JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY

FAKE NEWS DETECTION USING ARTIFICAL INTELLIGENCE

Submitted By:
Vanshika Mittal 16103012
Abhishek Verma 16103237
B-5
G-06

Submitted To:
Dr. Suma Dawn

Introduction:

Fake news on social media has experienced a resurgence of interest due to the recent political climate and the growing concern around its negative effect. For example , in January 2017 , a spokesman for the German government stated that they“are dealing with a phenomenon of a dimension that [they] have not seen before”,referring to the proliferation of fake news . Not only does it provide a source of spam in our lives , but fake news also has the potential to manipulate public perception and awareness in a major way . Detecting misinformation on social media is an extremely important but also a technically challenging problem.

The difficulty comes impart from the fact that even the human eye cannot accurately distinguish true from false news ; for example, one study found that when shown a fake news article, respondents found it “ ‘somewhat’ or ‘very’ accurate 75% of the time”,and another found that 80% of high school students had a hard time determining whether an article was fake.In an attempt to combat the growing misinformation and confusion ,several fact-checking websites have been deployed to expose or confirm stories (e.g. snopes.com). These websites play a crucial role in combating fake news,but they require expert analysis which inhibits a timely response. As a response , numerous articles and blogs have been written to raise public awareness and provide tips on differentiating truth from falsehood. While each author provides a different set of signals to look out for ,there are several characteristics that are generally agreed upon, relating to the text of an article , the response it receives,and its source. The most natural characteristic is the text of an article.

Advice in the media varies from evaluating whether the headline matches the body of the article , to judging the consistency and quality of the language . Attempts to automate the evaluation of text have manifested in sophisticated natural language processing and machine learning techniques that rely on hand -crafted and data -specific textual features to classify a piece of text as true or false. These approaches are limited by the fact that the linguistic characteristics of fake news are still not yet fully understood. Further,the characteristics vary across different types of fake news ,topics,and media platforms.

Fake news undermines serious media coverage and makes even more difficult for journalists to cover true news stories. According to an analysis done by Buzzfeed , it was found that top20 fake news stories about 2016 U.S. presidential election received more engagement on facebook than top 20 election stories from 19 major media outlets.
So,it becomes really important to know if a piece of information we are reading is even truly reliable or not.

Literature Survey:

1. **A Deep Neural Network for Fake News Detection**

   Sneha Singhania , Nigel Fernandez, and Shrisha Rao
   2017

   Algorithms Used and how the modules work:

   Bag of Words/ n- grams with Hierarchial Neural
   Network, SVM, CNN, LSTM ,GRU,Sequence Encoder using GRU, Word Encoder, Word Attention, Sentence Encoder, Sentence Attention, Headline Encoder.

Author splited sentences of bodies and tokenized sentences and headlines into words using Stanford CoreNLP. Lower cased and cleaned tokens are retaining alphabets, numerals and significant punctuation marks. When building the vocabulary we retained words with frequency more than 5. He treated words appearing exactly 5 times as a special single unknown token (UNK) and used 100 dimensional GloVe embeddings to initialize word embedding matrix and allowed it to be fine tuned. For missing words in GloVe, initialized their word embedding from a uniform distribution on $(-0.25,0.25)$. We padded (or truncated) each sentence and headline to an average word count of 32 and each article to an average sentence count of 21 . Hyperparameters are tuned on the validation set. 100 dimensional GloVe embeddings and 50 dimensional GRU annotations giving a combined annotation of 100 dimensions. The relevance vector at word, sentence and headline-body level are of 100 dimensions trained as a parameter of our model. SGD with a learning rate of 0.01, momentum of 0.9 and mini batch size of 32 to train all neural models. Accuracy was our evaluation metric since our data set is balanced.

Data Sets used by Author:

PolitiFact a respected fact-checking website released a list of sites manually investigated and labelled. Author used those sites from this list labelled fake. Forbes compiled a list of popular genuine sites across US demographics. To maintain a similar distribution as fake articles, genuine articles from January 1, 2016 to June 1, 2017 are used , with 65% coming from the 2016 US elections and politics, 15% from world news, 15% from regional news and 5% from entertainment.

Comparison criteria of their results

Author used a train, validation and test split of 20% | 10% | 70% for neural models and a train and test split of 30% | 70% for word count based models. In 3HAN-Ave vectors are composed using average, in 3HAN-Max vectors are composed using max pooling, 3HAN is their proposed model with an attention mechanism for composition and 3HAN+PT denotes pre-trained 3HAN model. Neural models using the hierarchical structure (HAN and variants, 3HAN and variants) give a higher accuracy than other baselines. The attention mechanism is a more effective composition operator than average or max pooling.

This is demonstrated by the higher accuracy of 3HAN against 3HAN-Ave and 3HAN-Max and headline premise is valid since 3HAN which devotes a separate third level in the hierarchy for the

headline performs better than HAN. HAN is indifferent to the headline and focuses its two hierarchical levels only on words and sentences. Pre-training helps in better initialization of 3HAN with 3HAN+PT outperforming 3HAN.

Merits of the methods that authors have proposed

Author demonstrated the superior accuracy of 3HAN over other state-of-the-art models.They highlighted the use of visualization of the attention layers and planned to deploy a web application based on 3HAN which provides detection of fake news as a service and learns in a real time online manner from new manually fact-checked articles.

Future Works:

In this paper, he presented 3HAN which creates news vector, an effective representation of an article for detection as fake news. We demonstrated the superior accuracy of 3HAN over other state-of-the-art models. We highlighted the use of visualization of the attention layers. We plan to deploy a web application based on 3HAN which provides detection of fake news as a service and learns in a real time online manner from new manually fact-checked articles.

2. "The Pope Has a New Baby!" Fake News Detection Using Deep Learning

By Samir Bajaj , 2017

Algorithms Used and how the modules work:

While there exist tools and products to detect sources of fake news (e.g., whether a web site publishes misleading news),they approached this problem as an instance of text classification, using only the content of the article as the source of features. Doing so affords us to focus on NLP-related algorithms, thereby allowing us to explore in depth the performance of a variety of models on a particular task.

Several different models were implemented as mentioned below. All models use pre-trained 300-dimensional GloVe embeddings, which are not updated during training.

Logistic Regression, Two-layer feedforward Neural Network, Recurrent Neural Network, Long Short-Term Memories , Grated Recurrent Units,Bidirectional RNN with LSTMs ,Convolutional Neural Network with Max Pooling, Attention - Augmented Convolutional Neural Network.

All code are written in Python 2.7, using TensorFlow r0.12.1 and NumPy. However, TensorFlow's implementation of LSTM and GRU cells was not used , instead they were coded from scratch. Xavier initialization was used for all variables. Dropout was employed as a regularization mechanism for the feedforward and the recurrent neural networks.

Experiments were run on local CPU-only machines, as well as on GPU-enabled Azure clusters provided by Microsoft.The hyperparameters used for the various models are listed in table 1.

Model	Embedding Size	Learning Rate	Optimizer	Hidden Size	Dropout	Time Steps	Epochs
Logistic Regression	300	10^{-1}	SGD	—	—	—	100
Feedforward Network	300	10^{-3}	Adam	200	0.5	—	100
RNN (Vanilla)	300	10^{-3}	Adam	100	0.5	200	30
GRUs	300	10^{-4}	Adam	100	0.5	200	30
LSTMs	300	10^{-4}	Adam	100	0.5	200	30
BiLSTMs	300	10^{-3}	Adam	100	0.5	200	20
CNN with Max Pooling	300	10^{-2}	Adam	100	—	—	100
CNN with Max Pooling and Attention	300	10^{-2}	Adam	100	—	—	10

Table 1: Hyperparameters for the Various Models

Data Sets used by Author:

The data used for this project was drawn from two different sources, both in public domain. Fake news articles were procured from an open Kaggle dataset made up of 13,000 articles that span multiple subjects. Authentic news articles (negative examples for the classifier) were extracted from the Signal Media News dataset reservoir and sampling was used to select 50,000 entries uniformly at random from the corpus. Binary 0/1 labels were assigned to the real/fake news articles, respectively. The collection of 63,000 articles was subsequently shuffled, and split into 60% training, 20% dev/validation,and 20% testsets. The test set was stashed away for the final performance evaluation of the models, and the validation set was used for hyperparameter selection.

Comparison criteria of their results

MODEL	PRECISION	RECALL	F_1
Logistic Regression	0.96	0.49	0.65
Feedforward Network	0.89	0.74	0.80
RNN (Vanilla)	0.91	0.56	0.70
GRUs	0.89	**0.79**	**0.84**
LSTMs	0.93	0.72	0.81
BiLSTMs	0.88	0.75	0.81
CNN with Max Pooling	0.87	0.44	0.58
CNN with Max Pooling and Attention	**0.97**	0.03	0.06

Table 2: Model Performance on the Test Set

The precision numbers are high for all models because the dataset is (deliberately) skewed, with authentic news examples (i.e., negative examples) outnumbering the fake news examples by 4 to 1. Consequently, any model can deliver high accuracy simply by guessing every test example as negative. However, only the feedforward network and the RNNs with complex activation units achieved a recall greater than 0.7, indicating that the models were able to extract some subset of relevant features to classify more of the positive examples correctly.

Merits of the methods that authors have proposed

There were two major takeaways from these experiments. First, the RNN architecture with GRUs outperformed one with LSTM cells in the task at hand. This result holds despite the fact that a positive bias was added to the LSTM's forget gate. Not only did the RNN with GRUs achieve the better F1 score (and the best overall), it converged faster than the one using LSTM units. While the faster convergence time can be attributed to the fact that GRUs have fewer gates, and hence fewer computations to generate the output, the fact that the GRU exposes its full content without any control likely helped it perform better overall, given that examples in the data set were long news stories, averaging 515 words across both classes. Further, the bidirectional RNN model equipped with LSTMs didn't do significantly better than the unidirectional network, reporting almost the same F1 score.

Demerits and future scope

A complete, production-quality classifier will incorporate many different features beyond the vectors corresponding to the words in the text. For fake news detection, we can add as features the source of the news, including any associated URLs, the topic (e.g., science, politics, sports, etc.), publishing medium (blog, print, social media), country or geographic region of origin, publication year, as well as linguistic features are not exploited in this exercise—use of capitalization, fraction of words that are proper nouns (using gazetteers), and others.

Also believe that the idea of attention-enabled CNNs is a promising avenue worth exploring further. Finally, more sophisticated models—for example, pointer and highway networks—definitely merit investigation.

3. **Evaluating Machine Learning Algorithms for Fake News Detection**
By Shlok Gilda, 2017

Algorithms Used and how the modules work:

This paper explores the application of natural language processing techniques for the detection of 'fake news', that is, misleading news stories that come from non-reputable sources., author applied term frequency-inverse document frequency (TF-IDF) of bi-grams and probabilistic context free grammar (PCFG) detection to a corpus of about 11,000 articles. He then tested dataset on multiple classification algorithms -Support Vector Machines, Stochastic Gradient Descent, Gradient Boosting, Bounded Decision Trees, and Random Forests. He fount the TF-IDF of bi-grams fed into a Stochastic Gradient Descent model identifies non-credible sources with an accuracy of 77.2%, with PCFGs having slight effects on recall.

Data Sets used by Author

Author used a dataset obtained from Signal Media and a list of sources from OpenSources.co

Comparison criteria of their results

1. Baseline Models for Comparison: Naive and Random

As a baseline comparison for understanding the performance of our models, Author looked at two methods. First, a Naive Bayes model that predicts all majority class in this case, all articles are from reliable news sources. Second, a model that randomly selects a classification for each article as either reliable or unreliable based on the posterior probability of that class in the training set. These are the Naive and Random models, respectively. We detail their performance in Table 3.

Model	Accuracy	Precision	Recall
Naive	67.89%	54.22%	54.22%
Random	56.42%	32.18%	32.18%

TABLE 3: Naive and random model performance across metrics.

2. Combining PCFG and TF-IDF bi-gram features

Combining both feature sets, our models perform well above our baseline as seen in Table 4. Author noted that their best models tended to be Stochastic Gradient Descent (SGD) models, which, given that they tend to perform well with sparse and highly dimensional data, is not surprising. In particular, SGDs far outperform on precision while retaining a high recall, meaning that these models would work well both as identification of high priority articles in addition as 'fake news' filters.

Model	Area Under Curve	Precision	Recall	Accuracy
Bounded Decision Trees	65.9%	66.9%	37.9%	67.6%
Gradient Boosting	75.6%	40.2%	16.1%	65.7%
Random Forests	80.0%	84.2%	18.4%	64.8%
Stochastic Gradient Descent	87.5%	74.1%	71.7%	65.7%
Support Vector Machine	84.3%	80.9%	44.5%	73.6%
Baseline	-	32.18%	32.18%	67.89%

TABLE 4: Average model performance with both PCFG and TF-IDF bi-gram features at 0.7 score threshold for categorization.

3. TF-IDF Bigram Only Model Performance

Removing the PCFG features allows us to understand in more depth the value of those features in achieving these combined feature results. The results from this more limited feature run are displayed in Table 5.

The removal of PCFGs improves most of the metrics across our models. This is surprising, indicating that the PCFG features add little predictive value to the models. Indeed, the only noticeable decrease in performance is in our recall figures for Decision Trees and SGDs.

Model	Area Under Curve	Precision	Recall	Accuracy
Bounded Decision Trees	60.7%	58.5%	23.3%	66.1%
Gradient Boosting	79.4%	41.0%	22.3%	68.7%
Random Forests	78.8%	82.9%	25.3%	67.6%
Stochastic Gradient Descent	88.3%	88.8%	45.3%	77.2%
Support Vector Machine	85.6%	81.3%	48.1%	76.2%
Baseline	-	32.18%	32.18%	67.89%

TABLE 5: Average model performance with only TF-IDF bi-gram features at 0.7 score threshold for categorization.

4. PCFG Only Model Performance

The removal of TF-IDF bi-gram features allowed to isolate the predictive value of PCFGs for application. The results are displayed in Table 6.

Surprisingly, all of author's models give the same result. Diving into the individual predictions, he found that all models produce the same rank order of scores. He switched from a 0.70 threshold for classification to a top-k of 0.05 because the distribution of scores for these models have a particularly low mean with a tight range, such that determining an appropriate threshold for categorization was tedious and the 0.70 results weren't illuminating. All this goes to indicate that in the case of this classification task, PCFGs do not add a strong source of information for classification on their own.

Model	Area Under Curve	Precision	Recall	Accuracy
Bounded Decision Trees	50.0%	40.9%	10.8%	60.1%
Gradient Boosting	50.0%	40.9%	10.8%	60.1%
Random Forests	50.0%	40.9%	10.8%	60.1%
Stochastic Gradient Descent	50.0%	40.9%	10.8%	60.1%
Support Vector Machine	50.0%	40.9%	10.8%	60.1%
Baseline	-	32.18%	32.18%	67.89%

TABLE 6: Average model performance with only PCFG features, classifying the top 5% of scores as positive (k = 0.05).

Merits of the methods that authors have proposed

The results obtained above are very promising. This method demonstrates that term frequency is potentially predictive of fake news - an important first step toward using machine classification for identification. The best performing models by overall ROC AUC are Stochastic Gradient Descent models trained on the TF-IDF feature set only. We observe that PCFGs do not add much predictive value, but balance the Recall for our top performing model. This indicates that PCFGs are good for a Fake-News Filter type implementation versus, say, targeting fake news sites for review. TF-IDF shows promising potential predictive power, even when ignoring named entities, but we remain skeptical that this approach would be robust to changing news cycles. However, this would require a more complete corpus.

Demerits and Future scope

Despite the high performance of our classifier, there is definitely scope for improvement. Author evaluated models using absolute probability thresholds, which may not be the most reliable for models where probability scoring is not well-calibrated. While TF-IDF performs better, we are possibly overfitting to topics/terms important in the ongoing news cycle. Also, a vectorized approach makes it technically hard to see which individual features are most important, thus hampering analysis. These issues limit analysis and thus prevent broader generalizability. Author plan to address these issues in a future work.

METHODOLOGY:

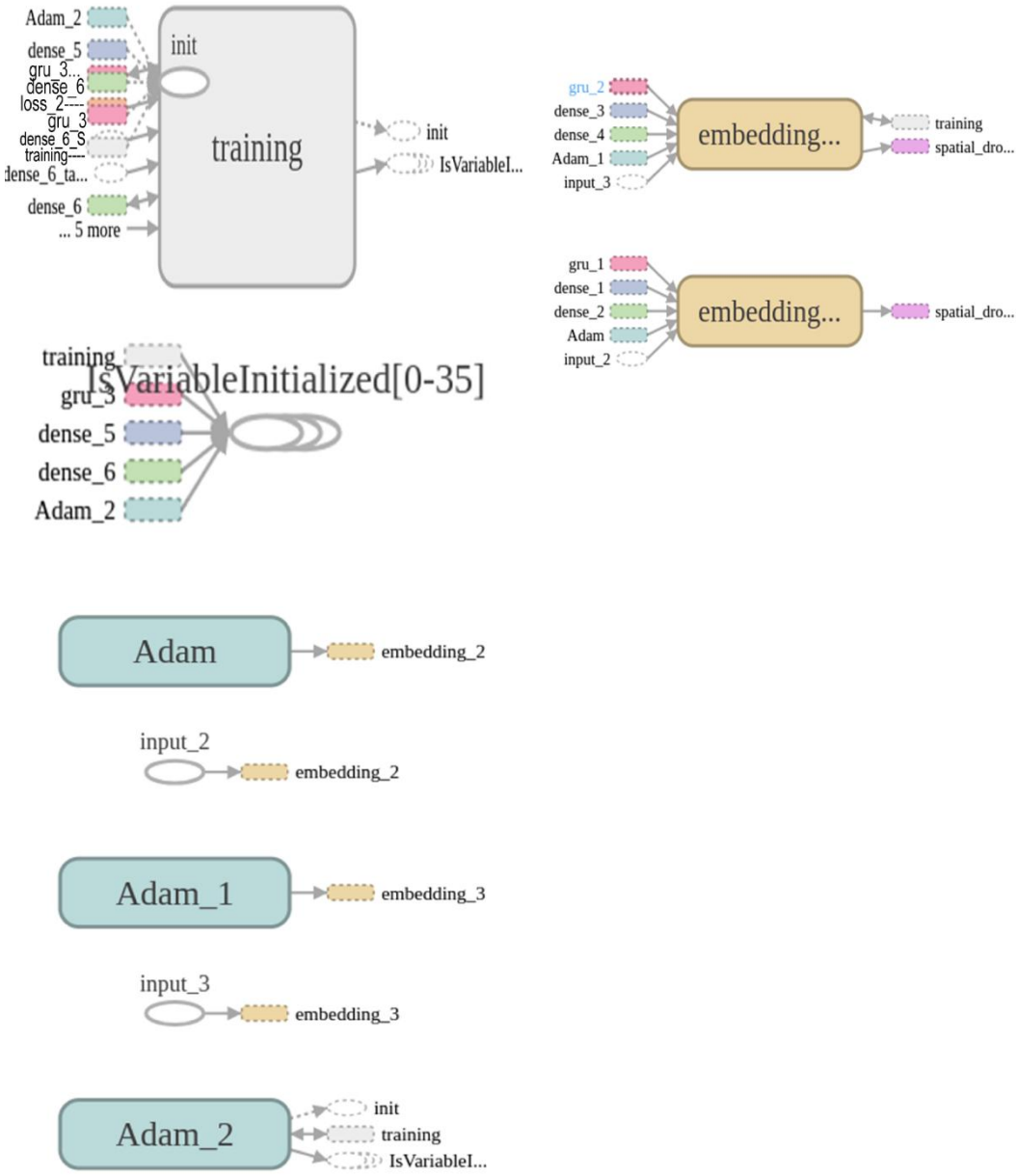


Figure 1. Neural Net Architecture using Tensorboard

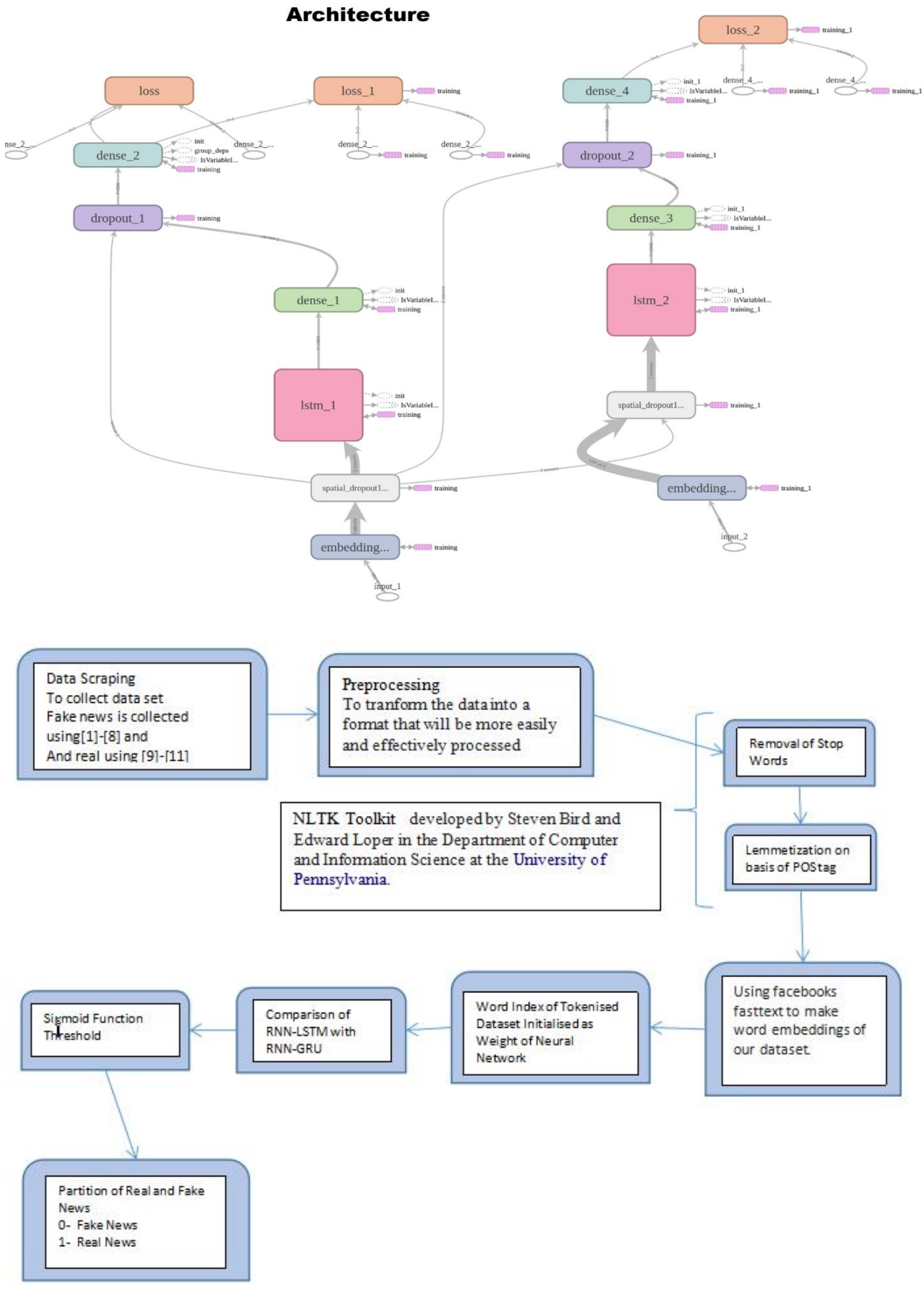


Figure 2: Modular Diagram

As shown in Figure 2. The dataset of news articles obtained needs to be made computationally acceptable to the algorithms and this is where the python's Natural Language Toolkit (NLTK) comes into play. To understand the data we make use of sentiment analysis which refers to the use of natural language processing, text analysis /classification , to systematically identify, extract, quantify, and study affective states and subjective information.

Textual data comprises of a lot of useless words which are not related to the context nor can they be of help in classyfing text contexts of the paragraph but helps us humans understand the proper meaning (words like "this", "that", "or" , "of", and "the"). These are called stop words. Our stopwords corpus consists of python's nltk library's stopwords, python's string library's punctuations and various other lists of such words gathered manually.

The news article dataset comprises of paragraphs where each paragraph is split into words (word tokenized) and iterated through to remove all such stop words. Then the words are lemmatized (stemmed as in 'happier' , 'happiness' and 'happy' are stemmed to their "origin" word 'happy'). But words like 'superb' and 'awesome' cannot be lemmatized to a common stem. So we computed the Part of Speech tag (noun, adjective , verb ,etc.) on the word and custom lemmatized them.

This took around 64 mins for 72,000 items.

The year, month, heading and the news article were then combined in a format to be fed into the artificial neural network.

We used fasttext to make word embeddings of our dataset.

FastText was proposed by Facebook in 2016. Instead of feeding individual words into the Neural Network, FastText breaks words into several n-grams (sub-words). For instance, the tri-grams for the word "delhi" is "del","elh" and "lhi" (ignoring the starting and ending of boundaries of words). The word embedding vector for apple will be the sum of all these n-grams. After training the Neural Network, we will have word embeddings for all the n-grams given the training dataset. Rare words can now be properly represented since it is highly likely that some of their n-grams also appears in other words.

We make use of keras' preprocessing's text's tokenizer. The dataset was tokenized (allows to vectorize a text corpus, by turning each text into either a sequence of integers (each integer being the index of a token in a dictionary) or into a vector where the coefficient for each token could be binary, based on word count, based on tf-idf...) and its word index (dictionary mappings of words (str) to their rank/index (int) which is only set after tokenizer was trained on the text) was used to make a matrix of embedded words (integer matrix) which is to be used as the default weights for the neural network.

The dataset was padded (given same length to all news articles) to be fed into the Recurrent Neural Network(RNN) with Long Short Term Memory (LSTM) units.

A recurrent neural network (RNN) is a class of artificial neural network where connections between nodes form a directed graph along a sequence. This allows it to exhibit temporal dynamic behavior for a time sequence. Unlike feedforward neural networks, RNNs can use their internal state (memory) to process sequences of inputs.

Gated Recurrent Units(GRU) and Long short-term memory (LSTM) units are units of a recurrent neural network (RNN). An RNN composed of LSTM units is often called an LSTM network. A

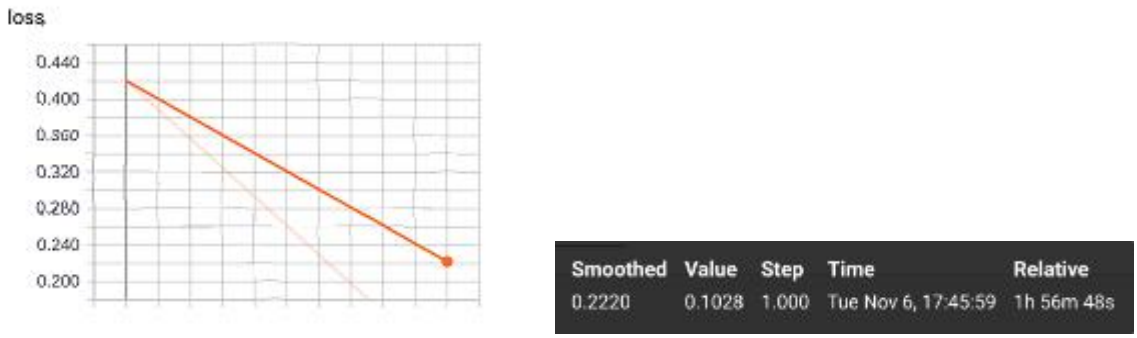
common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell. LSTM networks allow us to solve the problem of vanishing /exploding gradient prolem. Exploding gradient problem can be easily fixed by gradient clipping(limiting it). Vanishing gradient descent problem is faced when output is generated by passing input to a large number of hidden layers. The problem occurs because output of RNN layer is decided by back propagation on all its initial values resulting in multiple weight updates. So the "earliest" memory is "forgotten". This issue solved by LSTM and GRU units which make use of gates.

The LSTM-RNN consists of input layer (this layer takes the input), embedding layers (used for neural networks on text data. It requires that the input data be integer encoded, so that each word is represented by a unique integer), spatial dropout 1D layer (Dropout consists in randomly setting a fraction rate of input units to 0 at each update during training time, which helps prevent overfitting. 1D drops entire 1D feature maps instead of individual elements), lstm layer, dense layer (with activation relu), dropout layer and the output dense (sigmoid) layer with adam optimiser (Adam is an optimization algorithm that can used instead of the classical stochastic gradient descent procedure to update network weights iterative based in training data.)and binary cross-entropy loss(we have 2 classes, cross-entropy can be calculated as:

$$loss=-(y*log(p)+(1-y)*log(1-p))$$

where y is the true output and p is the predicted output)
We are using tensorboard to provide visualisations for the neural network.

Loss= 0.4206



Loss =0.1028

Figure 3. Binary Cross Entropy Loss Minimized Using Adam Optmiser

Results:

Software Specifications: Google Colaboratory

The project is made and trained on google colab service. After every 12hrs Disk, RAM, VRAM, CPU cache etc. data that is on our allotted virtual machine will get erased.

Hardware specifications :

GPU: 1xTesla K80 , having 2496 CUDA cores, compute 3.7, 12GB(11.439GB Usable) GDDR5 VRAM

CPU: 1xsingle core hyper threaded i.e(1 core, 2 threads) Xeon Processors @2.3Ghz (No Turbo Boost) , 45MB Cache

RAM: ~12.6 GB Available

Disk: ~33 GB Available

The data used for this project was drawn from different sources. Fake news articles were scraped from various fake news sites like Indian Satire . Authentic news articles were extracted from The Times of India and The Hindu . International news dataset is obtained from kaggle. Binary 0/1 labels were assigned to the real/fake news articles, respectively. This dataset is self accumulated from web scraping using beautiful soup in python. The dataset is specific to Indian news and may contain traces of International news also.

Comparisons

GRU vs LSTM

GRU

```
mode1.fit(train_seq_x, ytrain,epochs=4)

Epoch 1/4

14532/14532 [==============================]  995s 68ms/step - loss:  1.3480

Epoch 2/4

14532/14532 [==============================1  993s 68ms/step - loss:  0.5156

Epoch 3/4

14532/14532 [==============================]  9935 68ms/step - loss:  0.3387

Epoch 4/4

14532/14532 [==============================]   9945 68ms/step - loss:  0.2092
```

<keras.cal1backs.History at 0x7fa236e69240>

(Time – 1000s per epoch)

confusion_matrix

classification_report

accuracy_score

[[2852 214]

[290 2873]]

	Precision	recall	fl-score	support
0	0.91	0.93	0.92	3066
1	0.93	0.91	0.92	3163
avg / total	0.92	0.92	0.92	6229

0.9190881361374217 Accuracy – 92%

LSTM

Epoch 1/4

14532/14532 [==============================] - 12915 89ms/step loss: 0.3117

Epoch 2/4

14532/14532 [==============================]- 1292s 89ms/step loss: 0.0722

Epoch 3/4

14532/14532 [==============================]- 12925 89ms/step loss: 0.0246

Epoch 4/4

14532/14532 [==============================]- 12945 89ms/step loss: 0.0112

(Time — 1300 s per epoch)

confusion_matrix

classification_report

accuracy_score

[[2844 222]

[130 3033]]

	precision	recall	fl-score	support
0	0.96	0.93	0.94	3066
1	0.93	0.96	0.95	3163
avg / total	0.94	0.94	0.94	6229

0.9434901268261359

Accuracy – 94 %

RNN with LSTM units trained on train:test ratio of 70:30

Epoch 1/1

50104/50104 [==============================] - 4884s 97ms/step - loss: 0.1165

<keras.ca11backs History at 0x7f43e83ed128>

confusion_matrix

classification_report

accuracy_score

[[5784 142]

[63 15485]]

	Precision	recall	fl-score	support
0	0.99	0.98	0.98	5926
1	0.99	1.00	0.99	15548
avg / total	0.99	0.99	0.99	21474

0.9904535717611996

Now, the neural net is trained on all entries and neural net is saved to be deployed. To check the working of algorithm all entries are predicted and classification seems to be working fine.

```
model fit(train_seq_x, Y,epochs=2,callbacks=[tensorboard])
Epoch 1/2
71578/71578 [==============================] - 7058s 99ms/step - loss: 0.4206
Epoch 2/2
71578/71578 [==============================] - 7009s 98ms/step - loss: 0.1028
<keras.callbacks.History at 0x7f43c8ba8710>
```

```
confusion_matrix
classification_report
metrics.accuracy_score
```

```
[[19540  71]
[ 327     51640]]
                precision  recall  fl-score  support
        0   0.98     1.00   0.99    19611
        1   1.00     0.99   1.00    51967
avg / total  0.99     0.99   0.99    71578
0.9944396322892509
```

Initially test cases are:

heading	article	year	month	value	text
Customs officials bust drug racket	Seize 12 kg of methaqualone at international a...	2018	4	1	Customs officials bust drug racket Seize 12 kg...
PCB mulls over legalizing match fixing, to off...	Lahore, Pakistan. After realizing that its pla...	2010	8	0	PCB mulls over legalizing match fixing, to off...

Extended list of stop words

```
fsw=open('stopwordsultimate.txt','r')
stop_words=fsw.read().split('\n')
print(len(stop_words))
#stop_words
stops=stopwords.words('english')
punctuations=list(string.punctuation)
stops.extend(punctuations)
stops.extend(stop_words)
print(len(stops))
stops
```

```
1565
1776
```

Figure 4.

In Fig. 4 Punctutations are added to the list of stop words .

Word tokenizing

```
documents=[]
for doc in testing2['text']:
    txt=word_tokenize(doc)
    documents.append(txt)
```

Figure 5

In Figure 5 both text documents are word tokenized , that is it is converted to an array of words.

Output For 1st article:

['Customs',
'officials',
'bust',
'drug',

'racket',
'Seize',
'12',
'kg',
'of',
'methaqualone',
'at',
'international',
…….……']

Output for 2nd article:
['PCB',
'mulls',
'over',
'legalizing',
'match',
'fixing',
',',
'to',
'offer',
'it',
'as',
'service',
'Lahore',
……..']

Lemmatization and stop word removal

```
lemmatizer=WordNetLemmatizer()

def simple_pos(tag):
    if tag.startswith('N'):
        return wordnet.NOUN
    elif tag.startswith('V'):
        return wordnet.VERB
    elif tag.startswith('J'):
        return wordnet.ADJ
    elif tag.startswith('R'):
        return wordnet.ADV
    else:
        return wordnet.NOUN
```

```
def clean_review(words):
    #print(codeworker)
    op=[]
    for w in words:
        if w not in stop_words:
            p=pos_tag([w])
            cw=lemmatizer.lemmatize(w,pos=simple_pos(p[0][1]))
            op.append(cw)
    #codeworker+=1
    return op
```

```
import time
start=time.time()
documents=[(clean_review(doc)) for doc in documents]
end=time.time()
print(end-start)
```

```
0.2568540573120117
```

Figure 6

In Figure 6 For each word in document if it is not a stop word then that words tag is taken from pos_tag and word is lemmatized. Then, this collection of words is appended to documents.

Output for 1st article
['Customs',
 'official',
 'bust',
 'drug',
 'racket',
 'Seize',
 '12',
 'kg',
 'methaqualone',
 'international',
 ……]

Output for 2st article
['PCB',
 'mull',
 'legalize',
 'match',
 'fix',
 'offer',
 'service',
 'Lahore',
 'Pakistan',
 'After',
 ……]

Predicting

```
start=time.time()
preds=loaded_model.predict(valid_seq_x)
end=time.time()
print(end-start)
```

```
8.707987785339355
```

```
preds
```

```
array([[0.79674107],
       [0.00576266]], dtype=float32)
```

```
preds3=np.zeros(len(preds))
for i in range(len(preds)):
  if preds[i]>0.5:
    preds3[i]=1
  else:
    preds3[i]=0
#preds3
#print(sum(ytrue==preds3))
print(preds3)
```

```
[1. 0.]
```

Figure 7

In Figure 7 For both preprocessed testing data result is predicted.

Test Cases:

Test Case Variables	Test Data Values	Output	How critical is the Test Case
Customs officials bust drug racket Seize 12 kg of methaqualone at international airport Customs officials at Kempegowda International Airport seized 12.9 kg of the banned drug methaqualone on Wednesday. Its value in the international market is around Rs. 6.45 crore. According to Customs officials, the consignment was to be shipped to Kuala Lumpur in Malaysia by an export firm based in Chennai, Tamil Nadu. The goods were declared as 'palm sugar' for personal use, with a declared value of Rs. 6,000. Methaqualone is a sedative and hypnotic medication. Officials believe that the consignment is part of an international drug racket. Drugs concealed in packets of palm sugarThe drugs were packed in plastic packets, which were inserted in larger packets of palm sugar. Officials of the Courier Section of Air Cargo detected the drugs while assessing the consignment. Acting on a hunch that something was not quite right with the consignment, they opened a few packets and found smaller pouches within filled with a white crystalline powder, which was later identified as methaqualone. The consignment was booked by the exporter through a courier service in Chennai and sent to the Bengaluru hub for shipment to Kuala Lumpur. In the first week of January 2018, ketamine worth over Rs. 1 crore, hidden in sweet packages, was seized. That consignment, too, was marked for Kuala Lumpur, by an exporter in Chennai. “There are glaring similarities in the two cases. In both cases, the courier company is the same. The Customs Intelligence Unit of Bangalore Air Cargo is investigating the case,” an official said.	1	1	4
PCB mulls over legalizing match fixing, to offer it as service “It’s ancient wisdom that if you can’t fight them, join them.” said Butt, justifying his decision to officially get into match-fixing. “Furthermore, there was always a risk that BCCI or ICC could legalize it before we do it.” he added, arguing that the step could bring in the much needed cash for the cash strapped board.As part of CaaS, an individual or organization would now have the option of dialing in an official representative of PCB for services like No-Ball-On-Demand, Wide-Ball-On-Demand, Wicket-On-Demand, Ball-Tampering-On-Demand, Injury-On-Demand, etc. An online bidding system would be in place and the service would be offered to	0	0	5

the party offering largest amount for a particular event.“A wealthy fan, a rival cricket board, or any other interested party can avail of these services.” Butt informed, adding, “Such services would be cheaper than fixing the whole match as an interested party can try out with fixing a few balls or events to see how the match progresses.”“It thus retains the glorious uncertainties of the game as you never know who would fix the next ball.” he pointed out.“Anyone can participate; say a wealthy Sehwag fan can buy a No-Ball-On-Demand to deny a Sri Lankan batsman a century even though he might not have any interest in fixing the whole match.” Butt further elaborated upon the benefits of CaaS and how it could make cricket matches even more interesting.The PCB Chairman claimed that there was nothing illegal in such an arrangement and it was as innovative as IPL with a much higher earning potential, although he agreed that things would get complicated if and when other boards also starts offering Cricket-On-Demand, just as there is confusion over which player represents which team in Champions League T20.“We could have a situation where we have sold a No-Ball-On-Demand and the rival board had sold Wicket-On-Demand on the same ball. A middle way is that the batsman gets run-out, but I guess a clearing house for such conflicting demands will have to be set up by the ICC in future.” Butt said.Cricket lovers in Pakistan, who are hell irritated with recurring charges of match-fixing, had mixed reactions over the announcement, but many of them welcomed the decision.“I am happy. After giving the world first computer virus, it’s so nice that we are back to our innovative ways.” said Basit Alvi, a cricket lover and SaaS provider, who also offered to set up an online bidding platform for PCB to offer CaaS.(based on inputs by special correspondent Vinod Patil)			

Conclusion and Future Work:

In accordance with the observations, the RNN architecture with LSTM outperformed one with GRU. In future we can automate this model . We can also collect news tested on the model and retrain our model after a fixed duration of time. We can make the algorithm real time and host it on a server. Shift the algorithm from keras (running tensorflow backend made in C++)) to pytorch (made on torch library of lua).

References:

[1]. Faking News http://www.fakingnews.com/
[2]. The Indian Satire http://www.theindiasatire.com/
[3]. Teekhi Mirchi http://teekhimirchi.in/
[4]. The Unreal Times http://www.theunrealtimes.com/
[5]. The Spoof https://www.thespoof.com/
[6]. Fun Funny Khez http://www.funfunnykhez.com/
[7]. Dapaan http://dapaan.com/
[8]. Farzi News http://farzinews.com/
[9]. The Hindu https://www.thehindu.com/
[10]. The Times of India https://timesofindia.indiatimes.com/
[11]. Rising Kashmir http://www.risingkashmir.com/
[12]. https://machinelearningmastery.com/
[13]. CSI: A Hybrid Deep Model for Fake News Detection by Natali Ruchansky, Yan Liu and Sungyong Seo.
[14]. Fake News Detection with Deep Diffusive Network Model by Jiawei Zhang,Limeng Cui , Yanjie Fu , Fisher B. Gouza.
[15]. "The Pope Has a New Baby!" Fake News Detection Using Deep Learning by Samir Bajaj.
[16]. A Non-Genuine Message Detection Method based on Unstructured Datasets by Marcello Trovati, Richard Hill and Nik Bessis.
[17]. Evaluating Machine Learning Algorithms for Fake News Detection by Shlok Gilda.
[18]. 3HAN: A Deep Neural Network for Fake News Detection by Sneha Singhania, Nigel Fernandez, and Shrisha Rao.