just pay attention to the frequency of the word in a document. It also pays attention to how many times the word appears in the whole corpus. The main motive behind this is the more frequent a word appears in a document, the more relevant the word is with the document. Here, TF scores the frequency of the term in an article and IDF scores the frequency of the whole dataset. The equation

of TF for a term or word **t** and document **d** is as below,

$$tf(t,d) = \log(1 + f(t,d) \tag{2.1}$$

And the equation of IDF for a whole dataset **D** is as follows.

$$idf(t,D) = \log\left(\frac{N}{f(t,D)}\right) \tag{2.2}$$

Here, **N** is the number of documents in the dataset **D**. So, finally the TF-IDF score is calculated using the following formula,

$$\text{tfidf} = tf(t,d) * idf(t,D) \tag{2.3}$$

2.1.1.2 Countvectorizer

Like **TF-IDF**, **CountVectorizer** also converts the whole document into vector. But instead of storing the TF-IDF score, it stores the frequency of the word in a vector. The dimension of the vector is simply the unique words. And count vectorizer then count every occurrence of a word in the dataset and store the count in the respective field of that word in the vector. For an example, Suppose the following two lines are in the dataset.

- Karim can not solve a problem correctly. (V1)
- Rahim can solve a problem very quickly. (V2)

So, the unique words are ['Karim', 'Rahim', 'can', 'not', 'solve', 'a', 'problem', 'correctly', 'very', 'quickly'] Now, the vector will be as follows.

	Karim	Rahim	can	not	solve	a	problem	correctly	very	quickly
V1	1	0	1	1	1	1	1	1	0	0
V2	0	1	1	0	1	1	1	0	1	1

Table 2.1: CountVectorizer

2.1.2 Prediction based Word Embedding

Following the frequency-based word embedding techniques, a new idea called Word2Vec (Tomas Mikolov) was introduced in 2013. This idea fundamentally alters the way NLP is currently practised. Following the publication of this algorithm, we will be able to build intelligent chatbots. Even Google grew in power after its creation. While creating embeddings, it was able to capture the context. Word2Vec is a method for predictive word embedding. Indeed, Word2Vec is a pre-trained Prediction-based Embedding Model (which includes algorithms and training on its own data). It was honed using the Google news corpus. To be honest, I included these fascinating facts in the first paragraph to pique your attention. This article will discuss Word Embedding Techniques Based on Prediction.

2.1.2.1 Word2vec

Word2Vec is a technique for generating such an embedding. It may be obtained in one of two ways (both of which use Neural Networks): exclude the grammatical and colloquial bag of words (CBOW) Model CBOW: This technique takes the context of each word as input and attempts to predict the word that is most closely related to the context.

Word2vec is a natural language processing method that was first published in 2013. The word2vec method learns word connections from a huge corpus of text using a neural network model. As the name suggests, word2vec assigns a unique collection of integers called a vector to each individual word.

Word2vec is a two-layer neural network that analyses text via the process of "vectorizing" words. It takes a text corpus as input and returns a collection of vectors: feature vectors representing the words in the corpus. While Word2vec is not a deep neural network, it converts text to a numerical format understandable by deep neural networks.

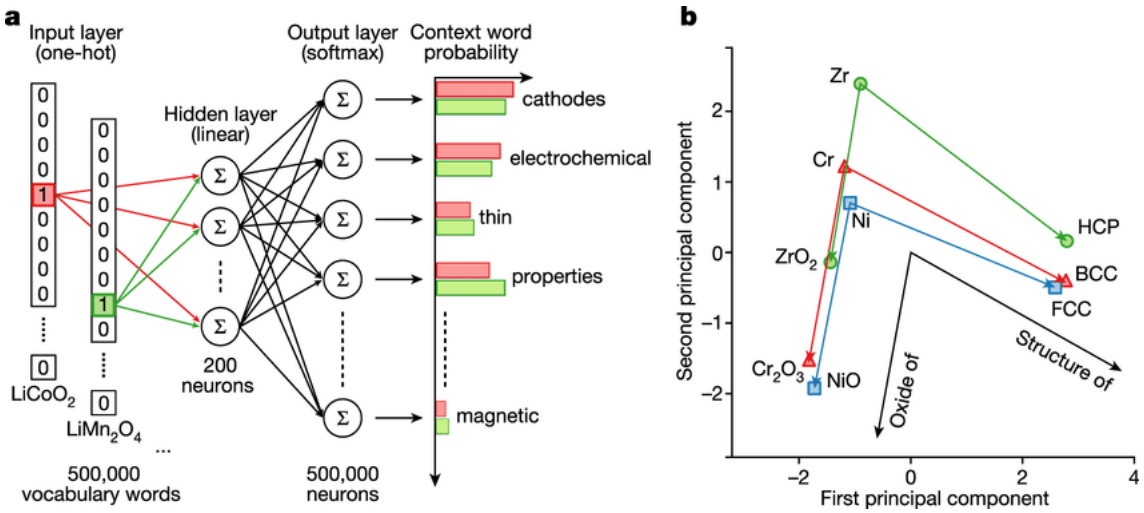


Figure 2.1: Word 2 vector [4]

2.2 Traditional Model

2.2.1 Support Vector Machine

Support Vector Machines known as (SVMs) are one of the most extensively used techniques for Supervised Learning and are also known as support vector machines. They may be utilized for both classification and regression tasks, depending on the situation. However, it is most often used in Machine Learning to solve problems involving classification. The Support Vector Machine, or SVM, is a linear model that is used to solve classification and regression issues. It is capable of solving linear and non-linear problems and is effective for a wide variety of practical situations. SVM is a straightforward concept: The method generates a line or hyperplane that classifies the data. A hyperplane denotes this optimal choice boundary. SVM determines the hyperplane's extreme points/vectors. Support vectors are used to refer to these extreme circumstances, and so the method is dubbed a Support Vector Machine. Consider the figure below, which illustrates two distinct categories separated by a decision boundary or hyperplane.

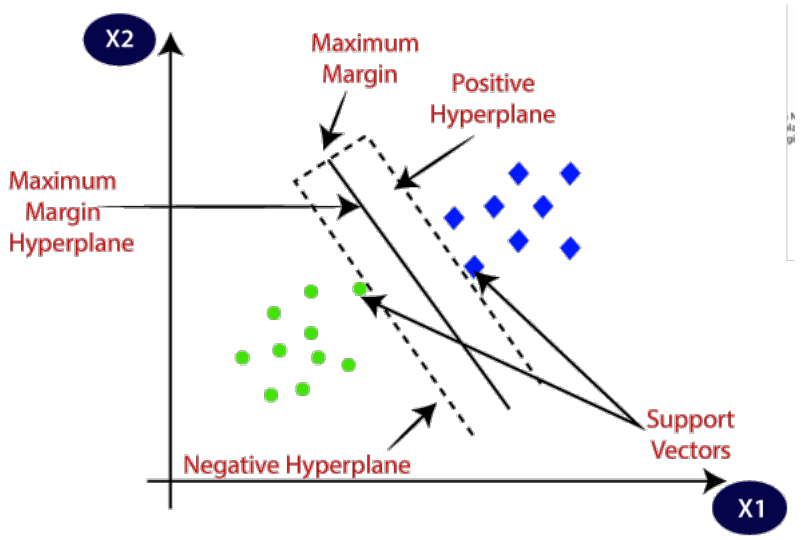


Figure 2.2: Support Vector Machines [5]

2.2.2 Random Forest Classifier

Random forests, also known as random decision forests, are a kind of ensemble learning method for classification, regression, and other problems that work by training a large number of decision trees. While random forests outperform decision trees in most cases, they are less accurate than gradient-boosted trees. Random Forest is a very effective technique for developing predictive models for classification and regression problems. Its default hyperparameters already provide great results, and it is very effective at avoiding overfitting. Additionally, it provides a pretty accurate estimate of the importance of your features. However, as previously stated, a random forest is just a collection of trees. Random forests are a robust modeling technique that outperforms a single decision tree. They integrate several decision trees in order to reduce overfitting and bias-induced inaccuracy and therefore provide meaningful results.

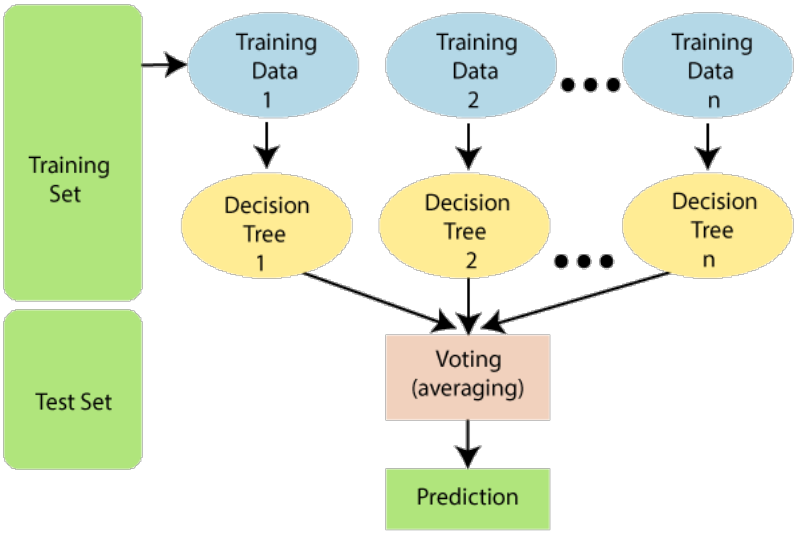


Figure 2.3: Random Forest Classifier [6]

2.2.3 Logistic Regression

When the dependent variable is dichotomous (binary), logistic regression is the proper regression technique to use. Logistic regression is used to describe data and explain the connection between one dependent binary variable and one or more independent variables at the nominal, ordinal, interval, or ratio levels. In the presence of many explanatory variables, logistic regression is used to calculate the odds ratio. The method is quite similar to multiple linear regression, except the response variable is binomial. The outcome is the effect of each variable on the observed event's odds ratio. The primary constraint for Logistic Regression is the linear relationship between the dependent and independent variables. It not only indicates the appropriateness of a predictor (coefficient size), but also indicates the direction of connection (positive or negative)

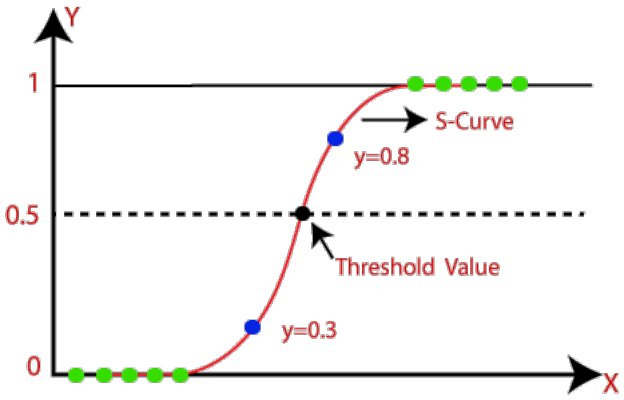


Figure 2.4: Logistic Regression [7]

2.2.4 Decision Tree

A decision tree is a decision assistance tool that employs a tree-like representation of choices and their potential repercussions, which may include chance event outcomes, resource costs, and utility. This is one method of displaying an algorithm composed entirely of conditional control statements. Decision trees assist you in weighing your choices. Decision trees are great tools for guiding you through the process of choosing between many alternative courses of action. They offer a very effective framework for outlining choices and investigating the potential consequences of selecting those choices. Because the objective of a decision tree is to make the best option at the end of each node, it requires an algorithm capable of doing this. Hunt's algorithm is a greedy and recursive algorithm. The choice or test is made based on the characteristics of the provided dataset. It is a graphical depiction used to get all feasible answers to a problem/decision under certain parameters. It is named a decision tree because, like a tree, it begins with the root node and grows via additional branches to form a tree-like structure. We employ the CART algorithm, which stands for Classification and Regression Tree algorithm to construct a tree. A decision tree is a simple diagram that asks a question and then splits the tree into sub-trees based on the answer (yes/no). The following graphic depicts the fundamental structure of a decision tree.

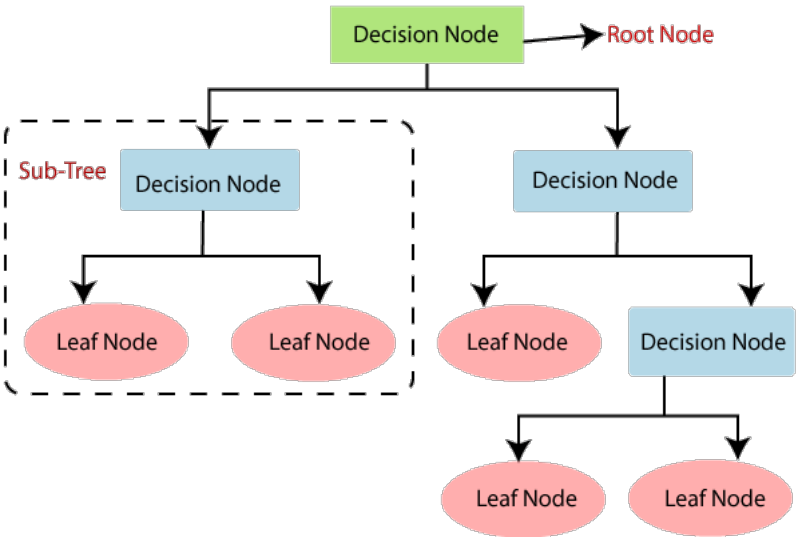


Figure 2.5: Decision Tree Classifier [8]

2.2.5 Multinomial Naive Bias (MNB)

Multinomial Naive Bayes is the name given to the Multivariate Event model. Naive Bayes is based on Bayes theorem, with the term Naive indicating that the dataset's characteristics are mutually independent. The presence of one characteristic has no bearing on the likelihood of the occurrence of the other. Multinomial Naive Bayes is a probabilistic learning technique that is often used in Natural Language Processing (NLP). The Multinomial Naive Bayes classifier is a collection of many methods that all adhere to a single guiding principle: each feature being categorized is unrelated to any other feature. The name Multinomial Naive Bayes merely indicates that each p(fi|c) is a multinomial distribution, as opposed to another kind of distribution. This is particularly useful for data that can be readily converted to counts, such as word counts in text.

2.2.6 K-Nearest Neighbour

KNN is one of the simplest accessible classification algorithms and one of the most often used learning algorithms. KNN is a technique for lazy learning that is non-parametric. Its aim is to forecast the classification of a new sample point using a database of previously categorized data points. Due to the highly exact predictions generated by the KNN algorithm, it can compete with the most accurate models. As a consequence, the KNN technique may be utilized in situations where great accuracy is required but a human-readable model is not required. The quality of the

prediction is depending on the distance metric employed. Numerous benefits of KNN include the following:

- Quick calculation time.
- Simple interpretation owing to the method's simplicity.
- Adaptable – suited for regression and classification.
- Superior accuracy – no need to compare to more supervised learning models.
- Quick calculation time.

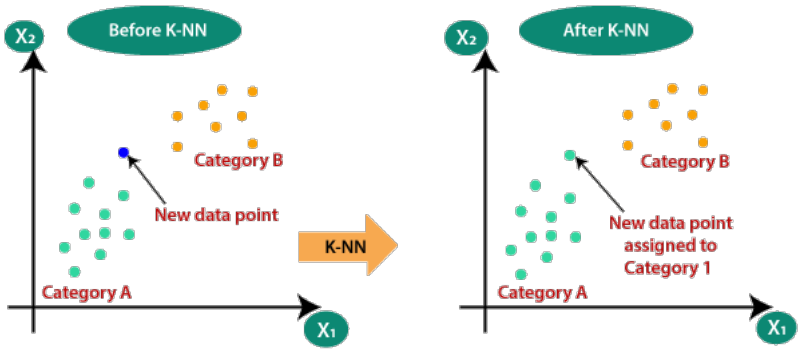


Figure 2.6: K-Nearest Neighbour [9]

2.2.7 SGD Classifier

Stochastic Gradient Descent-Classifier (SGD-Classifier) may cause some users to believe SGD is a classifier. However, this is not the case! The SGD Classifier is an SGD-optimized linear classifier (SVM, logistic regression, etc.). These are two very distinct ideas. SGD is an optimization technique, while Logistic Regression or linear Support Vector Machine is a machine learning algorithm/model. Consider that a machine learning model provides a loss function, which the optimization technique attempts to reduce or maximise.

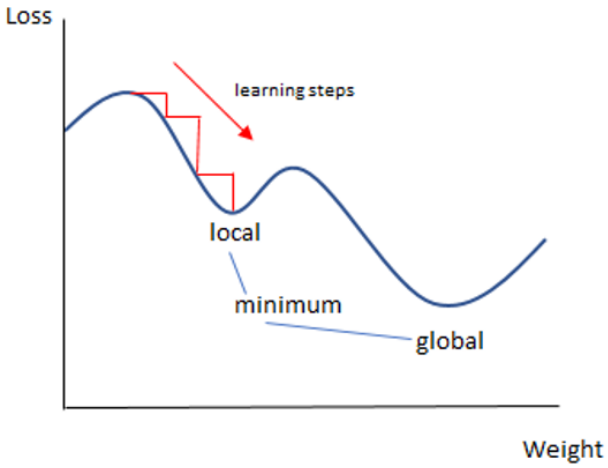


Figure 2.7: SGD Classifier Classifier [10]

2.2.8 Passive Aggressive Classifier

The Passive-Aggressive algorithms are a subset of machine learning algorithms that are unfamiliar to beginners and even intermediate enthusiasts. They may, however, be very helpful and efficient in some situations. The majority of the time, passive-aggressive algorithms are employed for large-scale learning. It is one of the few 'algorithms for online learning'. In contrast to batch learning, which uses the whole training dataset at once, online machine learning algorithms take the input data in sequential order and update the machine learning model step by step. This is very helpful in cases when there is a large quantity of data and training the whole dataset would be computationally impossible due to the sheer magnitude of the data. Simply put, an online-learning system will acquire a training instance, update the classifier, and then discard the instance. Passive: If the forecast is accurate, let the model alone. I.e., the data in the case is insufficient to alter the model. Aggressive: If the forecast proves to be wrong, modify the model. I.e., a modification to the model may resolve the issue.

2.2.9 MLP Classifier

MLPClassifier is an acronym for Multi-layer Perceptron classifier, which is connected to a Neural Network by its name. Unlike other classification methods such as Support Vector Machines or Naive Bayes Classifier, MLPClassifier does classification using an underlying Neural Network.

However, one similarity to Scikit-other Learn's classification methods is that implementing MLP-Classifier requires the same amount of work as implementing Support Vectors, Naive Bayes, or any other Scikit-Learn classifier.

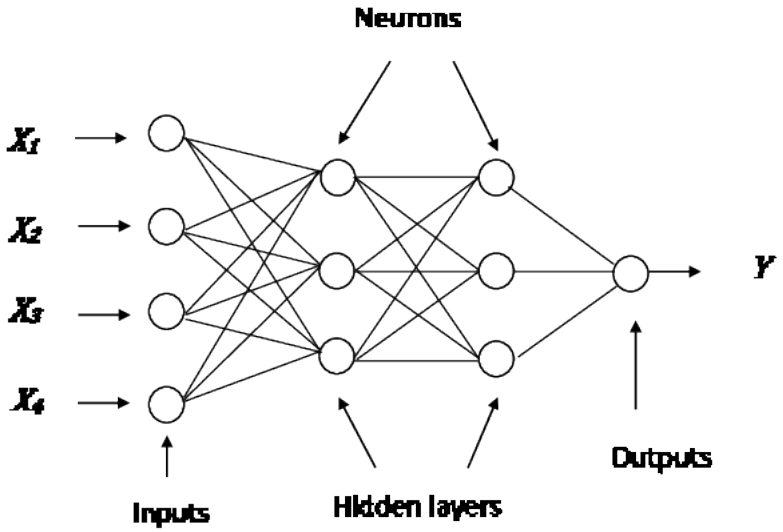


Figure 2.8: MLP Classifier [11]

2.2.10 XG Boost Classifier

Gradient boosting classifiers are a family of machine learning algorithms that integrate a large number of weak learning models to produce a strong prediction model. When doing gradient boosting, decision trees are often utilised. XGBoost is a distributed gradient boosting toolkit that has been tuned to be very efficient, versatile, and portable. It utilises Gradient Boosting to build machine learning techniques. XGBoost is a parallel tree boosting algorithm (also known as GBDT or GBM) that solves a wide variety of data science issues quickly and accurately. The same algorithm works on a variety of distributed computing platforms (Hadoop, SGE, and MPI) and is capable of solving problems with billions of instances.

2.3 Deep Learning Based Models

2.3.1 Neural Network

The name "Artificial Neural Network" derives from biological neural networks, which are responsible for developing the human brain's structure. Like the real brain, which is composed of neurons linked to one another, artificial neural networks are composed of neurons connected at different levels. These neurons are collectively referred to as nodes. Our research used various neural network classifiers like CNN, RNN, LSTM, MLP etc to train and test our dataset.

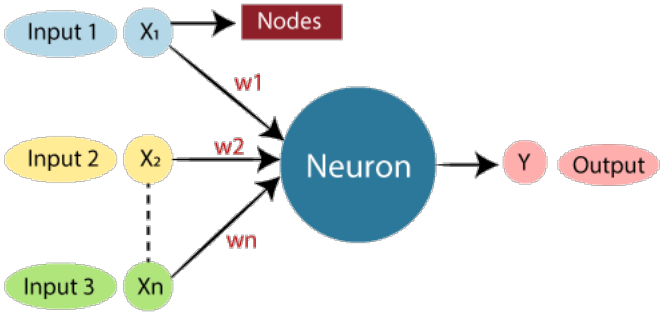


Figure 2.9: Neural Network

2.3.2 Recurrent Neural Network (RNN)

Recurrent Neural Networks (RNNs) are a subclass of Neural Networks in which the output of the previous step is used as the input for the next phase. While all inputs and outputs in conventional neural networks are independent, in certain instances, such as when predicting the next word in a phrase, the prior words are needed, necessitating the requirement to remember the previous words. Thus, RNN was born, which addressed this problem via the use of a Hidden Layer. The primary and most significant property of RNNs is their hidden state, which stores information about a sequence. RNNs have a "memory" that retains all information about the calculation. It utilises the same settings for each input since it produces the result by doing the same job on all inputs or hidden layers. In comparison to other neural networks, this lowers the complexity of the parameters.

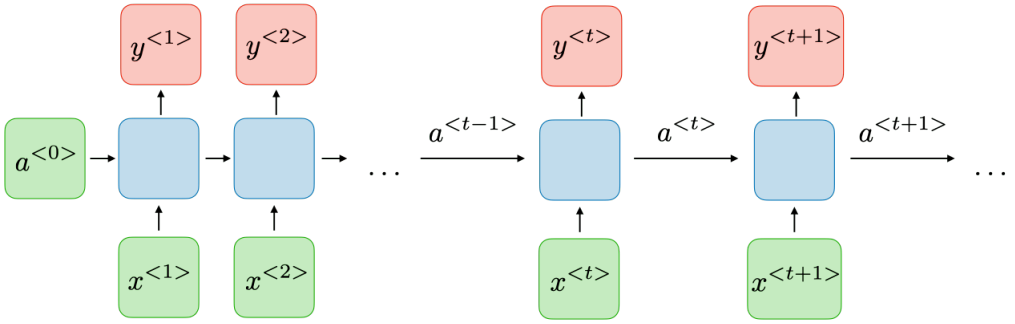


Figure 2.10: Recurrent Neural Network [12]

2.3.3 Convolution Neural Network (CNN

Convolutional neural networks (CNNs), a subclass of artificial neural networks that has proven dominant in a number of computer vision tasks, are gaining attention in a variety of disciplines, including radiology. CNNs are intended to automatically and adaptively learn spatial hierarchies of features through backpropagation, using a variety of building blocks including as convolutional layers, pooling layers, and fully connected layers. This review article covers the fundamental principles of CNN and how they are used to different radiological tasks, as well as the difficulties and future prospects of CNN in the area of radiology. Two additional difficulties associated with applying CNN to radiological tasks will be discussed in this paper, including limited datasets and overfitting, as well as methods for mitigating them. Understanding the principles and benefits of CNN, as well as its limits, is critical for maximising its potential in diagnostic radiology, with the aim of enhancing radiologists' performance and improving patient care.

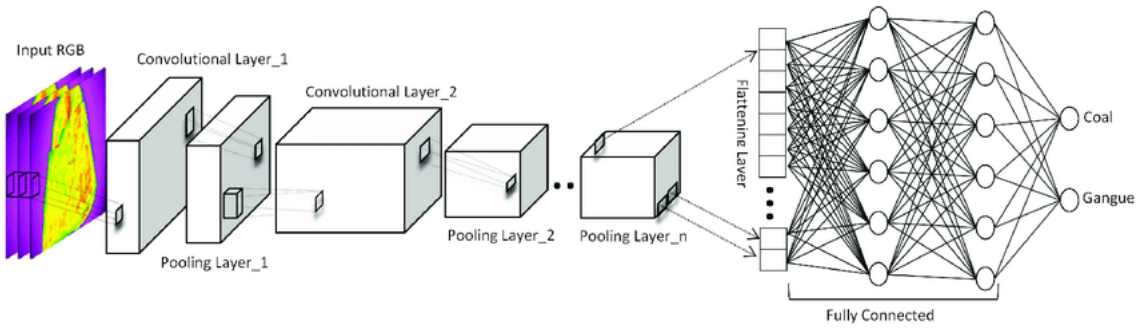


Figure 2.11: convolutional Neural Network [13]

2.3.4 Long short-term memory (LSTM)

Long Short-Term (LSTM) networks are a kind of recurrent neural network that may learn to depend on sequence prediction issues. This is necessary in a variety of difficult problems, including machine translation, voice recognition, and others. LSTMs are a deep-learning subfield.

An LSTM control flow resembles a recurrent neural network. It processes data, sending propagating information. Differences exist in processes inside LSTM cells. These techniques enable LSTM to retain or forget data.

RNNs (LSTMs) are extremely capable of extracting patterns from input function space, even if data input is lengthy. Since LSTMs have a gated design and the capacity to alter memory status, they are suitable for such situations. Using Long Short Term Memory (LSTM) networks, forecast sequential data like stocks' future. Forecasting is the practise of making future predictions based on past facts. The primary difficulty is to identify patterns in a data collection, then apply it to future study.

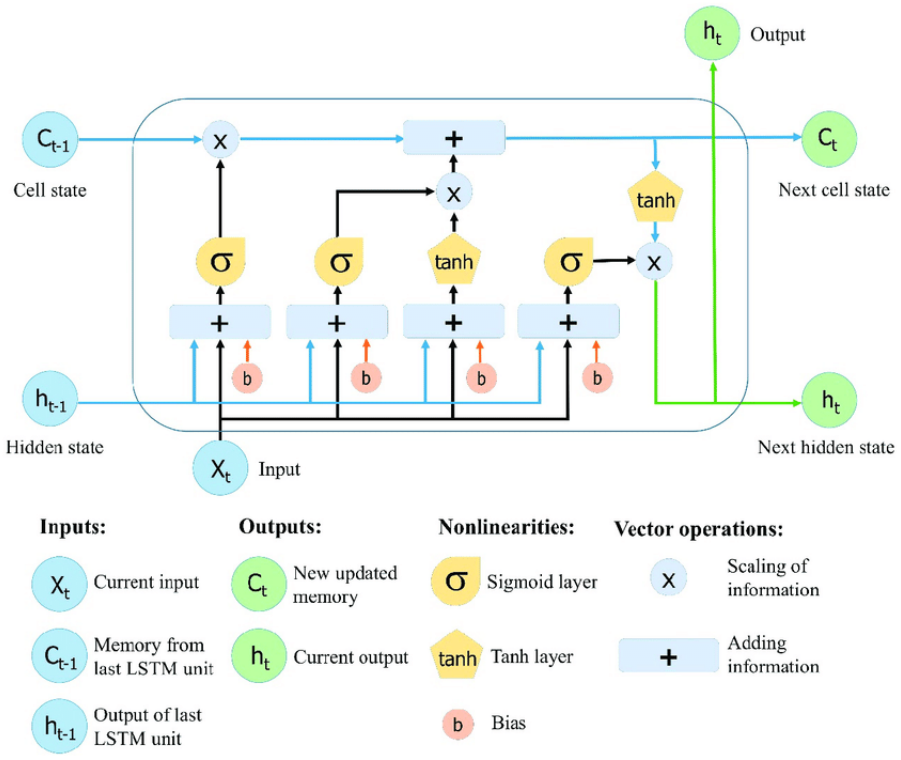


Figure 2.12: Long Short-Term Memory [14]

2.3.5 Bidirectional LSTM (BiLSTM)

Bidirectional LSTMs are an extension of standard LSTMs that significantly improve model performance when it comes to sequence classification. In circumstances when all time steps are available, bidirectional LSTMs train two instead of one LSTM on the input sequence. Bidirectional processing runs your inputs in two directions, one from past to future and one from future to past, and what distinguishes it from unidirectional processing is that in the LSTM that goes backwards, you can save knowledge from the future and combine the two hidden states at any point in time. Bidirectional LSTMs, or biLSTMs, are a kind of sequence processing model that consists of two LSTMs: one that takes the input forward and one that takes it backward. BiLSTMs effectively increase the amount of information available to the network, thus enhancing the context for the algorithm (e.g. knowing what words immediately follow and precede a word in a sentence). Cornegruta, Bidirectional Long-Term Radiological Language Modeling, etc. Bidirectional RNN (BRNN) repeats the RNN processing chain in order to process inputs in both directions. This allows a BRNN to consider future context as well. LSTM outperforms RNN in capturing long-term dependencies.

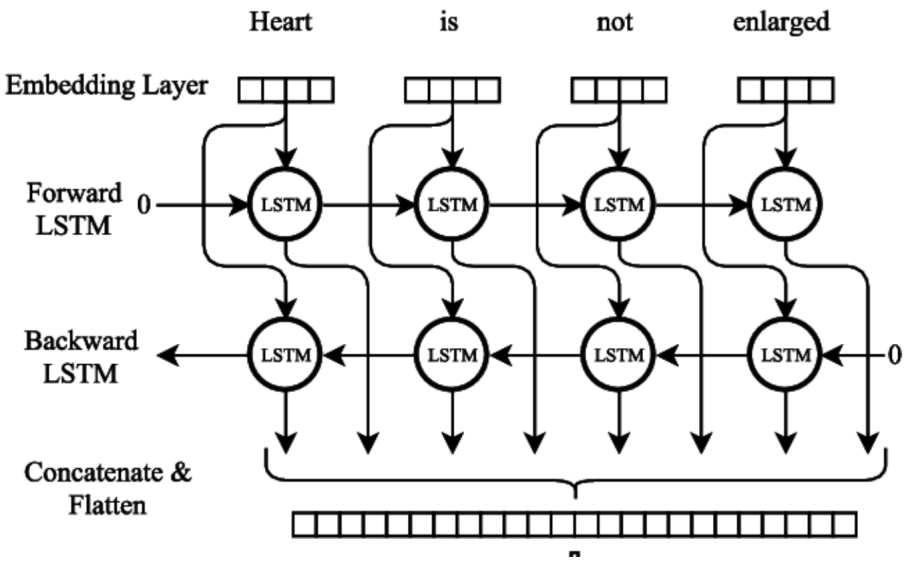


Figure 2.13: Bi-Directional Long Short-Term Memory [15]

2.4 Pre-Trained Models

Simply stated, a pre-trained model is one that has been developed by another individual to address a comparable issue. Rather than beginning from scratch with a new model to tackle a comparable issue, you utilise the model trained on the previous problem as a starting point. For instance, suppose you want to construct a self-learning automobile. In general, supervised pretraining entails gradually increasing the number of hidden layers in a model trained on a supervised learning task. Unsupervised pretraining entails building an unsupervised autoencoder model using the greedy layer-wise method, which is subsequently supplemented with a supervised output layer. Additionally, many pre-trained models for different platforms are available at https://www.gradientzoo.com. Additionally, if you are interested in a certain network architecture, authors sometimes offer pre-trained models themselves, for example, ResNeXt.

2.4.1 BERT

BERT is an open source platform for machine learning-based natural language processing (NLP). BERT is intended to assist computers in deciphering the meaning of ambiguous language in text by establishing context via the use of surrounding material. BERT was pre-trained on Wikipedia text and may be fine-tuned using question and answer datasets.

BERT is an acronym for Bidirectional Encoder Representations from Transformers. It is based on Transformers, a deep learning model in which each output element is linked to each input element and their weightings are dynamically computed depending on their relationship. (In NLP, this is referred to as attention.)

Historically, language models could only interpret text in sequential order â either left-to-right or right-to-left â but not both. BERT is unique in that it is capable of reading in both directions simultaneously. Bidirectionality is a capacity provided by the advent of Transformers.

BERT is pre-trained on two distinct but related NLP tasks using this bidirectional capability: Masked Language Modeling and Next Sentence Prediction.

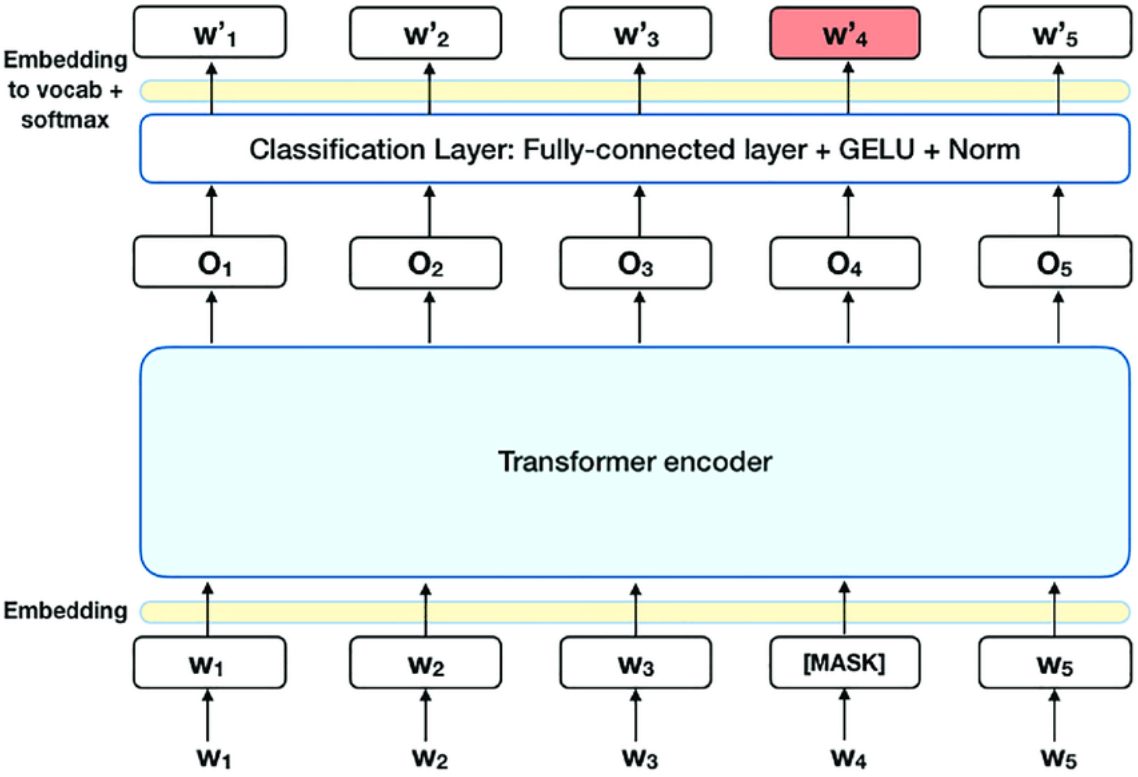


Figure 2.14: Bidirectional Encoder Representations from Transformers [16]

2.5 Confusion Matrix

For a given collection of test data, the confusion matrix is a matrix that may be used to assess the overall performance of classification models. A determination may only be made if the real values for the test data are known in advance. The matrix itself is straightforward to comprehend, but the associated terminology may be difficult to grasp. A typical confusion matrix have the following instances.

- True Negative: The model predicted No, and the corresponding true or actual value was likewise No.
- True Positive: The model predicted yes, and the actual value confirmed this prediction.
- False Negative: The model predicted no, but the observed result was Yes; this is referred to as a Type-II mistake.

- False Positive: Although the model predicted Yes, the actual result was No. Additionally, it is referred to as a Type-I mistake.

In our analysis, we'll be thoroughly use this confusion matrix to measure our model performances, classification accuracies, error rates and other measurements.

2.6 Literature Review

Many research projects are devoted to detecting false news, most of which focus on English news. There is some work being done in the area of false news identification in the Bangla language. The majority of instances concentrate on a single characteristic to evaluate the outcome. The conventional methods include the following:

2.6.1 Linguistic-based Approaches

2.6.1.1 Deep Syntax Analysis

This is accomplished via the use of Probability Context-Free Grammars (PCFG). Certain rewrite rules are created by transforming sentential forms. These rewriting rules are then utilized to construct a parse tree with an associated probability. Finally, the parse tree is utilized to evaluate if the incoming text has a regular syntactical structure or not. At times, deliberately generated false news deviates from natural syntax. Thus, using the parse tree to validate the syntax of false news is quite reasonable. Additionally, this technique has an accuracy of 85-91 percent. [19]

2.6.1.2 Deep Syntax Analysis

This technique evaluates the veracity of the news's body text using reviewers. An honest reviewer is more likely to offer accurate observations regarding current events. Additionally, a fake news writer with no understanding of the incident/event being covered may add inconsistencies or omissions of relevant information. This technique needs the existence of information regarding a review writer's honesty in the dataset. Though this is a concern, this technique is very accurate. It is capable of correctly predicting falsity in about 91 percent of cases. [19]

2.6.1.3 Categorization

To identify false news, this method requires the completion of four subtasks for three distinct kinds of false news: fabrication, hoaxing, and satire.

- **Fabrication**: This is the kind of false news in which criminal tales are included. Celebrity gossip columns, rumors concerning unhealthy foods, and so on. The following are the outcomes of Yellow Journalism.

- **Hoaxing**: This refers to news that has been incorrectly verified by conventional news organizations and is a kind of intentional fabrication. These are more severe pranks that go beyond being simply amusing and inflict the victim with substantial loss or harm

- **Satire**: This is a technique for exposing and condemning people's folly or vices via the use of comedy, mockery, or exaggeration. Politics and the media are the primary subjects of satire. Satire is also a highly common technique for fabricating news.

- **Clickbait**: A clickbait headline is a hyperbolic title designed to get you to click on a link to an article, picture, or video. Rather than conveying objective information, clickbait headlines often appeal to your emotions and curiosity. Clickbait is mainly used to increase the number of page views on websites, whether for their own objectives or to generate income from online advertising. Additionally, it may be used in phishing attempts to distribute malicious files or steal user information. As a consequence, many studies have been conducted to independently identify various kinds of news, ultimately culminating in the detection of false news.[20]

2.6.2 Network-based Approaches

2.6.2.1 Linked Data

A common method is to conduct fact checks utilizing the knowledge networks created by linking the connected data. A well-constructed knowledge network reduces fact-checking to a straightforward network analysis job. A news item may be judged as truthful or fake by determining the shortest route between two connected words. The closer a word is to a known connected term,

the more likely it is that the associated assertion is true. This method achieves an accuracy of between 60% and 95% for various topic areas. [19]

2.6.2.2 Twitter Graphs

Certain research projects are conducted solely for the purpose of generating Twitter news. Facts were verified by analyzing the writers' characteristics and creating a graph linking comparable individuals who have been active in disseminating false news. Additionally, the depth of the discussion tree was a significant factor in the erroneous detection. A greater depth implies a greater degree of conflict in the story. Additionally, A strong possibility is that the conflicting news is false. By analyzing the relationships and features of the people engaged in the discussion, a forecast for the news is produced, and the outcome is very good. [21]

2.6.2.3 Credibility Network

This is also a strategy for Twitter news. Tweets on a certain subject are linked. There are two types of connections: supportive and antagonistic. Tweets with similar (positive/negative) perspectives are labeled as supporting, whereas tweets with divergent perspectives are labeled as opposing. A link is given a weight depending on its strength. Then, a result is computed using the value discovered in the credibility network after the convergence of credibility value propagation. This technique has an overall accuracy of about 84%. [22]

2.6.3 Naive Bayes

Certain methods simply classified news using a naÃ¯ve bayes classifier. This is a probabilistic method in its entirety. To begin, 1-grams have been extracted from the context of the story. After that, a little amount of pre-processing has been performed. The pre-processing step eliminates stop-words (words that make no sense when used to describe the news, such as a, the, to, etc.) to create the bag of essential words. To improve accuracy, a paper suggested stemming (considering make/making, etc. to be the same word). Then, the terms are put into a naÃ¯ve bayes classifier, which uses them to determine if the news is false or genuine. The system achieves approximately 70% accuracy without pre-processing and 78.6% accuracy with pre-processing in Indonesian Language. [23] [24]

2.6.4 CSI Model

Capture Score and Integrate is an acronym for Capture Score and Integrate. This is a hybrid approach that emphasizes the context, the response, and the source of the news rather than the more conventional selection of a single characteristic. In this method, sense-making words from the news's body text are extracted and put into the RNN to determine if the news is authentic or not. Again, the reaction is interpreted as a feature and is used to categorize the news. Finally, the source of the news plays a significant role in categorization. The dataset for this assignment should include information about different news sources (such as whether they usually offer accurate or misleading news). Then, this information is very useful for detecting false news. The CSI model is a synthesis of these three concepts. This model has an accuracy of 89.2 percent when using Twitter data and 95.3 percent when using Weibo data. [25]

2.6.5 Fact-Checking Using Likes

This method makes use of a collection of news articles and the identification of people who shared them. Individuals are assigned a weight based on their overall preference for genuine or false news. Thus, users are classified as genuine likers or fake likers. Now, based on the number of genuine and false likes received by a news item, the item is classified as fake or genuine. In this instance, an issue emerges. If a person is not included in the dataset, he or she is ignored, and if news about people who are not included in the dataset is received, it is classified. An extra step has been made to resolve this issue. A bipartite graph is created, which keeps the users together. and the postings are in a separate batch. The graph's connections are created by linking all of the node pairs. Then, the users' weights are allowed to propagate until convergence. Thus, posts with followers who are not included in the dataset establish a connection to posts with known users, resolving the previously mentioned issue. This process is referred to as the Harmonic Algorithm. The first method achieves an astounding accuracy of approximately 99 percent, while the Harmonic method improves this to 99.4 percent. [26]

Chapter 3

Dataset

3.1 Dataset Description

The dataset has been collected from from Kaggle. [27] We then have modified the dataset and added our self labelled 130+ unique fake news. The dataset contains mainly three types of fake news.

- **Misleading News**: Propaganda based news that is created to mislead people.
- **Clickbait**: They are types of news where the titles are not related to the body contents.
- **Satire**: Fake News that is created for entertainment purposes only.

The dataset contains **10000** labelled authentic news and **1440** labelled fake news. The below figure shows the target distribution of the dataset.

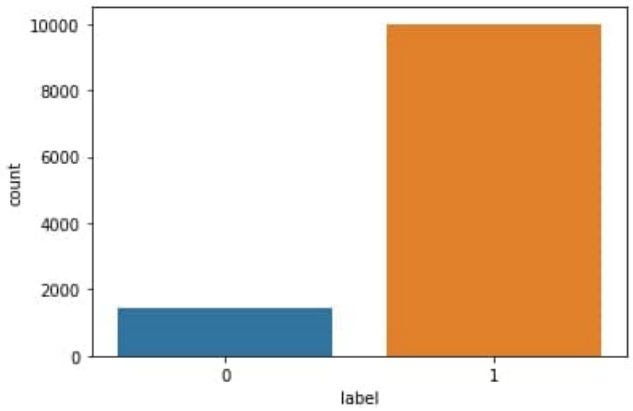


Figure 3.1: Number of Fake and True News

The dataset mainly contains **10 features** and they are as the following List.

- **articleID**: Unique id of the news.
- **domain**: The originating domain for the news.
- **date**: News publication date.
- **category**: Category of the news.
- **source**: The person or source who have verified or published the news.
- **relation**: News is related with headline or not.
- **headline**: Headline of the news.
- **content**: Body text of the news.
- **label**: Target of the news. The value is 0 for fake news and 1 for true.
- **F-type**: Fake news type whether it is clickbait, satire or misleading fake news.

Label	1 (Authentic)
Domain	channelionline.com
Published Time	2018-09-19 18:15:40
Category	আন্তর্জাতিক
Source	বিবিসি
Headline-article relation	related
Headline	মুক্তি পেলেন নওয়াজ শরীফ
Article	পাকিস্তানের সাবেক প্রধানমন্ত্রী নওয়াজ শরীফকে মুক্তি দিয়েছে দেশটির উচ্চ আদালত। কথিত দুর্নীতির মামলায় ১০ বছরের সাজা পেয়ে দুই মাস কারাভোগের পর তিনি মুক্তি পান। বুধবার দেশটির আদালত পাকিস্তান মুসলিম লিগের শীর্ষ এ নেতাকে মুক্তির আদেশ দেন। আদালত একইসঙ্গে নওয়াজের মেয়ে মরিয়ম শরীফকেও মুক্তির আদেশ দিয়েছেন। তাদের আবেদনের পরিপ্রেক্ষিতে আদালত এই রায় দিয়েছেন বলে বিবিসি জানায়। পাকিস্তানের জাতীয় নির্বাচনের আগে দুর্নীতির মামলায় নওয়াজ শরীফকে ১০ বছর এবং তার মেয়ে মরিয়মকে ৭ বছরের কারাদণ্ড দেন আদালত।

Figure 3.2: Dataset

We then further evaluated the dataset to determine the distribution of fake news and true news on each category. The follwing table shows the number of news on each category.

Name of Category	News	Name of Category	News
Miscellaneous	1107	Crime	187
Political	744	Lifestyle	284
National	3566	Finance	239
Entertainment	663	Technology	174
Sports	1660	Editorial	673
Education	211	International	1526

Table 3.1: Number of News on each category

We then further evaluate and count only the fake news from the dataset.The following figure shows distribution of only fake news in the dataset according to category.

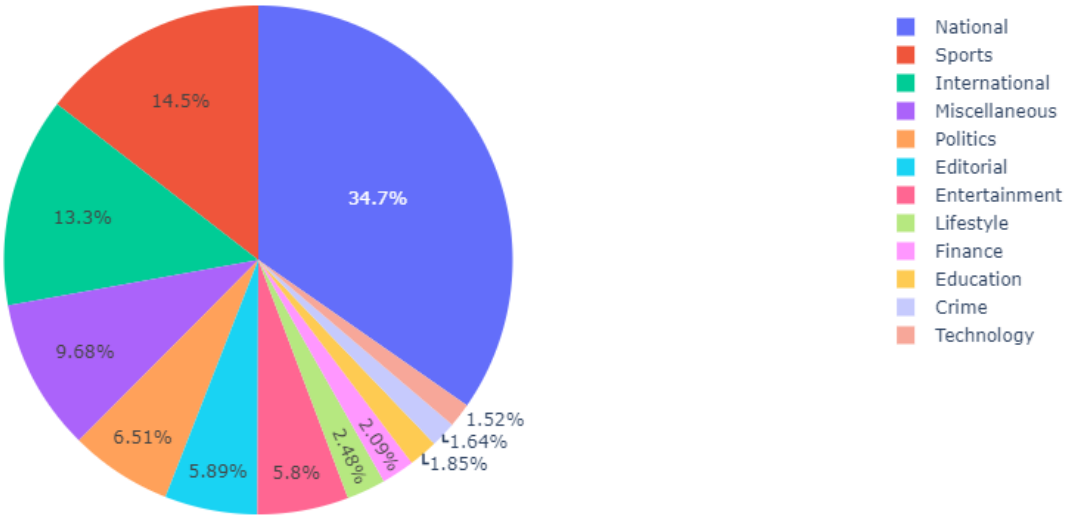


Figure 3.3: Number of Fake News on Each Category

From the figure we can see that most fake news are from "Miscellaneous" category. That means they donât target any particular topic. They are more broad and distributed to multiple topics. Also we have found that the entertainment category is another big sector of fake news. Thatâs because of the increasing satirical fake news.

3.1.1 Average Length of Words

We have then further evaluated the dataset and compared the average length of words for both fake and true news and found the following.

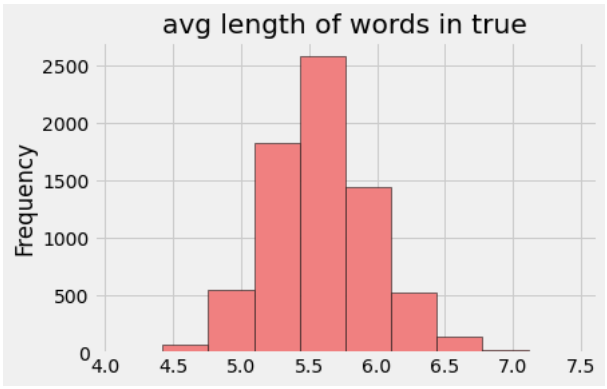


Figure 3.4: Average length of words in True News

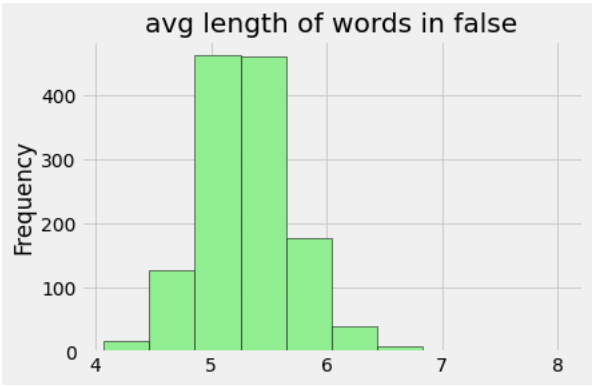


Figure 3.5: Average length of words in False News

If we closely look at this then we can clearly see that true news have more words frequency then the fake news. That means fake news are not too long. They are typically shorter. Also if we look at the average length of the words then we can see true news have more dynamic length of words then the fake news. That means fake news typically restricted to short and almost static length of words. Thatâs why there are almost 400 words in the same length range.

3.1.2 Most Frequent Words

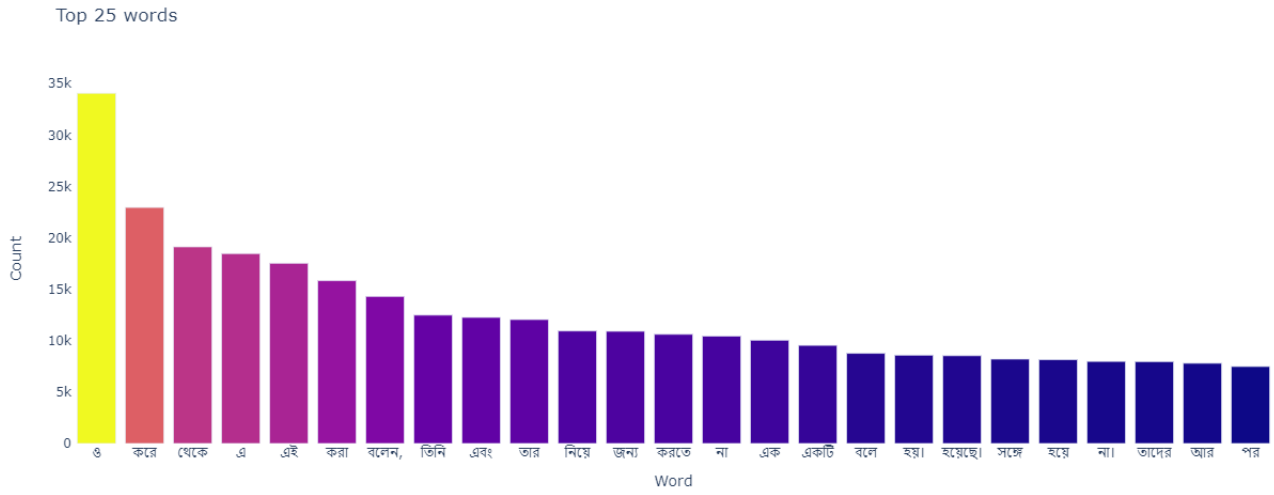


Figure 3.6: Most 25 frequent word

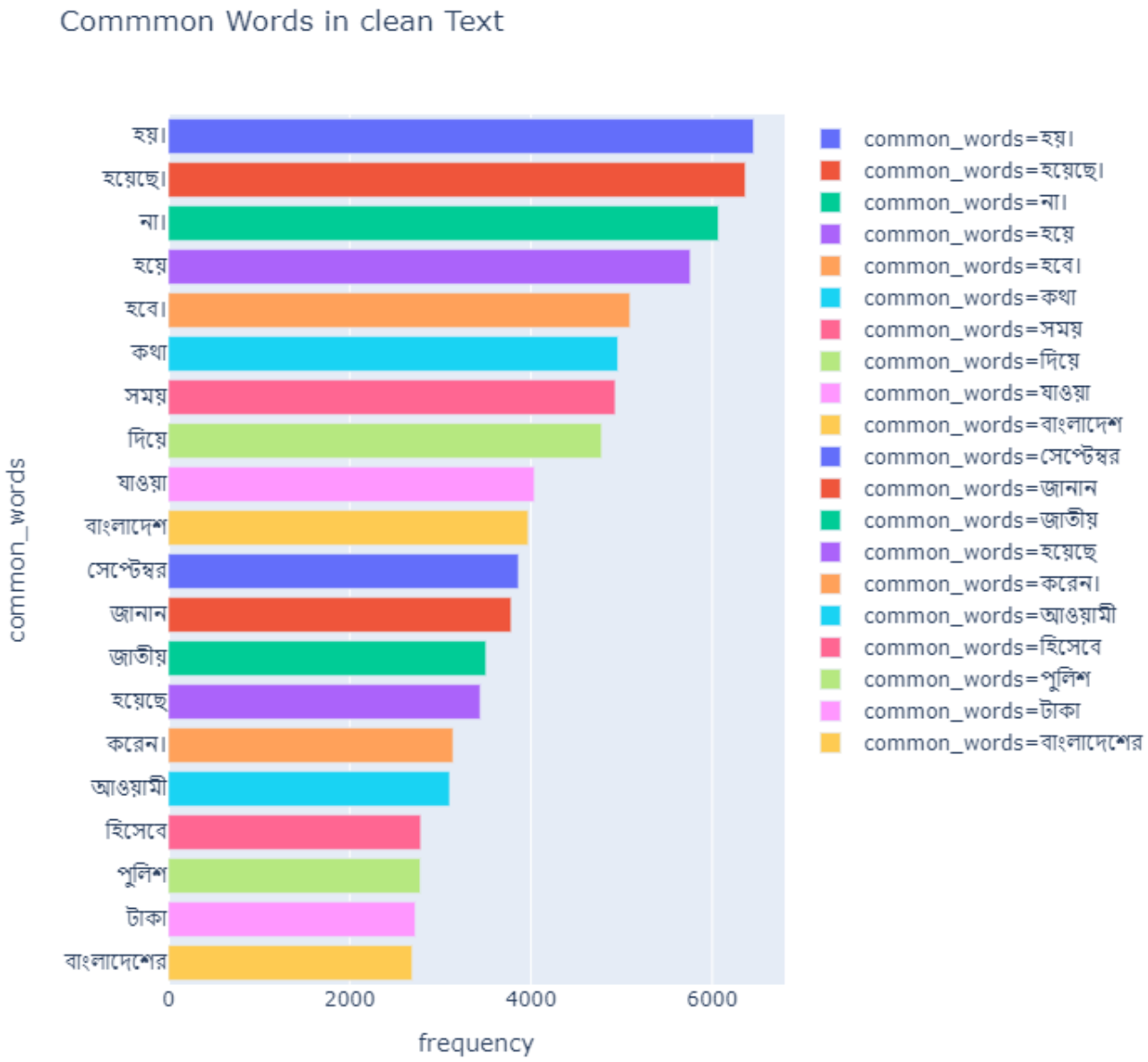


Figure 3.7: Most common words after cleaning stop wards

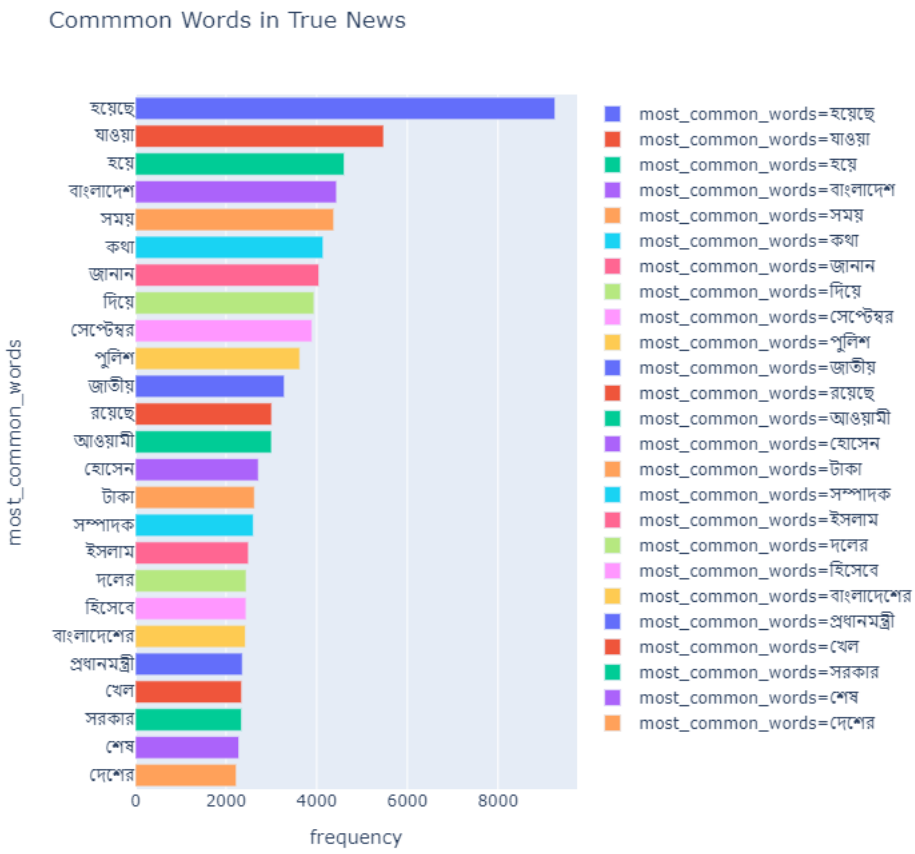


Figure 3.8: Most Common Words in True News

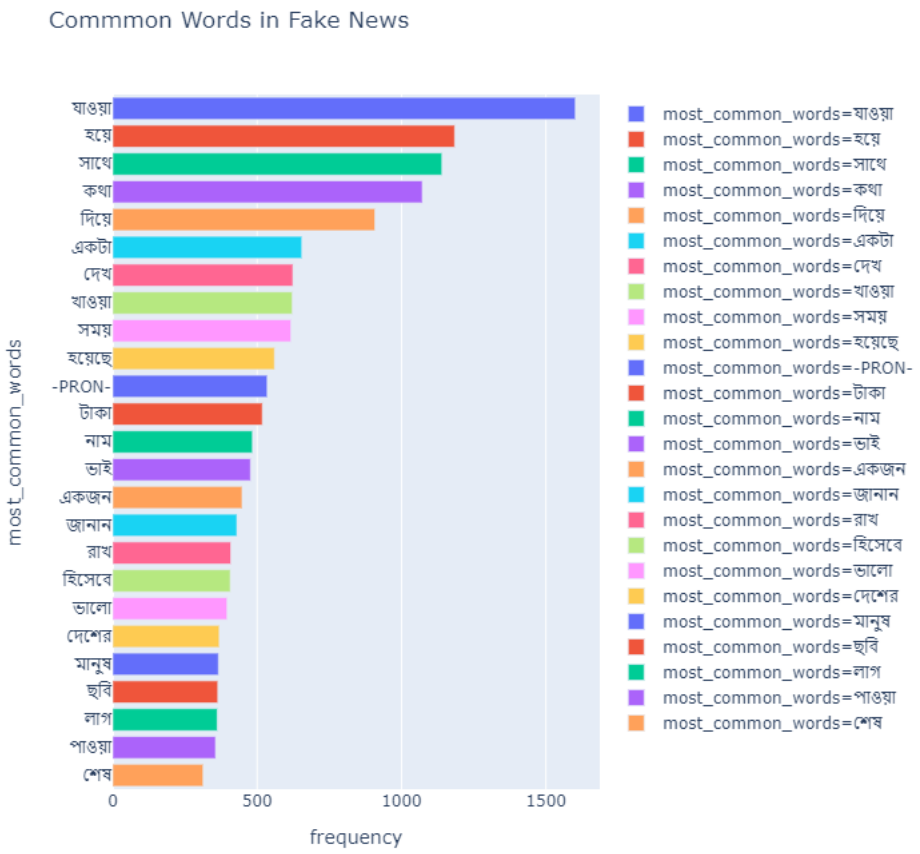


Figure 3.9: Most Common Words in Fake News

3.1.3 Top Punctuation Marks

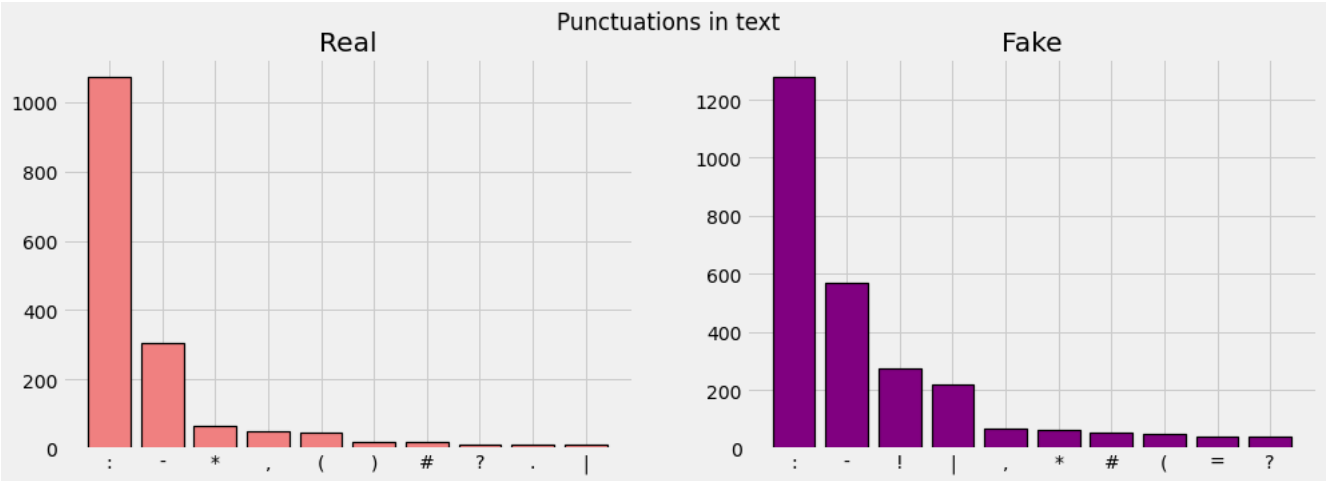


Figure 3.10: Distribution of punctuation marks

After cleaning the dataset we have again evaluated it to check changes in the most common words and got the following results.

3.1.4 Word Cloud

Figure 3.11: wordcloud of the clean dataset

3.2 N-gram analysis

An n-gram is a continuous sequence of n elements from a given sample of text or voice in the areas of computational linguistics and probability.[1] Typically, the n-grams are extracted from a text or voice corpus. When the elements are words, n-grams are often referred to as shingles.

3.2.1 Bi-gram

Bigrams are utilised in the majority of effective voice recognition language models.[28] They are a subset of the N-gram. In cryptography, bigram frequency attacks may be used to decrypt cryptograms. Consult the frequency analysis section. We discover bigrams in the Bigram Language Model, which are two words that combine in the corpus (the whole collection of words/sentences). For instance: Edpresso is great, and the bigrams are as follows: "Edpresso is" "is wonderful"

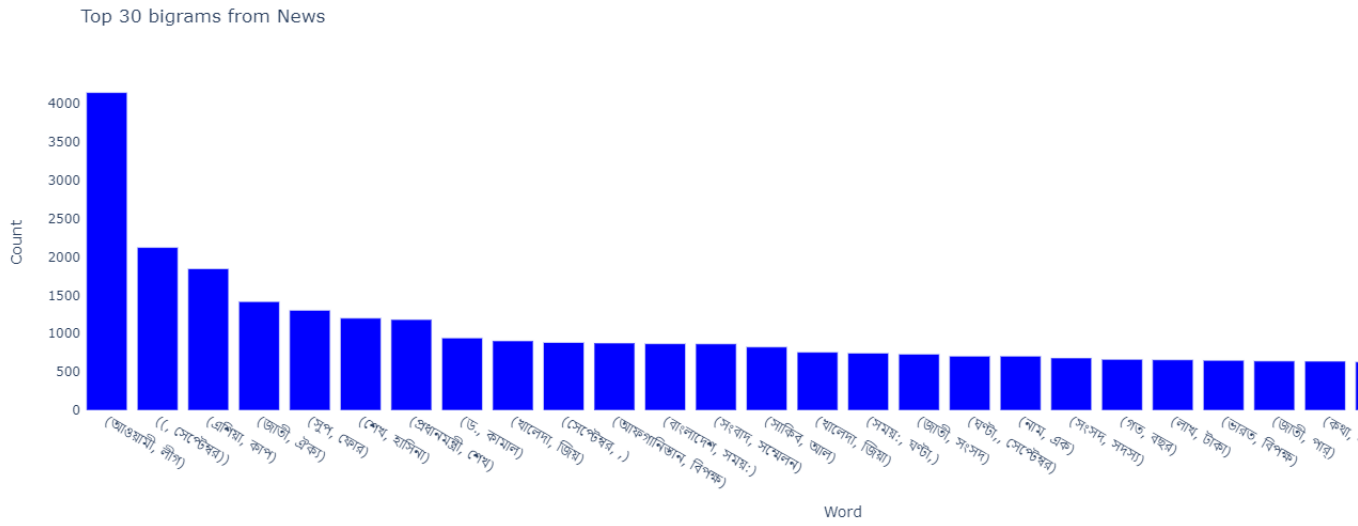


Figure 3.12: Top bigram from merge dataset

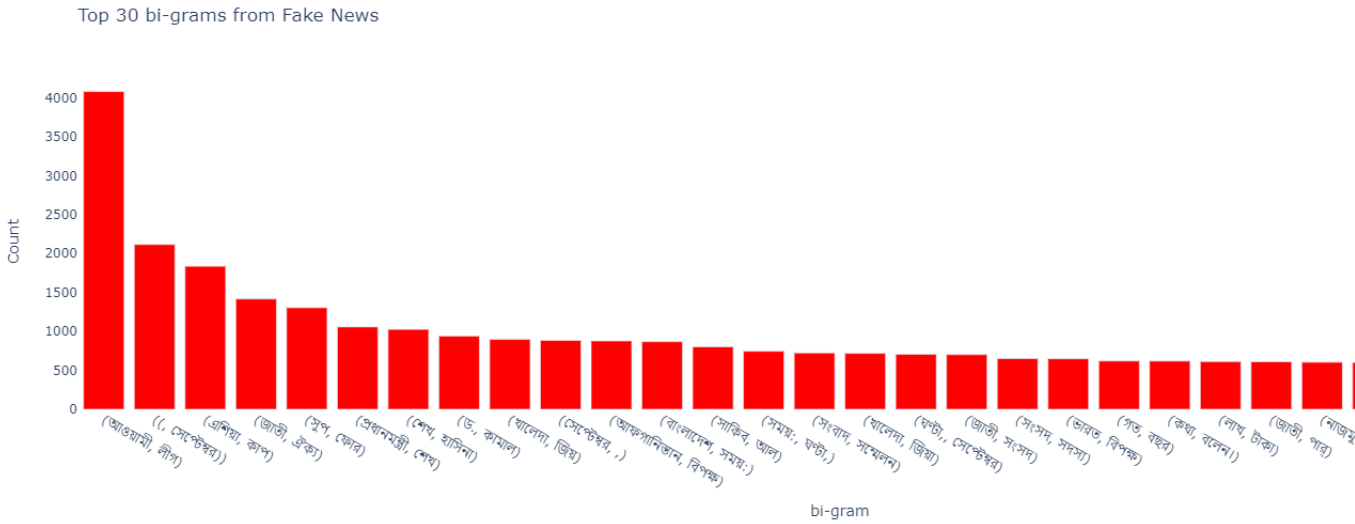


Figure 3.13: Top 30 bigram for Fake News

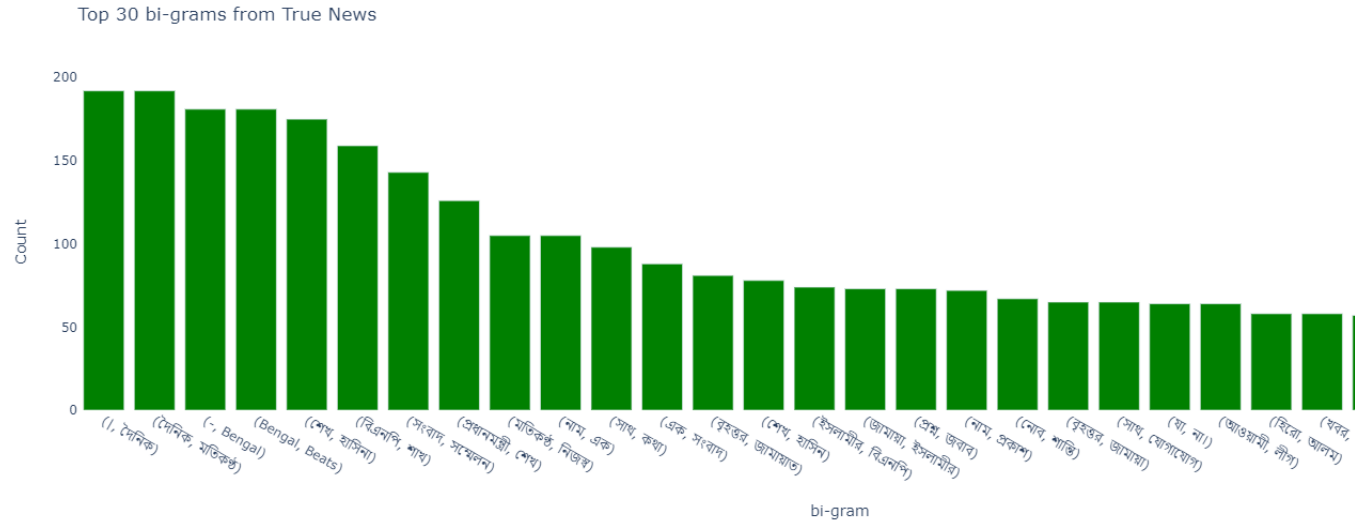


Figure 3.14: Top 30 Bigram from True News

3.2.2 Tri-gram

Trigrams are a subset of the n-gram, where n equals three. They are often used in natural language processing to conduct statistical analysis on texts and in cryptography to regulate the usage and control of cyphers and codes.

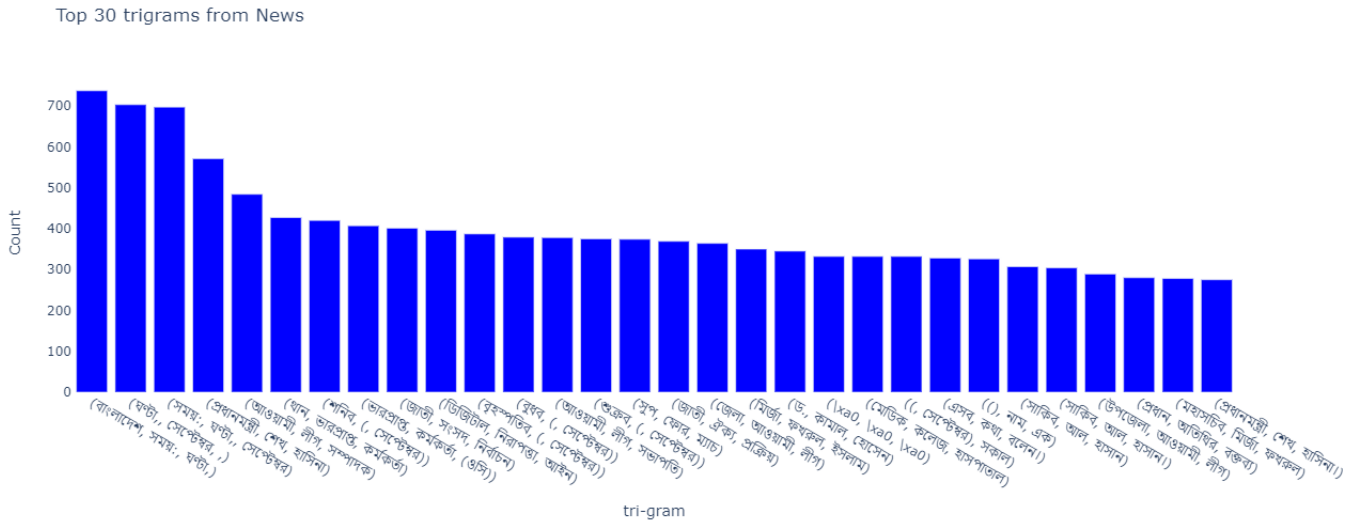


Figure 3.15: Top trigram from merge dataset

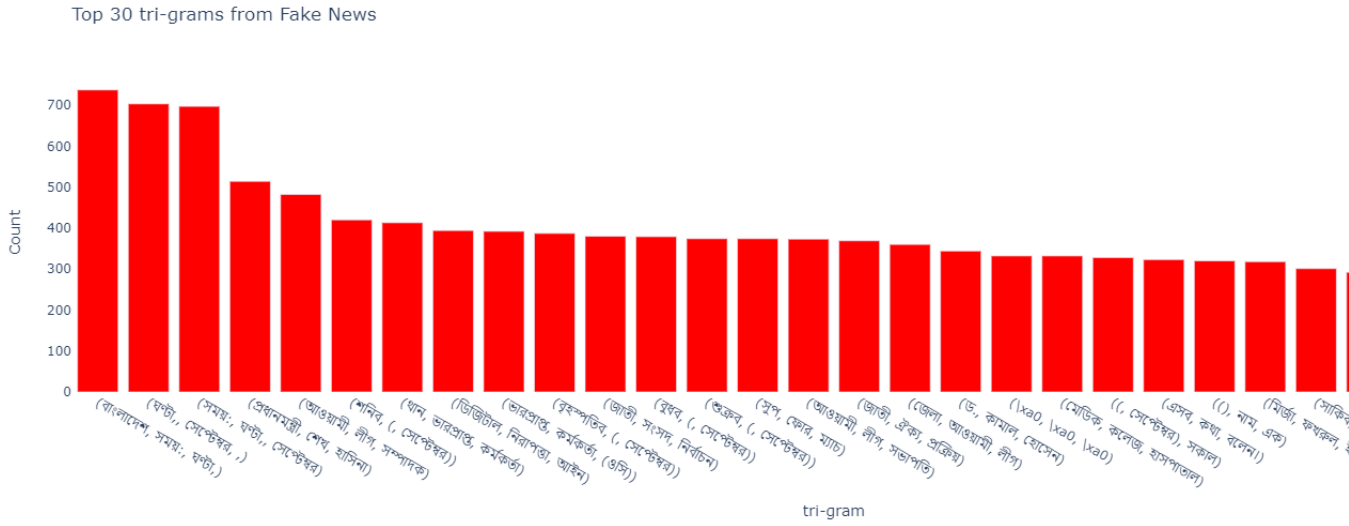


Figure 3.16: Top trigram from fake dataset

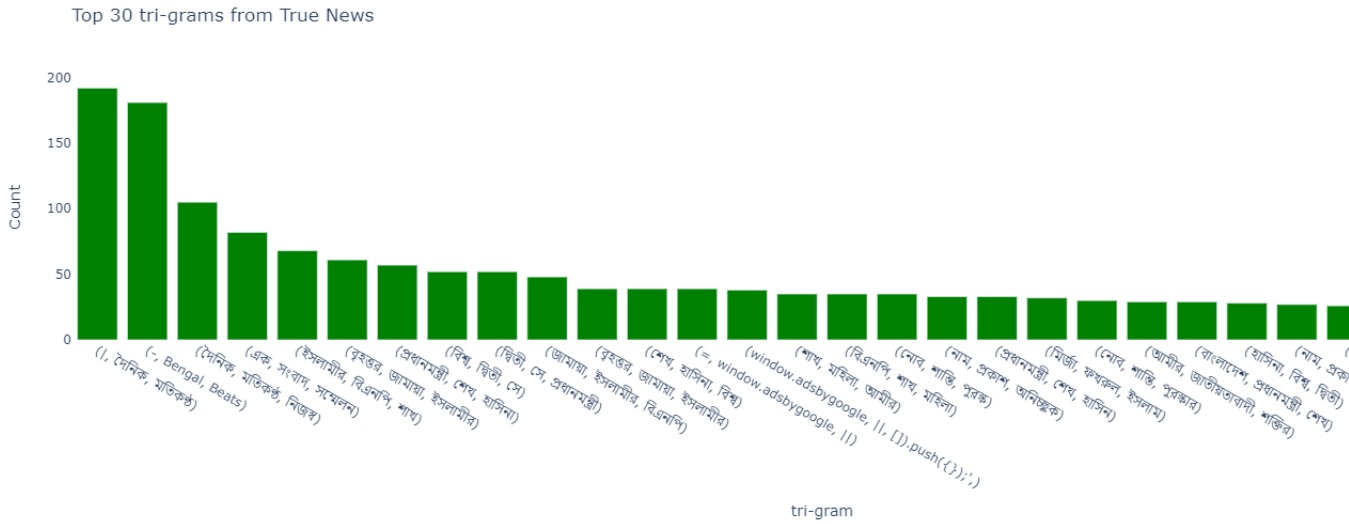


Figure 3.17: Top trigram from true dataset

3.3 Data Pre-processing

3.3.1 Removing stopwords

Stopwords are the most commonly occurring words in a language, and they often don't contain any relevance with the context. That's why to get better classification results, removing them from the dataset is necessary. Now, as NLTK and other popular NLP libraries have no built-in Bengali stopwords corpus, we have used our custom-made built-in stopwords list and then used it to remove the stopwords from the dataset.

3.3.2 Cleaning dataset

Unnecessary characters and punctuation marks often make the training process biased. As they don't have any meaning, we have further checked the whole dataset for punctuation marks and other invalid characters and then removed them.

3.3.3 POS Tagging

Pos tagging means tagging all the words according to their parts of speech. It is a very crucial task. It helps the machine learning models to identify the true meaning and the context of a

language. It also helps to lemmatize words properly. To POS Tag the words, we have used bnltk POS tagger. [29]

3.3.4 Stemming

Stemming is the method of getting the word's stem or root form. Due to the fact that we will be using semantic word embedding and pre-trained word vectors, it is important to get the root form of the words in our dataset. We have used the bnltk [30] word stemmer to get the stems.

3.3.5 Lemmatizing

Lemmatization is a method for grouping words that exist in more than one form in a language. For instance, suppose one of the words is "Make," and the others are "Makes" and "Made." "Makes" and "made" are alternative spellings of the word Make. They are classified as Make by lemmatization. Thus, since lemmatization enables us to query root terms and get more relevant results, we choose to lemmatize the words. We utilised banglakit lemmatizer to lemmatize the terms. [29]

3.4 Feature Extraction

Feature extraction is the method to extract the features from our dataset. Mainly In the case of natural language processing, we have to convert our textual data to word vectors. We do that task in the feature extraction process. There are many feature extraction methods but among them, we have choosen the n-gram feature extraction model to extract the lexical features and then word2vec to extract the semantic features

3.4.1 N-Gram[1]

An n-gram is a contiguous sequence of n elements drawn from a particular sample of text or voice in the disciplines of computational linguistics and probability. The elements can be anything like phonemes, syllables, letters, words, or base pairs. Typically, the n-grams are extracted from a text or voice corpus. Here, we have to use it to extract features from our textual dataset. We'll use TF-IDF and Countvectorizer as our n-gram model.

3.4.2 word2vec

While the TF-IDF vectors also create vectors from texts, they cannot adequately capture the context, while the word2vec vector attempts to extract the document's semantic connections. [4] To determine similarity, the cosine of the angle between two vectors was applied, which is referred to as Cosine Similarity. And the word2vec algorithm is implemented using the CBOW (Continuous Bag Of Words) and Skip-gram models. Additionally, we experiment with a second set of pre-trained 100-dimensional word vectors trained using Word2Vec on 20K Bengali news. [31] Also Later, we utilised the Bengali Glove Vector, which has been trained on 20 million Wikipedia tokens.[32]

3.5 Classification Models

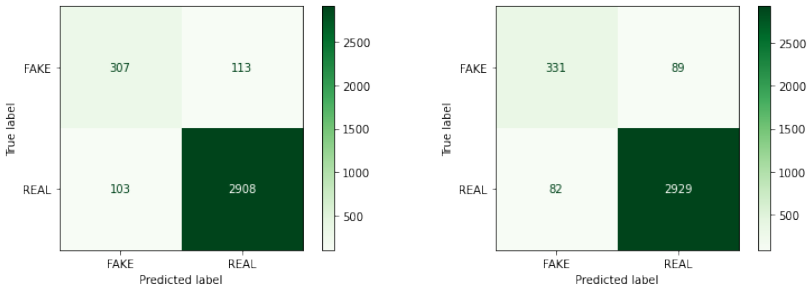
We began by extracting the stemmed terms from our dataset and evaluating the models against the stemmed text. The models were then tested on the lemmitized text in order to compare their results. We utilised tfidf and countvectorizers to embed words in both processes. Following that, we analysed the dataset without stemming or lemmatizing it. Lastly, we used character-based tfidf embedding to assess the models.

3.5.1 Support Vector Machine (SVM)

Initially, We have used the liniear kernel of SVM. We used C=100 as our tweaking parameter. The C parameter instructs the SVM optimization to mitigate the misclassification. For large values of C, the optimization will choose a smaller-margin hyperplane if it does a better job of accurately classifying all of the training points. [5] We have evaluated the model the following contexts.

3.5.1.1 Using Stemmed text

At first we evaluated the model for Stemmed text. We have evaluated for both TFIDF and Countvectorizer and got 93.7% accuracy for countvectorizer and 95.02% accuracy for TF-IDF vectorizer. The confusion matrix is the following.



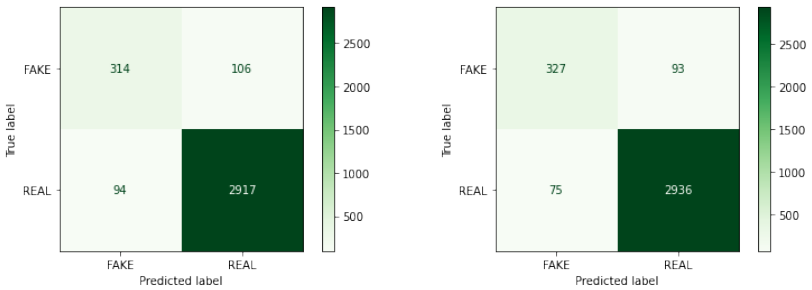
(a) Confusion Matrix for CountVectorizer

(b) Confusion Matrix for TF-IDF

Figure 3.18: Confusion Matrix of SVM for Stemmed Text

3.5.1.2 Using Lemmatized text

We evaluated the model with the same settings for lemmatized text and got 94.17% accuracy for CountVectorizer and 95.1% accuracy for TF-IDF. The confusion matrix for the both processes is the following.



(a) Confusion Matrix for CountVectorizer

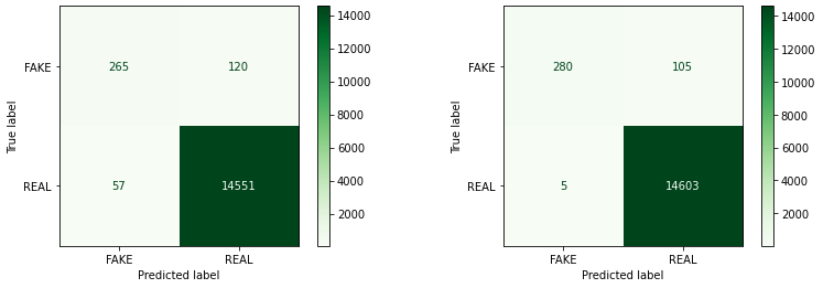
(b) Confusion Matrix for TF-IDF

Figure 3.19: Confusion Matrix of SVM for Lammatized Text

3.5.1.3 Using Word-Based Embedding

We later evaluated the model without builtin bengali lemmatizer and stemmer and used sklearn based tf-idf and countvecxtorizers and got 97.32% accuracy for countvectorizers. We have further evaluated the same model for tf-idf vectorizer as our word embedding model to see if we could find any better result and to our surprise, we have seen that tf-idf even gave us 98.41% accuracy.

The confusion matrix for both models is the following.



(a) Confusion Matrix for CountVectorizer

(b) Confusion Matrix for TF-IDF

Figure 3.20: Confusion Matrix of SVM for Word Embedding

3.5.1.4 Using Character-Based Embedding

Then we have used character based embedding instead of word embedding with TF-IDF to see if we could find any improvement in our result and we have got 98.45% accuracy. But to our surprise, we have seen that the number of true negative boosted to 402 out of 442 fake news. Also, the true positive rate was much better. The confusion matrix is the following.

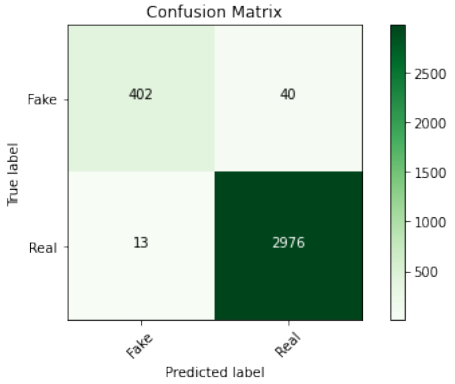


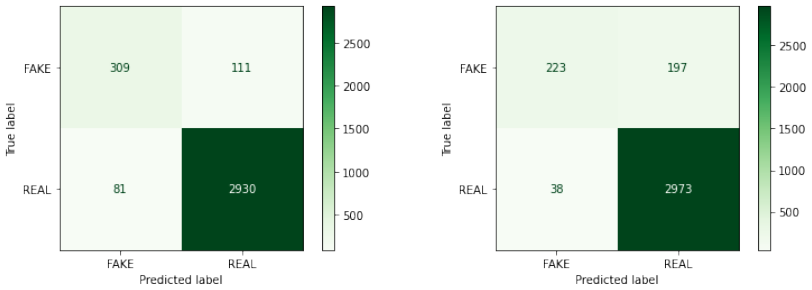
Figure 3.21: Confusion Matrix of SVM for Character-Embedding

3.5.2 Logistic Regression (LR)

As our dataset is showing better results for liner models, we tried our luck with Logistic Regression. We have done the same for logistic regression also to see the perfomance of this model for stemmed text, Lemmatized text, word-based embedding and character-based embedding.

3.5.2.1 Using Stemmed text

In the case of stemmed text, logistic regression gives us an accuracy of 94.4% and 93.15% for countvectorizer and TF-IDF vectorizer respectitively. The confusion matrix is the following.



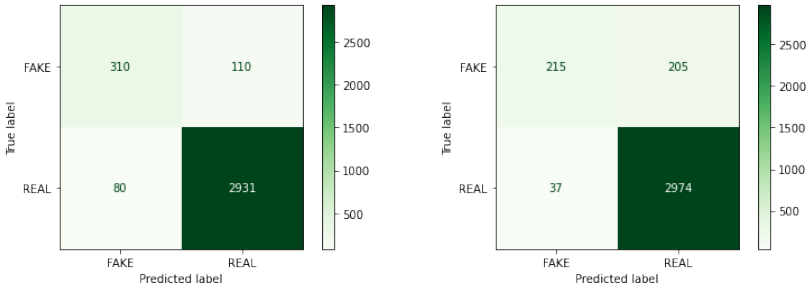
(a) Confusion Matrix for CountVectorizer

(b) Confusion Matrix for TF-IDF

Figure 3.22: Confusion Matrix of LR for Stemmed Text

3.5.2.2 Using Lemmatized text

In case of lemmatized text the highest accuracy was for countvectorizer and that was 94.46%. So it's almost gives similar performances for both stemmed text and lemmatized text.



(a) Confusion Matrix for CountVectorizer

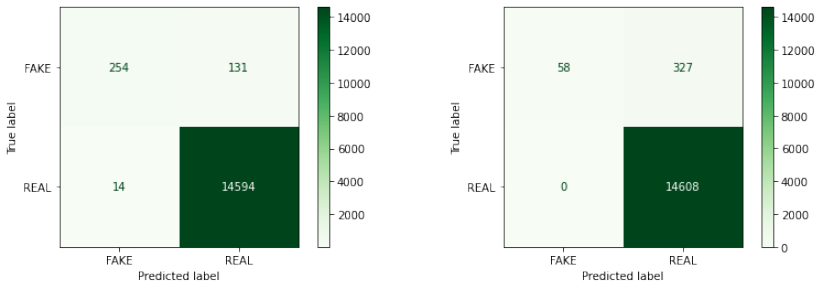
(b) Confusion Matrix for TF-IDF

Figure 3.23: Confusion Matrix of LR for Lemmatized Text

3.5.2.3 Using Word-Based Embedding

Then we have further evaluated Logistic Regression classifier without builtin stemmer and lemmatizer and have found an accuracy of 98.12%. We have then tried the model with tf-idf word

vectorizer and got an accuracy of 97%. So, here CountVectorizer performs well for model. The confusion matrix for both cases are the following.



(a) Confusion Matrix for CountVectorizer

(b) Confusion Matrix for TF-IDF

Figure 3.24: Confusion Matrix of LR for Word Embedding

3.5.2.4 Using Character-Based Embedding

Now, we have evaluated Logistic Regression on the character-based embedding model and used TF-IDF for char vector and got an accuracy of 94.87%. So, The Logistic Regression only performs well for Counvectorizer and with only word tokenizer.

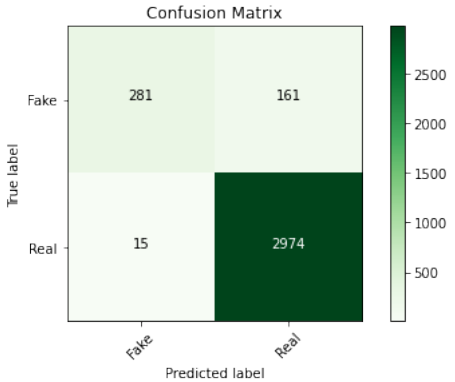


Figure 3.25: Confusion Matrix of LR for Character-Embedding

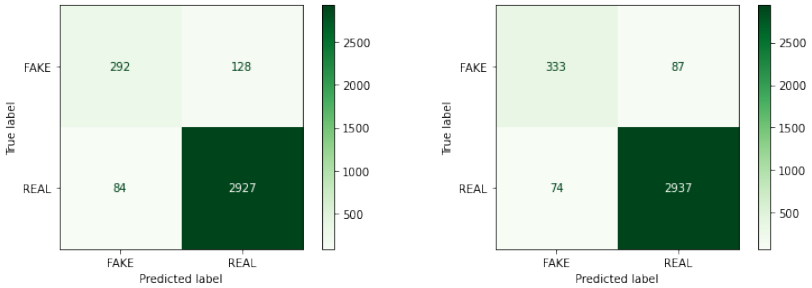
3.6 Passive Agressive Classifier

Passive Agressive Classifier often performs well for NLP based tasks and learning. So, We have used passive aggressive classifier and used max_iter = 10 as our tweaking parameter and

evaluated this for the following methods.

3.6.0.1 Using Stemmed text

Passive Agressive also shows moderate performance for the stemmed text that we have found using bnltk word stemmer. It gives us 95.31% accuracy for TF-IDF and 93.82% for countvectorizer.



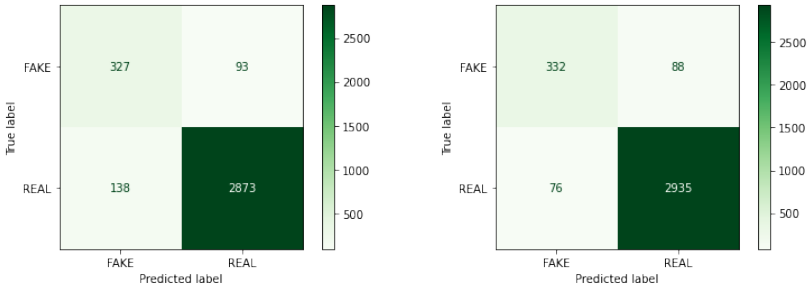
(a) Confusion Matrix for CountVectorizer

(b) Confusion Matrix for TF-IDF

Figure 3.26: Confusion Matrix of PAC for Stemmed Text

3.6.0.2 Using Lemmatized text

In case of lemmatized text the accuracy was 93.27% and 95.22% for countvectorizer and TF-IDF respectatively. So, it also performs almost similar like as stemmed text.



(a) Confusion Matrix for CountVectorizer

(b) Confusion Matrix for TF-IDF

Figure 3.27: Confusion Matrix of PAC for Lemmatized Text

3.6.0.3 Using Word-Based Embedding

We have further evaluated Passive Agressive Classifier without using the builtin stemmer and lemmatizer and found a boost of 99.33% accuracy for TF-IDF vectorizer.

3.6.0.4 Using Character-Based Embedding

We evaluated the model on character-based embedding and again got very good reasult with an accuracy of 98.39%. It also detects the 2nd higher amount of ture negatives now and it is **400**. The confusion matrix is the following.

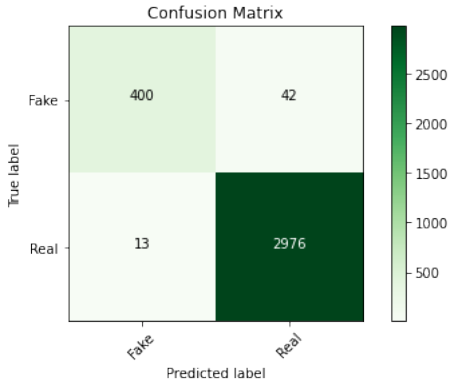


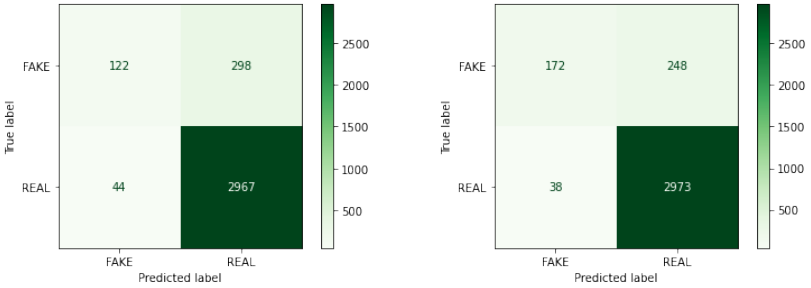
Figure 3.28: Confusion Matrix of PAC for Character-Embedding

3.6.1 K-nearest Neighbour (KNN)

K-nearest Neighbour is just another linear classifier which performs well for text based data. So, we evaluated it for both stemmed text and lemmatized text and observed the performances. We have used k=3 as our tweaking parameter for the model.

3.6.1.1 Using Stemmed text

For stemmed text, KNN gives us an accuracy of 91.66% for TF-IDF vectorizer and 90.03% accuracy for countvectorizer. So, it performs poorly on the stemmed data.



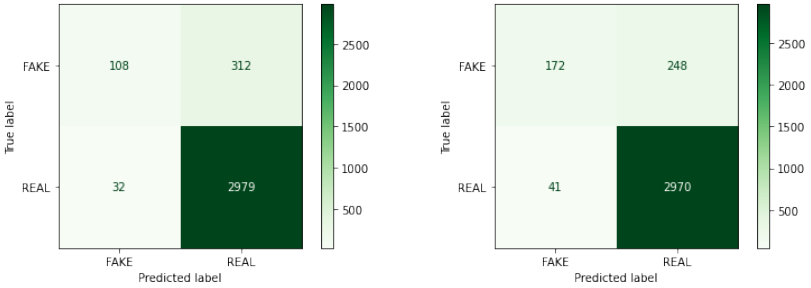
(a) Confusion Matrix for CountVectorizer

(b) Confusion Matrix for TF-IDF

Figure 3.29: Confusion Matrix of KNN for Stemmed Text

3.6.1.2 Using Lemmatized text

We further evaluated it to see if it can give us any improvement for the lemmatized data and it gives us an highest accuracy of 91.58% in case of TF-IDF vectorizer.



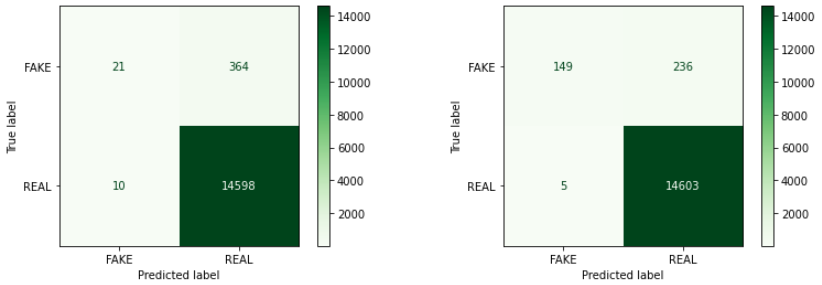
(a) Confusion Matrix for CountVectorizer

(b) Confusion Matrix for TF-IDF

Figure 3.30: Confusion Matrix of KNN for Lemmatized Text

3.6.1.3 Using Word-Based Embedding

Now Similarly, we have evaluated it without stemmed text and lemmatizer on CountVectorizer model and got accuracy of 99.20% accuracy. Then, we have evaluated it using tf-idf as our feature extraction model and got 99.23%. We got almost the same result as the CountVectorizer model but more improved result than the previous methods. The confusion matrix for both evaluations is as follows.



(a) Confusion Matrix for CountVectorizer

(b) Confusion Matrix for TF-IDF

Figure 3.31: Confusion Matrix of KNN

3.6.1.4 Using Character-Based Embedding

Now as the previous models are performing so well for character-based embedding, We have also evaluted the same for KNN but this time KNN gives an accuracy of 92.33%. So, KNN performs only well for the countvectrizer word-embedding models.

3.6.2 Multinomial Naive Bayes (MNB)

After KNN, we have used multinomial Naive Bayes as our classifier. Multinomial naive Bayes also have a tuning parameter α. We trained the model using different values of α from range (0 to 1) and have found that α = 0.01 gave us a satisfactory result. Below is shown the different alpha values we have tried to tune the classifier and their outputs.

Alpha: 0.00 Score: 98.23

Alpha: 0.01 Score: 98.45

Alpha: 0.05 Score: 98.15

Alpha: 0.10 Score: 98.02

Alpha: 0.15 Score: 98.06

Alpha: 0.20 Score: 98.15

Alpha: 0.25 Score: 98.13

Alpha: 0.30 Score: 98.03

Alpha: 0.35 Score: 97.91

Alpha: 0.40 Score: 97.87

Alpha: 0.45 Score: 97.81

Alpha: 0.50 Score: 97.75

Alpha: 0.55 Score: 97.65

Alpha: 0.60 Score: 97.63

Alpha: 0.65 Score: 97.63

Alpha: 0.70 Score: 97.59

Alpha: 0.75 Score: 97.59

Alpha: 0.80 Score: 97.59

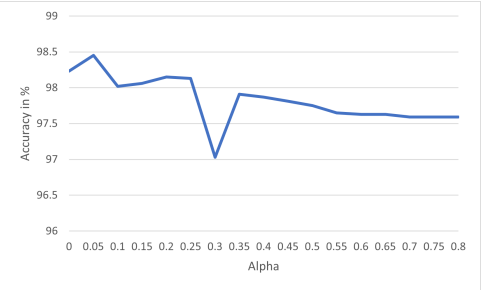
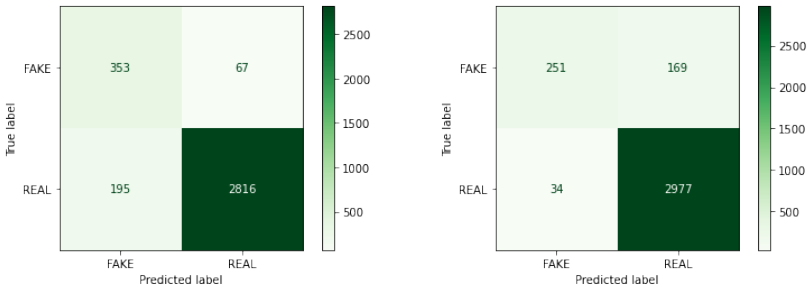


Figure 3.32: Change of Accuracies against Alpha Values

So, we have used the same parameter for all the following conditions.

3.6.2.1 Using Stemmed text

Again the the stemmed text we have found using the builtin stemmer doesn't perform well and gives us an accuracy of 92.36% in case of TF-IDF vectorizer.



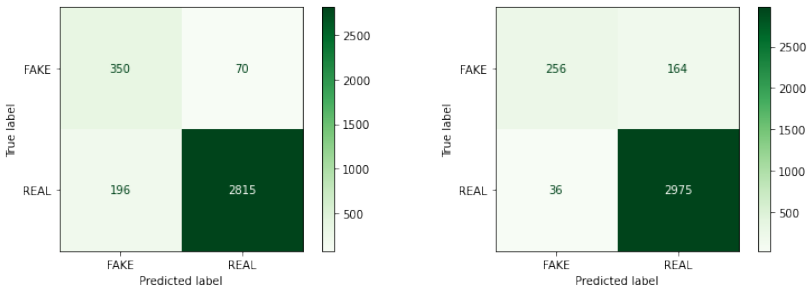
(a) Confusion Matrix for CountVectorizer

(b) Confusion Matrix for TF-IDF

Figure 3.33: Confusion Matrix of MNB for Stemmed Text

3.6.2.2 Using Lemmatized text

We have evaluated it for the lemmatized text that we have got using builtin lemmatizer and got an accuracy of 92.25% accuracy for countvectorizer which is almost similar to stemmed text.



(a) Confusion Matrix for CountVectorizer

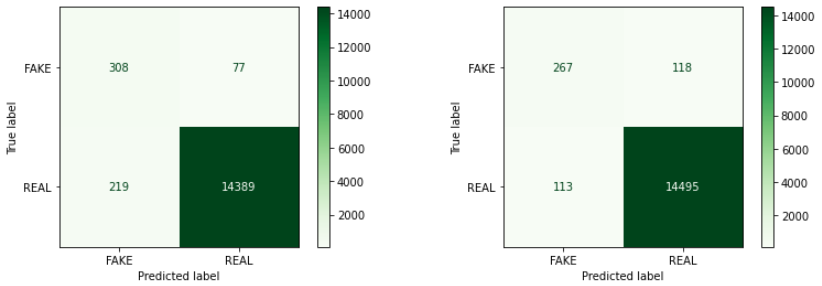
(b) Confusion Matrix for TF-IDF

Figure 3.34: Confusion Matrix of MNB for Lemmatized Text

3.6.2.3 Using Word-Based Embedding

Now we have evaluated the same model with our self pre-processor and tokenizer and as expected we have got 98% accuracy in the case of CountVectorizer and 98.45% accuracy in case

of tf-idf vectorizer. So, in that case tf-idf slightly performs better. The confusion matrix for both cases is as follows



(a) Confusion Matrix for CountVectorizer

(b) Confusion Matrix for TF-IDF

Figure 3.35: Confusion Matrix of MNB for Word Embedding

3.6.2.4 Using Character-Based Embedding

We have further evaluated the model for character-based embeddings and got an accuracy of 95.31%. it detects 372 true negatives from 440 total fake news. The confusion matrix is the following.

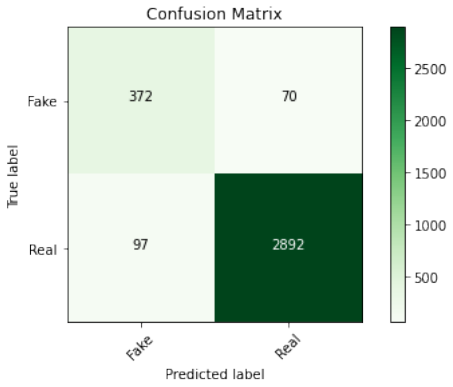


Figure 3.36: Confusion Matrix of MNB for Character-Embedding

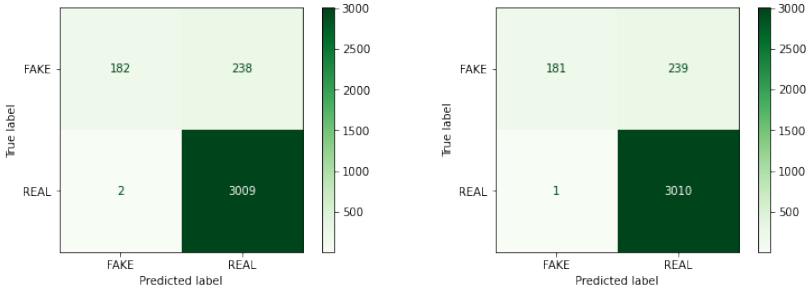
3.6.3 Random Forests

Next we have tried the ensemble learning model Random Forests. Rondom Forest is basically a combination of decision trees. Like other models Random Forest also can be slightly tweaked. We have used different tree depth to see how it performs on our dataset and have found that when

the max_depth parameter is 400 it gave us the best result. We have also used **criterion** as our entropy as it slowly increases the performances of the model according to the max_depth.

3.6.3.1 Using Stemmed text

In case of stemmed text the model gives us an accuracy of 93.82% in case of countvectorizer and 95.31% in case of TF-IDF.



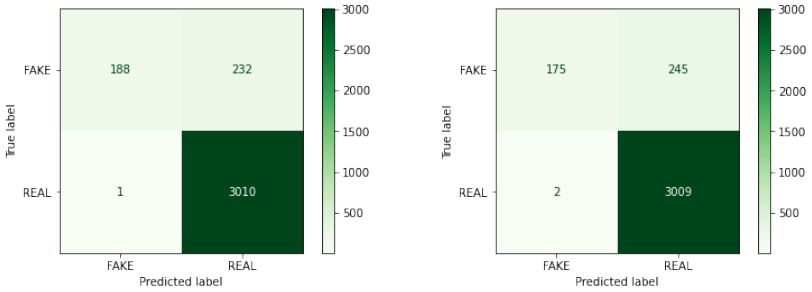
(a) Confusion Matrix for CountVectorizer

(b) Confusion Matrix for TF-IDF

Figure 3.37: Confusion Matrix of RF for Stemmed Text

3.6.3.2 Using Lemmatized text

In case of stemmed text the model gives us an accuracy of 93.27% in case of countvectorizer and 95.22% in case of TF-IDF. So, lemmatized text almost performs the similar to stemmed text.



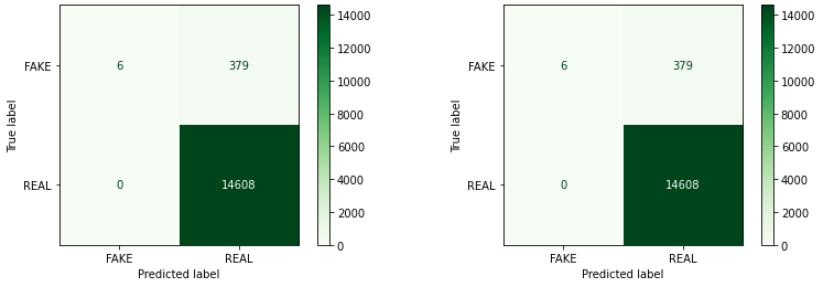
(a) Confusion Matrix for CountVectorizer

(b) Confusion Matrix for TF-IDF

Figure 3.38: Confusion Matrix of RF for Lemmatized Text

3.6.3.3 Using Word-Based Embedding

Now in case of our self pre-processed data without using any builtin stemmer and lemmatizer, We have got 98.65% accuracy in case of CountVectorizer as our feature extractor. In case of tf-idf vectorizer, we have also got 98.92% accuracy almost the same as the CountVectorizer. The confusion matrix for both model is as follows.



(a) Confusion Matrix for CountVectorizer

(b) Confusion Matrix for TF-IDF

Figure 3.39: Confusion Matrix of RF for Word-Embedding

3.6.3.4 Using Character-Based Embedding

In case of character-based embedding Random Forest gives us an accuracy of 93.87%. So it performs only well for word-based embedding and our self processed data.

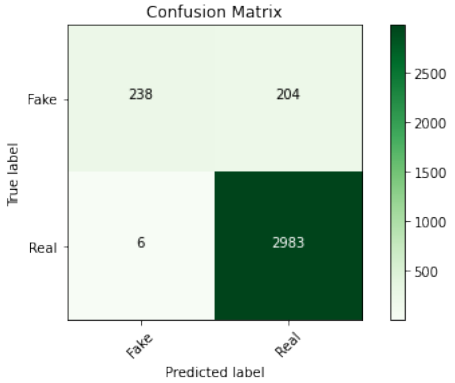


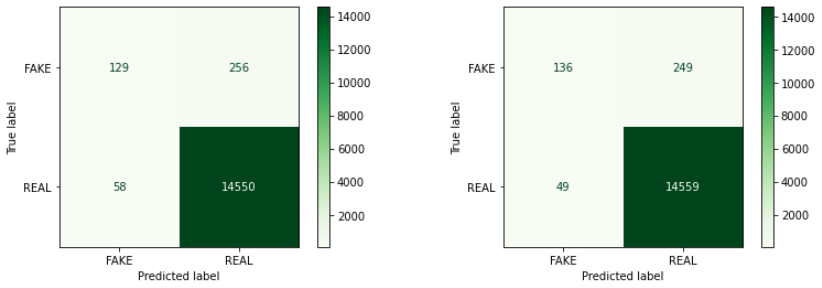
Figure 3.40: Confusion Matrix of MNB for Character-Embedding

3.6.4 AdaBoost

Adaboost is just another ensemble algorithm with adaptive boosting. This type of boosting algorithms can learn from their mistakes and then can build better models to reduce bias errors. Also, Adaboost has much less parameter tweaking in comparison to SVM algorithms. Thatâs why we tried this model to see how it performs on the bengali dataset. We have evaluated AdaBoost classifier only to our self processed data and it gives the following perfomances according to our method.

3.6.4.1 Using Word-Based Embedding

We have got 97.90% accuracy in case of countvectorizer and 98% accuracy in case of tf-idf vectorizer. The confusion matrix of the tasks are as follows.



(a) Confusion Matrix for CountVectorizer

(b) Confusion Matrix for TF-IDF

Figure 3.41: Confusion Matrix of AdaBoost for Word-Embedding

3.6.4.2 Using Character-Based Embedding

For character-based embedding AdaBoost gives us an accuracy of 94.75% in case of TF-IDF vectorizer. So, AdaBoost performs only well for word-based embedding.

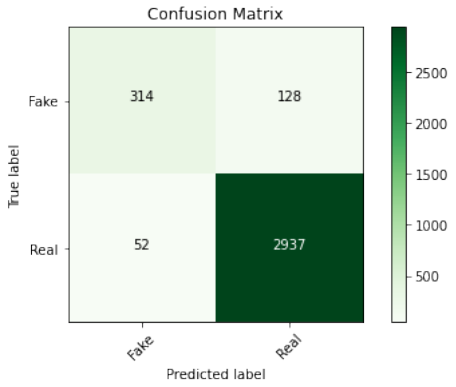


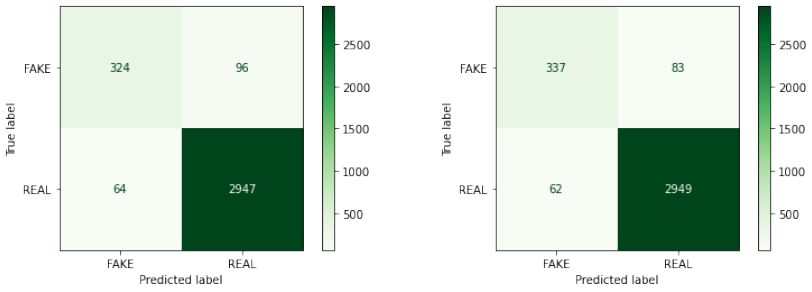
Figure 3.42: Confusion Matrix of AdaBoost for Character-Embedding

3.6.5 Neural Network

Here, we have used the builtin multilayer perceptron or MLP classifier of sklearn library as our Neural Network classifier. We have used hidden_layer_sizes=(33,),max_iter=500 as out parameters to tweak the model and evaluated the model in the following cases.

3.6.5.1 Using Stemmed text

In case of stemmed text the neural netowork gives us an accuracy of 95.77% in case TF-IDF and 95.34% in case of countvectorizer.



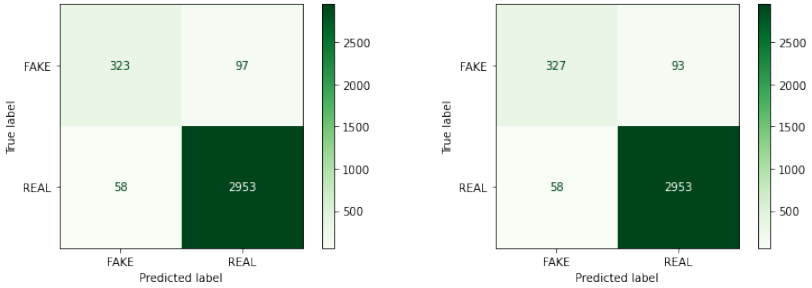
(a) Confusion Matrix for CountVectorizer

(b) Confusion Matrix for TF-IDF

Figure 3.43: Confusion Matrix of NN for Stemmed Text

3.6.5.2 Using Lemmatized text

In the case of lemmatized text, we have got an accuracy of 95.48% accuracy and 95.6% accuracy for countvectorizer and TF-IDF respectively.



(a) Confusion Matrix for CountVectorizer

(b) Confusion Matrix for TF-IDF

Figure 3.44: Confusion Matrix of NN for Lemmatized Text

3.6.5.3 Using Character-Based Embedding

Now we have evaluated the model with the same settings for character-based embedding and got an accuracy of 98% for TF-IDF. The model also detects **385** true fake news form 440 fake news. The confusion matrix is in below.

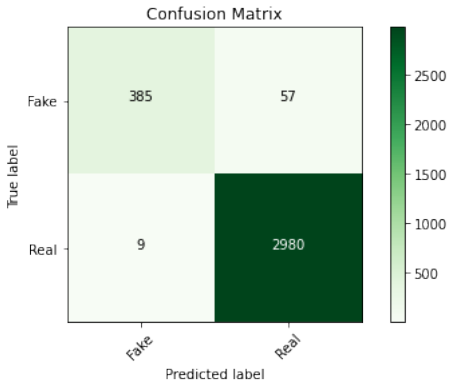


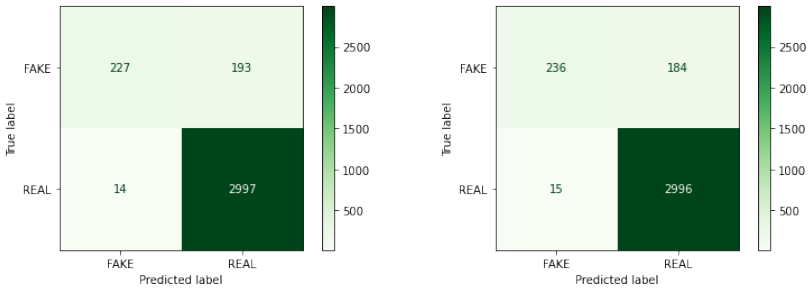
Figure 3.45: Confusion Matrix of NN for Character-Embedding

3.6.6 XGBoost

Now we have used another gradient boosting algorithms that called XGBoost on our dataset and it gives us the following results according to our methods.

3.6.6.1 Using Stemmed text

In case of stemmed text, XGBoost gives us an accuracy of 93.97% in case of countvectorizer and 94.2% in case of TF-IDF vectorizer.



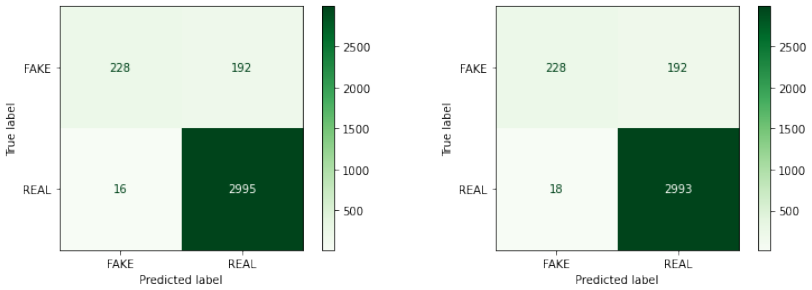
(a) Confusion Matrix for CountVectorizer

(b) Confusion Matrix for TF-IDF

Figure 3.46: Confusion Matrix of XGB for Stemmed Text

3.6.6.2 Using Lemmatized text

For lemmatized text, XGBoost gives an accuracy of 93.94% accuracy in case of countvecotrizer and 93.88% accuracy in case of TF-IDF vectorizer. The accuracy is slight lower than the stemmed text.



(a) Confusion Matrix for CountVectorizer

(b) Confusion Matrix for TF-IDF

Figure 3.47: Confusion Matrix of XGB for Lemmatized Text

3.6.6.3 Using Character-Based Embedding

We have further evaluated the same model with character-based embedding with our self processed data to see if it can improve the accuracy or not. We have got an accuracy of 95.22% which is slight better than the stemmed text case. But in the case of stemmed text, the rate of detection true negatives was much higher than this one.

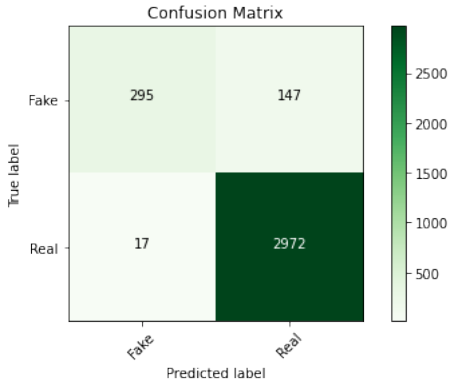


Figure 3.48: Confusion Matrix of XGB for Character-Embedding

3.6.7 CatBoost

We have further evaluated another gradient boosting algorithm catboost on character-base embedding to see any improvement and we have got a better accuracy than XGBoost.

3.6.7.1 Using Character-Based Embedding

In case of CatBoost, the character-based embedding model gives us an accuracy of 96.41% which is slightly better than XGBoost. The confusion matrix the given below.

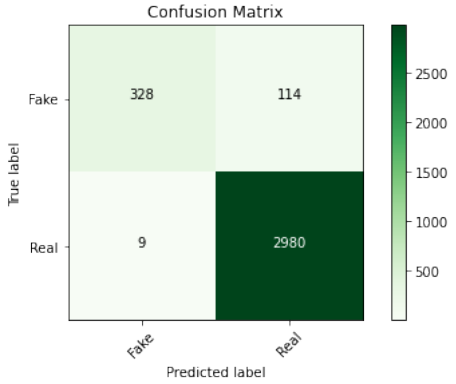


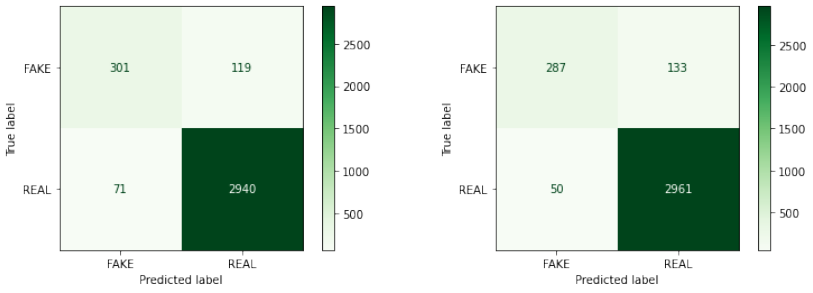
Figure 3.49: Confusion Matrix of CatBoost for Character-Embedding

3.6.8 Stochastic Gradient Descent (SGDC)

We have implemented another Gradient descent algorithm and evaluate on the stemmed and lemmatized text to compare the performances.

3.6.8.1 Using Stemmed text

In case of stemmed text, The model gives an accuracy of 94.67% in case of TF-IDF vectrizer and 94.46% accuracy in case of countvectorizer.



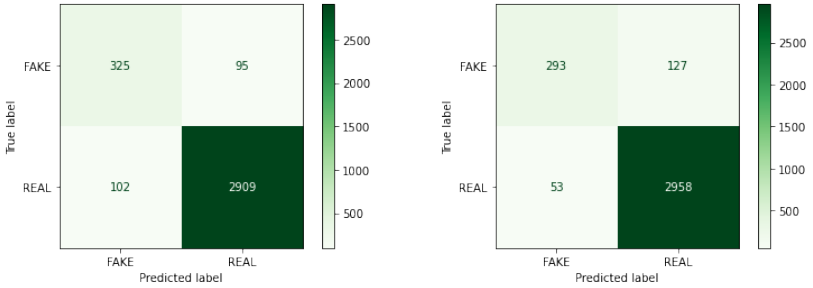
(a) Confusion Matrix for CountVectorizer

(b) Confusion Matrix for TF-IDF

Figure 3.50: Confusion Matrix of SGDC for Stemmed Text

3.6.8.2 Using Lemmatized text

For lemmatized text, The model gives an accuracy of 94.75% in case of TF-IDF vectrizer and 94.26% accuracy in case of countvectorizer. So, for both cases the accuracy is almost 94%.



(a) Confusion Matrix for CountVectorizer

(b) Confusion Matrix for TF-IDF

Figure 3.51: Confusion Matrix of SGDC for Lemmatized Text

3.7 Deep Learning Models

After evaluating the traditional model and observing the performances, we have further implemented some very well known deep learning models and evaluated them against our dataset.

3.7.1 LSTM with One Hot Vector[2]

Long Short Term Memory (LSTM) networks are one of the most extensively utilised models in text categorization and generation challenges due to their ability to capture sequential information in an effective manner. [14] That's we have used sequential LSTM model for our dataset. We have taken 600 embedding vector features and **sigmoid** as our acivation function. Keeping the epochs=20 and batch_size = 256 we have evaluated the model for both stemmed and lemmatized text respectively.

3.7.1.1 Using Stemmed text

For stemmed text, the model gave us an accuracy of 93.5%. Moreover it detecs 285 true fake news out of 420 fake news. The confusion metrix is shown below.

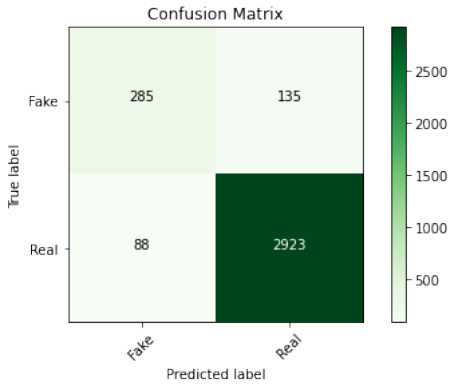


Figure 3.52: Confusion Matrix of LSTM for Stemmed Text

3.7.1.2 Using Lemmatized text

In case of lemmatized text, the model gave us an accuracy of 94.11% accuracy. it detects 126 false negative where the number of false positive rate was 76. The confusion matrix is shown below.

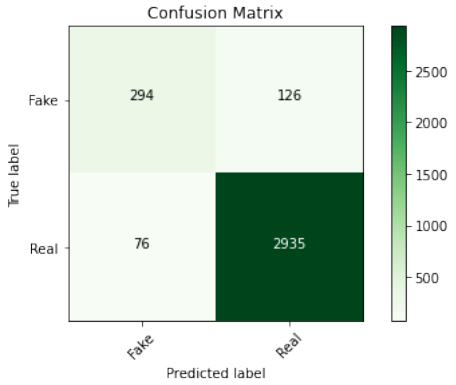


Figure 3.53: Confusion Matrix of LSTM for Lemmatized text

3.7.2 LSTM with word2vec

Now after lexical based features we have tried the LSTM model with pre-trained word2vec model. The model was trained with 22K bengali news dataset.[31] Moreover we chose 100 for our embedded vector dimension and also have used activated function like **Relu**, **sigmoid**. We have kept our dataset in 70:30 ratio. The number of epochs were 12. After evaluation, the model gave us an accuracy of 94.5%. So, the accuracy slightly improves than the lexical based features.

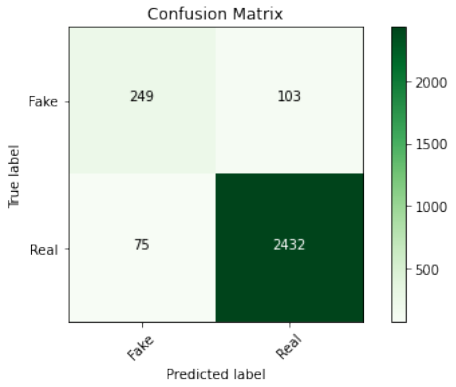


Figure 3.54: Confusion Matrix of LSTM with word2vec

3.7.3 LSTM with Glove Vector

After evaluating with pre-trained bengali word2vec model, we have now used pre-trained bengali golve vector model which is trained over 20M wikipedia data.[32] We have used the same settings and our emdding dim as 100. This time, the model gave us an accuracy of 94.49%. Though the accuracy was almost similar to pre-trained word2vec model but the number of false negative is now 40 which is much better comparing to word2vec model. The confusion matrix is given below.

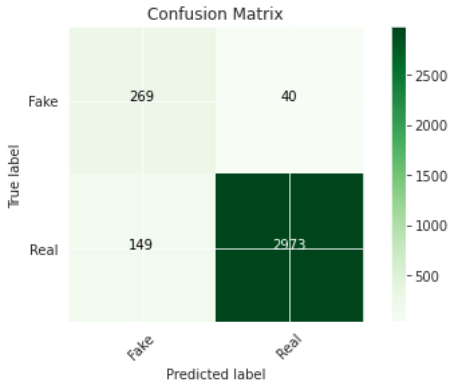


Figure 3.55: Confusion Matrix of LSTM with Glove Vector

3.7.4 Recurrent Neural Network

In this case, we have used Recurrent Neural Network with the builtin functions of tensorflow. We have also used sequential model and relu as our activation function as we have used before. We have also used Bidirectional LSTM and our self implemented tokenizer to tokenize the words.

After 12 epochs, the model gave an accuracy of 96.55%, which is clearly better than our previous approaches. The training and validation accuracy against different number of epochs is shown in below graph.

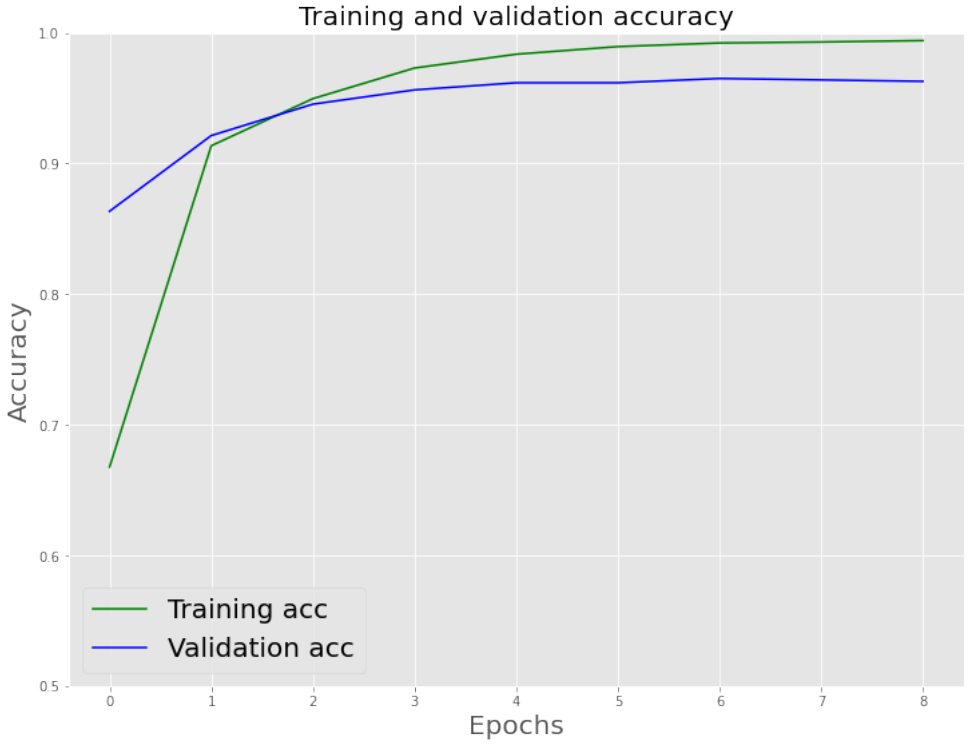


Figure 3.56: Accuracy Against Epochs for RNN

The model also detects 227 true fake news from 249 fake news, which is much better than the previous approches. The confusion matrix is given below.

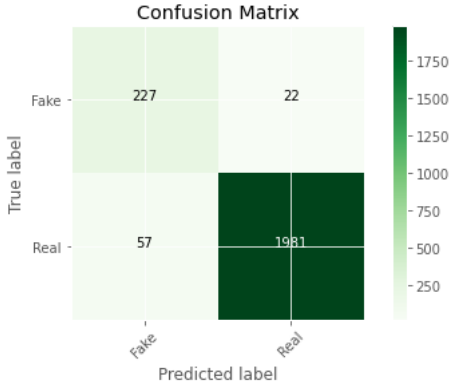


Figure 3.57: Confusion Matrix of RNN

3.8 Pre-Trained Language Models

Pre-trained language models such as OpenAI GPT [33], BERT[16], and ULMFiT[34] have recently made significant progress in a variety of NLP tasks. BERT and its variant models, in particular, have outperformed the GLUE benchmark for Natural Language Understanding (NLU).[35] That's why we have decided to use BERT to observe it's performance over the dataset.

3.8.1 BERT

We have used the multilingual cased pre-trained BERT model for our evaluation. It is trained for over 107 languages.[36] We have used BERT wrapper over sklearn library as it slightly performs better from the original hubbingface hosted BERT. [37] We kept 70% of our dataset as train dataset and 30% as our test dataset and used epoch=3 to train the BERT model. We have also used our custom made vocab for the bert. Additionaly, we have used learning_rate=2e-05, num_mlp_hiddens=500, random_state=42, max_seq_length=64 as our model tweaking parameter. We have got an accuracy of 93.35% after the 3rd epoch. May be accuracy will slightly improve for more epochs as the model was performing slightly better in terms of number of epochs.The confusion matrix is shown below.

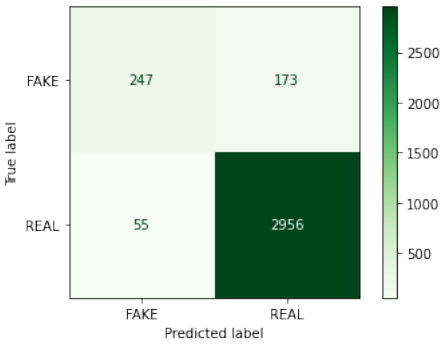


Figure 3.58: Confusion Matrix of BERT

Chapter 4

Results and Discussions

In this section, we'll show and discuss the overall perfomances of the models and also the drawbacks of some methods.

4.1 Overall Performances of the traditional Models

After evaluating all the traditional models with every possible case, we have found that Support Vector Machine and Passive Agressive Classifier perform best for frequency based word embedding methods. For word embedding, the Passive Agressive classifier gave an accuracy of 99.33% and Support Vector Machines gave an accuracy of 98.45% accuracy. But, however, they still couldn't detect a good rate of true negatives compared to true positives. After evaluating them against the character-based embedding method, we can see that the Support Vector Machine and Passive Agressive classifier both outperform all models by detecting more than 400 true negatives out of 440 total fake news. That means we can conclude that both models work more well for character-based embedding rather than word-based embedding. The third model which gave us a better performance is Multinomial Naive Bayes. We can see that after tweaking it, it gave us an accuracy of 98%. But that doesn't end. If gave an even better accuracy in the case of countvectorizer, which is 98.45%. Below is an table which shows the best performing models with their performances.

	SVM	LR	Passive Agressive	MNB	Random Forests	Neural Nework
CV	97.32%	98.12%	99%	98.45%	98.65%	99%
TF-IDF	98.45%	97%	99.33%	98%	98.92%	98%

Table 4.1: Traditional Models with their Best Performance

4.2 Overall Performances of the deep learning Models

As only lexical features can't derive true context of the news, so we have used word2vec model with deep learning algoriths to get more better classification. After evaluation of deep learning models like LSTM with different word vectors, RNN and pre-trained multilingual BERT, RNN gave us the best accuracy which is 96.5%. Moreover, It detect 227 fake news. The number of false negative was also pretty much low which is 22. Among 2038 true news it only detects 57 wrong true news. The second best performing model was LSTM with pre-trained word2vec embedding model. The model gave an accuracy of 94.5%. The multilingual BERT model gave an accuracy 93.11% after 3rd epoch. Below is shown the perfomances of deep learning models on the dataset.

Model	Accuracy
LSTM+One Hot+Stemmer	93.50%
LSTM+One Hot+Lemmatizer	94.11%
LSTM+word2vec	94.5%
LSTM+Glove	94.49%
RNN	96.55%
BERT	93.35%

Table 4.2: Deep Learning Models with their Performances

4.3 Failed Cases

4.3.1 Pre-Processors

If we look our results closely, we can see that the traditional models can't gave us a good accuracy when we are using word processors like lemmatizer, stemmer and POS tagger. That's

because there is still no standard POS Tagger, Stemmer and Lemmatizer. After using the banglakit lemmatizer we have seen that instead of getting the root form of the word it sometimes breaks the word and can't handle the conjoint letters properly. That's why after processing with this builtin pre-processors the model in showing such poor results.

4.3.2 Deeplearning Models

After the traditional models, we expected to get better results when using semantic word vectors like word2vec and deep learning models like LSTM, RNN, BERT. But after evaluation, the models couldn't show expected results. That may be because the dataset is imbalanced and doesn't contain a sufficient amount of fake news. We know that deep learning and neural network models need a huge amount of data to train properly, but after we have added some propaganda based fake news, the dataset still lacks a sufficient amount of fake news.

Chapter 5

Conclusion

This paper discusses an extensive research on different apporaches and models to detect fake news in bengali language. After a lot of evaluation and using different methods, we conclude that there is a limitation to detect fake news from only textual analyses. It also conludes that there's a scope to improve the dataset to evaluate the deep learning models to find the real context. It also concludes that there's still missing standard POS Tagger, Stemmer and Lemmatizer for bengali languages. We also have found that only semantic word2vec models can't perform well in detecting fake news. May be extra features are needed. There's still missing a standard pre-trained word2vec model for bengali language.

References

[1] W. Cavnar and J. Trenkle, "N-gram-based text categorization," *Ann Arbor MI*, vol. 48113, no. 2, pp. 161–175, 1994.

[2] P. Cerda and G. Varoquaux, "Encoding high-cardinality string categorical variables," 2019, cite arxiv:1907.01860. [Online]. Available: http://arxiv.org/abs/1907.01860

[3] rappler. What is fake news. [Online]. Available: https://r3.rappler.com/newsbreak/investigative/185560-mocha-uson-posts-news

[4] word2vec. Word2vec. [Online]. Available: https://www.researchgate.net/publication/334209824_Unsupervised_word_embeddings_capture_latent_knowledge_from_materials_science_literature

[5] S. vector. Support vector machine. [Online]. Available: https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm

[6] classifier. Random forest classifier. [Online]. Available: https://www.javatpoint.com/machine-learning-random-forest-algorithm

[7] javapoint. Logistic regression. [Online]. Available: https://www.javatpoint.com/logistic-regression-in-machine-learning

[8] ——. Decission tree classifier. [Online]. Available: https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm

[9] ——. K-nearest neighbour. [Online]. Available: https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning

[10] another. Logistic regression. [Online]. Available: https://michael-fuchs-python.netlify.app/2019/11/11/introduction-to-sgd-classifier/

[11] javapoint. Neural network. [Online]. Available: https://www.analyticsvidhya.com/blog/2020/12/mlp-multilayer-perceptron-simple-overview/

[12] RNN. Rnn. [Online]. Available: https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

[13] CNN. Cnn. [Online]. Available: https://www.researchgate.net/figure/The-construction-of-the-CNN-model_fig2_340909278

[14] LSTM. Lstm. [Online]. Available: https://www.researchgate.net/figure/The-structure-of-the-Long-Short-Term-Memory-LSTM-neural-network-Reproduced-from-Yan_fig8_334268507

[15] BiLSTM. Bilstm. [Online]. Available: https://paperswithcode.com/method/bilstm

[16] BERT. Bert. [Online]. Available: https://www.researchgate.net/figure/BERT-model_fig6_332543716

[17] Science. The science of fake news. [Online]. Available: https://science.sciencemag.org/content/359/6380/1094.summarys

[18] Wikipedia. Fake news. [Online]. Available: https://en.wikipedia.org/wiki/Fake_news

[19] V. L. R. Nadia K. Conroy and Y. Chen, "Automatic deception detection: Methods for finding fake news," *Proceedings of the Association for Information Science and Technologys*, vol. 52, pp. 1–4, jan 2015. [Online]. Available: https://doi.org/10.1002/pra2.2015.1450520100827

[20] Y. C. Victoria L. Rubin and N. K. Conroy, "Deception detection for news: Three types of fakes," *Proceedings of the Association for Information Science and Technologys*, vol. 52, pp. 1–4, nov 2015. [Online]. Available: https://doi.org/10.1002/pra2.2015.145052010083

[21] C. Buntain and J. Golbeck, "Automatically identifying fake news in popular twitter threads," in *2017 IEEE International Conference on Smart Cloud (SmartCloud)*, 2017, pp. 208–215.

[22] Z. Jin, J. Cao, Y. Zhang, and J. Luo, “News verification by exploiting conflicting social viewpoints in microblogs,” in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, ser. AAAI’16. AAAI Press, 2016, p. 2972â2978.

[23] 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, pp. 900–903.

[24] I. Y. R. Pratiwi, R. A. Asmara, and F. Rahutomo, “Study of hoax news detection using naÃ¯ve bayes classifier in indonesian language,” in *2017 11th International Conference on Information Communication Technology and System (ICTS)*, 2017, pp. 73–78.

[25] N. Ruchansky, S. Seo, and Y. Liu, “Csi,” *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, Nov 2017. [Online]. Available: http://dx.doi.org/10.1145/3132847.3132877

[26] E. Tacchini, G. Ballarin, M. L. D. Vedova, S. Moret, and L. de Alfaro, “Some like it hoax: Automated fake news detection in social networks,” 2017.

[27] M. Z. Hossain. Banfakenews: A dataset for detecting fake news in bangla. [Online]. Available: https://www.kaggle.com/cryptexcode/banfakenews

[28] X. Zhu, A. B. Goldberg, M. Rabbat, and R. Nowak, “Learning bigrams from unigrams,” in *Proceedings of ACL-08: HLT*. Columbus, Ohio: Association for Computational Linguistics, Jun. 2008, pp. 656–664. [Online]. Available: https://aclanthology.org/P08-1075

[29] BanglaKit. Banglakit lemmatizer. [Online]. Available: https://github.com/banglakit/lemmatizer

[30] A. Hossain. Bnltk: Bangla natural language toolkit. [Online]. Available: https://github.com/ashwoolford/bnltk

[31] A. Ahmad and M. Amin, “Bengali word embeddings and it’s application in solving document classification problem,” *2016 19th International Conference on Computer and Information Technology (ICCIT)*, pp. 425–430, 2016.

[32] sagorbrur. Bengali glove vector. [Online]. Available: https://github.com/sagorbrur/GloVe-Bengali

[33] V. Cohen and A. Gokaslan, "Opengpt-2: Open language models and implications of generated text," *XRDS*, vol. 27, no. 1, p. 26â30, Sep. 2020. [Online]. Available: https://doi.org/10.1145/3416063

[34] J. Howard and S. Ruder, "Fine-tuned language models for text classification," *CoRR*, vol. abs/1801.06146, 2018. [Online]. Available: http://arxiv.org/abs/1801.06146

[35] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. Bowman, "GLUE: A multi-task benchmark and analysis platform for natural language understanding," in *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Brussels, Belgium: Association for Computational Linguistics, Nov. 2018, pp. 353–355. [Online]. Available: https://aclanthology.org/W18-5446

[36] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018. [Online]. Available: http://arxiv.org/abs/1810.04805

[37] charles9n. Bert sklearn. [Online]. Available: https://github.com/charles9n/bert-sklearn