News Aggregation & Fraud News Detection Using Deep Learning Techniques

Submitted in partial fulfilment of the requirements for the SET

Bachelors of Technology

in

Electronics And Communication Engineering

by

Nithin Thota 17BEC0051
Asutosh Mohapatra 17BEC0141

Under the guidance of

Prof. Prakasam P (Project guide)

SENSE,
VIT,
Vellore.

May,2021

DECLARATION

I hereby declare that the thesis entitled "News Aggregation & Fraud News Detection Using Deep Learning Techniques" submitted by me, for the award of SET to VIT is a record of bonafide work carried out by us under the supervision of Prof. Prakasam P.

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place : Vellore

Date :

Signature of the Candidates

Nithin Thota – 17BEC0051

Asutosh Mohapatra – 17BEC0141

CERTIFICATE

This is to certify that the thesis entitled "News Aggregation & Fraud News Detection Using Deep Learning Techniques" submitted by Nithin Thota (17BEC0051), Asutosh Mohapatra (17BEC0141), School of Electronics And Communication Engineering (SENSE), VIT University, for the award SET is a record of bonafide work carried out by him under my supervision during the period, 01. 01. 2021 to 05.05.2021, as per the VIT code of academic and research ethics.

The contents of this report have not been submitted and will not be submitted eitherin part or in full, for the award of any other degree or diploma in this institute or any other institute or university. The thesis fulfills the requirements and regulations of the University and in my opinion meets the necessary standards for submission.

Place : Vellore

Date :25/11/2020

Prof. Prakasam P.
Signature of the Guide

Internal Examiner **External Examiner**

Prof. Prakasam P.
Head of the Department
Department of Electronics And Communication Engineering, SENSE

ACKNOWLEDGEMENTS

While we have taken effort for the completion of the project, it would have not been possible without the kind supporting help of many individuals and organizations. We would like to extend our sincere thanks to all of them.

We are highly indebted to VIT faculty for their guidance and constant supervision as well as for providing necessary information regarding the project and also for their support in completing the project.

In particular, we would like to express our sincere and profound gratitude towards Prof. Prakasam P, for his guidance and constant supervision. Also we would like to thank him for his cooperation and encouragement. In addition to that we would like to thank the VIT Administration, who allowed us to research this topic and gave us the opportunity to pursue this project.

Finally, we are grateful to our parents and family without whose support we would not have been able to complete this project.

Signature of the Candidate

-Nithin Thota

(17BEC0051)

-Asutosh Mohapatra

(17BEC0141)

Executive Summary

Living in 21st century, from shopping to reading news articles everything has changed, everything has become online. Anyone can access most of everything by a single touch from the cell phone. Internet is the new normal, everyone is very much attached to it. Reading news online is something very common around people of all age groups, thousands of articles are being published on various online media portals online every hour. These articles are not necessarily genuine always, sometimes false information is written knowingly and sometimes knowingly. It is very much needed to keep these articles away from the users. Many researches have been conducted using traditional mathematical models and sequential neural networks to detect these fraud news online.

Any person who prefers to reads articles online has a certain genre of choice, but the vast amount of articles published on daily basis makes it impossible for any one to go through all articles to find the articles of preference. Even news datasets show that different organisations give different tags to similar articles, it solely depends on the writers and the publication.

In this project we are training two models to detect fake news and to aggregate news by genres with the help of two Kaggle datasets. We are computing accuracy, precision and recall of the two Deep Learning models and will compare it with existing models.

Supervised learning is the technique of accomplishing a task by providing training, input and output patterns to the systems whereas unsupervised learning is a self-learning technique in which system has to discover the features of the input population by its own and no prior set of categories are used.

The algorithms in Deep Learning:-

- Bidirectional Recurrent Neural Networks
- Dense Neural Networks
- Self- Attention

Preprocessing Concepts-:

- Lematizaton
- Tokenization
- Word Embeddings

We are using nearly 20,000 news articles for fraud news detection of 10,000 fake news and Real news articles each. For news aggregation we are using nearly 49,000 news articles and each classes are having 6,150 news articles.

Table Of Contents

Serial Number	**Content**	**Page Number**
	Acknowledgment	iv
	Executive Summery	v
	Table Of Contents	vi
	List of Figures	vii
	List of Abbreviations	viii
1	Introduction	1
1.1	Objective	1
1.2	Motivation	1
1.3	Background	1
1.4	Literature Surve	2
2	Project Description And Goals	3
3	Technical Specifications	4
4	Methodology	7
5	Results	19
6	Conclusion And Future Scope	22
7	References	23
	Appendix Code A	24
	Appendix Code B	31

List of Figures

Figure No.	Title	Page Number
1	Fake News Dataset	4
2	Word Cloud of Fake News Dataset	5
3	News Aggregation	5
4	Word Cloud Of News Aggregation	6
5	Lemmatization Working	7
6	Types of Embeddings	8
7	LSTM Architecture	9
8	Bi-LSTM Architecture	10
9	Self-Attention Mechanism	10
10	Dropout Mechanism	11
11	ReLU Activation Graph	12
12	Softmax Activation Formula and Graph	13
13	Types Of Fake News Classification Approaches	14
14	Architecture of both models	15
15	Datasets After Preprocessing 1	16
16	Datasets After Preprocessing 2	16
17	Datasets After Tokenization	16
18	News Aggregation Model	17
19	Fake News Classification Model	18
20	Fake News Classification Result	19
21	Fake News Model Accuracy and Loss Graphs	19
22	Precision, Recall, F1-Score of Fake News Classification Model	19
23	Fake News Model Confusion Matrix	20
24	News Aggregation Result	20
25	News Aggregation Accuracy and Loss Graphs	20
26	Precision, Recall, F1-Score of Fake News Aggregation Model	21
27	News Aggregation Model Confusion Matrix	21

List of Abbreviations

ML	Machine Learning
LSTM	Long Shrot Term Memory
Bi-LSTM	Bidirectional Long Short Term Memory
GRU	Gated Recurrent Units
RNN	Recurrent Neural Network
CNN	Convolutional Neural Networks

1. INTRODUCTION

1. OBJECTIVE

The main objective of our project is to develop an efficient deep learning models for aggregation of news articles based on their genres To develop an efficient deep learning models classify fake news articles. To develop the two hybrid deep learning models for both aggregation and fake news detection. To fuse advanced deep learning models to create a model that can more effectively classify the news articles.

2. MOTIVATION

Most of the researchers were concentrated on fake news classification of social media posts and rather than aggregating the news articles based on their genres. Therefore, it motivated us to develop an efficient deep learning based models to classify the genres of news articles as well as to detect fake news articles.

These days millions of articles published worldwide based on various issues or phenomenons happening in our society. Sometimes the articles are illinformed either knowingly or unknowingly. It happens every day. Thousands of published articles are also withdrawn from web after public outrage because of the false information it's carrying on daily basis. These kind of articles can harm someone else's repute or can start public riots. These are very harmful for our society. No user thinks the content he is seeing is false directly and that creates a misconception of certain thing in his mind. So it is absolutely needed to have a system that purifies the content published on web. If something is found fishy it should straight away remove it. This motivated us to develop a model that can more effectively classify genuine and fraud news articles than existing models by processing the text in it.

Everyone have their genre of preference. People generally more interested in reading articles of certain genre. Because millions of articles being published everyday on social media it is not in someone scope to go through all the articles to filter out those they want to read. So this motivated us to develop a system that can identify genres of articles just by processing the text in it.

3. BACKGROUND

When deep learning models were in the preliminary stage the accuracy obtained by these models compared to complex mathematical models were less. No one was able to describe the relationship between input and output hence it was not preferred for these kind of classification tasks. These days with advancement the accuracy of deep learning models are way better than those models. LSTM models first designed in 1997 and further improvised gives very good results on time series data. Thousands of papers are being published every year on hybrid LSTMs.

4. LITERATURE SURVEY

- **Fake news detection: A hybrid CNN-RNN based deep learning approach [7]-:** As the name of paper indicates authors have used an hybrid recurrent convolutional net model on two different type of datasets to check for fake news. One dataset had tweets and the other had news articles. ISOT dataset consisting of news articles is a large dataset and FAKEs is a small one. Authors have got very good accuracy on ISOT dataset.

- **Transformer based Automatic COVID-19 Fake News Detection System [6] -**: Authors have tried to detect fake news came in social media during covid pandemic time. They use several models out of which 3 transfer models BERT, XLBERT, XLNET and ensemble model gave very good results. The ensemble model got and F1 score of 0.9855.

- **A Deep Learning Approach for Automatic Detection of Fake News [4]-:** Authors have used BI-GRU model with attention and Embedding For Language Model(ELMo) with MLP net. They have used smaller datasets of different genres. ELMo based model gave slightly better accuracy in all the cases.

- **Fake News Detection Using A Deep Neural Network [8]**-: In this paper the authors have tried to detect fake news on social media using LSTM, CNN-LSTM and other models. Out of all the models CNN-LSTM model had best F1 score.

- **Fake News Detection Using Deep Learning Techniques [1]** -: In this paper the authors have compared different machine learning techniques like random forest, SVM, Deep Neural Network and measured how accurately the models are working and how much time those are taking and how much memory those are using. DNN has performed way better among these rest models.

- **Fake News Detection Using Sentiment Analysis [3] -:** In this paper the authors have tried to detect fake tweets as well as news articles. They have used different concepts of bi-grams, tri-grams, tf-idf vectorizer with cosine similarity for text preprocessing. The result obtained with tf-idf vectorizer with cosine similarity is very impressive.

2. PROJECT DESCRIPTION AND GOALS

1. Description

This project aims to detect fraud news articles and aggregate news articles based on the genres by reading the heading and main text of the articles. We will be using advanced deep learning models to create a very efficient system that can seamlessly detect fake news and aggregate news on the go. We will be using pretrained embedding libraries in which similar word have nearer embedding values hence it will be more of a feature based models.

We have used Bi-LSTM nets to process time series data effectively. The self-attention mechanism is also used that enhances the word learning of model. We have stacked minimal layers of cells and tried to modify the hyperparameters and added certain regularization methods to enhance the performance. By adding less cells we have decreased the total parameters required to be computed for each cycle and hence it consumes less memory and processes the data faster. We have also used different preprocessing mechanism that indeed helps the model in learning better.

2. Goals

In this project our aim is to develop two models for fraud news detection and news aggregation. The models should have high accuracy and less loss values. It should require less computational power and less memory and at the same time these should give outcome fast. These models should be easily modifiable and employable in a system that can use these as core. We want to fuse existing models and increase their efficiency to the highest level.

3. TECHNICAL SPECIFICATIONS

3.1 Technical Specifications

- Lemmatization
- Embedding
- Word2Vec
- Glove
- Self-Attention
- Artificial Neural network
- Bi-directional Long Short Term Memory (BI-LSTM) model

3.2 Software Used

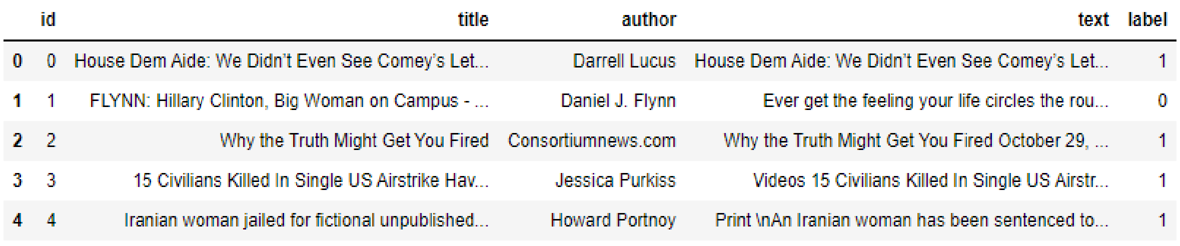
OS	:	Windows
Python IDE	:	python 3.6.x and above
	:	Anaconda IDE

setup tools and pip to be installed for 3.6.x and above

3.3 Datasets Description

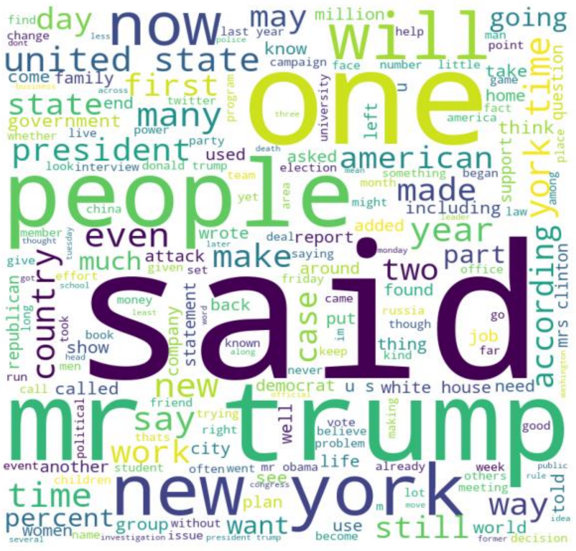
We have used two different Kaggle datasets. One for fraud news detection part and other one is for news aggregation part.

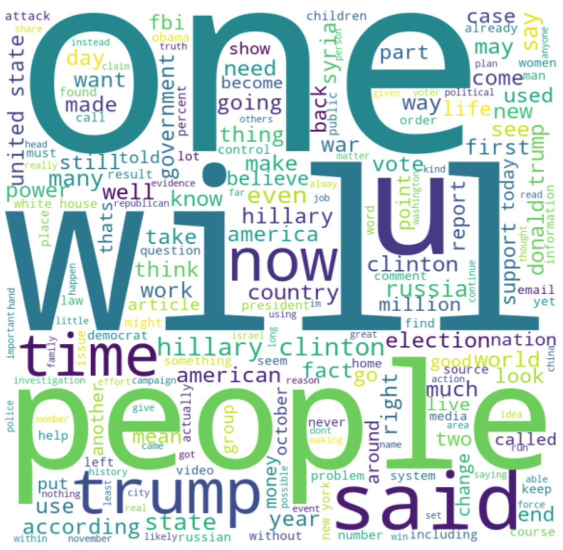
The fake news dataset has 2 different labels either 0 or 1 for each news article where 0 equals to true and 1 equivalent to fraud news articles. There were more than 10,000 articles for each class. This dataset had article id, title, authors name, text and labels for all news articles. Some articles were very sort of length and some are very large.



	id	title	author	text	label
0	0	House Dem Aide: We Didn’t Even See Comey’s Let...	Darrell Lucus	House Dem Aide: We Didn’t Even See Comey’s Let...	1
1	1	FLYNN: Hillary Clinton, Big Woman on Campus - ...	Daniel J. Flynn	Ever get the feeling your life circles the rou...	0
2	2	Why the Truth Might Get You Fired	Consortiumnews.com	Why the Truth Might Get You Fired October 29, ...	1
3	3	15 Civilians Killed In Single US Airstrike Hav...	Jessica Purkiss	Videos 15 Civilians Killed In Single US Airstr...	1
4	4	Iranian woman jailed for fictional unpublished...	Howard Portnoy	Print \nAn Iranian woman has been sentenced to...	1

Fig 1: Fake News Dataset



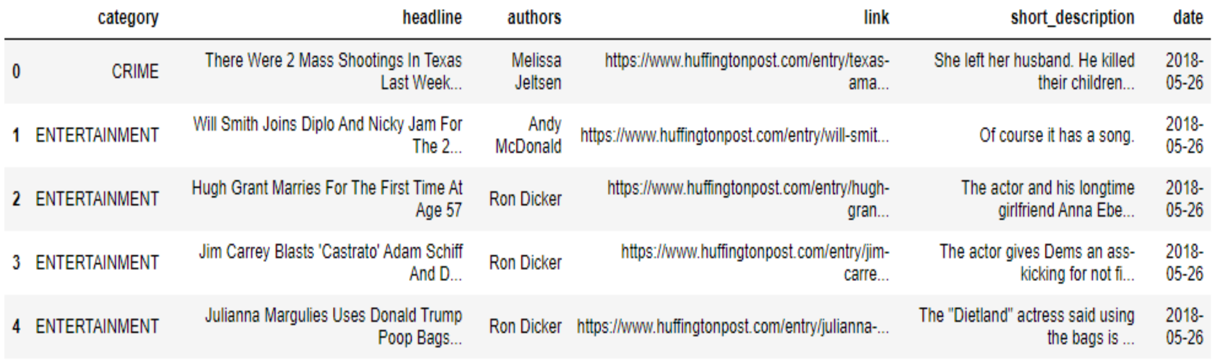


a) Genuine News Word Cloud b) Fake News Word Cloud

Fig 2: Word Cloud Of Fake News Dataset

The figures above show two word clouds of frequently used words in genuine news and in fake news articles. Words like “Trump”,”People”, “Said”, “New York”,”United States” etc appeared in genuine news articles and “One”, “People”, “Trump”, “Will” , “Said” etc appeared in fake news articles most times. Hence we can see that similar words are appearing in both cases. It won’t be possible to classify dataset based on frequent words hence it makes classification task complex.

The news aggregation dataset has a total of 41 different genres. There were a total of 2,02,372 articles clubbed in this dataset. The dataset had article id, category, headline, authors name, text, article link and date of publishment for each article. For each genres corresponding number of articles were not evenly distributes. For some genres total articles were in the range of 1,000 and for some it was in the range of 15,000. The articles were published in the period of 2016-2019. Some genres had similar article content but different category names. The Word Cloud of most frequently used words across articles of all genres given below. Words like “Photo”, “One”, “New”, “Trump”, “Say”, “Will”, “You” etc are appearing the most in comparison to other words. It is very clear that similar words are appearing across all the articles hence the classification task is very complex for this dataset.



	category	headline	authors	link	short_description	date
0	CRIME	There Were 2 Mass Shootings In Texas Last Week...	Melissa Jeltsen	https://www.huffingtonpost.com/entry/texas-ama...	She left her husband. He killed their children...	2018-05-26
1	ENTERTAINMENT	Will Smith Joins Diplo And Nicky Jam For The 2...	Andy McDonald	https://www.huffingtonpost.com/entry/will-smit...	Of course it has a song.	2018-05-26
2	ENTERTAINMENT	Hugh Grant Marries For The First Time At Age 57	Ron Dicker	https://www.huffingtonpost.com/entry/hugh-gran...	The actor and his longtime girlfriend Anna Ebe...	2018-05-26
3	ENTERTAINMENT	Jim Carrey Blasts 'Castrato' Adam Schiff And D...	Ron Dicker	https://www.huffingtonpost.com/entry/jim-carre...	The actor gives Dems an ass-kicking for not fi...	2018-05-26
4	ENTERTAINMENT	Julianna Margulies Uses Donald Trump Poop Bags...	Ron Dicker	https://www.huffingtonpost.com/entry/julianna-...	The "Dietland" actress said using the bags is ...	2018-05-26

Fig 3: News Aggregation Dataset

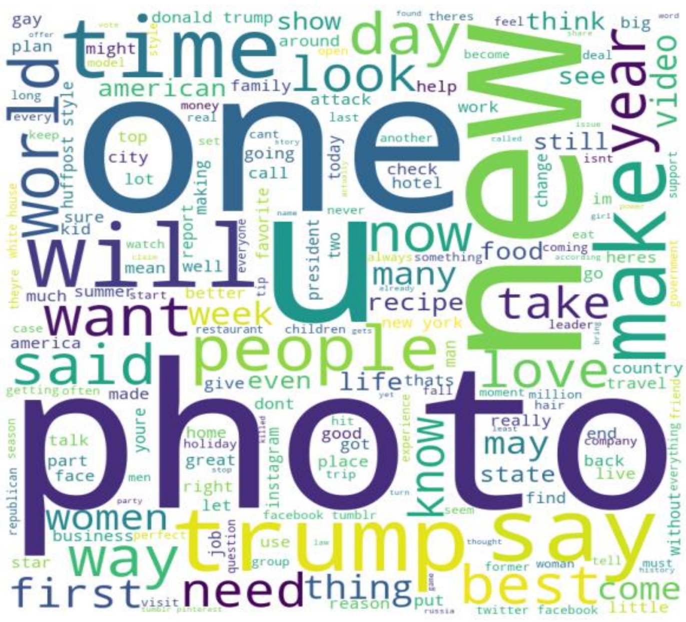


Fig 4: Word cloud of News Aggregation Dataset

4. METHODOLOGY

In this part, we will be giving an overview of lemmatizer , Porter stemmer ,bi-directional lstm model, embedding, word2vec embedding, glove embeddings, self-attention, full connected layer, dropout and batch normalization, relu and softmax activation, Loss functions and L1-L2 regularization . We will be discussing our proposed model and available models in this section.

4.1 LEMMATIZER

Lemmatization is a procedure that helps us to club together the inflected types of words such that it can be analyzed as a single thing. Lemmatizer is an additional pipe like component that can be adjoined to your pipeline, and not a hidden item of the vocabulary that works behind the scenes. This helps us to easy the customization procees such as how lemmas should be assigned in your pipeline. If the lemmatization mode is given as "rule", which requires POS (Token.pos) to be assigned, make sure a Tagger, Morphologizer or another such component assigning POS is available in the pipeline and works before the lemmatizer.

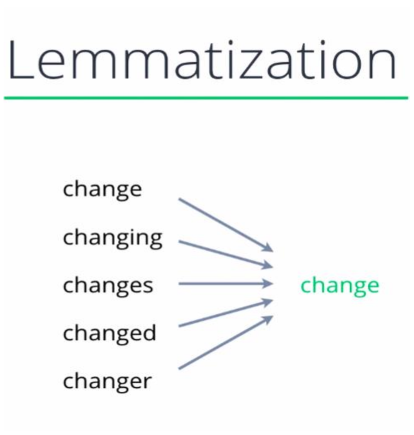


Fig 5: Lemmatization Working

4.2 EMBEDDING

Embedding is a relatively low-dimensional space into which you can translate high-dimensional vectors. Embeddings make it easier to do machine learning on large inputs like sparse vectors representing words**.** Embedding refers to the integration of links, images, videos, gifs and other content into social media posts or other web media. Embedded content appears as part of a post and supplies a visual element that encourages increased click through and engagement.

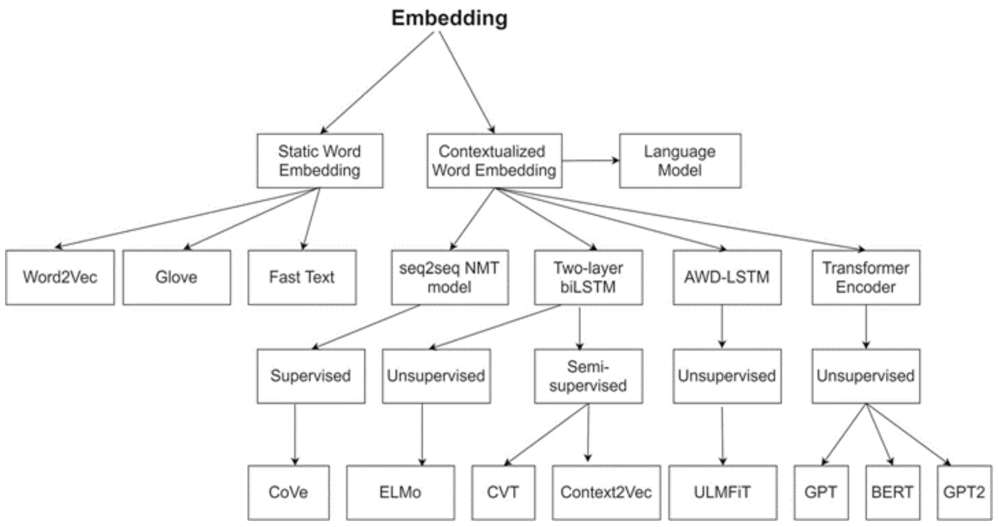


Fig 6: Types of Embedding

4.3 WORD2VEC EMBEDDING

Word embeddings are popularly utilised in both machine learning as well as deep learning models. These models perform well in cases such as reduced training time and improved overall classification performance of the model. Pre-trained representations can also either be static or contextual Contextual models generate a representation of each word that is based on the other words in the sentence. This tool provides an efficient implementation of the continuous bag-of-words and skip-gram architectures for computing vector representations of words. These representations can be subsequently used in many natural language processing applications and for further research.The word2vec tool takes a text corpus as input and produces the word vectors as output. It first constructs a vocabulary from the training text data and then learns vector representation of words.A simple way to investigate the learned representations is to find the closest words for a user-specified word. The distance tool serves that purpose. For example names of politicians will have smaller distance between them. Similarly words of similar domains will have smaller distance between them.

4.4 PORTER STEMMER

The Porter stemmer is a process for removing the commoner morphological and inflexional endings from words in English. Its main use is as part of a term normalisation process that is usually done when setting up Information Retrieval systems. Stemming is the process of reducing a word to its word stem that affixes to suffixes and prefixes or to the roots of words known as a lemma. For example: words such as “Likes”, ”liked”, ”likely” and ”liking” will be reduced to “like” after stemming.

4.5 GLOVE EMBEDINGS

The GloVe is a weighted least square model that train the model using co-occurrence counts of the words in the input vectors. It effectively leverages the benefits of the statistical information by training on the non-zero elements in a word-to-word co-occurrence matrix. The

GloVe is an unsupervised training model that is useful to find the co-relation between two words with their distance in a vector space. These generated vectors are known as word embedding vectors. We have used word embedding as semantic features in addition to n-grams because they represent the semantic distances between the words in the context. The smallest package of embedding is 822Mb, called “glove.6B.zip”. GloVe model is trained on a dataset having one billion words with a dictionary of 400 thousand words.

4.6 LSTM

Long Short Term Memory models a.k.a LSTM models are very powerful recurrent neural network models. These are designed to process sequential time series data very effectively. They have various sigmoid and tanh activation gates inside which helps the to choose which word to keep in memory and which to forget very effectively.

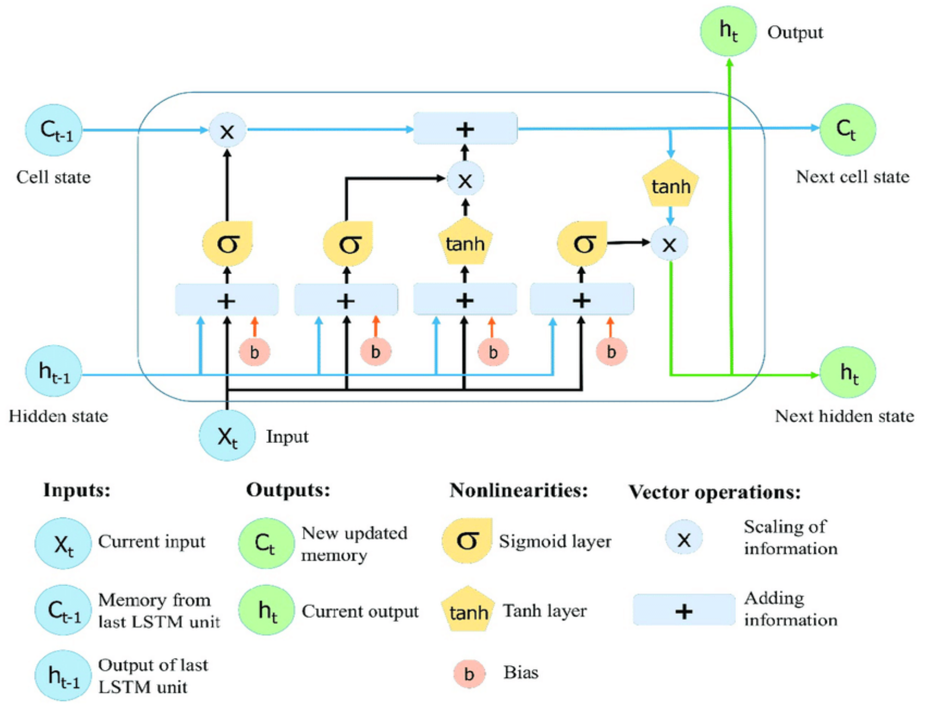


Fig 7: LSTM Architecture

4.7 BI-LSTM

Bidirectional LSTM models comprises of an additional layer of LSTMs. While in one layer of LSTM the input information travels in forward direction in the other layer backward direction.Because the information travels in both directions it more efficiently detects main major features of data.This type of stacking of LSTM layers also effectively reduces the loss and helps in improving the accuracy.

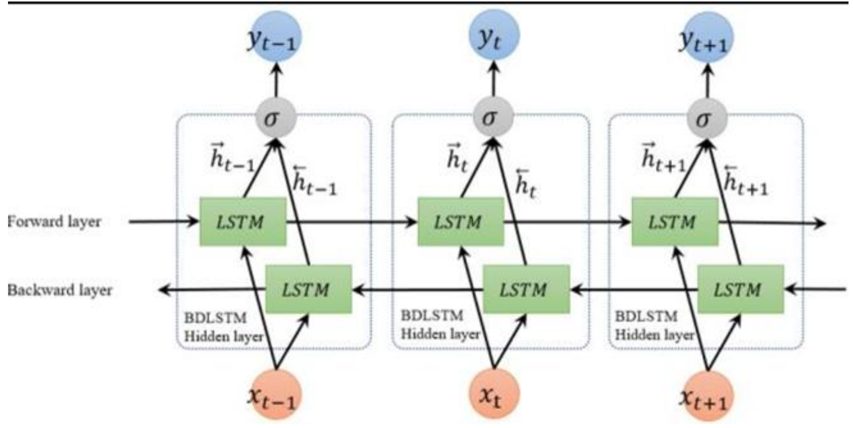
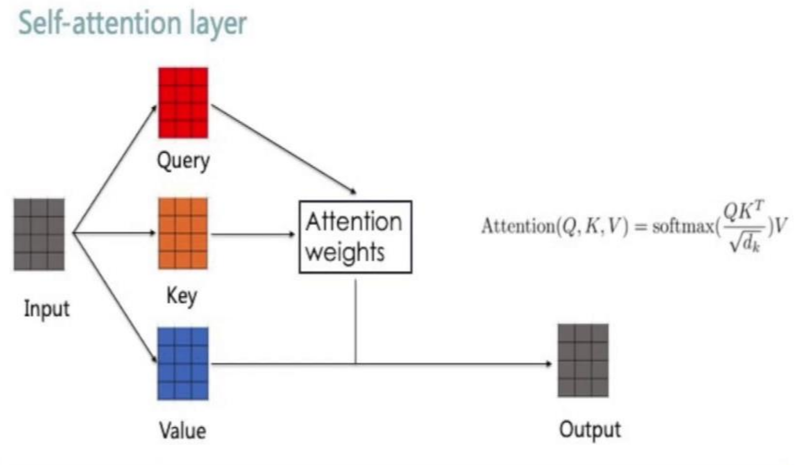


Fig 8: Bi-LSTM Architecture

4.8 SELF ATTENTION

When we process large news datasets, it has very large vocabulary and it is not possible for a normal lstm model to associate a primary word to a secondary word.For example in the sentence "The animal didn't cross the street because it was too tired", 'it' refers to 'animal' and it's very simple for an average human to understand it but it's not so simple for the sequential model. When we are developing a sequential model these minor details can help decreasing the accuracy. Hence we need the help of self-attention here. Self- attention associate 'it' with the 'animal'. As the model processes each word (each position in the input sequence), self attention allows it to look at other positions in the input sequence for clues that can help lead to a better encoding for this word**.**



Self-attention layer
Query
Input
Key
Value
Attention weights
Output
$\text{Attention}(Q,K,V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$

Fig 9: Self Attention Mechanism

4.9 FULL CONNECTED LAYER

Fully Connected layers in a neural networks are those layers where all the inputs from one layer are connected to every activation unit of the next layer. In most popular machine learning models, the last few layers are full connected layers which compiles the data extracted by previous layers to form the final output. Fully connected layers connect every neuron in one layer to every neuron in another layer. It is the same as a traditional multi-layer perceptron neural network (MLP). The flattened matrix goes through a fully connected layer to classify the images. Fully-connected layers have weights connected to all of the outputs of the previous layer.

4.10 DROPOUT

Dropout is a technique used to prevent a model from overfitting. Dropout works by randomly setting the outgoing edges of hidden units (neurons that make up hidden layers) to 0 at each update of the training phase. Dropout is where randomly selected neurons are ignored during training. They are "dropped-out" randomly. This means that their contribution to the activation of downstream neurons is temporally removed on the forward pass and any weight updates are not applied to the neuron on the backward pass. Technically you can add the dropout layer at the ending of a block, for instance either after the convolution or after the RNN ENCODING. With dropout (dropout rate less than 0.25), the accuracy will gradually increase and loss will gradually decrease first. Then, accuracy will suddenly drop to 1/(# of class), and loss will also stay close to a small constant. However, applying dropout to a neural network typically increases the training time. Moreover, the improvement of training speed increases when the number of fully-connected layers increases.

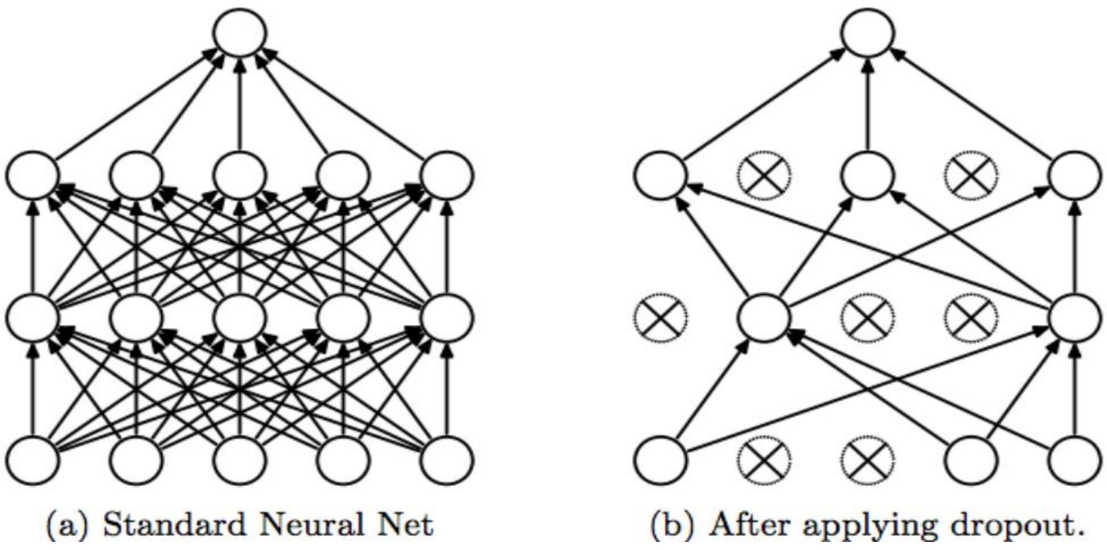


Fig 10: Dropout Mechanism

4.11 BATCH NORMALIZATION

Data normalization gets rid of a number of anomalies that can make analysis of the data more complicated. Some of those anomalies can crop up from deleting data, inserting more information, or updating existing information. It is a technique for training very deep neural networks that standardizes the inputs to a layer for each mini-batch. This has the effect of stabilizing the learning process and dramatically reducing the number of training epochs required to train deep networks. Batch normalisation normalises a layer input by subtracting the mini-batch mean and dividing it by the mini-batch standard deviation.To fix this, batch normalisation adds two trainable parameters, gamma γ and beta β, which can scale and shift the normalised value. In practical coding, we add Batch Normalization after the activation function of the output layer or before the activation function of the input layer. Mostly researchers found good results in implementing Batch Normalization after the activation layer.

Advantages Of Batch Normalization

1. Reduces internal covariant shift.
2. Reduces the dependence of gradients on the scale of the parameters or their initial values.
3. Regularizes the model and reduces the need for dropout, photometric distortions, local response normalization and other regularization techniques.

4.12 ReLU

ReLU stands for rectified linear unit, and is a type of activation function. Mathematically, it is defined as y = max(0, x). ... ReLU is the most commonly used activation function in neural networks, especially in CNNs. If you are unsure what activation function to use in your network. It is the most used activation function in the world right now. Since, it is used in almost all the convolutional neural networks or deep learning. As you can see, the ReLU is half rectified (from bottom). f(z) is zero when z is less than zero and f(z) is equal to z when z is above or equal to zero. The main advantage of using the ReLU function over other activation functions is that it does not activate all the neurons at the same time. ... Due to this reason, during the backpropogation process, the weights and biases for some neurons are not updated. This can create dead neurons which never get activated. The Rectified linear activation function or ReLU for short is a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero. The rectified linear activation function overcomes the vanishing gradient problem, allowing models to learn faster and perform better.

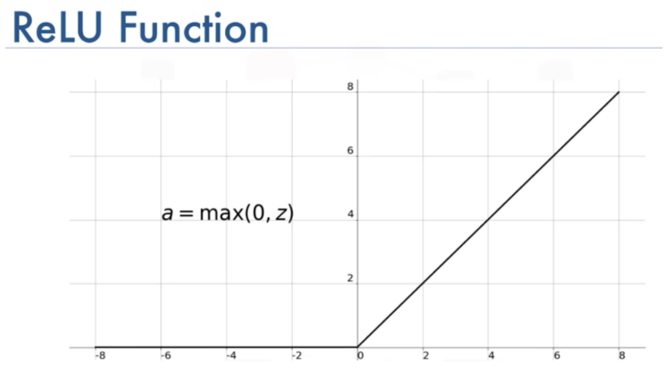


Fig 11: ReLU Activation Graph

4.13 SOFTMAX ACTIVATION

The softmax function, also known as softargmax or normalized exponential function, is a generalization of the logistic function to multiple dimensions. The softmax function is used as the activation function in the output layer of neural network models that predict a multinomial probability distribution. By definition, the softmax activation will output one value for each node in the output layer. It is often used as the last activation function of a neural network to normalize the output of a network to a probability distribution over predicted output classes. Softmax is an activation function that scales numbers/logits into probabilities. The output of a Softmax is a vector (say v) with probabilities of each possible outcome. The probabilities in vector v sums to one for all possible outcomes or classes.

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

σ =	softmax
$\vec{z}$ =	input vector
e^{z_i} =	standard exponential function for input vector
K =	number of classes in the multi-class classifier
e^{z_j} =	standard exponential function for output vector
e^{z_j} =	standard exponential function for output vector

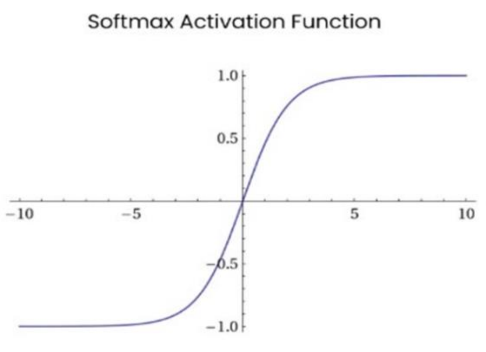


Fig 12: Softmax Activation Formula and Graph

4.14 LOSS FUNCTION

The Loss Function is one of the important components of Neural Networks. Loss is nothing but a prediction error of Neural Net. And the method to calculate the loss is called Loss Function. In simple words, the Loss is used to calculate the gradients. And gradients are used to update the weights of the Neural Net. Machines learn by means of a loss function. It's a method of evaluating how well specific algorithm models the given data. If predictions deviates too much from actual results, loss function would cough up a very large number. At its core, a loss function is a measure of how good your prediction model does in terms of being able to predict the expected outcome(or value). We convert the learning problem into an optimization problem, define a loss function and then optimize the algorithm to minimize the loss function.

4.15 L1-L2 REGULARIZATION

Regularization is a set of techniques that can prevent overfitting in neural networks and thus improve the accuracy of a Deep Learning model. A regression model that uses L1 regularization technique is called Lasso Regression and model which uses L2 is called Ridge Regression. The key difference between these two is the penalty term. Ridge regression adds "squared magnitude" of coefficient as penalty term to the loss function. Neural network regularization is a technique used to reduce the likelihood of model overfitting. There are several forms of regularization. The most common form is called L2 regularization. ... L2 regularization tries to reduce the possibility of overfitting by keeping the values of the weights and biases small. L1 and L2 are the most common types of regularization. Due to the addition of this regularization term, the values of weight matrices decrease because it assumes that a neural network with smaller weight matrices leads to simpler models. Therefore, it will also reduce overfitting to quite an extent. The main intuitive difference between the L1 and L2 regularization is that L1 regularization tries to estimate the median of the data while the L2 regularization tries to estimate the mean of the data to avoid overfitting.That value will also be the median of the data distribution mathematically. Regularization is one of the important prerequisites for improving the reliability, speed, and accuracy of convergence, but it is not a solution to every problem.

4.16 AVAILABLE DEEP LEARNING MODELS FOR FAKE NEWS DETECTION

In the 90's when deep learning concept was new no one is preferring over complex mathematical models and its accuracy was also low. With time and adequate amount of research it has proven to be better in this field. It is evident [1] that a basic deep neural is more powerful than conventional random forest, SVM models in classifying text data. Time series data is effectively processed in recurrent neural networks. In various papers discussed above it is evident that various researches have been conducted to check the efficiency of deep learning based models. To detect fake news from photos, from social media posts and from news articles various studies have been conducted. From all these studies we understand that fusing existing models can give better results. One of the interesting hybrid model is fused CNN-LSTM models. Convolutional Neural Network Models are generally used to filter image data efficiently. But several studies conducted by stacking CNN on top of LSTM or stacking LSTM on top of CNNs gave surprisingly better results.

Except from these models XLnet, Bert and assemble net models were also used to classify fake news. These models also better accuracy. Focus of studies in some papers are mainly based on preprocessing task and how data is fed to the deep learning model. In some studies it is found that instead of sending on word at a time sending n-grams to the model can be more effective. Some studies have focused on how to effectively tokenize the data like how to club similar data together using their cosine similarities. By clubbing similar data together it will help the models to better optimize its hyperparameters that will help the model to learn more efficiently.

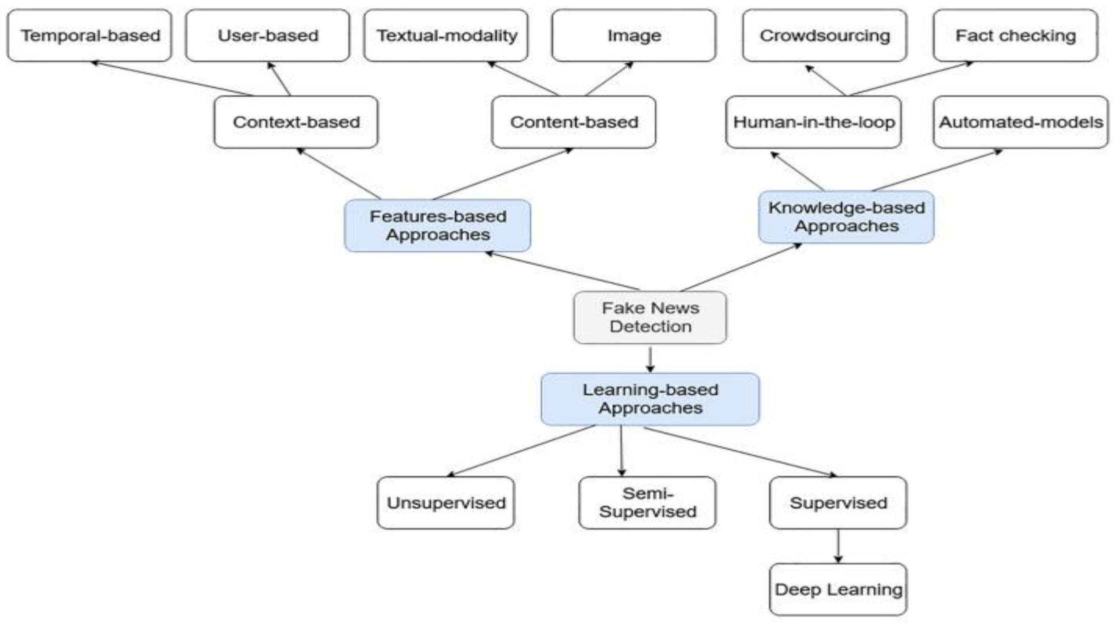


Fig 13: Types Of Fake News Classification Approaches

The figure above tells about various ways to detect fake news. In unsupervised mechanisms the model learns by finding patterns between in input data there is no learning process. Unsupervised models are not quite able to detect fake news because of the verity of data present. There is not much research about semi-supervised learning mechanisms. And supervised learning mechanisms are widely used for detecting fake news and classifying text data mostly.

4.17 PROPOSED HYBRID BI-LSTM MODEL WITH SELF-ATTENTION

We have proposed two hybrid bi-lstm models with self-attention with different embeddings and different preprocessings sections. While majority of models are same there ae some minor differences in both that helps to gain better result in both cases.

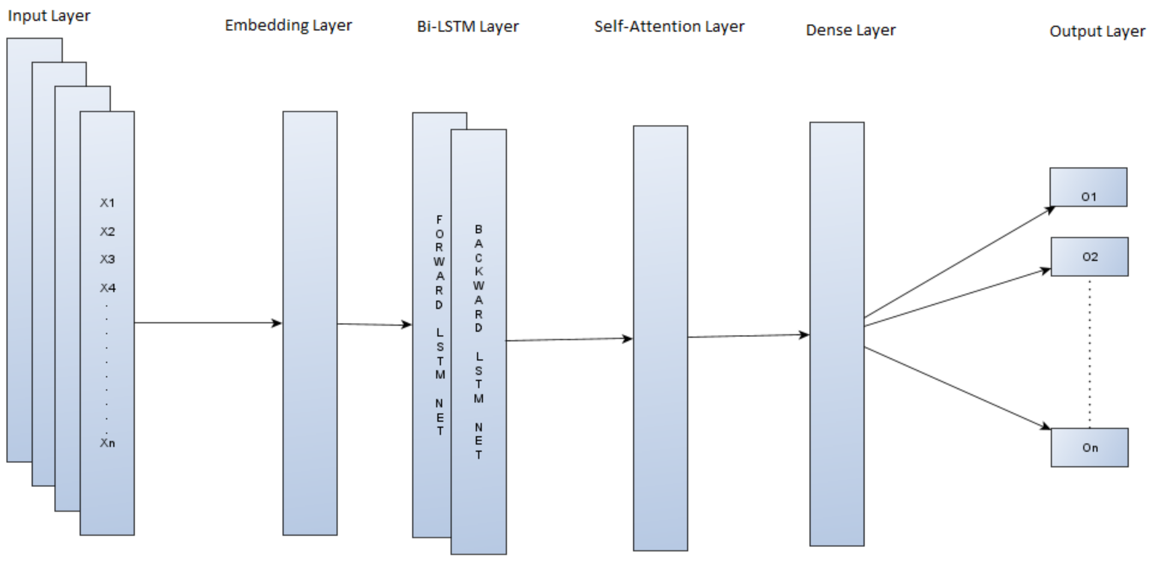


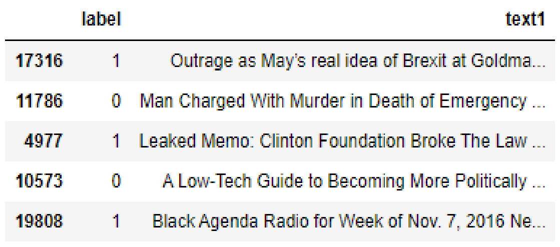
Fig 14: Architecture of both models

The most fundamental idea behind selecting a bi-lstm over a lstm is that in bi-lstms data simultaneously goes through lstm usint in both directions hence the model efficiently correlates words of text data. Using self-attention on top helps the model in learning the relation between different words more effectively. And on top of attention layer there is a fully connected dense layer that again processes the data hence stacking so many layers make the model very complex and because the data is complex it is easier for model to understand hence distinguish between different classes of data.

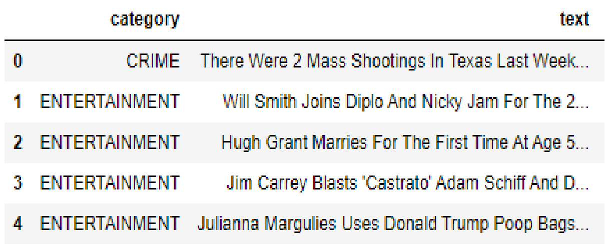
We have used lemmatizer of fake news dataset. Because this dataset has a very large vocabulary it was needed to reduce the vocabulary by making all same words in different forms as one word. Lemmatization was preferred over stemming because stemmers were not accurately detecting words and it was making different words as on word and one word of different form to different words. We haven’t applied lemmatization to news aggregation model because the vocab size for this model was already small so by reducing it further it was difficult for the model to learn.We have used glove embeddings for fake news detection dataset and word2vec embeddings for news aggregation dataset.

Data Preprocessing

The first part of both models was data preprocessing. Initially we have removed unnecessary columns and joined article heading and text part and then removed some articles having length less than 50.



	label	text1
17316	1	Outrage as May's real idea of Brexit at Goldma...
11786	0	Man Charged With Murder in Death of Emergency ...
4977	1	Leaked Memo: Clinton Foundation Broke The Law ...
10573	0	A Low-Tech Guide to Becoming More Politically ...
19808	1	Black Agenda Radio for Week of Nov. 7, 2016 Ne...

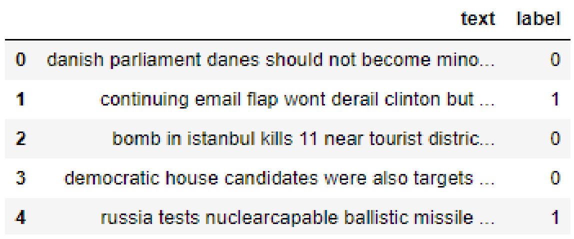


	category	text
0	CRIME	There Were 2 Mass Shootings In Texas Last Week...
1	ENTERTAINMENT	Will Smith Joins Diplo And Nicky Jam For The 2...
2	ENTERTAINMENT	Hugh Grant Marries For The First Time At Age 5...
3	ENTERTAINMENT	Jim Carrey Blasts 'Castrato' Adam Schiff And D...
4	ENTERTAINMENT	Julianna Margulies Uses Donald Trump Poop Bags...

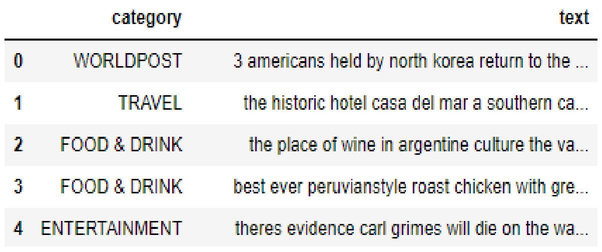
Fake News Preprocessing 1 News Aggregation Preprocessing 1

Fig 15 : Datasets After Preprocessing 1

After this we lower all text and remove any special characters in both models. And we remove artices from classes having excess articles to equalize the number of articles for each class.



	text	label
0	danish parliament danes should not become mino...	0
1	continuing email flap wont derail clinton but ...	1
2	bomb in istanbul kills 11 near tourist distric...	0
3	democratic house candidates were also targets ...	0
4	russia tests nuclearcapable ballistic missile ...	1

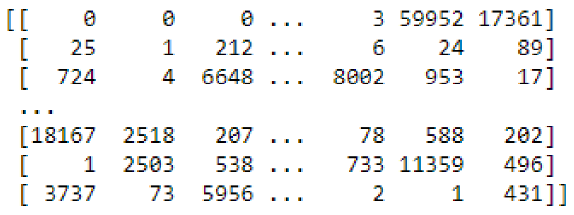


	category	text
0	WORLDPOST	3 americans held by north korea return to the ...
1	TRAVEL	the historic hotel casa del mar a southern ca...
2	FOOD & DRINK	the place of wine in argentine culture the va...
3	FOOD & DRINK	best ever peruvianstyle roast chicken with gre...
4	ENTERTAINMENT	theres evidence carl grimes will die on the wa...

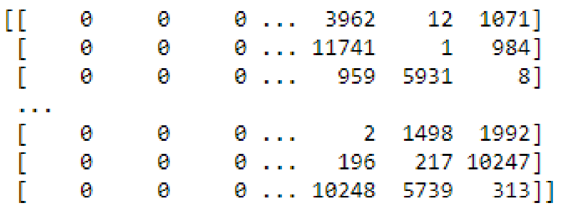
Fake News Preprocessing 2 News Aggregation Preprocessing 2

Fig 16 : Datasets After Preprocessing 2

After this we lemmatize the data in fake news detection model. For both the models we tokenize the text and the pad the data from the left. If any article is having lesser than 250 words in it then it adds extra 0's to the text.



```
[[    0     0     0 ...     3 59952 17361]
 [   25     1   212 ...     6    24    89]
 [  724     4  6648 ...  8002   953    17]
 ...
 [18167  2518   207 ...    78   588   202]
 [    1  2503   538 ...   733 11359   496]
 [ 3737    73  5956 ...     2     1   431]]
```



```
[[    0     0     0 ...  3962    12  1071]
 [    0     0     0 ... 11741     1   984]
 [    0     0     0 ...   959  5931     8]
 ...
 [    0     0     0 ...     2  1498  1992]
 [    0     0     0 ...   196   217 10247]
 [    0     0     0 ... 10248  5739   313]]
```

Fake News Preprocessing 3 News Aggregation Preprocessing 3

Fig 17 : Datasets After Tokenization

The above diagrams are of articles after tokenization. After this we have given id to categories and spllited the dataset into train and validation sets.

4.18Model Development

- In the first part we find the embedding value of available tokens of tokenized text by glove embeddings for fake news prediction model and word2vec embeddings for news aggregation model. Then we create embedding matrix.

- We apply dropout after adding embedding layer, this helps in reducing variance of model.
- The we keep a Bi-LSTM layer with relu activation function and recurrent dropout. We stack batch normalization on top of it. Batch normalization helps in reducing overfitting problem.
- We have added self-attention model on top of it to help the model understand the meaning of words better with batch normalization and dropout
- On top of it we have stacked a dense layer with dropout and batch normalization and relu activation function.
- Finally there is output layer with L1-L2 regularization and softmax activation function.
- After this we compile the model, we use Adams optimizer, catergorical cross entropy for loss and accuracy matrix.

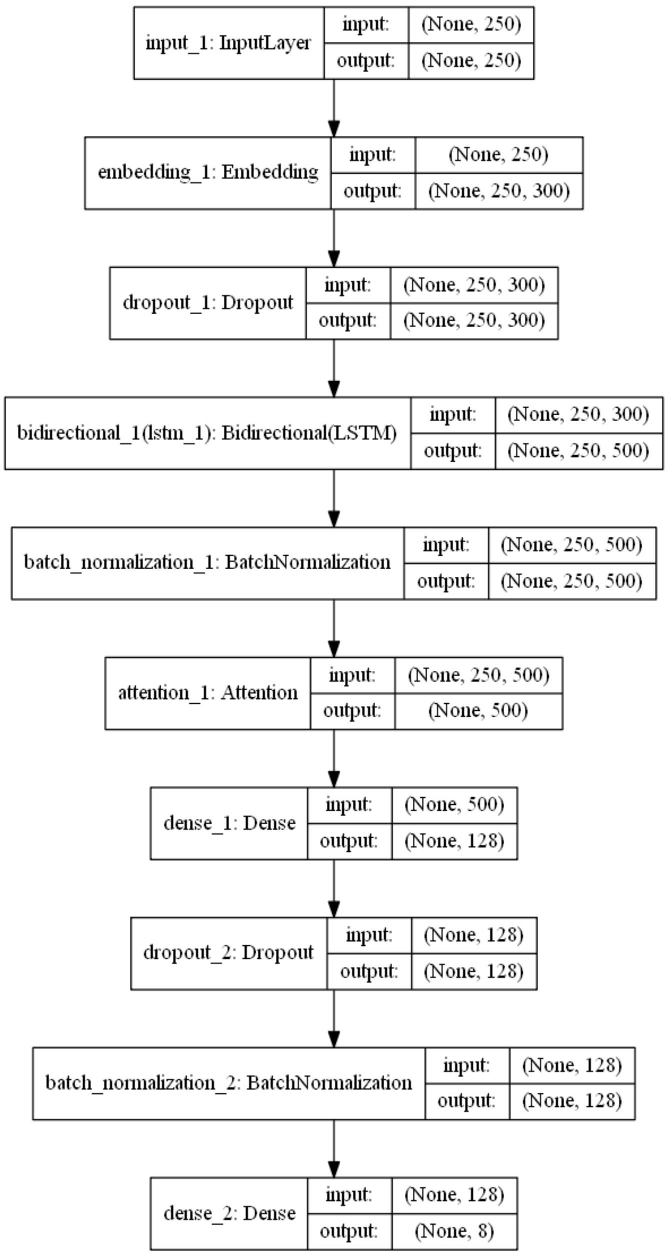


Fig 18: News Aggregation Model

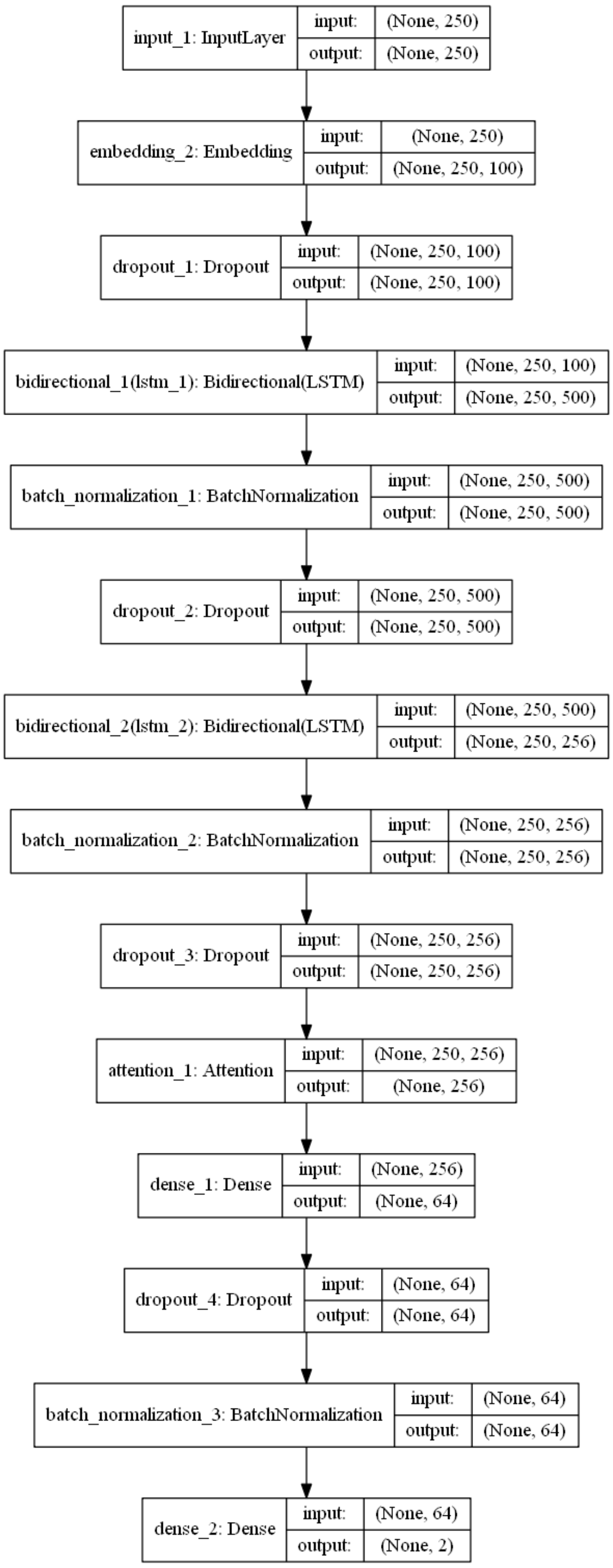
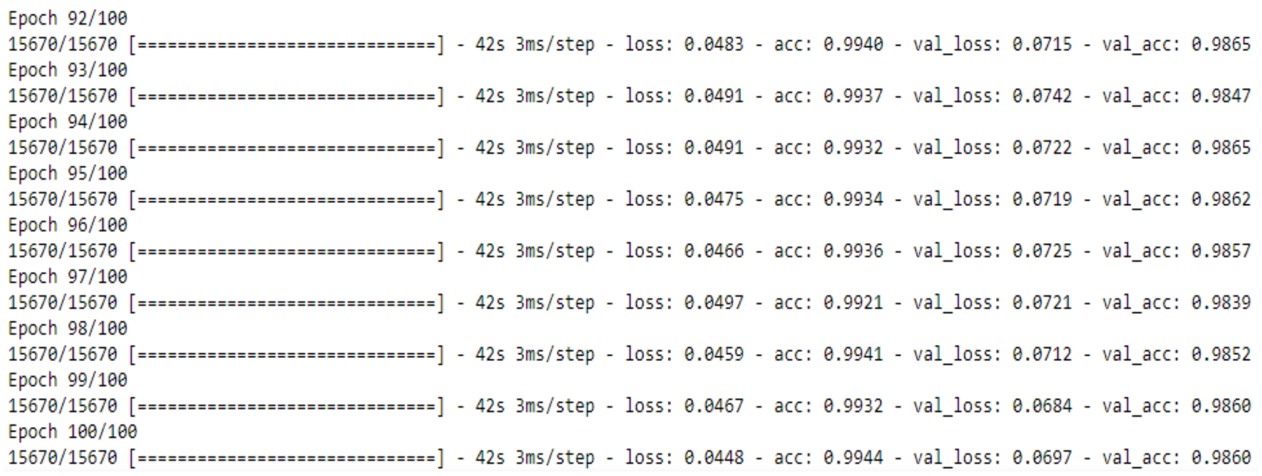

input_1: InputLayer | input: (None, 250) | output: (None, 250)
embedding_2: Embedding | input: (None, 250) | output: (None, 250, 100)
dropout_1: Dropout | input: (None, 250, 100) | output: (None, 250, 100)
bidirectional_1(lstm_1): Bidirectional(LSTM) | input: (None, 250, 100) | output: (None, 250, 500)
batch_normalization_1: BatchNormalization | input: (None, 250, 500) | output: (None, 250, 500)
dropout_2: Dropout | input: (None, 250, 500) | output: (None, 250, 500)
bidirectional_2(lstm_2): Bidirectional(LSTM) | input: (None, 250, 500) | output: (None, 250, 256)
batch_normalization_2: BatchNormalization | input: (None, 250, 256) | output: (None, 250, 256)
dropout_3: Dropout | input: (None, 250, 256) | output: (None, 250, 256)
attention_1: Attention | input: (None, 250, 256) | output: (None, 256)
dense_1: Dense | input: (None, 256) | output: (None, 64)
dropout_4: Dropout | input: (None, 64) | output: (None, 64)
batch_normalization_3: BatchNormalization | input: (None, 64) | output: (None, 64)
dense_2: Dense | input: (None, 64) | output: (None, 2)

Fig 19: Fake News Classification Model

5. RESULT

We have obtained very good results for both models. For the fraud news we have obtained training accuracy of 99.4% and loss of 4.48%. On this model we have receive validation accuracy of 98.6% and validation loss of 6.97%. After performing 100 epochs we got this result. Received precision, accuracy and f1-score of value 99%.

Proof of results-:



```
Epoch 92/100
15670/15670 [==============================] - 42s 3ms/step - loss: 0.0483 - acc: 0.9940 - val_loss: 0.0715 - val_acc: 0.9865
Epoch 93/100
15670/15670 [==============================] - 42s 3ms/step - loss: 0.0491 - acc: 0.9937 - val_loss: 0.0742 - val_acc: 0.9847
Epoch 94/100
15670/15670 [==============================] - 42s 3ms/step - loss: 0.0491 - acc: 0.9932 - val_loss: 0.0722 - val_acc: 0.9865
Epoch 95/100
15670/15670 [==============================] - 42s 3ms/step - loss: 0.0475 - acc: 0.9934 - val_loss: 0.0719 - val_acc: 0.9862
Epoch 96/100
15670/15670 [==============================] - 42s 3ms/step - loss: 0.0466 - acc: 0.9936 - val_loss: 0.0725 - val_acc: 0.9857
Epoch 97/100
15670/15670 [==============================] - 42s 3ms/step - loss: 0.0497 - acc: 0.9921 - val_loss: 0.0721 - val_acc: 0.9839
Epoch 98/100
15670/15670 [==============================] - 42s 3ms/step - loss: 0.0459 - acc: 0.9941 - val_loss: 0.0712 - val_acc: 0.9852
Epoch 99/100
15670/15670 [==============================] - 42s 3ms/step - loss: 0.0467 - acc: 0.9932 - val_loss: 0.0684 - val_acc: 0.9860
Epoch 100/100
15670/15670 [==============================] - 42s 3ms/step - loss: 0.0448 - acc: 0.9944 - val_loss: 0.0697 - val_acc: 0.9860
```

Fig 20 : Fake News Classification Result

Accuracy Graph-:

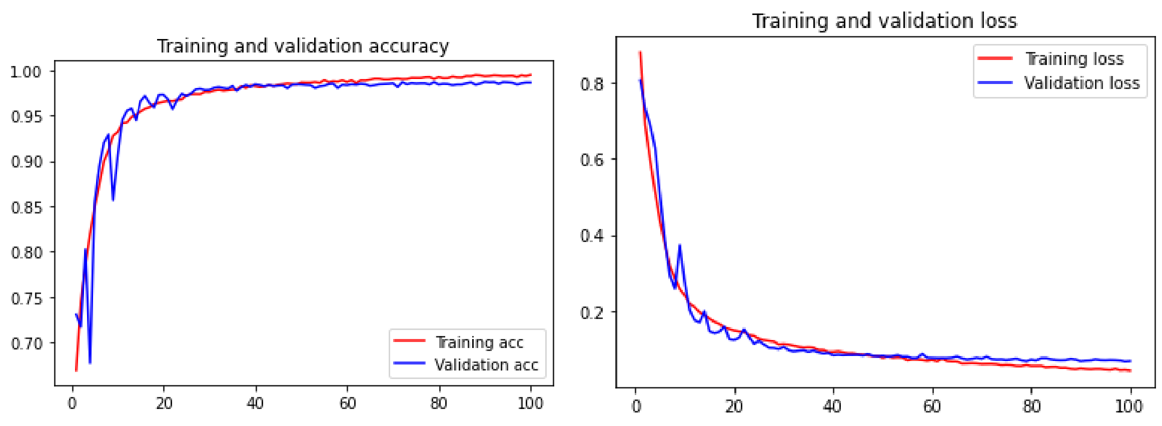
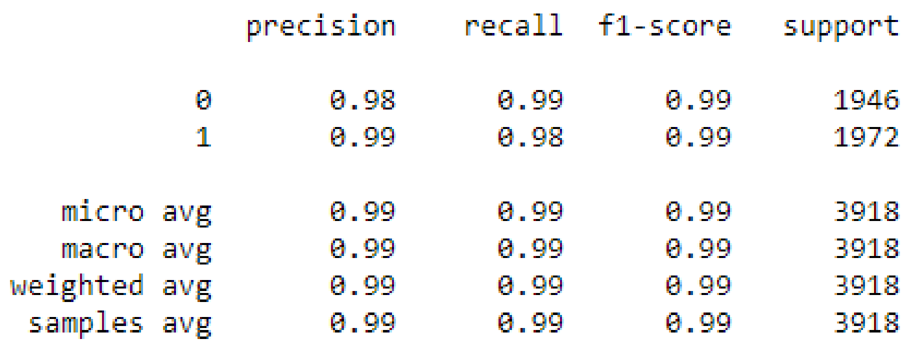


Fig 21 : Fake News Models Accuracy and Loss Graph

Precision, Recall and F1-Score -:



```
              precision    recall  f1-score   support

           0       0.98      0.99      0.99      1946
           1       0.99      0.98      0.99      1972

   micro avg       0.99      0.99      0.99      3918
   macro avg       0.99      0.99      0.99      3918
weighted avg       0.99      0.99      0.99      3918
 samples avg       0.99      0.99      0.99      3918
```

Fig 22: Precision, Recall, F1-Score of Fake News Classification Model

Confusion Matrix-:

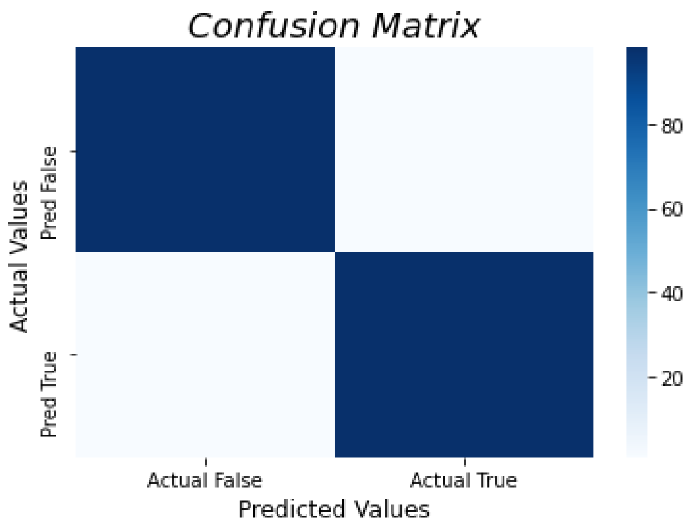
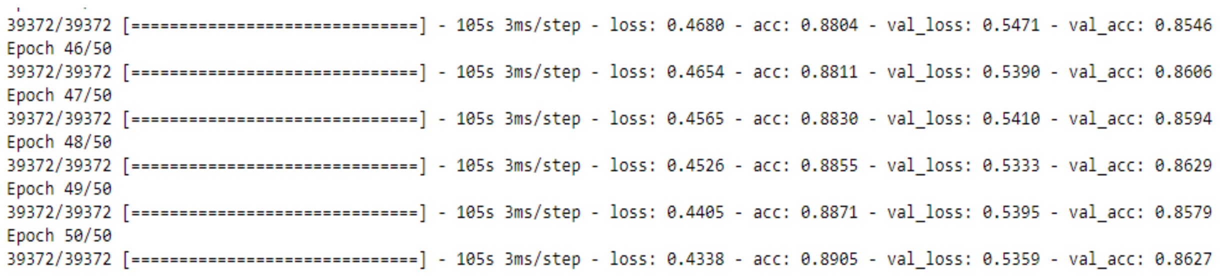


Fig 23: Confusion Matrix of Fake News Model

In the second model developed for news aggregation we have got training accuracy of 89.05% and training loss of 43.38% where as we received validation accuracy of 86.27% and validation loss of 53.59%. The precision, f1-score, recall obtained by this model is 0.89, 0.86, 0.84 respectively.

Proof of results-:



```
39372/39372 [==============================] - 105s 3ms/step - loss: 0.4680 - acc: 0.8804 - val_loss: 0.5471 - val_acc: 0.8546
Epoch 46/50
39372/39372 [==============================] - 105s 3ms/step - loss: 0.4654 - acc: 0.8811 - val_loss: 0.5390 - val_acc: 0.8606
Epoch 47/50
39372/39372 [==============================] - 105s 3ms/step - loss: 0.4565 - acc: 0.8830 - val_loss: 0.5410 - val_acc: 0.8594
Epoch 48/50
39372/39372 [==============================] - 105s 3ms/step - loss: 0.4526 - acc: 0.8855 - val_loss: 0.5333 - val_acc: 0.8629
Epoch 49/50
39372/39372 [==============================] - 105s 3ms/step - loss: 0.4405 - acc: 0.8871 - val_loss: 0.5395 - val_acc: 0.8579
Epoch 50/50
39372/39372 [==============================] - 105s 3ms/step - loss: 0.4338 - acc: 0.8905 - val_loss: 0.5359 - val_acc: 0.8627
```

Fig 24: Result of News Aggregation Model

Accuracy and Loss Graph-

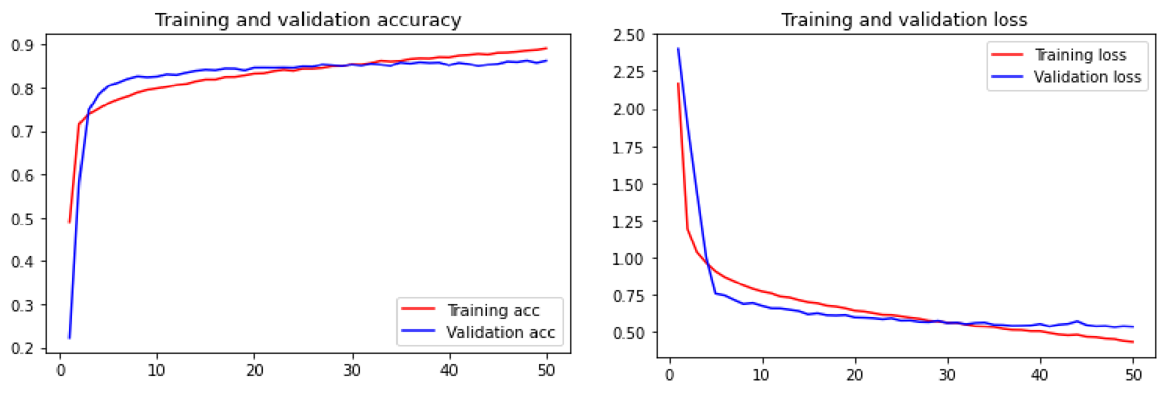


Fig 25-: Accuracy And Loss Graph of News Aggregation Model

Precision, Recall and F1-score-:

```
              precision    recall  f1-score   support

           0       0.87      0.78      0.82      1213
           1       0.81      0.88      0.84      1197
           2       0.92      0.94      0.93      1264
           3       0.84      0.80      0.82      1269
           4       0.92      0.79      0.85      1232
           5       0.94      0.89      0.91      1258
           6       0.92      0.83      0.87      1228
           7       0.90      0.82      0.86      1183

   micro avg       0.89      0.84      0.86      9844
   macro avg       0.89      0.84      0.86      9844
weighted avg       0.89      0.84      0.86      9844
 samples avg       0.84      0.84      0.84      9844
```

Fig 26-: Precision, Recall, F1-scores of News Aggregation Model

Confusion Matrix-:

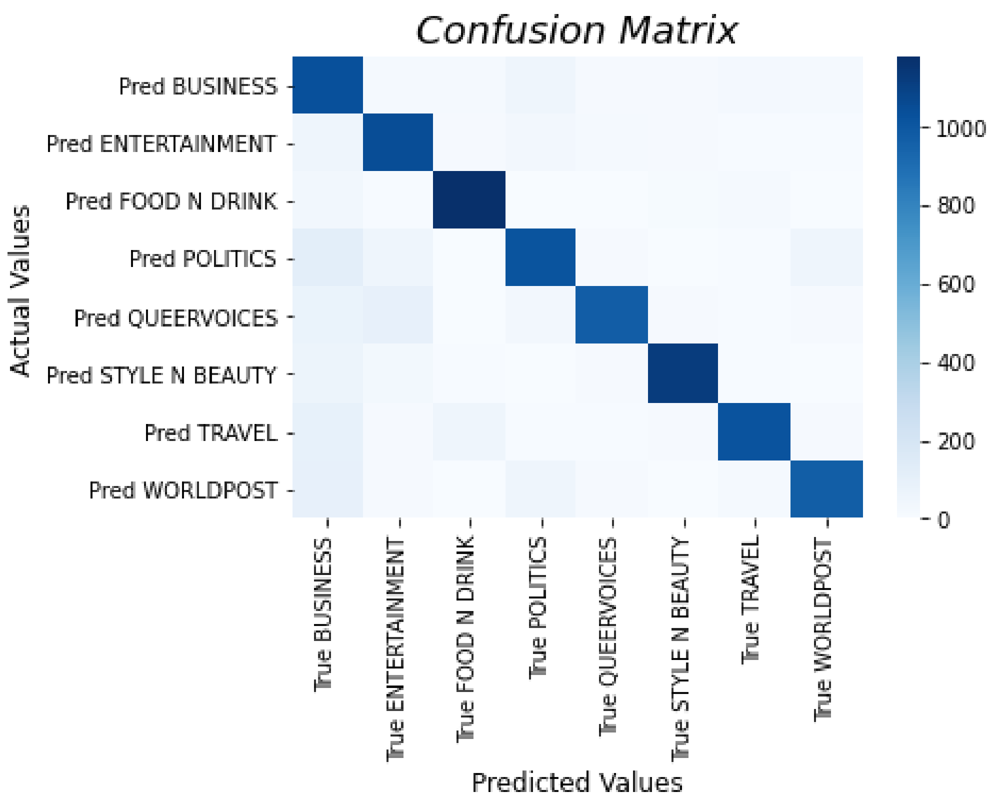


Fig 27-: Confusion Matrix of News Aggregation Model

6. Conclusion and Future Scope

In this project we have purposed two models for classification of true and fakes news articles as well as news categories. We have received 99.6% accuracy and 6.97% loss for fraud news detection model and 86.27% accuracy and 53.5% loss for our news aggregation model. Both of our models have performed very well. We have hybridized existing models and tried too many different combinations of models as well as embedding layers and also discussed the usability of available techniques for natural language processing.

The result and performance are excellent and it is similar to human performance. In all respects we can conclude by saying that, deep learning is one of the best techniques to analyze news data. In deep learning models a very minor modification can change the outcomes completely which is evident from our project.

With a larger dataset the performance of news aggregation model can be enhanced. These models can be used as core of a major system that can provide preferred news articles to a user of his genre of choice. This can reduce the need of fact checker portals. We can keep track of news publishers and authors who have written false news. We can also create a system to check how much dangerous are articles published and we can give confidence scores to authors and the publications accordingly.

7. References

[1] Chaitra K Hiramath , Prof. G.C Deshpande "Fake News Detection Using Deep Learning Techniques" in proceedings of 2019 1st International Conference on Advances in Information Technology.

[2] Ethar Qawasmeh, Mais Tawalbeh, Malak Abdullah "Automatic Identification of Fake News Using Deep Learning" in proceedings of 2019 Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS).

[3] Bhavika Bhutani Neha Rastogi Priyanshu Sehgal Archana Purwar "Fake News Detection Using Sentiment Analysis " Published in: 2019 Twelfth International Conference on Contemporary Computing (IC3)

[4] Tanik Saikh, Arkadipta De, Asif Ekbal, Pushpak Bhattacharyya "A Deep Learning Approach for Automatic Detection of Fake News" arXiv:2005.04938v1 [cs.CL] 11 May 2020

[5] Md Rafqul Islam, Shaowu Liu1, Xianzhi Wang2, Guandong Xu "Deep learning for misinformation detection on online social networks: a survey and new perspectives" Social Network Analysis and Mining (2020) 10:82 https://doi.org/10.1007/s13278-020-00696-x

[6] Sunil Gundapu and Radhika Mamidi "Transformer based Automatic COVID-19 Fake News Detection System " arXiv:2101.00180v3 [cs.CL] 21 Jan 2021

[7] Jamal Abdul Nasir , Osama Subhani Khanb , Iraklis Varlamis c "Fake news detection: A hybrid CNN-RNN based deep learning approach" https://doi.org/10.1016/j.jjimei.2020.100007

[8] Rohit Kumar Kaliya "Fake news detection using a deep neural network" in proceedings of 2018 4th International Conference on Computing Communication and Automation (ICCCA).

[9] N. J. Conroy, V. L. Rubin, and Y. Chen, "Automatic deception detection: Methods for finding fake news," Proceedings of the Association for Information Science and Technology, vol. 52, no. 1, pp. 1–4, 2015.

[10] S. Gilda, "Evaluating machine learning algorithms for fake news detection," IEEE 15th Student Conference on Research and Development (SCOReD), 2017.

[11] ChaoweiZhang, AshishGupta, ChristianKauten, Amit V.Deokar, XiaoQin "Detecting fake news for reducing misinformation risks using analytics approaches" European Journal of Operational Research Volume 279, Issue 3, 16 December 2019, Pages 1036-1052

Appendix-code

Code for Fake News Classification::

By using supervised Learning:

```
#IMPORTING SET 1 REQUIRED LIBRARIES
import pandas as pd
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import string
import nltk
import re
from nltk.stem.porter import *
import tensorflow as tf
from tensorflow.keras.preprocessing.sequence import pad_sequences
import random
import matplotlib.pyplot as plt
from matplotlib.ticker import StrMethodFormatter
from nltk.stem import WordNetLemmatizer

#CONFIGURING GPUS
config = tf.compat.v1.ConfigProto()
config.gpu_options.allow_growth = True
session =tf.compat.v1.InteractiveSession(config=config)

#CREATING A DATAFRAME AND IMPORTING DATASET
fake_df = pd.DataFrame()
fake_df = pd.read_csv(f"train_fake.csv")
fake_df.head()

#DROPING UNNECESSARY COLUMNS
fake_df.drop(labels=[f"id",f"author"],axis=1,inplace=True)
fake_df.head()

#CHECING IF ANY ROWS WITH NO VALUES ARE PRESENT
fake_df = fake_df.dropna()
fake_df.isnull().sum()

#PRINTING TOTAL NUMBER OF ARTICLES AVAILABLE FOR
DIFFERENT CLASSES
print("Total Genuine News Present =",end="   " )
print(fake_df[fake_df['label']==1].size//3)
print("Total Fake News Present= ", end="   ")
print(fake_df[fake_df['label']==0].size//3)
```

```
#ADDING A LENGTH COLUMN BY ADDING ARTICLE HEAD
AND BODY AND PRONTING A HISTOGRAM OF ARTICLE
LENGTHS
length = []
fake_df['text1'] = fake_df.title+" "+fake_df.text#df.headline  + "  " +
df.short_description
[length.append(len(str(text))) for text in fake_df['text1']]
fake_df['length'] = length
fake_df[f"length"].hist(bins=80)
plt.xlim(0,500)
plt.xlabel("Text Lengths")
plt.ylabel("Number of Articles")

#DROPING ARTICLES WITH LENGTH LESS THAN 50
fake_df = fake_df[fake_df.length > 50]
fake_df.drop(labels=[f"title",f"length",f"text"],axis=1,inplace=True)
fake_df.head()

# shuffle the DataFrame rows
fake_df = fake_df.sample(frac = 1)
fake_df = fake_df.sample(frac = 1)
fake_df.head()

#PRINTING TOTAL NUMBER OF ARTICLES AVAILABLE FOR
DIFFERENT CLASSES
print("Total Genuine News Present =",end="   " )
print(fake_df[fake_df['label']==0].size//2)
print("Total Fake News Present= ", end="   ")
print(fake_df[fake_df['label']==1].size//2)

def preprocessing_1(data):

   # LOWER CASING ARTICLE TEXT
   data[f"text1"] = data[f"text1"].apply(lambda x: x.lower())
   #removing special characters
   data['text1'] = data['text1'].apply((lambda x: re.sub('[^a-zA-Z0-
9\s]','',x)))

   #THE FOLLOWING CODE REMOVES EXTRA ARTICLES FROM
CLASS HAVING MORE ARTICLES AND QUALIZES NUMBER OF
ARTICLES FOR ALL CLASSES
   fake=[]
   true =[]

   for i in data.index:
      if data['label'][i]==0:
         fake.append([data['text1'][i],data['label'][i]])
      else:
         true.append([data['text1'][i],data['label'][i]])

   lower = min(len(true),len(fake))
```

```
    fake=fake[:lower]
    true=true[:lower]

    allData = fake+true
    random.shuffle(allData)
    random.shuffle(allData)
    new_df = pd.DataFrame()
    text1=[]
    label=[]
    for i,j in allData:
        text1.append(i)
        label.append(j)
    new_df['text'] = text1
    new_df['label'] = label
    return new_df

#CALLING PREPROCESSING 1 FUNCTION
fake_df[f"text1"] = fake_df[f"text1"].astype(str)
fake_df = preprocessing_1(fake_df)
fake_df.head(10)

def preprocessing_2(data):
    text = data[f"text"]
    #lEMMATIZING,TOKENIZING, PADDING TEXT DATA
    text = [[lemmatizer.lemmatize(word) for word in sentence.split(" ")]
for sentence in text]

    tokenizer.fit_on_texts(text)
    X = tokenizer.texts_to_sequences(text)
    X = pad_sequences(X,maxlen=250,padding='pre')
    """
    tokenizer.fit_on_texts(text.values)
    X = tokenizer.texts_to_sequences(text.values)
    X = pad_sequences(X,maxlen=256,padding='pre')
    """
    return X

#CALLING PREPROCESSING 2 FUNCTION
lemmatizer = WordNetLemmatizer()
fake_df = fake_df.sample(frac = 1) #shuffling data set
max_features = 70000
tokenizer = tf.keras.preprocessing.text.Tokenizer(filters='!"#$%&()*+,-
./:;<=>?@[\\]^_`{|}~\t\n',num_words=max_features,split=' ')
X = preprocessing_2(fake_df)
print(X)

# category to id

categories = fake_df.groupby('label').size().index.tolist()
category_int = {}
```

```
int_category = {}
for i, k in enumerate(categories):
    category_int.update({k:i})
    int_category.update({i:k})

fake_df['c2id'] = fake_df['label'].apply(lambda x: category_int[x])

#iMPORTING SECOND SET OF LIBRARIES
import tensorflow.keras
from tensorflow.keras.models import Sequential
from sklearn.feature_extraction.text import CountVectorizer

from keras.layers import Dense, Dropout, LSTM,
BatchNormalization,Embedding,SpatialDropout1D,CuDNNLSTM,Bidir
ectional,GRU
from keras.layers import Conv1D, MaxPooling1D, Flatten,Dense


from sklearn.model_selection import train_test_split
from keras.engine.topology import Layer

from nltk.corpus import stopwords
from nltk.stem import SnowballStemmer
from string import punctuation

from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.layers import Dense, Input, LSTM, Embedding, Dropout,
Activation
from keras.layers.merge import concatenate
from keras.models import Model
from keras.layers.normalization import BatchNormalization
from keras.callbacks import EarlyStopping, ModelCheckpoint

from keras import backend as K
from keras import initializers, regularizers, constraints

#SPLITTING DATA TO TRAIN SET AND VALIDATION SET
from keras.utils import np_utils
X = np.array(X)
Y = np_utils.to_categorical(list(fake_df.c2id))
seed = 29
x_train, x_val, y_train, y_val = train_test_split(X, Y, test_size=0.2,
random_state=seed)

#SELF-ATTENTION CLASS
class Attention(Layer):
    def __init__(self, step_dim,
                 W_regularizer=None, b_regularizer=None,
                 W_constraint=None, b_constraint=None,
                 bias=True, **kwargs):
```

```
        self.supports_masking = True
        self.init = initializers.get('glorot_uniform')
        self.W_regularizer = regularizers.get(W_regularizer)
        self.b_regularizer = regularizers.get(b_regularizer)
        self.W_constraint = constraints.get(W_constraint)
        self.b_constraint = constraints.get(b_constraint)
        self.bias = bias
        self.step_dim = step_dim
        self.features_dim = 0
        super(Attention, self).__init__(**kwargs)

    def build(self, input_shape):
        assert len(input_shape) == 3
        self.W = self.add_weight(shape=(input_shape[-1],),
                                 initializer=self.init,
                                 name='{}_W'.format(self.name),
                                 regularizer=self.W_regularizer,
                                 constraint=self.W_constraint)
        self.features_dim = input_shape[-1]
        if self.bias:
            self.b = self.add_weight(shape=(input_shape[1],),
                                     initializer='zero',
                                     name='{}_b'.format(self.name),
                                     regularizer=self.b_regularizer,
                                     constraint=self.b_constraint)
        else:
            self.b = None
        self.built = True

    def compute_mask(self, input, input_mask=None):
        return None

    def call(self, x, mask=None):
        features_dim = self.features_dim
        step_dim = self.step_dim
        eij = K.reshape(K.dot(K.reshape(x, (-1, features_dim)),
K.reshape(self.W, (features_dim, 1))), (-1, step_dim))
        if self.bias:
            eij += self.b
        eij = K.tanh(eij)
        a = K.exp(eij)
        if mask is not None:
            a *= K.cast(mask, K.floatx())
        a /= K.cast(K.sum(a, axis=1, keepdims=True) + K.epsilon(),
K.floatx())
        a = K.expand_dims(a)
        weighted_input = x * a
        return K.sum(weighted_input, axis=1)

    def compute_output_shape(self, input_shape):
        return input_shape[0],  self.features_dim
```

```
# Converting data to matrix of shape (194902,100) where 194902 is
number of words in vocaboulry + 1 and 100 word vector for each word
word_index = tokenizer.word_index

EMBEDDING_DIM = 100

embeddings_index = {}
f = open('glove.6B.100d.txt', encoding="utf8")
for line in f:
    values = line.split()
    word = values[0]
    coefs = np.asarray(values[1:], dtype='float32')
    embeddings_index[word] = coefs
f.close()

print('Found %s unique tokens.' % len(word_index))
print('Total %s word vectors.' % len(embeddings_index))

#CREATING EMBEDDING LAYER
from keras.initializers import Constant
embedding_matrix = np.zeros((len(word_index) + 1,
EMBEDDING_DIM))
for word, i in word_index.items():
    embedding_vector = embeddings_index.get(word)
    if embedding_vector is not None:
        embedding_matrix[i] = embedding_vector

embedding_layer = Embedding(len(word_index)+1,
                            EMBEDDING_DIM,
                            embeddings_initializer=Constant(embedding_matrix),
                            input_length=150,
                            trainable=False)

#CREATING MODEL
l = 250
inp = Input(shape=(l,), dtype='int32')
x = Embedding(194902, 100,input_length =
X.shape[1],weights=[embedding_matrix],trainable=False)(inp)

x = Dropout(0.3)(x)
x = Bidirectional(LSTM(l, dropout=0.3, recurrent_dropout=0.3,
return_sequences=True))(x)


x = Dropout(0.3)(x)


x = BatchNormalization()(x)
```

```
merged = Attention(l)(x)
merged = Dense(64, activation='relu')(merged)
merged = Dropout(0.3)(merged)
merged = BatchNormalization()(merged)
outp = Dense(len(int_category),
activation='softmax',kernel_regularizer='l1_l2')(merged)

AttentionLSTM = Model(inputs=inp, outputs=outp)
AttentionLSTM.compile(loss='categorical_crossentropy',
optimizer='adam', metrics=['acc'])

AttentionLSTM.summary()#TRAINING MODEL
attlstm_history = AttentionLSTM.fit(x_train,
                        y_train,
                        batch_size=256,
                        epochs=100,
                        validation_data=(x_val, y_val))

#PRINTING ACCURACY AND LOSS OF TRAING AND
VALIDATION SET FOR EVERY EPOCH
acc = attlstm_history.history['acc']
val_acc = attlstm_history.history['val_acc']
loss = attlstm_history.history['loss']
val_loss = attlstm_history.history['val_loss']
epochs = range(1, len(acc) + 1)

plt.title('Training and validation accuracy')
plt.plot(epochs, acc, 'red', label='Training acc')
plt.plot(epochs, val_acc, 'blue', label='Validation acc')
plt.legend()

plt.figure()
plt.title('Training and validation loss')
plt.plot(epochs, loss, 'red', label='Training loss')
plt.plot(epochs, val_loss, 'blue', label='Validation loss')
plt.legend()

plt.show()


#PRINTING CONFUSION MATRIX
y_labels = ['Pred False' ,'Pred True']
x_labels = ['Actual False','Actual True']

from sklearn.metrics import confusion_matrix

pred = AttentionLSTM.predict(x_val)

pred= np.round(pred)
```

```
cf_matrix=confusion_matrix(y_val.argmax(axis=1),pred.argmax(axis=1)
)

pos = 0
neg = 0
for i in range(len(y_val)):
    pos += y_val[i][1]
    neg += y_val[i][0]



cf_matrix[1][1] = cf_matrix[1][1]/pos*100
cf_matrix[0][1] = cf_matrix[0][1]/pos*100
cf_matrix[0][0] = cf_matrix[0][0]/neg*100
cf_matrix[1][0] = cf_matrix[1][0]/neg*100

sns.heatmap(cf_matrix,xticklabels=x_labels, yticklabels=y_labels ,
cmap='Blues',fmt=".2g")
plt.xlabel('Predicted Values',fontsize = 12)
plt.ylabel('Actual Values',fontsize = 12)

plt.title("Confusion Matrix", fontsize = 18, color='Black',
fontstyle='italic')
plt.show()

#PRINTING PRECISION, RECALL, F1-SCORE
from sklearn.metrics import classification_report

y_pred = AttentionLSTM.predict(x_val)
pred= np.round(y_pred)
print(classification_report(y_val, pred))
```

Code for News Aggregation::

By using supervised Learning:

```
#IMPORTING SET 1 REQUIRED LIBRARIES
import pandas as pd
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import string
import nltk
import re
from nltk.stem.porter import *
import tensorflow as tf
from tensorflow.keras.preprocessing.sequence import pad_sequences
```

```
import random
import matplotlib.pyplot as plt
from matplotlib.ticker import StrMethodFormatter
from nltk.stem import WordNetLemmatizer

#CONFIGURING GPUS
config = tf.compat.v1.ConfigProto()
config.gpu_options.allow_growth = True
session =tf.compat.v1.InteractiveSession(config=config)

#CREATING A DATAFRAME AND IMPORTING DATASET
fake_df = pd.DataFrame()
df = pd.read_json(f"News_Category_Dataset_v2.json " , lines=True)
fake_df.head()

#DROPING UNNECESSARY COLUMNS
df.drop(labels=[f"date",f"authors",f"link"],axis=1,inplace=True)
df.head()

#CHECING IF ANY ROWS WITH NO VALUES ARE PRESENT
df = df.dropna()
df.isnull().sum()

#ADDING A LENGTH COLUMN BY ADDING ARTICLE HEAD
AND BODY AND PRONTING A HISTOGRAM OF ARTICLE
LENGTHS
length = []
df['text'] = df.headline  + "  " + df.short_description
[length.append(len(str(text))) for text in df['text']]
df['length'] = length
df[f"length"].hist(bins=80)
plt.xlim(0,500)
plt.xlabel("Text Lengths")
plt.ylabel("Number of Articles")

#dROPING ARTICLES WITH LENGTH LESSER THAN 50 AND
UNNECESSARY COLUMNS
df = df[df.length > 50]
df.drop(labels=[f"short_description",f"headline"],axis=1,inplace=True)
df.head()

#DROPING CLASSES WITH VERY LESS ARTICLES AND
CLUBING SIMILAR ARTICLES
df.category = df.category.map(lambda x: "WORLDPOST" if x == "THE
WORLDPOST" else x)
df.category = df.category.map(lambda x: "WORLDPOST" if x ==
"WORLD NEWS" else x)

df.category = df.category.map(lambda x: "BUSINESS" if x ==
"MONEY" else x)
```

```
df.category = df.category.map(lambda x: None if x == "EDUCATION"
else x)
df.category = df.category.map(lambda x: None if x == "CULTURE &
ARTS" else x)
df.category = df.category.map(lambda x: None if x == "ARTS &
CULTURE" else x)
df.category = df.category.map(lambda x: None if x == "LATINO
VOICES" else x)
df.category = df.category.map(lambda x: None if x == "COLLEGE" else
x)
df.category = df.category.map(lambda x: None if x ==
"ENVIRONMENT" else x)
df.category = df.category.map(lambda x: None if x == "GOOD NEWS"
else x)
df.category = df.category.map(lambda x: None if x == "FIFTY" else x)
df.category = df.category.map(lambda x: None if x == "ARTS" else x)
df.category = df.category.map(lambda x: None if x == "MONEY" else x)
df.category = df.category.map(lambda x: None if x == "TECH" else x)
df.category = df.category.map(lambda x: None if x == "TASTE" else x)
df.category = df.category.map(lambda x: None if x == "WORLD
NEWS" else x)
df.category = df.category.map(lambda x: None if x == "SCIENCE" else
x)
df.category = df.category.map(lambda x: None if x == "STYLE" else x)
df.category = df.category.map(lambda x: None if x == "RELIGION" else
x)
df.category = df.category.map(lambda x: None if x == "GREEN" else x)
df.category = df.category.map(lambda x: None if x == "WEIRD NEWS"
else x)
df.category = df.category.map(lambda x: None if x == "MEDIA" else x)
df.category = df.category.map(lambda x: None if x == "CRIME" else x)
df.category = df.category.map(lambda x: None if x == "DIVORCE" else
x)
df.category = df.category.map(lambda x: None if x == "WOMEN" else
x)
df.category = df.category.map(lambda x: None if x == "WEDDINGS"
else x)
df.category = df.category.map(lambda x: None if x == "PARENTS" else
x)
df.category = df.category.map(lambda x: None if x == "BLACK
VOICES" else x)
df.category = df.category.map(lambda x: None if x == "IMPACT" else x)
df.category = df.category.map(lambda x: None if x == "HOME &
LIVING" else x)
df.category = df.category.map(lambda x: None if x == "PARENTING"
else x)
df.category = df.category.map(lambda x: None if x == "HEALTHY
LIVING" else x)
df.category = df.category.map(lambda x: None if x == "WELLNESS"
else x)
```

```
df.category = df.category.map(lambda x: None if x == "COMEDY" else
x)
df.category = df.category.map(lambda x: None if x == "SPORTS" else x)

df.dropna(inplace=True)



def preprocessing_21(data):


   data[f"text"] = data[f"text"].apply(lambda x: x.lower())
   #removing special characters
   data['text'] = data['text'].apply((lambda x: re.sub('[^a-zA-Z0-9\s]','',x)))
   #THE FOLLOWING CODE REMOVES EXTRA ARTICLES FROM
CLASS HAVING MORE ARTICLES AND QUALIZES NUMBER OF
ARTICLES FOR ALL CLASSES

   POLITICS = []
   ENTERTAINMENT = []
   BEAUTY = []
   TRAVEL= []
   WORLDPOST = []
   BUSINESS =[]
   FOOD = []
   VOICES = []



   for i in data.index:

      if data['category'][i]=='POLITICS':
         POLITICS.append([data['category'][i],data['text'][i]])
      elif data['category'][i]=='ENTERTAINMENT':
         ENTERTAINMENT.append([data['category'][i],data['text'][i]])
      elif data['category'][i]=='TRAVEL':
         TRAVEL.append([data['category'][i],data['text'][i]])
      elif data['category'][i]=='STYLE & BEAUTY':
         BEAUTY.append([data['category'][i],data['text'][i]])
      elif data['category'][i]=='WORLDPOST':
         WORLDPOST.append([data['category'][i],data['text'][i]])
      elif data['category'][i]=='FOOD & DRINK':
         FOOD.append([data['category'][i],data['text'][i]])
      elif data['category'][i]=='QUEER VOICES':
         VOICES.append([data['category'][i],data['text'][i]])

      elif data['category'][i]=='BUSINESS':
         BUSINESS.append([data['category'][i],data['text'][i]])
```

```
    lower =
min(len(POLITICS),len(ENTERTAINMENT),len(TRAVEL),len(BEAU
TY),len(WORLDPOST),len(FOOD),len(VOICES),len(BUSINESS))

    POLITICS = POLITICS[:lower]
    FOOD = FOOD[:lower]
    WORLDPOST = WORLDPOST[:lower]
    TRAVEL = TRAVEL[:lower]
    ENTERTAINMENT =ENTERTAINMENT[:lower]
    BUSINESS =BUSINESS[:lower]
    BEAUTY =BEAUTY[:lower]
    VOICES = VOICES[:lower]



    allData = POLITICS+FOOD + WORLDPOST + TRAVEL +
ENTERTAINMENT + BEAUTY + VOICES+ BUSINESS

    random.shuffle(allData)
    random.shuffle(allData)
    new_df = pd.DataFrame()

    text=[]
    category=[]

    for i,j in allData:
        category.append(i)
        text.append(j)
    print(lower)
    new_df['category'] = category
    new_df['text'] = text


    return new_df

#cALLING PREPROCESSING FUNCTION 1
df[f"category"] = df[f"category"].astype(str)
df = preprocessing_21(df)
df.head(10)

#PRINTING DIFFERENT CLAASES AVAILABLE AND
CORRESPONDING ARTICLES FOR THOSE CLASSES
cates = df.groupby('category')
print("TOTAL DIFFERENaT CATEGORIES:", cates.ngroups)
print("ALL THE DIFFERENT GENRES AVAILABLE")
df.category.value_counts()

def preprocessing_2(data):
    text = data[f"text"]
    # TOKENIZING AND PADDING ARTICLE TEXT
    #text = [[lemmatizer.lemmatize(word) for word in sentence.split(" ")]
for sentence in text]

    tokenizer.fit_on_texts(text)
    X = tokenizer.texts_to_sequences(text)
    X = pad_sequences(X,maxlen=250,padding='pre')


    return X

from nltk.stem import WordNetLemmatizer

#CALLING SECOND PREPROCESSING FUNCTION
df[f"text"] = df[f"text"].astype(str)
lemmatizer = WordNetLemmatizer()


df = df.sample(frac = 1) #shuffling data set
max_features = 100000
tokenizer = tf.keras.preprocessing.text.Tokenizer(filters='!"#$%&()*+,-
./:;<=>?@[\\]^_`{|}~\t\n',num_words=max_features,split=' ')
X = preprocessing_2(df)

# category to id

categories = df.groupby('category').size().index.tolist()
category_int = {}
int_category = {}
for i, k in enumerate(categories):
    category_int.update({k:i})
    int_category.update({i:k})

df['c2id'] = df['category'].apply(lambda x: category_int[x])

#IMPORTING 2ND SET OF REQUIRED LIBRARIES
import tensorflow.keras
from tensorflow.keras.models import Sequential
from sklearn.feature_extraction.text import CountVectorizer

from keras.layers import Dense, Dropout, LSTM,
BatchNormalization,Embedding,SpatialDropout1D,CuDNNLSTM,Bidir
ectional,GRU
from keras.layers import Conv1D, MaxPooling1D, Flatten,Dense


from sklearn.model_selection import train_test_split
from keras.engine.topology import Layer

from nltk.corpus import stopwords
from nltk.stem import SnowballStemmer
from string import punctuation
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.layers import Dense, Input, LSTM, Embedding, Dropout,
Activation
from keras.layers.merge import concatenate
from keras.models import Model
from keras.layers.normalization import BatchNormalization
from keras.callbacks import EarlyStopping, ModelCheckpoint

from keras import backend as K
from keras import initializers, regularizers, constraints

from keras.utils import np_utils
X = np.array(X)
Y = np_utils.to_categorical(list(df.c2id))

# SPLITING TO TRAINING SET AND VALIDATION SET

seed = 29
x_train, x_val, y_train, y_val = train_test_split(X, Y, test_size=0.2,
random_state=seed)

#SELF-ATTENTION CLASS
class Attention(Layer):
    def __init__(self, step_dim,
                 W_regularizer=None, b_regularizer=None,
                 W_constraint=None, b_constraint=None,
                 bias=True, **kwargs):
        self.supports_masking = True
        self.init = initializers.get('glorot_uniform')
        self.W_regularizer = regularizers.get(W_regularizer)
        self.b_regularizer = regularizers.get(b_regularizer)
        self.W_constraint = constraints.get(W_constraint)
        self.b_constraint = constraints.get(b_constraint)
        self.bias = bias
        self.step_dim = step_dim
        self.features_dim = 0
        super(Attention, self).__init__(**kwargs)

    def build(self, input_shape):
        assert len(input_shape) == 3
        self.W = self.add_weight(shape=(input_shape[-1],),
                                 initializer=self.init,
                                 name='{}_W'.format(self.name),
                                 regularizer=self.W_regularizer,
                                 constraint=self.W_constraint)
        self.features_dim = input_shape[-1]
        if self.bias:
            self.b = self.add_weight(shape=(input_shape[1],),
                                     initializer='zero',
                             name='{}_b'.format(self.name),
                             regularizer=self.b_regularizer,
                             constraint=self.b_constraint)
        else:
            self.b = None
        self.built = True

    def compute_mask(self, input, input_mask=None):
        return None

    def call(self, x, mask=None):
        features_dim = self.features_dim
        step_dim = self.step_dim
        eij = K.reshape(K.dot(K.reshape(x, (-1, features_dim)),
K.reshape(self.W, (features_dim, 1))), (-1, step_dim))
        if self.bias:
            eij += self.b
        eij = K.tanh(eij)
        a = K.exp(eij)
        if mask is not None:
            a *= K.cast(mask, K.floatx())
        a /= K.cast(K.sum(a, axis=1, keepdims=True) + K.epsilon(),
K.floatx())
        a = K.expand_dims(a)
        weighted_input = x * a
        return K.sum(weighted_input, axis=1)

    def compute_output_shape(self, input_shape):
        return input_shape[0],  self.features_dim

#IMPORTING WORD2VEC EMBEDDINGS
import gensim
wordembeddings = gensim.models.KeyedVectors.
load_word2vec_format('GoogleNews-vectors-negative300.bin',
binary=True)
# Converting data to matrix of shape (51262,300) where 51262 is number
of words in vocaboulry + 1 and 300 word vector for each word
word_index = tokenizer.word_index
unique_words = len(word_index)
total_words = unique_words + 1
skipped_words = 0
embedding_dim = 300
embedding_matrix = np.zeros((total_words, embedding_dim))
for word, index in tokenizer.word_index.items():
    embedding_vector = word_index.get(word)
    try:
        embedding_vector = wordembeddings[word]
    except:
        skipped_words = skipped_words+1
    pass
    if embedding_vector is not None:
        embedding_matrix[index] = embedding_vector
print("Embeddings Matrix shape : ",embedding_matrix.shape)

#BI-LSTM MODEL
l = 250
inp = Input(shape=(l,), dtype='int32')
x = Embedding(51262, 300,input_length =
X.shape[1],weights=[embedding_matrix],trainable=False)(inp)
x = Dropout(0.3)(x)
x = Bidirectional(LSTM(l, dropout=0.3, recurrent_dropout=0.3,
return_sequences=True))(x)
x = BatchNormalization()(x)
x = BatchNormalization()(x)

merged = Attention(l)(x)
merged = Dense(128, activation='relu')(merged)
merged = Dropout(0.3)(merged)
merged = BatchNormalization()(merged)
outp = Dense(len(int_category),
activation='softmax',kernel_regularizer='l1_l2')(merged)

AttentionLSTM = Model(inputs=inp, outputs=outp)
AttentionLSTM.compile(loss='categorical_crossentropy',
optimizer='adam', metrics=['acc'])

AttentionLSTM.summary()

#MODEL TRAINING
attlstm_history = AttentionLSTM.fit(x_train,
                            y_train,
                            batch_size=256,
                            epochs=40,
                            validation_data=(x_val, y_val))


#PRINTING ACCURACY AND LOSS OF TRAING AND
VALIDATION SET FOR EVERY EPOCH
acc = attlstm_history.history['acc']
val_acc = attlstm_history.history['val_acc']
loss = attlstm_history.history['loss']
val_loss = attlstm_history.history['val_loss']
epochs = range(1, len(acc) + 1)

plt.title('Training and validation accuracy')
plt.plot(epochs, acc, 'red', label='Training acc')
plt.plot(epochs, val_acc, 'blue', label='Validation acc')
plt.legend()

plt.figure()
plt.title('Training and validation loss')
plt.plot(epochs, loss, 'red', label='Training loss')
plt.plot(epochs, val_loss, 'blue', label='Validation loss')
plt.legend()

plt.show()



#CALCULATING RECALL,F1-SCORE,PRECISION
from sklearn.metrics import classification_report

y_pred = AttentionLSTM.predict(x_val)
pred= np.round(y_pred)
print(classification_report(y_val, pred))

#CALCULATING AND PRINTING CONFUSION MATRIX
y_labels = ['Pred BUSINESS' ,'Pred ENTERTAINMENT','Pred FOOD N
DRINK','Pred POLITICS','Pred QUEERVOICES','Pred STYLE N
BEAUTY','Pred TRAVEL','Pred WORLDPOST']
x_labels = ['Pred BUSINESS' ,'Pred ENTERTAINMENT','Pred FOOD N
DRINK','Pred POLITICS','Pred QUEERVOICES','Pred STYLE N
BEAUTY','Pred TRAVEL','Pred WORLDPOST']
from sklearn.metrics import confusion_matrix


pred= np.round(y_pred)

cf_matrix=confusion_matrix(y_val.argmax(axis=1),pred.argmax(axis=1)
)
sns.heatmap(cf_matrix ,xticklabels=x_labels, yticklabels=y_labels ,
cmap='Blues',fmt=".2g")
plt.xlabel('Predicted Values',fontsize = 12)
plt.ylabel('Actual Values',fontsize = 12)

plt.title("Confusion Matrix", fontsize = 18, color='Black',
fontstyle='italic')
plt.show()
```