Directed Research Project

Fake News Detection Using BERT

Group Members:

Mohammad Mohtashim Khan (21110290)

Muhammad Abdullah (21110246)

Rehan Waseem (21110294)

Mahrukh Khattak (21110083)

Mahnoor Malik (21110137)

Supervisor: Dr. Salma Zaman

Total Credit Hours: 2

Abstract:

In this paper, we first replicated the performance of an existing paper in which different classical machine algorithms' performance were analysed on the classification task of real and fake news. We created the same pipeline and replicated the procedure from start to end as proposed by the paper. However, then we analysed the state-of-the-art BERT which is the latest Machine Learning algorithm in the field of Natural Language Processing devised by google. BERT outperformed all the classical Machine Learning Algorithms and achieved outstanding accuracy. In the end, we concluded that BERT can be easily fined tuned to the task of classifying real and fake news.

Literature Review:

Social media spreads false news faster than traditional media sources. Anyone who has access to social media can initiate the process of spreading false news. Additionally, as compared to traditional forms of media, it is harder to track who initiated the spread of fake news on social media websites. Most of the news posts are intentionally designed to influence readers' opinions and ideas while others are hypothetical reports that are assumed to be true and released without confirmation. Therefore, an increasing number of researchers have designed different resources for the detection of fake news. Most of these sources rely on algorithms and artificial intelligence for the detection of fake news articles. Mykhailo Granik and Volodymyr Mesyura used Baye's classifier in their research. They used algorithm based on artificial intelligence for the detection of fake news on social media (Granik et al 2017). Their model was tested on Facebook news posts and had achieved 74% accuracy. Bayesian classification is a data mining technique based on Baye's theorem. It is a family of algorithms that is widely used in spam detection based on machine learning. Work on fake news detection has also been in the field of social website such as by Shu, Wang and Liu who constructed real-world datasets measuring users trust level on fake news and selected representative groups of both 'experienced' users who were able to recognize fake news items as false and 'naïve' users who were more likely to believe fake news and then performed a comparative analysis over explicit and implicit profile features between these user groups which revealed their potential to differentiate fake news(Shu et al. 2016).

Recently, many researchers have designed algorithms that determine the credibility of news on the basis of the identity of the users "liking" or "sharing" them. Eugenio Tacchini, Gabriele Ballarin, Marco L. Della

Vedova, Stefano Moret and Luca de Alfaro presented two classification techniques for classifying hoax posts and non-hoax posts on Facebook with more than 99% accuracy. One of the methods is based on logistic regression and the other on harmonic Boolean crowdsourcing algorithms. In Boolean label crowdsourcing, individuals are asked to determine whether a post is true or false and the results are used as input for the algorithm. Their objective was to design an algorithm that automatically detect fake posts on Facebook based on the users who "like" them, as fast as possible (Tacchini et al 2017). Many recent research papers have also focused on designing algorithms that can differentiate between satire and humour regarding a news and the true news itself. Victoria L. Rubin, Nial J. Conroy, Yimin Chen and Sarah Cornwell have published a paper on SVM based algorithm that can differentiate between satire and true news on a subject with 90% precision and 84% recall. Their algorithm is a combination of Natural Language Processing (NLP) and machine learning techniques and detect "language patterns, topicality, sentiment, rhetorical devices and word occurrences" (Rubin et al 2016)

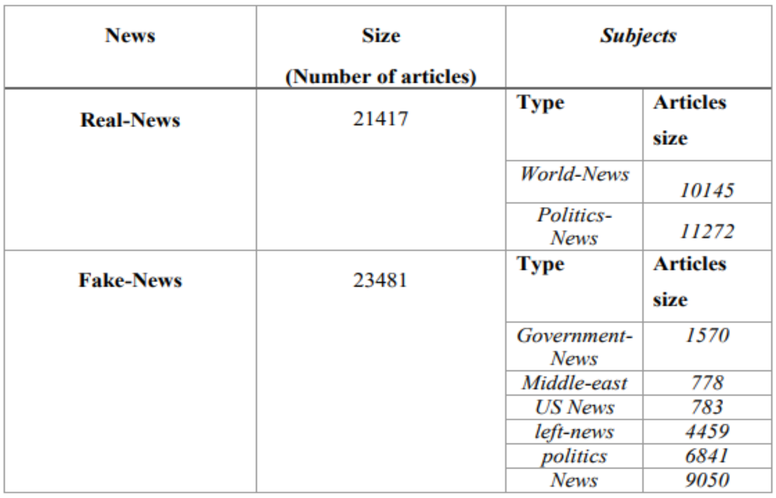
Main Objective:

The main purpose of our paper is to test different statistical and machine learning techniques to solve the challenging task of detecting whether a given piece of news is fake or not. As we all know that in today's modern world, fake news can spread more quickly than real news which can potentially have long-term detrimental consequences as internet users might be made biased or might be subscribed to wrong point of view. Consequently, devising such an algorithm which can differentiate between fake and real news becomes all more important to avoid such harms. Consequently, we came up with the idea of replicating and beating the accuracy of the following paper on the same topic: "*Ahmed H, Traore I, Saad S. (2017) "Detection of Online Fake News Using N-Gram Analysis and Machine Learning Techniques. In: Traore I., Woungang I., Awad A. (eds) Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments. ISDDC 2017. Lecture Notes in Computer Science, vol 10618. Springer, Cham (pp. 127-138).*"

We would essentially first try to replicate the same procedure as the paper and then would reflect how the current-state-of -art NLP algorithm called BERT is able outperform the classical techniques proposed by the paper. This would reflect how much NLP algorithms have improved over the time while also showing how state-of the-art BERT can solve this challenging task.

Dataset-Used:

The data we used comes from Kaggle.com. The author of the paper (which we replicated) are the creator of this dataset and they then made the dataset public on Kaggle. This is a well-crafted dataset with two csv files. One for fake news and other one for real news. The authors of this paper crafted this dataset by crawling truthful articles from Reuters.com and fake news articles were collected from different sources such as unreliable website that were flagged by PolitiFact and Wikipedia (Ahmed Et al.). The following table describes the dataset which we used:



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

Table-1(a)

Replication of the Existing Paper

As described earlier, the basic purpose of our paper is to replicate the method of fake news detection proposed in *"Ahmed H, Traore I, Saad S. (2017) "Detection of Online Fake News Using N-Gram Analysis and Machine Learning Techniques"* and try to surpass its accuracy by introducing some improvements.

This paper proposes a fake news detection model using n-gram analysis and machine learning techniques. N-gram analysis is basically an approach for feature identification used in language modelling and Natural language processing fields. The paper uses six different supervised classification techniques for text analysis. These techniques, namely, are as follows:

- K-Nearest Neighbour (KNN)
- Support Vector Machine (SVM)
- Logistic Regression (LR)
- Linear Support Vector Machine (LSVM)
- Decision Tree (DT)
- Stochastic Gradient Descent (SGD)

Before passing the data through the n-gram model, the data is subjected through several refinements, such as stop-word removal (removing insignificant words), stemming (changing the words into their original form), etc. After the refinement is complete, the data is subjected to features extraction using two techniques: Term Frequency (TF), and Term Frequency-Inverted Document Frequency (TF-IDF). TF is a technique that uses the counts of different words to identify the similarities between the documents. Whereas, TF-IDF is used to weigh or measure the importance of several terms or words appearing in a document. The terms that appear more frequently are weighed less than the terms that appear less frequently. Moreover, the paper also only focusses on the sub-sample of data in which author of paper essentially filter the data and ran their algorithm only on 2016 US election News Articles with

1000 fake articles and 1000 real articles. We also filtered in the same way and took the same sub-sample before feeding the data into the algorithm.

Next, the data is passed through six classifiers (as mentioned above), separately, to classify each set of news either as 'fake', or 'truthful'. The whole process can be summarized as follows:

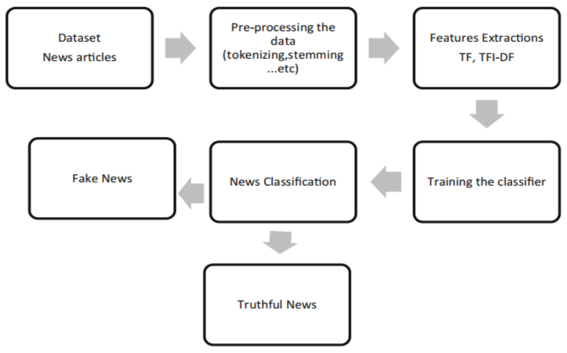


Fig.1

Now, let's have a look at the accuracy of results when we passed the dataset trough the same six classifiers. Each of the following table is output of our own code and also a replication of the original paper

Logistic Regression (LR):

LR is a machine learning technique (originally borrowed from statistics) for the binary classification of the problems. It predicts the probability that a given data entry belongs to a category either labelled as '1' or '0' using regression.

• **N-Gram Size**	**TF-IDF**				**TF**			
	1000	**5000**	**10,000**	**50,000**	**1000**	**5000**	**10,000**	**50,000**
Uni-gram	96.0	95.0	95.0	96.0	99.0	98.0	99.0	98.0
Bi-gram	94.0	94.0	95.0	93.0	96.0	96.0	97.0	97.0
Tri-gram	92.0	92.0	91.0	87.0	91.0	94.0	93.0	93.0
Four-gram	81.0	89.0	89.0	75.0	83.0	86.0	89.0	90.0

Table 1: Logistic Regression Accuracy Results. Accuracy values are in %.

Linear Support Vector Machine (LSVM):

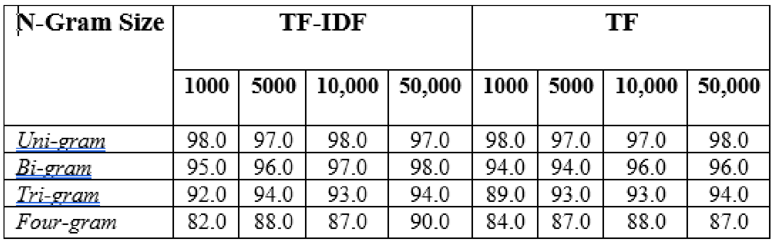
LSVM is similar to SVM, the difference being that in LSVM the hyperplane (or dividing line) identified to divide the dataset into two categories in linear in nature.

N-Gram Size	**TF-IDF**				**TF**			
	1000	**5000**	**10,000**	**50,000**	**1000**	**5000**	**10,000**	**50,000**
Uni-gram	98.0	98.0	97.0	97.0	99.0	99.0	99.0	99.0
Bi-gram	95.0	97.0	97.0	97.0	93.0	96.0	97.0	97.0
Tri-gram	89.0	94.0	93.0	95.0	90.0	91.0	92.0	93.0
Four-gram	84.0	88.0	90.0	92.0	83.0	86.0	88.0	88.0

Table 2: Linear SVC Accuracy Results. Accuracy values are in %.

Stochastic Gradient Descent (SGD):

SGD is an optimization technique which works to minimize an objective function in this case hinge loss. SGD at each iteration takes one sample from the dataset and then calculates feature's weight of this sample with respect to loss. This gradient is multiplied by a constant called learning rate (which is specified as hyper-parameter). Finally, gradient value multiplied by learning rate is subtracted from current feature weight's value. SGD then repeat this process of sampling and gradient calculation until the cost has converged or it has reached a maximum number of iterations specified by the user as hyperparameter.

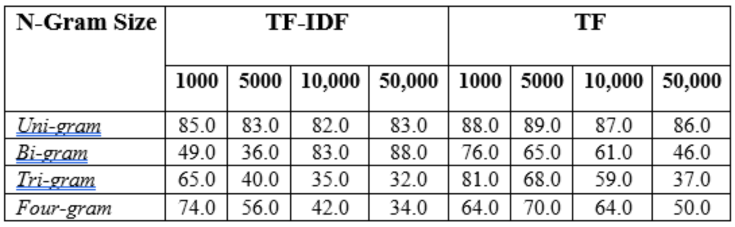


N-Gram Size	TF-IDF				TF			
	1000	5000	10,000	50,000	1000	5000	10,000	50,000
Uni-gram	98.0	97.0	98.0	97.0	98.0	97.0	97.0	98.0
Bi-gram	95.0	96.0	97.0	98.0	94.0	94.0	96.0	96.0
Tri-gram	92.0	94.0	93.0	94.0	89.0	93.0	93.0	94.0
Four-gram	82.0	88.0	87.0	90.0	84.0	87.0	88.0	87.0

Table 3: SGD Classifier Accuracy Results. Accuracy values are in %.

K- Nearest Neighbour (KNN):

KNN is a learning algorithm that uses the database in which the data-points are separated into several classes to predict the classification of new data entry. It is a non-parametric supervised method in which new point are allocated to the desired cluster through kth majority voting with Euclidean distance as the metric.

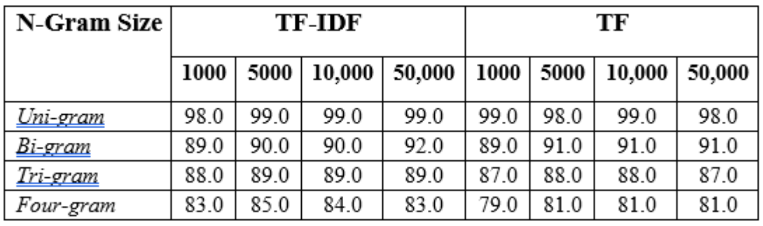


N-Gram Size	TF-IDF				TF			
	1000	5000	10,000	50,000	1000	5000	10,000	50,000
Uni-gram	85.0	83.0	82.0	83.0	88.0	89.0	87.0	86.0
Bi-gram	49.0	36.0	83.0	88.0	76.0	65.0	61.0	46.0
Tri-gram	65.0	40.0	35.0	32.0	81.0	68.0	59.0	37.0
Four-gram	74.0	56.0	42.0	34.0	64.0	70.0	64.0	50.0

Table 4: K Neighbours Classifier Accuracy Results. Accuracy values are in %.

Decision Tree classifier (DT):

DT is a supervised machine learning algorithm that splits the dataset continuously according to several certain parameters to classify it into a specific category. It learns how different features can be used to split the dataset at different levels such that separation is most efficient.



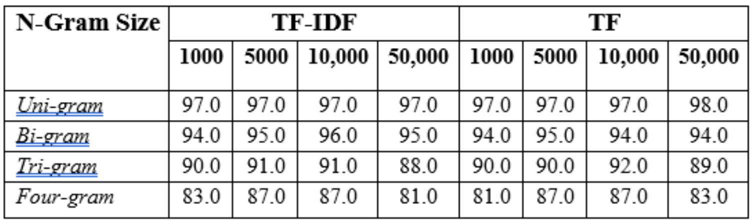
N-Gram Size	TF-IDF				TF			
	1000	5000	10,000	50,000	1000	5000	10,000	50,000
Uni-gram	98.0	99.0	99.0	99.0	99.0	98.0	99.0	98.0
Bi-gram	89.0	90.0	90.0	92.0	89.0	91.0	91.0	91.0
Tri-gram	88.0	89.0	89.0	89.0	87.0	88.0	88.0	87.0
Four-gram	83.0	85.0	84.0	83.0	79.0	81.0	81.0	81.0

When a new data is entered, it tests it against those parameters one-by-one to identify its category.

Table 5: Decision Tree Classifier Accuracy Results. Accuracy values are in %

Support Vector Machine (SVM):

SVM is a supervised machine learning algorithm used for the classification of datasets. It works on the idea of finding a hyperplane (or a dividing line) that best divides the sets into two classes. It works by identifying whether a given data entry belongs to one side of the hyperplane or the other.



N-Gram Size	TF-IDF				TF			
	1000	5000	10,000	50,000	1000	5000	10,000	50,000
Uni-gram	97.0	97.0	97.0	97.0	97.0	97.0	97.0	98.0
Bi-gram	94.0	95.0	96.0	95.0	94.0	95.0	94.0	94.0
Tri-gram	90.0	91.0	91.0	88.0	90.0	90.0	92.0	89.0
Four-gram	83.0	87.0	87.0	81.0	81.0	87.0	87.0	83.0

Table 6: SVM Accuracy Results. Accuracy values are in %.

BERT (Bidirectional Encoder Representations from Transformers)-Beating the Paper:

Having replicated the paper, we decided to use state of the art of NLP model named BERT to solve the given task in order to see how BERT's performance compares to the traditional machine learning model used by the author of the paper which we replicated in the previous section. BERT was developed by researchers at Google in 2019 as a tool to improve the state-of-the-art performance of NLP tasks. **Please note that the whole dataset was used for this algorithm rather any sort of sub-sample from the data.**

Now, we will explain the architecture of the BERT and then we would go into the details of how it performs on our task. BERT is an improvement over the traditional Recurrent Neural Network architecture which tended to work in a sequential manner as shown below:

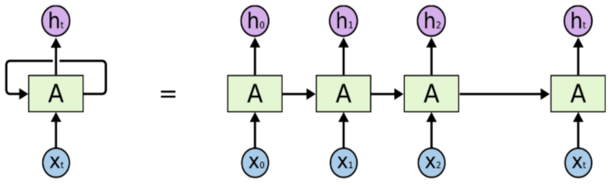


Fig.2-from Using LSTM to Implement a Sector Rotation Trading Strategy Part I by Christopher

As, shown in the traditional RNN model, the State A(t) is dependent on A(t-1) which mean that with long sentence when encoded into long token feature vector , would cause time dimension of RNN to become very large which in turn not only causes , the RNN not able to capture relationship between words which are really far apart but also causes the problem of gradient vanishing/explosion during backpropagation. Aligning the positions to steps in computation time, they generate a sequence of hidden states ht, as a function of the previous hidden state ht−1 and the input for position t. This inherently sequential nature precludes parallelization within training examples, which becomes critical at longer sequence lengths, as memory constraints limit batching across examples (Vaswani and et al 2017).

Back-Bone of Bert-TRANSFORMER ARCHITECTURE:

Therefore, in order to overcome this long-term time dependency problem, Google Brain Team in 2017 proposed ‘Transformers’ Model. It is imperative for us to first understand the transformed model as BERT is just several layers of Transformers stacked together. The **Transformer Model** followed the architecture as follow:

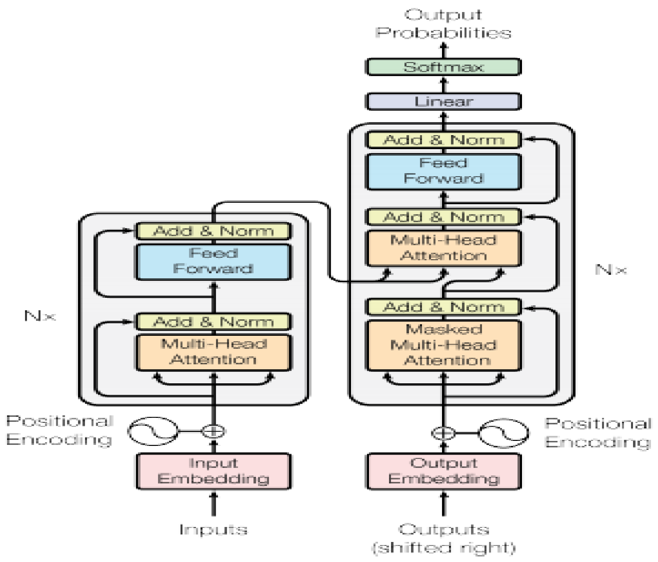
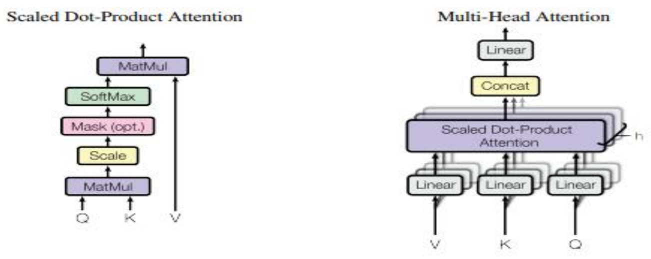


Fig.3 from Attention is all you need by Vaswani et al.

Encoder:

The Transformer has two main parts: Encoder (Left Rectangle-Fig.3) and Decoder (Right Rectangle-Fig.3). The encoder part receives tokenize sentence as input and then uses multi-head attention (See Fig.3(b)) to compute a matrix representing the embedding of the sentence.

Fig.3(a) **fig.3(b)**



-from Attention is all you need by Vaswani et al.

The Multi-Head Attention is concatenation of output of h (any scalar) scaled-dot product attention layers (See Fig.3(a)). Therefore, let us understand scaled-dot product attention. Let us suppose you have **N words** in the given sentence. You would embed each sentence into a feature vector of any size(this is only necessary for the first multi-head attention layer of Transformer), then you would form three matrices of sizes: V-Value(Nx64), K-Key(Nx64) and Q(Nx64).The reason of using matrix is to make the computation parallel for whole sentence rather than iteratively computing attention for each word. The scaled dot product attention using these matrices would compute the Attention Matrix for the complete sentence (Equation Below):

where dk is scalar value of 64

$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

fig.4- from Attention is all you need by Vaswani et al.

The final output of this dot-product is a matrix z with size of Nx64(this is the attention matrix for the sentence). Now, multi-head attention would have h layers of scaled-dot product attention layers and then it will concatenate the final output of all the h layers of scaled-dot product giving us the matrix A of Nx(64*h)-remember each scaled dot product attention gives z matrix with size Nx64 consequently h such matrixes are concatenate together. This matrix is then multiplied by an additional matrix Wo of size ((64*h) x64). After, this multiplication the final output of matrix Z is of size Nx64-this matrix is essentially capturing the embedding of the sentence. The multi-head attention process is summarised as below:

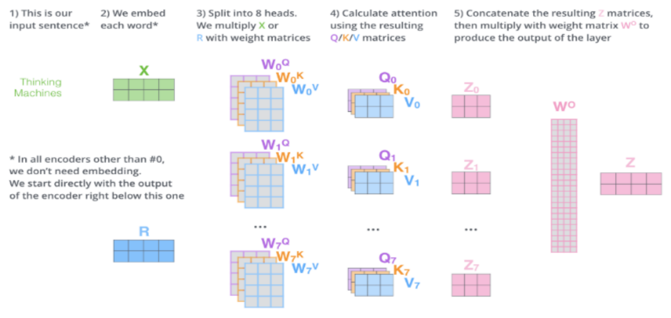

1) This is our input sentence*
2) We embed each word*
3) Split into 8 heads. We multiply X or R with weight matrices
4) Calculate attention using the resulting Q/K/V matrices
5) Concatenate the resulting Z matrices, then multiply with weight matrix W^O to produce the output of the layer
Thinking Machines
X
R
* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one
W0Q W0K W0V
W1Q W1K W1V
W7Q W7K W7V
Q0 K0 V0
Q1 K1 V1
Q7 K7 V7
Z0
Z1
Z7
WO
Z

Fig.5- from The Illustrated Transformer by Alammar

This final Matrix Z is then fed into a fully connected Neural Network. Now, the output of Fully Connected Neural Network is Normalised and then this output is fed into decoder (right rectangle-See Fig.2).

Decoder:

As shown in the Fig.3, decoder is fed the same sentence but **shifted towards rights** as this helps to capture the semantic meaning of sentence better and help to understand the embedding in a better way. The input of decoder is then fed into a Masked-Multi-Head Attention Layer which is essentially same as Multi-Head Attention(Explained above) but some words are masked/hidden, as explained by original authors of the paper, "We also modify the self-attention sub-layer in the decoder stack to prevent positions from attending to subsequent positions. This masking, combined with fact that the output embeddings are offset by one position, ensures that the predictions for position i can depend only on the known outputs at positions less than i." (Vaswani et al).

The output of Masked-Multi Head Attention is fed into another multi-head attention layer, but this multi-head attention layer is also given **the output of encoder**. Therefore, this multi-head attention layer does the same processing but now onto both the output of encoder layer and the current input of decoder layer. Finally, this output is fed into another fully connected neural network which generates a matrix of Nx512 where N is the number of words in the sentence.

Also just another additional detail to note in Transformed architecture, both encoder input and decoder output are first feed into Positional Encoding Layer (see the fig.2), the rationale is explained by the original authors, "Since our model contains no recurrence and no convolution, in order for the model to make use of the order of the sequence, we must inject some information about the relative or absolute position of the tokens in the sequence. To this end we add 'positional encodings' to the input embeddings at the bottoms of the encoder and decoder stacks. The positional encodings have the same dimension model as the embeddings, so that the two can be summed." (Vaswani et al)

BERT-STACKS of TRANSFORMERS:

It was important to understand TRANSFORMERS because BERT is just stack of transformers. However, the google team got rid of decoder part of the Transformer and stacked only encoder parts onto the top of each other. So essentially what they did was to follow the below architecture

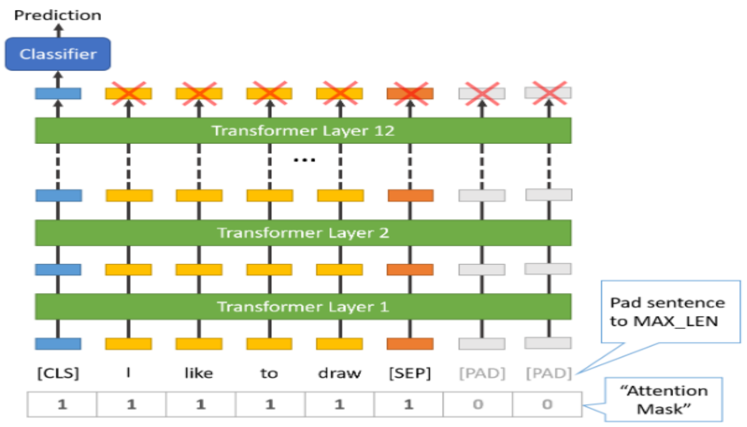


Fig.6

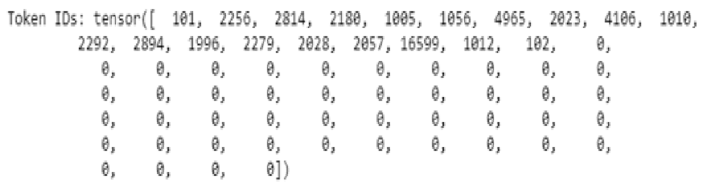
Each **Transformer Layer only has the encoder part**. The BERT we used had 12 such Transforms Layers. As you can see that BERT would calculate embedding of each word individually and in parallel therefore getting rid of the problem of Time-dependency of RNN.

What we did?

We took a pre-trained BERT of 12 Layers which was trained by the authors on both unsupervised and supervised NLP tasks. The original author of BERT states, "During pre-training, the model is trained on unlabelled data over different pre-training tasks. For finetuning, the BERT model is first initialized with the pre-trained parameters, and all of the parameters are fine-tuned using labelled data from the downstream tasks. Each downstream task has separate fine-tuned models, even though they are initialized with the same pre-trained parameters".(Devlin et al) The

BERT was trained on different tasks by original authors with a lot of data therefore the weights of BERT is essentially designed to work on different NLP tasks.

Therefore, we took a 12-Layer Pretrained BERT model and used BERT tokens to encode our news classification data (since BERT has its own special tokens). Plus, we had padded our encoded sentence to the maximum length of the sentence in whole corpus (since size of each input must be fixed-see fig.6). So essentially, each instance fed into BERT was as follow, where each integer is a token generated by Bert tokens:

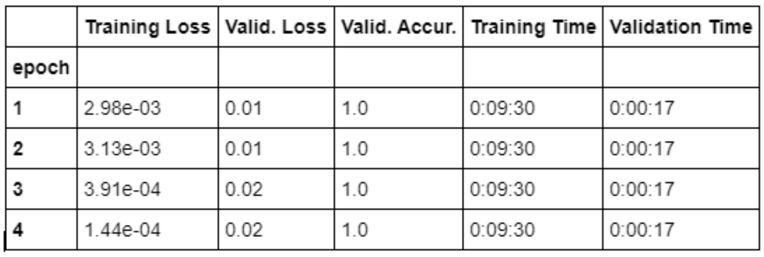


```
Token IDs: tensor([  101,  2256,  2814,  2180,  1005,  1056,  4965,  2023,  4106,  1010,
          2292,  2894,  1996,  2279,  2028,  2057, 16599,  1012,   102,     0,
             0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0,     0])
```

Fig.7

Then this tokenize sentence was feed into the BERT Architecture as in fig.6. As you can see, in fig.6 that each sentence was also given a special <CLS> token to indicate BERT that it is a classification task. Also the fig.6 also shows attention mask which is essentially used by BERT to differentiate between original tokens and padded zeros .Plus, you can see that we also get rid of output of final embedding for each word (red crosses in fig.6) since we wanted just the final output of our classifier layer. The BERT automatically make sure under the hood that all embedding weights are used for the classification layer. This classification layer is essentially a SoftMax layer which predicts whether the given text is fake news or not.

We then initiated the training of our task with standard log-loss and the result were truly remarkable. The BERT model was well above the accuracy of the paper whose results we originally replicated.



	Training Loss	Valid. Loss	Valid. Accur.	Training Time	Validation Time
epoch					
1	2.98e-03	0.01	1.0	0:09:30	0:00:17
2	3.13e-03	0.01	1.0	0:09:30	0:00:17
3	3.91e-04	0.02	1.0	0:09:30	0:00:17
4	1.44e-04	0.02	1.0	0:09:30	0:00:17

Fig.8

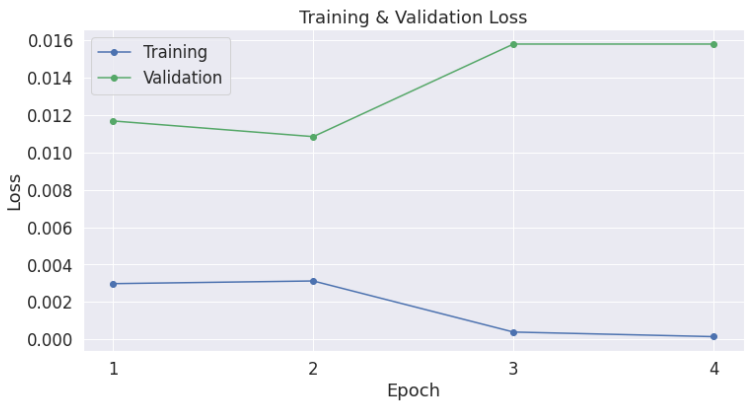


Fig.9

Our Validation Loss was minimal, and our validation accuracy was 1.0 which meant that our model was perfectly differentiating the categories of news. We thought it might be overfitting but given the fact that Validation data was data which BERT did not see while training, this initial worriedness had no justification.

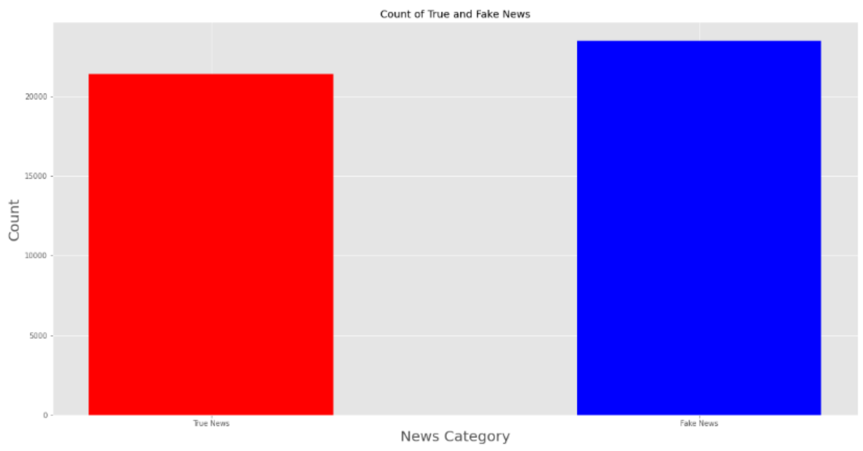
Final Take about Bert:

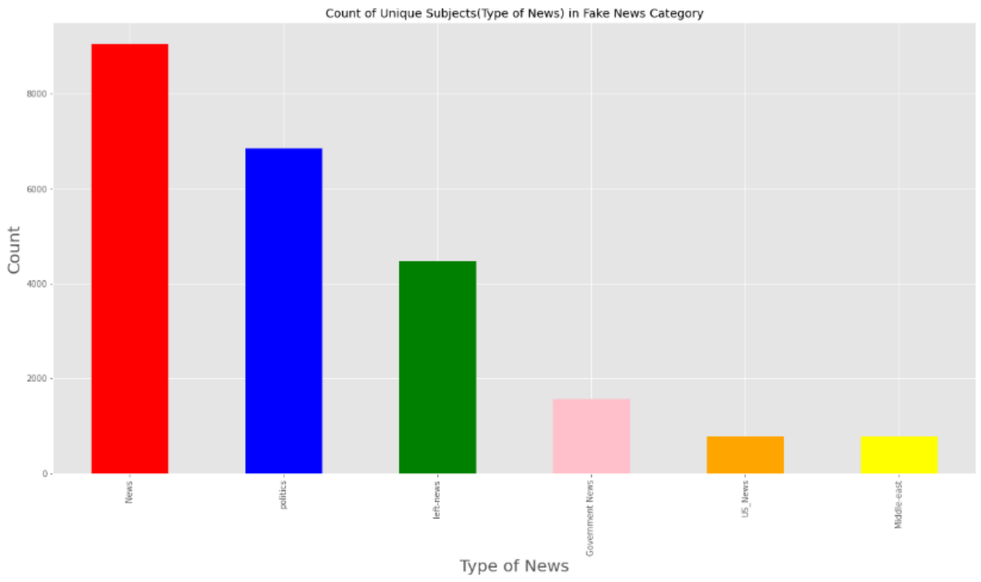
The results clearly reflect as how BERT is truly a remarkable model which through transfer learning can be adapted to various tasks of Natural Language Processing. Plus, BERT is clearly ahead of traditional Machine Learning Models. Finally, we also want to make it clear that BERT was trained on the whole NEWS data and there was no filtering done like done in the paper which we replicated since we wanted to get the maximum performance.

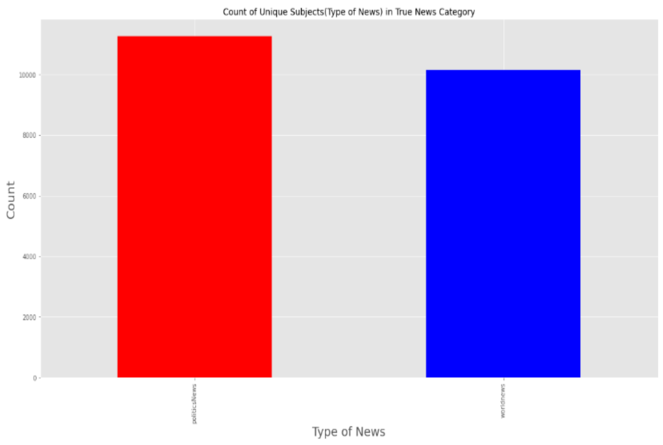
Conclusion:

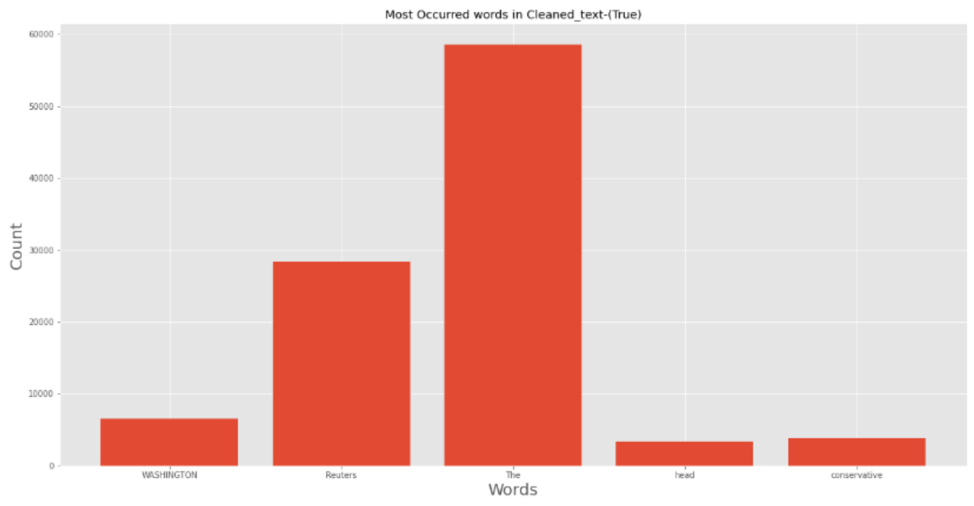
In the end, we arrived at the conclusion through this paper that NLP field has been progressing at rapid rate. As BERT outperformed all the classical Machine Learning algorithms which in turn showed that BERT could be easily fined tuned to the task of classifying fake and real news. Moreover, our paper also showed the inner working of the BERT. We explained how BERT is based on the transformer architecture. This in turn make our paper unique since we not only showed you that BERT was able to beat the existing papers but justified its performance by showcasing the sophisticated and deep architecture that BERT possesses as compare to the simple architecture of classical Machine Learning algorithms.

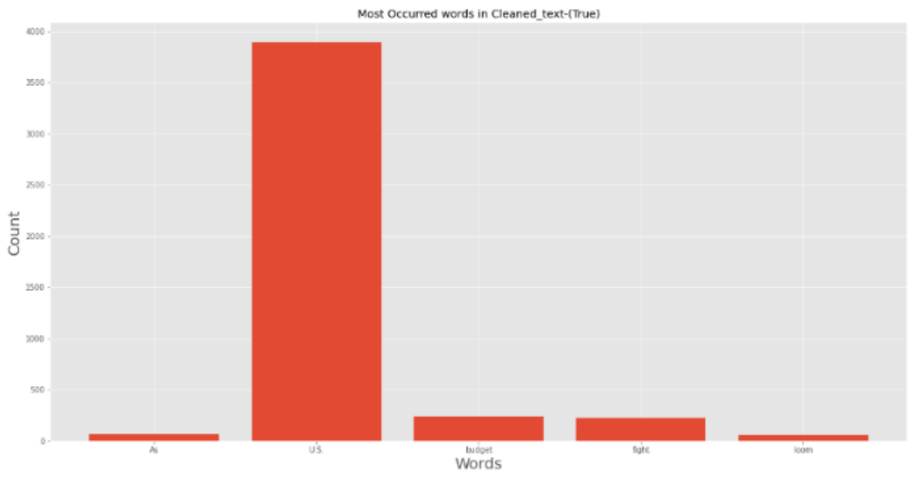
Exhibits

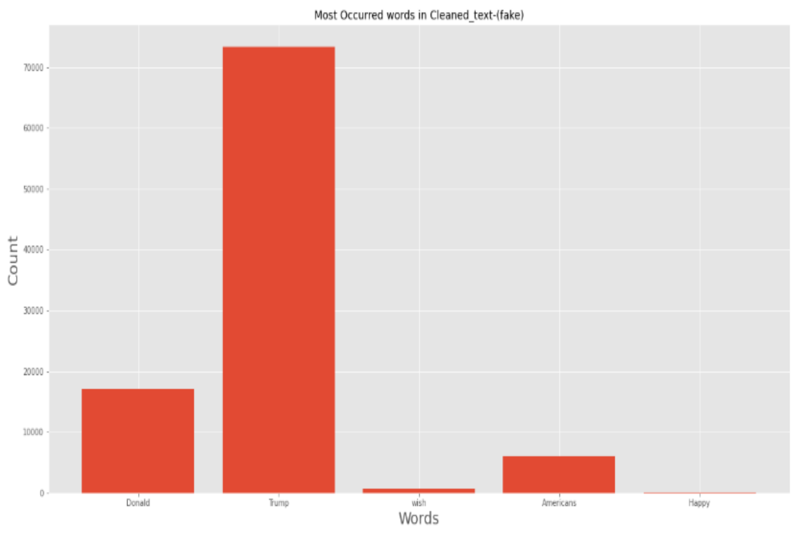

Count of True and Fake News
Count
0
5000
10000
15000
20000
True News
Fake News
News Category


Count of Unique Subjects(Type of News) in Fake News Category
Count
0
2000
4000
6000
8000
News
politics
left-news
Government News
US_News
Middle-east
Type of News


Count of Unique Subjects(Type of News) in True News Category
Count
0
2000
4000
6000
8000
10000
politicsNews
worldnews
Type of News


Most Occurred words in Cleaned_text-(True)
Count
60000
50000
40000
30000
20000
10000
0
WASHINGTON
Reuters
The
head
conservative
Words


Most Occurred words in Cleaned_text-(True)
Count
4000
3500
3000
2500
2000
1500
1000
500
0
As
U.S.
budget
fight
loom
Words


Most Occurred words in Cleaned_text-(fake)
Count
70000
60000
50000
40000
30000
20000
10000
0
Donald
Trump
wish
Americans
Happy
Words

Work Cited

Ahmed H, Traore I, Saad S. “Detecting opinion spams and fake news using text classification”, Journal of Security and Privacy, Volume 1, Issue 1, Wiley, January/February 2018

Ahmed H, Traore I, Saad S. (2017) “Detection of Online Fake News Using N-Gram Analysis and Machine Learning Techniques. In: Traore I., Woungang I., Awad A. (eds) Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments. ISDDC 2017. Lecture Notes in Computer Science, vol 10618. Springer, Cham (pp. 127- 138).

Alammar, Jay. “The Illustrated Transformer.” *The Illustrated Transformer – Jay Alammar – Visualizing Machine Learning One Concept at a Time.*, jalammar.github.io/illustrated-transformer/.

Devlin, Jacob, et al. “BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding.” *ArXiv.org*, 24 May 2019, arxiv.org/abs/1810.04805.

Granik, Mykhailo and Mesyura, Volodymyr, "Fake news detection using naive Bayes classifier," *2017 IEEE First Ukraine Conference on Electrical and Computer Engineering (UKRCON)*, Kiev, 2017, pp. 900-903.

Huh, Minyoung, Andrew Liu, Andrew Owens, and Alexei A. Efros, “Fighting Fake News: Image Splice Detection via Learned Self Consistency”, *The European Conference on Computer Vision (ECCV)*, 2018, pp. 101-117

Riggio, Christopher. “Using LSTM to Implement a Sector Rotation Trading Strategy Part I.” *Medium*, Analytics Vidhya, 29 Oct. 2019, medium.com/analytics-vidhya/using-lstm-to-implement-a-sector-rotation-trading-strategy-part-i-5481e2b00a71.

Rubin, Victoria L., Nial J. Conroy, Yimin Chen and Sarah Cornwell, “Fake News or Truth? Using Satirical Cues to Detect Potentially Misleading News”, *Association for Computational Linguistics*, pp. 7-17, 2016

Shu, et al. “The Role of User Profile for Fake News Detection.” *ArXiv.org*, 30 Apr. 2019, arxiv.org/abs/1904.13355.

Tacchini, Eugenio, Gabriele Ballarin , Marco L. Della Vedova, Stefano Moret, and Luca de Alfaro, “Some Like it Hoax: Automated Fake News Detection in Social Networks”, 2017

Vaswani, Ashish, et al. “Attention Is All You Need.” *ArXiv.org*, 6 Dec. 2017, arxiv.org/abs/1706.03762.