FAKE NEWS DETECTION USING MACHINE LEARNING

A BTP Report

by

Names

Mohammed Mohsin Ali

Mohit Kumar

Gurrapu Jaideep

Roll Nos. :

S20180010105

S20180010106

S20180010063

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY SRICITY

DATE - 15-12-2021

Final Report

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY SRI CITY

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the BTP entitled "**Fake News Detection Using Deep Learning**" in the partial fulfillment of the requirements for the award of the degree of B. Tech and submitted in the Indian Institute of Information Technology SriCity, is an authentic record of my own work carried out during the time period from January 2021 to December 2021 under the supervision of Dr. Ameet Praseed, Indian Institute of Information Technology SriCity, India.

The matter presented in this report has not been submitted by me for the award of any other degree of this or any other institute

[illegible]

Signature of the student with date
(Mohammed Mohsin Ali)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Signature of BTP Supervisor with date
(Dr. Ameet Praseed)

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY SRI CITY

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the BTP entitled "**Fake News Detection Using Deep Learning**" in the partial fulfillment of the requirements for the award of the degree of B. Tech and submitted in the Indian Institute of Information Technology SriCity, is an authentic record of my own work carried out during the time period from January 2021 to December 2021 under the supervision of Dr. Ameet Praseed, Indian Institute of Information Technology SriCity, India.

The matter presented in this report has not been submitted by me for the award of any other degree of this or any other institute.

Mohit Kumar

Signature of the student with date
(Mohit Kumar)

_ _

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Signature of BTP Supervisor with date
(Dr. Ameet Praseed)

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY SRI CITY

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the BTP entitled "**Fake News Detection Using Deep Learning**" in the partial fulfillment of the requirements for the award of the degree of B. Tech and submitted in the Indian Institute of Information Technology SriCity, is an authentic record of my own work carried out during the time period from January 2021 to December 2021 under the supervision of Dr. Ameet Praseed, Indian Institute of Information Technology SriCity, India.

The matter presented in this report has not been submitted by me for the award of any other degree of this or any other institute.

[illegible]

Signature of the student with date
(Gurrapu Jaideep)

_ _

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Signature of BTP Supervisor with date
(Dr. Ameet Praseed)

ABSTRACT

Fake news detection has been a challenging problem in deception detection and it has tremendous real world political and social impacts. A lot of research work is happening to tackle this problem and many papers also worked on building datasets. Here we have used the paper “Liar, Liar Pants on Fire” along with the dataset : A New Benchmark Dataset for Fake News Detection. It has not only collected the labelled fake/not fake tweets but also meta data of tweets like(author,state,party,count of fake tweets, count of real and some other information about tweet and author). This paper along with a dataset also proposed a hybrid Convolutional Neural Networks framework for integrating text and meta-data.

We first designed a different architecture that has three inputs (textual data, categorical data and numerical data) using a pre-trained BERT model. Before experimenting with this model we have also used classical machine learning models like K-Nearest Neighbour Classifier, Decision Trees, Gradient Boosting Decision Trees, Kernel-SVM. And for vectorizing the data before feeding to classical models we used TF-IDF weighted word2Vec, and some augmentation techniques to deal with imbalance of data, and to validate all classical models we used RandomSearchCV as a tuning algorithm to get best hyperparameters.

In addition to above models, we also designed a transformer model from scratch with different positional embedding and this type of changes in transformer gave some promising results compared to a large pretrained bert model. This shows how a simple transformer mode with some changes can achieve the same performance or greater compared to heavily trained bert model.

CONTENTS

INTRODUCTION

An increasing amount of our lives is spent interacting online through social media platforms and more and more people tend to seek out and consume news from social media rather than traditional news organizations. It is often more timely and less expensive to consume news on social media compared with traditional news media, such as newspapers or television; and it is easier to further share, comment on, and discuss the news with friends or other readers on social media.

Detecting fake news is a challenging task to accomplish as it requires models to summarize the news and compare it to the actual news in order to classify it as fake. Moreover, the task of comparing proposed news with the original news itself is a daunting task as it is highly subjective and opinionated.

Through experimental procedures, we propose a model which can detect fake news. We also studied how different hyperparameters affect the model performance and summarized the details . Our model performs reasonably well when classifying between all the stances with some variations in accuracy for disagreed stances.

Further, we have discussed problems with defining and identifying fake news, described the LIAR: A BENCHMARK data set which we used to perform the experiment, and we provided a primer on various techniques used in our experiments such as machine and deep learning. We explain the experimental design followed by our solution to solve the fake news detection problem. We present our results for different models and hyper parameter tuning for the models and we further conclude our findings.

Dataset Used: **LIAR**: A BENCHMARK data set

Short Statement	12,836
Labels	6
Other Details	● *Subject* ● *context/veue* ● *Speaker* ● *State* ● *party* ● *prior history*

table1→ about dataset

LITERATURE SURVEY

1. “Liar, Liar Pants on Fire

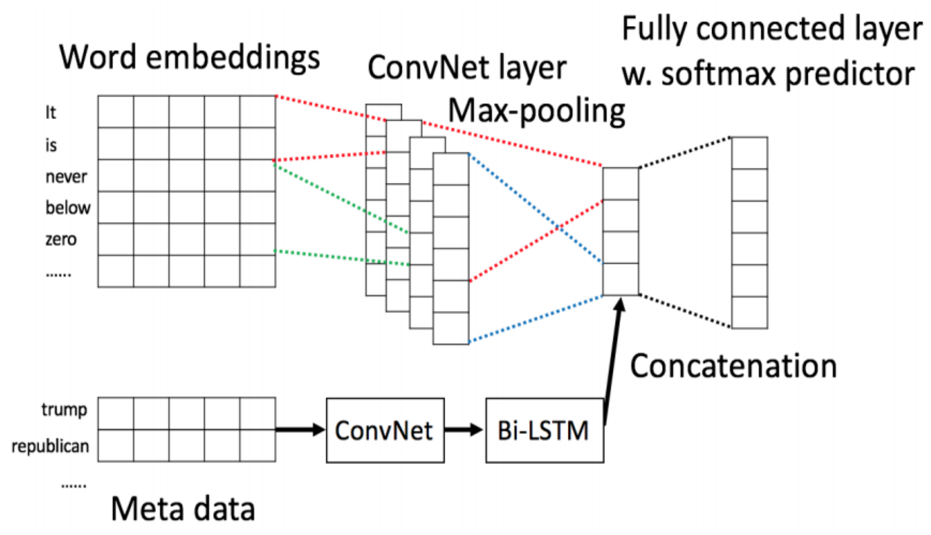
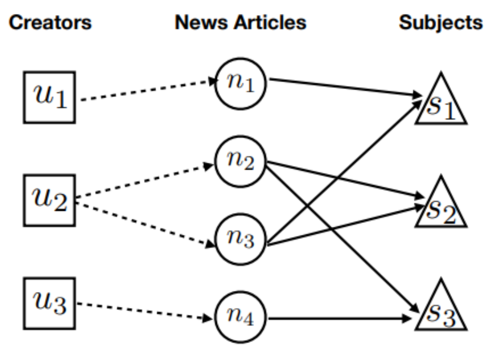


fig-1 → Hybrid Convolutional Neural Network

They randomly initialized a matrix of embedding vectors to encode the metadata embeddings. Used a convolutional layer to capture the dependency between the metadata vector. A standard max-pooling operation is performed on the latent space, followed by a bi-directional LSTM layer. Then concatenated the max-pooled text representations with the meta-data representation from the bi-directional LSTM, and fed them to a fully connected layer with a softmax activation function to generate the final prediction.

2. **FAKEDETECTOR:** Effective Fake News Detection With Deep Diffusive Neural Network

This paper introduces an automatic fake news inference model called FAKE DETECTOR. Based on a set of explicit and latent features extracted from the textual information, FAKEDETECTOR builds a deep diffusive network model to learn the representations of news articles, creators and subjects simultaneously.



$$\begin{cases} \text{\# Fusion Layer: } \mathbf{x}^l_{n,i} = \sigma(\sum_{t=1}^{q} \mathbf{W}_i \mathbf{h}_{i,t}), \\ \text{\# Hidden Layer: } \mathbf{h}_{i,t} = GRU(\mathbf{h}_{i,t-1}, \mathbf{x}_{i,t}; \mathbf{W}) \end{cases}$$

fig-2 → diffusive NN

3. RNN and LSTM:

- The RNN presented in the paper[1] by **Rumelhart et al**. described how recursively feeding the output itself into the neuron as input can help retain sequential memory

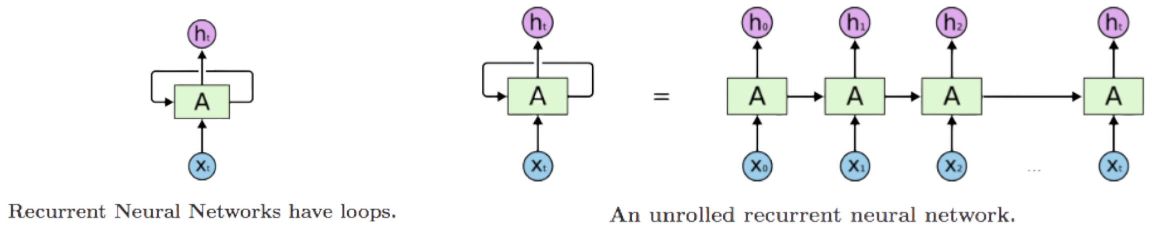

Recurrent Neural Networks have loops.
An unrolled recurrent neural network.
A
=

fig-3 → RNN Units

- But, for the longer sequences RNN had problems with vanishing gradient/gradient diminishing
- The problem of vanishing gradient for longer sequences was solved using the revolutionary LSTM cell
- LSTM However, solved the problem of vanishing gradient using the gates storing longer dependencies but still it was computationally expensive to train

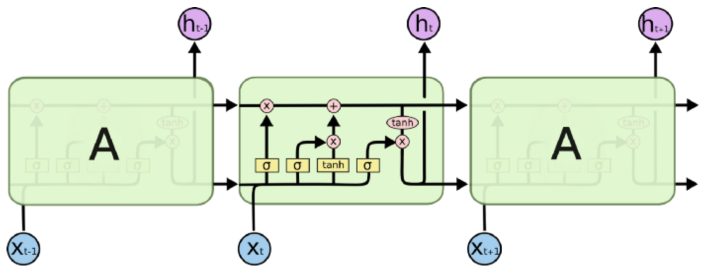


fig-4 → LSTM

4. Attention(Transformers)

- In the paper by Vaswani et al. came up with an encoder-decoder architecture that could be fed all the input tokens simultaneously.
- This results in two things:
    1. Faster computation
    2. Loss of positional information
- Thus the only downfall being losing positional information, it was resolved by the use of **Positional Encoding**.

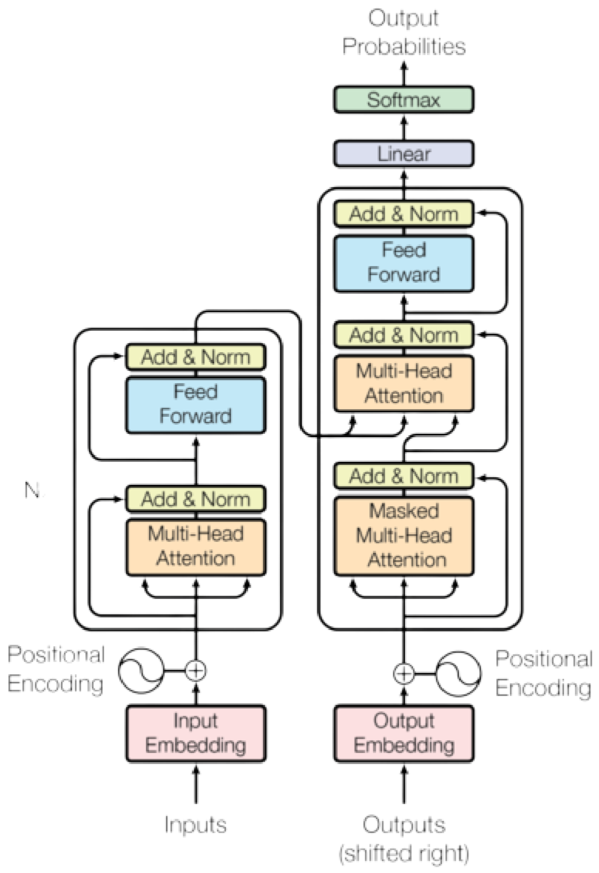


fig-5 → Transformer model

METHODOLOGY

Word2Vec + LSTM and ELMO Models

- Using Word2Vec embedding as the base model, experimented using LSTM and ELMO Models
- These models performed poorly, f1 score of 56 and 50 respectively

DistillBert

- DistilBERT is a small, fast and light Transformer model
- This model has shown a decent performance
- Distribution in which the BERT has been trained could be the major reason for the this performance
- This improvement over traditional LSTM/ELMO gave us the motivation to explore further with Transformer

Our Model - 1

- Designing our own encoder based transformer
- Using Multi Head attention in encoder layers
- Using pre trained Word Embedding instead of training because of complexity
- Incorporating word level and sentence level word embedding
- Our transformer model has 3 different types of inputs(text, cat, num)

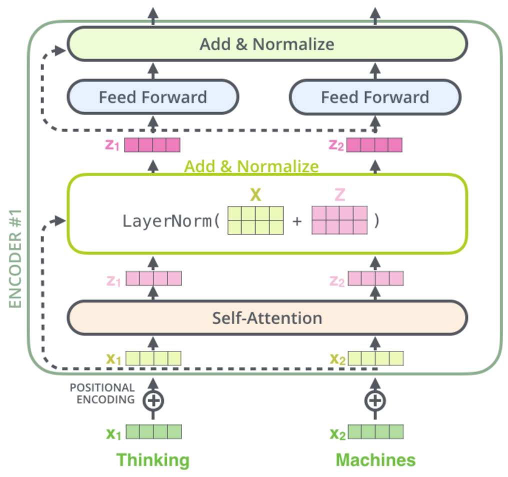


fig-6 → Encoder Layer

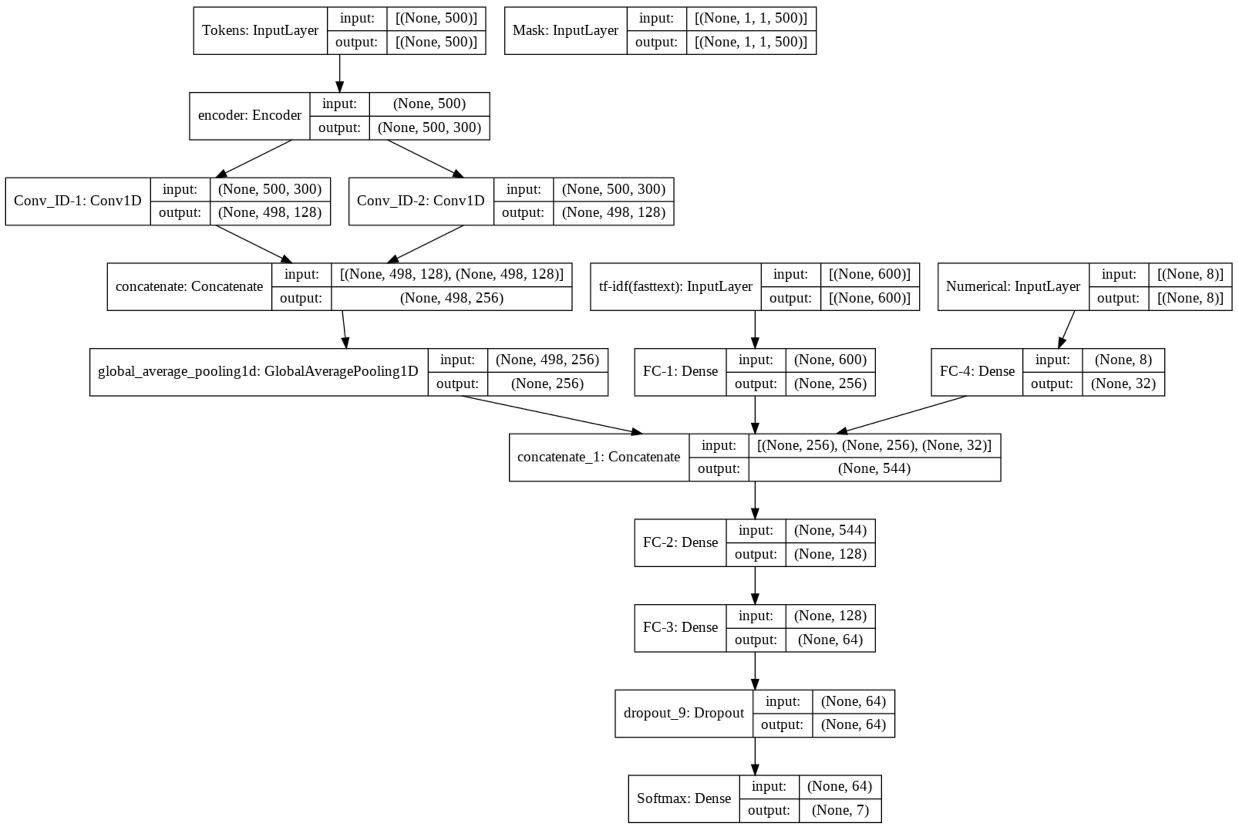
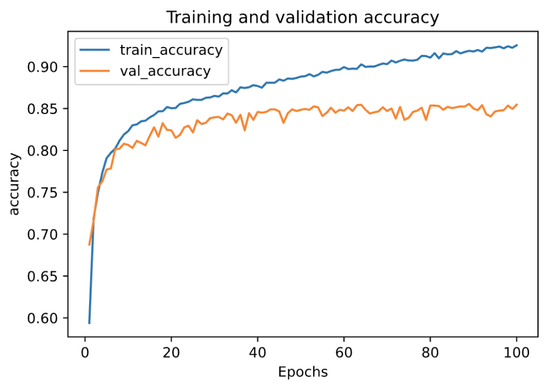

Tokens: InputLayer | input: [(None, 500)] | output: [(None, 500)]
Mask: InputLayer | input: [(None, 1, 1, 500)] | output: [(None, 1, 1, 500)]
encoder: Encoder | input: (None, 500) | output: (None, 500, 300)
Conv_ID-1: Conv1D | input: (None, 500, 300) | output: (None, 498, 128)
Conv_ID-2: Conv1D | input: (None, 500, 300) | output: (None, 498, 128)
concatenate: Concatenate | input: [(None, 498, 128), (None, 498, 128)] | output: (None, 498, 256)
tf-idf(fasttext): InputLayer | input: [(None, 600)] | output: [(None, 600)]
Numerical: InputLayer | input: [(None, 8)] | output: [(None, 8)]
global_average_pooling1d: GlobalAveragePooling1D | input: (None, 498, 256) | output: (None, 256)
FC-1: Dense | input: (None, 600) | output: (None, 256)
FC-4: Dense | input: (None, 8) | output: (None, 32)
concatenate_1: Concatenate | input: [(None, 256), (None, 256), (None, 32)] | output: (None, 544)
FC-2: Dense | input: (None, 544) | output: (None, 128)
FC-3: Dense | input: (None, 128) | output: (None, 64)
dropout_9: Dropout | input: (None, 64) | output: (None, 64)
Softmax: Dense | input: (None, 64) | output: (None, 7)

fig-7 → Our Model - 1



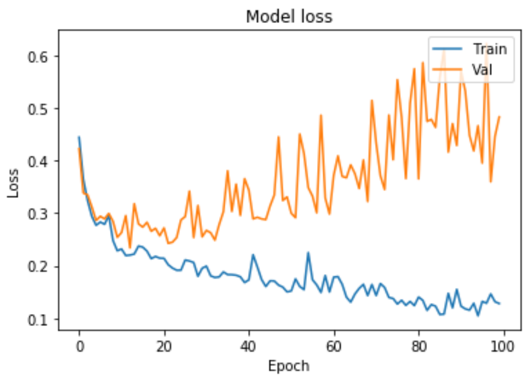


fig-8 → Our Model 1 Accuracy Plot ***fig-9 → Our Model 1 Loss Plot***

Our Model - 2:

Positional information:

- The brilliant idea was to append positional information with the embedding for each individual token before feeding it to the encoder-decoder architecture.
- This way, we could input all the tokens simultaneously and retain some information about the order of the tokens.

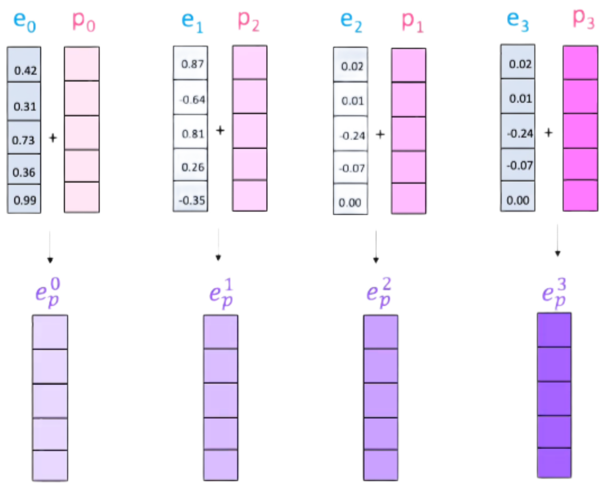


fig-10 → positional vectors

Appropriate positional Vectors:

- The vectors that were needed should have had:

- A fixed range so that values don't explode due to longer sentences.
- A metric that doesn't vary with variations in the length of sentences for the same position.
- Using geometrically progressive frequencies in sinusoidal functions adhered to both the needs.

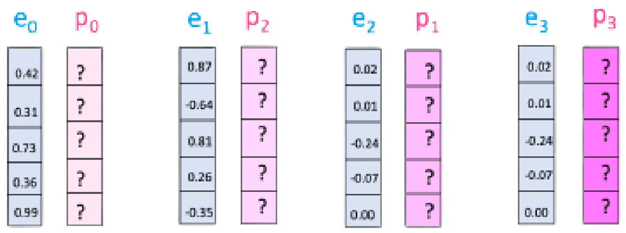


fig-11 → appropriate vectors

Sinusoidal Positional Encoding:

The use of multiple frequencies for the same position helps us retain the uniqueness of the positional vectors

$$\vec{p_t}^{(i)} = f(t)^{(i)} := \begin{cases} \sin(\omega_k . t), & \text{if } i = 2k \\ \cos(\omega_k . t), & \text{if } i = 2k+1 \end{cases} \qquad \omega_k = \frac{1}{10000^{2k/d}}$$

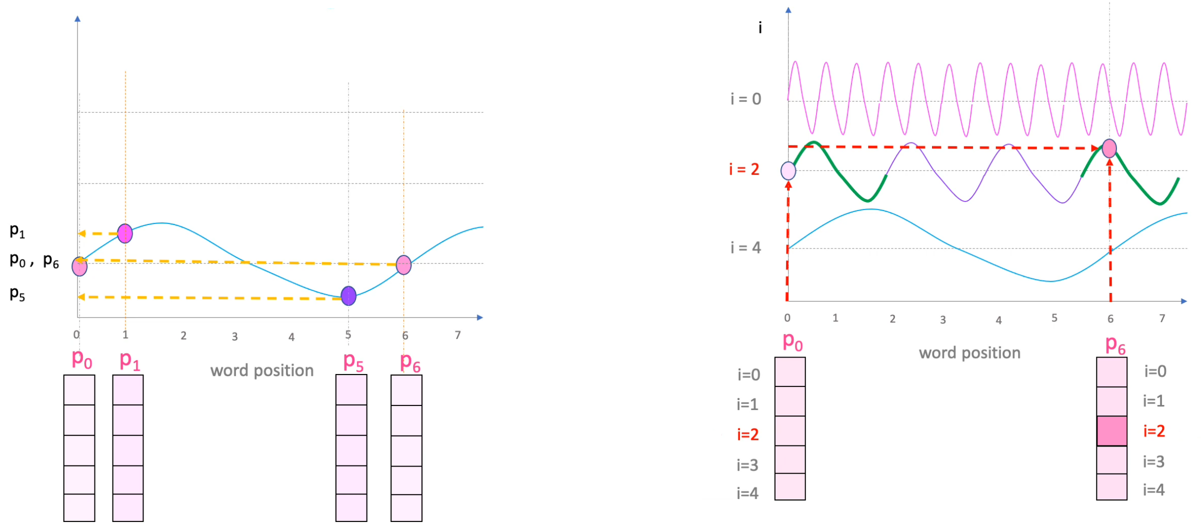


fig-12 → Sinusoidal Vectors

Why use both sine and cosine instead of just one ?

- Using only sin/cos will have effects on higher dimensions, to deal with this ambiguity, authors have introduced a way to get positional information according to odd and even positions. This way will differentiate words coming from higher dimensions

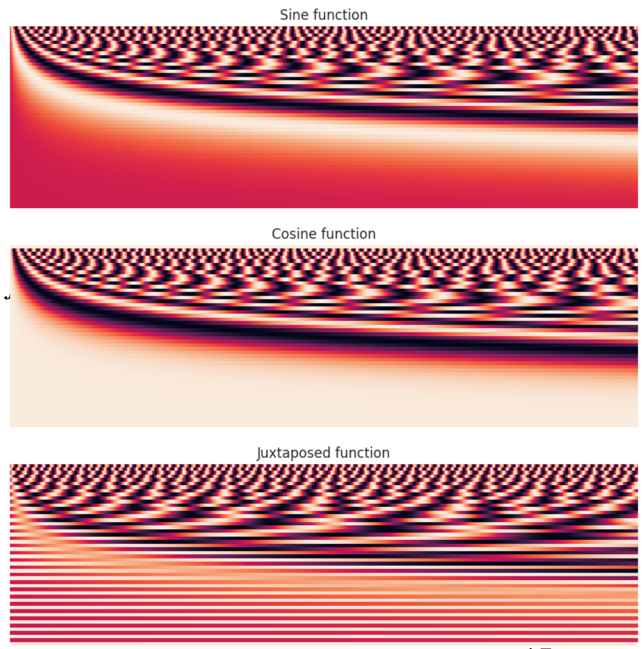


fig-13 → Sin/Cos and juxtaposed function

- Authors chose this function because they hypothesized it will allow the model to easily learn to attend by relative positions, sinc for any fixes offset, PE(pos+offset) can be represented as a linear function of PE(pos**)**
- Which basically means that for all **pos** (i.e. **k**), there is a linear transformation **M** that maps it to another position with a fixed offset.

$$M.\begin{bmatrix}\sin(\omega_k.\,t)\\ \cos(\omega_k.\,t)\end{bmatrix} = \begin{bmatrix}\sin(\omega_k.\,(t+\phi))\\ \cos(\omega_k.\,(t+\phi))\end{bmatrix} \qquad M_{\phi,k} = \begin{bmatrix}\cos(\omega_k.\,\phi) & \sin(\omega_k.\,\phi)\\ -\sin(\omega_k.\,\phi) & \cos(\omega_k.\,\phi)\end{bmatrix}$$

$$\alpha_{ij}^{abs} = \frac{1}{\sqrt{d}}\left((w_i + p_i)\,W^{Q,1}\right)\left((w_j + p_j)\,W^{K,1}\right)^T$$

- This means that not only does positional encoding retain information about the absolute positions of the terms but also retains relative positions of each term with respect to every other term.

FLOATER *(Learning to Encode Position for Transformer with Continuous Dynamical Model)*

- Lui et al. proposed a transformer model with change in positional representation
- Instead of predefined sinusoidal positional representation, learning positional representation at encoder layer(main idea)
- Brings dynamic behavior
- Learnable function (***Θ***)
- ‘ i ’ is index wrt word w_i in a sentence/tweet

$$\alpha_{ij} = \frac{1}{\sqrt{d}}\left((w_i + \theta(i))\,W^{Q,1}\right)\left((w_j + \theta(j))\,W^{K,1}\right)^T$$

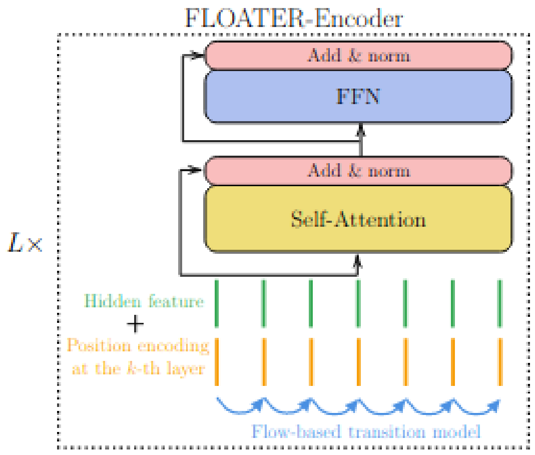


fig-14 → floater

Relative Positional Encoding (RPE)

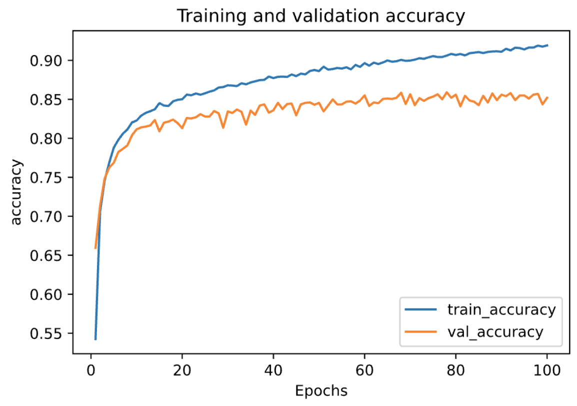
- A followup paper from Peter Shaw and Vaswani to improve the performance over longer sequences
- Introduced a learnable parameter a^{l}_{j-i}
- Capture Relative word orders
- On experimenting with dynamic and relative position encoding, this has shown an improvement of 3%

$$\alpha_{ij}^{rel} = \frac{1}{\sqrt{d}} \left((w_i)^l \, W^{Q,l} \right) \left((w_i)^l \, W^{K,l} + a^l_{j-i} \right)^T$$

Major Changes:

- Designing our own encoder based transformer
- Using Multi Head attention in encoder layers
- Using pre trained Word Embedding instead of training because of complexity
- Incorporating word level and sentence level word embedding
- Our transformer model has 3 different types of inputs(text, cat, num)
- Our model-2 uses dynamic and relative positional encoding instead of predefined position encoding(**Novel**)

$$\alpha_{ij} = \frac{1}{\sqrt{d}} \left((w_i + \theta\left(S(l_i)\right))^l \, W^{Q,l} \right) \left((w_j + \theta\left(S(l_j)\right))^l \, W^{K,l} + a^l_{j-1} \right)^T$$



f ig-15 → our model 2 Accuracy plot

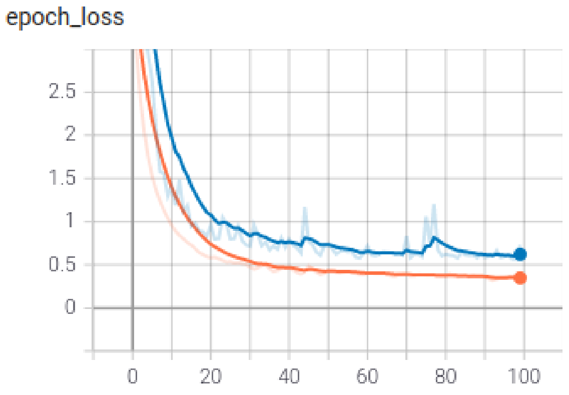


fig-16 → our model 2 loss plot

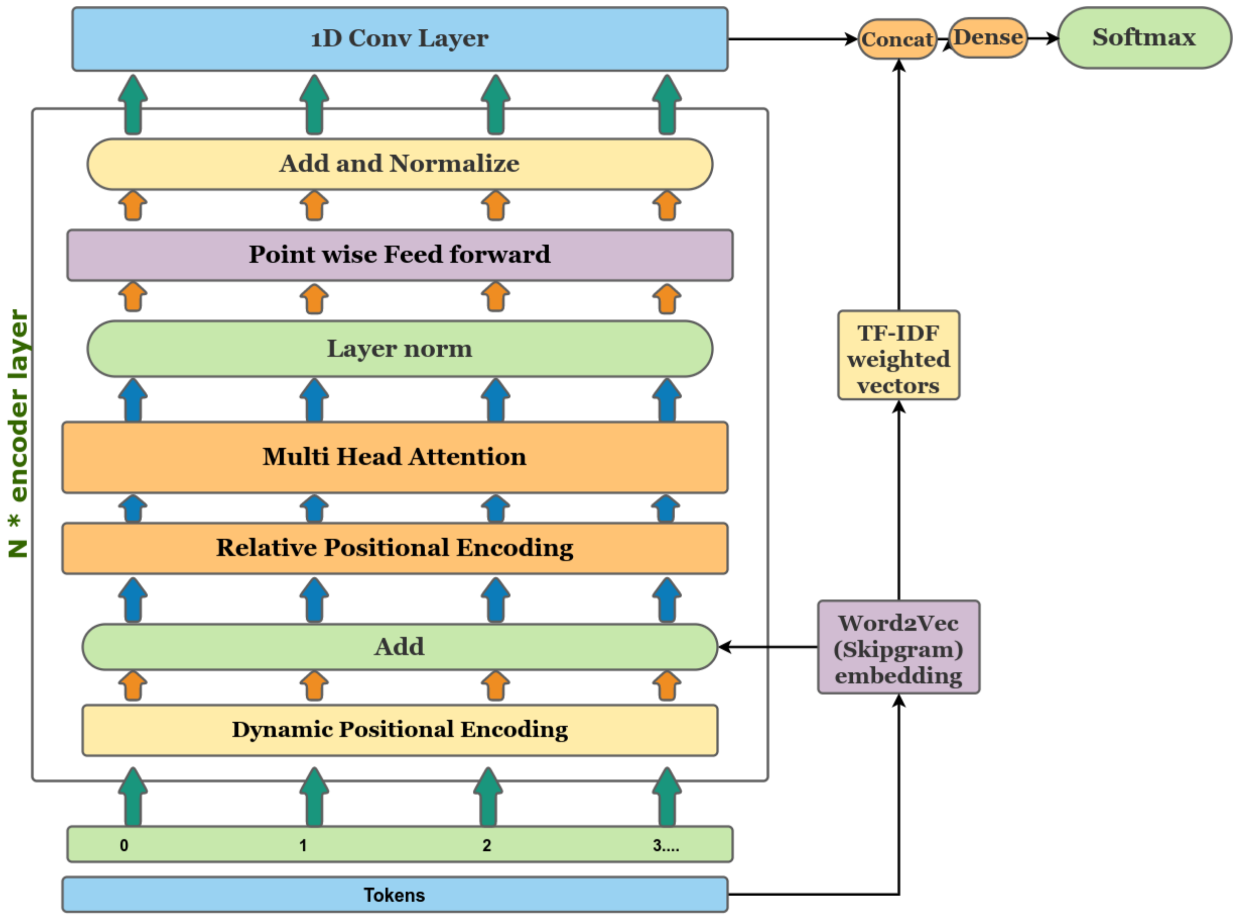


fig-17 → Our Model 2 Architecture

RESULTS

MODELS	Train Accuracy	Val Accuracy	Test Accuracy	Hyperparameters
Decision Trees	*64*	*51*	*50*	***max_depth = 50, min_sample_split = 100***
GBDT	*65*	*53*	*50*	***n_estimator = 200, max_depth = 3, loss=multclass_logloss Learning_rate = 0.1***
SVM	*65*	*61*	*55*	***alpha = 1, kernel=rbf, sigma=1***
Bert + fine tuning	*89*	*93*	*88*	***optimizer=Adam with lr=1e-3 weight_initialisation = he_normal, L2-regularization=1e-2 batch_size=128, epoch=200,loss=categorical_crossentropy***
Our Model - 1	*85*	*85*	*80*	***optimizer=Adam with lr=1e-4 weight_initialisation = he_normal, L2-regularization=1e-2 batch_size=128, epoch=200,loss=categorical_crossentropy***
Our Model - 2	*92*	*85*	*90*	***optimizer=Adam with lr=1e-4 weight_initialisation = he_normal, L2-regularization=1e-2 batch_size=128, epoch=200,loss=categorical_crossentropy***

table2→ Results

- We noticed DT, GBDT and SVM are performing same even after different hyperparameters, and more over DT and GBDT are good application for low latency system but due the poor performance we can not take to real word
- KNN have given some good results but, at the same time it is non-parametric and very slow at training as well as testing time
- We were needed of model like KNN but which take less computation power
- Bert+fine tuning model results are quite similar to KNN and also this model can be taken to real world application
- Our Model-1 is trained from scratch transformer model which has shown some promising results
- Our Model-2 is also a trained from scratch transformer model with modifications in positional information. Results show this model has some better performance than BERT though it is trained on a low scale.

DEPLOYMENT

Flask App:

- For deploying the model, I used the flask web framework.
- Used two html pages, one is responsible for taking data which is home.html and other is a prediction page named request.html
- We get data from home in a POST method and this data is again preprocessed and vectorized using the save vectorized file
- Making sure no data leakage is happening
- After vectorizing the data, passing it to a model which give class probabilities and final prediction
- These probabilities and prediction are rendered to request.html and it displaces the both class and its probabilities
- So, we are done with the app but need to deploy

Heroku:

- For Heroku, we first need a Procfile and its task it to identify which is the mail file in the project
- Need to commit the entire code in github and after logging in from Heroku website, we need to connect to our github repository and deploy with just a single click
- Here is the Heroku deployment link → https://mohsinaliapi.herokuapp.com/

AWS:

- For AWS deployment, I am using the same flask app
- We first need to create an EC2 instance. Get the key and convert the .pem key to .ppk
- Then, Need to download a third party software which connect our local system to the EC2 server we created
- We have to connect to server using .ppk key and transfer our entire project to server
- We also need to download a PuTTY SSH client software, which helps in running the code in server
- Putty SSH provides a kind of command link interface, where we can install all our dependencies and run the app in server
- Here is the AWS deployment link → http://ec2-3-128-18-227.us-east-2.compute.amazonaws.com:8080/
- Note* → AWS link may not work when I stop the instance and It has the same interface as Heroku**.**

CONCLUSION

We designed different ML models like KNN, DT, GBDT and SVM. Tuned all these models using RandomSearchCV to get best hyperparameters, KNN outperformed among these machine learning models. But KNN was super slow and not applicable for low-latency systems, then after referring to the “Liar, Liar Pants on Fire” paper we got to know about a hybrid CNN model that uses data as well as meta data for classification. We also designed a pre-trained Bert model which has three inputs, text, category and numerical. compared to both KNN and this model we can say that this model can be a good application for a low latency system. We also experimented with transformers and also did some modifications in positional encoding like, Incorporating the absolute and relative embeddings. For OurModel-1 we used the predefined Sinusoidal positional encoding and for OurModel-2 we used Dynamic and Relative positional encoding (Novel). From all the models we can say that OurModel 1 and 2 has shown better performance than the pre-trained Bert model in terms of parameters and accuracy.

LIST OF FIGURES

LIST OF TABLE

LIST OF ABBREVIATIONS

CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
LSTM	Long short term memory
KNN	K-Nearest Neighbors
GBDT	Gradient Boosted Decision Tree
DT	Decision Tree
SVM	Support Vector Machine
BN	Batch Normalization
FCN	Fully Connected Networks
re_relu	LeakyRelu
GRU	Gated Recurrent Networks
LR	Learning Rate
RBF	Radial Basis Function
PAD	Padding
BERT	Bidirectional Encoder Representations from Transformers
Tf-idf	Term Frequency - Inverse Document Frequency
W2Vec	Word to Vector
IG	Information Gain
NaN	Not a Number
CV	Cross Validation

ACKNOWLEDGEMENT

We would like to express our special thanks to our mentor
Dr. Ameet Praseed for his able guidance and support in completing our project.

We would also like to extend our gratitude to the evaluators **Dr. Shiv Ram Dubey, Dr. Rakesh Sanodiya** and **Dr. Piyush Joshi** for evaluating and giving further insights on how to overcome real world problems.

We would also like to thank ourselves for completing this project together.

REFERENCES

- “Liar, Liar Pants on Fire”:A New Benchmark Dataset for Fake News Detection
- FAKEDETECTOR: Effective Fake News Detection With Deep Diffusive Neural Network
- Supervised Learning for Fake News Detection - Nanyang Technological University, Singapore
- Using CountVectorizer to Extracting Features from Text- GeeksforGeeks
- https://arxiv.org/pdf/2003.09229.pdf
- https://arxiv.org/abs/1803.02155
- https://arxiv.org/abs/1706.03762
- https://www.kaggle.com/pierremegret/gensim-word2vec-tutorial
- https://scikit-learn.org/stable/
- https://www.tensorflow.org/
- https://towardsdatascience.com/covid-fake-news-detection-with-a-very-simple-logistic-regression-34c63502e33b
- https://www.analyticsvidhya.com/blog/2019/09/demystifying-bert-groundbreaking-nlp-framework/
- https://neptune.ai/blog/bert-and-the-transformer-architecture-reshaping-the-ai-landscape
- https://towardsdatascience.com/word-embeddings-for-nlp-5b72991e01d4
- https://www.onely.com/blog/what-is-tf-idf/