# Comparative Analysis of Transformers with Recurrent Neural Network and Long Short-Term Memory on Sentiment Analysis

Siyam Sajnan Chowdhury
*School of Computer Science*
*University of Windsor*
Windsor, Canada
chowdh1b@uwindsor.ca

Kasra Mojallal
*School of Computer Science*
*University of Windsor*
Windsor, Canada
mojalla@uwindsor.ca

Kimia Tahayori
*School of Computer Science*
*University of Windsor*
Windsor, Canada
tahayor@uwindsor.ca

***Abstract*—Sentiment analysis, which is also referred to as opinion mining, is a subset of Natural Language Processing (NLP) which is used to find out or extract emotions or perception towards a specific thing from text data which is subjective in nature [1]. The importance of NLP in general is immense. Using transformers, especially Bidirectional Encoder Representation from Transformers (BERT) [2], has yielded quite significant accuracy improvements over previously used models such as Recurrent Neural Network (RNN) [3] and Long Short-Term Memory (LSTM) [4]. In this paper, we have investigated the accuracy in classifying different types of sentiments using pre-trained model such as BERT and simulated the increased accuracy that can be achieved over using conventional models such as RNN and LSTM. We used the Play Store Application Reviews Dataset [5] and the IMDB Movie Reviews Dataset [6] for performing our comparative analysis of the different models.**
***Keywords: Sentiment Analysis, Natural Language Processing, Transformers, BERT, RNN, LSTM.***

## I. INTRODUCTION

Natural Language Processing has been on the rise for the past 5 years [7], and NLP has seen huge practical applications and implementations. One of the most substantial uses of natural language processing is a generative language chatbot ChatGPT which uses the pretrained model GPT4 [8]. ChatGPT currently has more than 100 million users [9]. This goes to show the immense potential and usefulness of NLP.

Sentiment Analysis is another segment of NLP which is extremely important for all kinds of tasks. Sentiment analysis is a process which is employed to extract and detect emotions, sentiments and /or feelings from textual data that are used in fields such as social media, or e-commerce reviews. It is a crucial element to ascertain a person's sentiments from their textual expressions from these sources as it can have huge applications both in terms of a researchers' perspective or commercial perspectives. Some possible examples of how important this is in these aspects could be researchers using sentiment analysis to detect people's moods or feelings which could then be translated into their state of mental health [13]. Conversely, in commercial cases, reviews left by users could give an idea as to how a product is and if there are certain issues with the mass perception of the product, it could then be altered accordingly [14]. Sentiments can be a wide spectrum but for our paper, the sentiments that we used are positive, negative, and neutral. These different types of sentiments are what enables us to help classify sentiments from given input strings or data. It is

these categories in which the analysis and then further classification takes place. There can be multiple other categories as well in sentiment analysis.

Transfer learning was first used in NLP back in 2018 with the advent of Universal Language Model Fine-tuning for Text Classification (ULMFiT) [10]. Building on this concept, transfer learning has since become a very popular means to train models in NLP and Sentiment Analysis in particular. As of now, there have been multiple transformer-based language models that have been developed and many more that are being developed and updated. Some of the most used and notable ones (as of July 2023) are:

Megatron – Nvidia [11]
GPT-4 – OpenAI [8]
BERT – Google Research [2]
Turing-NLG – Microsoft [12]

## II. PROBLEM STATEMENT

### *A. Problem Definition and Formulation*

Traditional models of sentiment analysis are by no means efficient. One of the key issues with traditional methods are that it misses out on the understanding of contexts which is a core capacity in extracting the sentiments. Those methods usually consider each texts and words as independent units not taking into account the importance surrounding words might have with regards to understanding the true meaning of any specific word. Also, an important part of understanding sentiments would require the understanding and filtering of sarcasms and ironic sentences. In these instances, disregarding or not considering contexts could end up with the model identifying a sentiment which is the exact opposite of the sentiment that is truly expressed.

Another issue with traditional models is their overreliance on lexicon-based approaches. This means that special emphasis is placed on certain words which falls withing the list of pre-determined words which bears sentiments. Those models also have issues working with negation words such as “not happy” which reverses the sentiment of a given word.

One other key issue with traditional models is that those models require a fully annotated dataset for those to be trained and this can be very expensive as well as time consuming because the annotation would generally require some sort of human intervention for it to be created. This also means that languages which are not as popular or as widely used as languages such as English could be negatively impacted due to the limitations of having inadequate labelled datasets. Another aspect of this issue is that data collected from social media or other user-generated content platforms could contain words and vocabularies that are not included in the dictionaries such as urban slangs, abbreviations, or shortened words. This results in limited accuracy of the models.

The use of models like RNN had brought about significant improvement over the traditional models in sentiment analysis tasks. However, even that had quite a lot of drawbacks. RNNs suffer from vanishing gradient or exploding gradient problems when the sequence becomes longer. This inability to capture long-term dependencies accurately means that the parameters of the model will not be updated accurately, and this ends up in the updates over the time steps either very nominal or very drastic.

RNNs are unable to compute in parallel which means that the process is totally sequential, and this becomes a huge disadvantage while working with larger datasets and massively decreases its computational capacity. This also causes RNNs to be quite slow because they cannot make use of the resources that we have at hand, such as modern GPUs which are capable of parallel computing.

RNNs also have difficulty in global context modeling as the sequential input means that the hidden state, at any given time, is only influenced or factored by just the previous state. They also have issues handling variable length sequences due to the fact that RNN models take sequential input, and this results in varying length sequences being truncated or padded which could result in important information being lost and ultimately cost the model accuracy.

LSTM model bridges a lot of the drawbacks that RNNs have. One of the most important drawbacks of RNN that LSTM overcomes is that it can capture long-term dependencies which means that the vanishing or exploding gradient problem face by RNNs is greatly minimized and longer sequences can now be used to accurately

predict sentiments. LSTMs can also handle variable length input sequences unlike RNN and thus does not require truncation or any sort of data loss which could cost the model's accuracy. This is done mostly by LSTMs memory cells. LSTMs can also understand contexts and negations better than RNNs. However, they still suffer from the same problem RNN does – unable to process data parallelly. The sequential processing means that it suffers from computational bottleneck when it comes to modern computational systems.

This paper is seeking to find the solution of the aforementioned problems via employing BERT, which is built on the transformers model. BERT has superb capability to under contexts because it does not take in the input sequentially but rather takes the whole input at once this enabling it to effectively understand the relationship between different words. This concept of having the whole input sequence at once to understand the high-level context is a significant element in sentiment analysis.

Transformers and BERT has the advantage of making use of transfer learning and fine-tuning. BERT comes already pre-trained with a large library of inputs which could potentially understand the different words that might not be included in conventional vocabularies.

Taking all the data input at once is also a feature of BERT as it has the capacity to compute in parallel rather than sequentially. This also means that the model is quite fast as it can make use of very powerful modern-day graphics processing unit to compute parallelly, utilizing all the available resources that we have at our disposal.

### *B. Motivations*

Massive amount of textual data is generated every single second across the globe from various different sources such as social media, blogs, articles etc. The inherent need to successfully and efficiently extract the meaning and sentiment from these texts is also very important. It would provide not just researchers, but also different business organizations understand human insights and opinions across multiple domains.

Real-time sentiment analysis is also crucial in certain domains such as customer support where understanding a customers' emotions and sentiment at a given time could mean the retention or losing of that customer. There are many other such examples where efficient sentiment analysis could play a pivotal role in bettering our lives.

### *C. Justifications*

Considering BERT as a probable solution to these problems is because of its proven capabilities. BERT works very well for solving complex tasks. It offers parallel computing making it faster to train compared to other models and works well with long sequence of input with higher accuracy.

## III. RELATED WORKS

We considered a few papers when working for this paper.

Hoang et al. applied BERT to develop an aspect-based sentiment analysis model which managed to predict different aspects related to a text by fine-tuning the BERT model to a sentence pair classification model. Their experimental results outperformed previous state-of-the-art models [29].

Zhang et al. used deep learning techniques such as bidirectional LSTM with Graph Convolutional Networks for the embedding for sentiment analysis [30].

Zimbra at al. found out that the average accuracy for sentiment analysis and classification tasks are around 61% in cases where the domain is quite generalized and in case of domain specific tasks, this accuracy was seen to increase up to about 11% on average [31].

Delvin et al. found out in their paper that BERT which uses attention mechanism achieved state-of-the-art results for eleven different tasks in the area of natural language processing and these state-of-the-art models were created by fine-tuning the pre-trained BERT model [2].

## IV. METHODOLOGY

### *A. Material and Data*

In order to carry out our analysis of deriving sentiments using the three different models RNN, LSTM and BERT, we used the Google Play Store Application Reviews Dataset [5] and IMDB Movie Reviews Dataset. [6]

The Google Play Store Reviews dataset contains 12000 data points with three labels – positive, neutral, and negative and it these data have been scraped from the Google Play Store using python libraries and some of these apps include Microsoft, Todoist, etc. [5]

The IMDB dataset contains data with two labels - positive and negative, among which 25,000 are for training and 25,000 for testing along with some other unlabeled data [6].

*B. Proposed Models*

i. Recurrent Neural Network

Recurrent Neural Network, or RNN, is a type of neural network which can make use of loops rather than just going in a unidirectional flow like conventional feedforward neural networks. RNNs can understand sequential data by understanding the context of the time-series data and by understanding the temporal dependencies [18].

The elemental element of an RNN is the recurrent unit which serves as a memory using the idea of a hidden state. At every iteration, the RNN uses the output from the previous hidden state and uses that in conjunction with the input and produces an output.

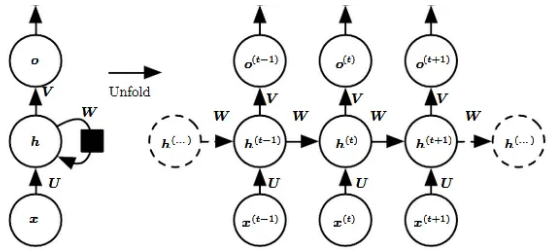


Figure 1: RNN ARCHITECTURE [19]

The left unit of Figure 1 is a single RNN unit and it is unfolded to provide multiple layers of the RNN. x(t) is the input that the hidden layer receives at any time step t. h(t) is considered to be the hidden memory state and it is calculated at each time step to be a function (any non-linear function e.g. ReLU) of the current input and the hidden step of the previous time step. RNNs have connections between the input to the hidden layer and this connection is denoted by the parameter U, connections between the hidden layers to the hidden layers which are denoted by the parameter W and connections between the hidden layer to the output layer which is parameterized by V. These parameters are all weight vectors which are shared across every time-step. o(t) refers to the output of the RNN. These weight vectors are updated with every pass of the RNN by backpropagation through time. With this, the models learn about the dependencies and the specific patterns that are to be learned.

II. Long Short-Term Memory

LSTM is built on the idea of RNN. It has quite a similar architecture to that of RNN but it overcomes the vanishing gradient problem that RNNs have by understanding the sequential data better saving the long term dependencies better. LSTM does this by employing a component called the memory cell which can choose to selectively hold on to information over multiple iterations so that it does not keep on diminishing and can forget such data as well.

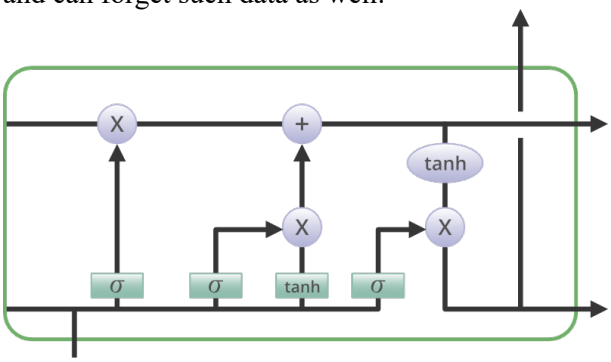


Figure 2: LSTM ARCHITECTURE [20]

LSTM consists of three gates, the Input Gate, the Forget Gate, and the Output Gate. The input gate consists of two functions – sigmoid and the tanh. It ascertains which values to use and which to not use to update the memory cell. The sigmoid function is there to determine which of the value will go through and the tanh functions assign weights to those values that do pass through. The values assigned are based off their importance and rated from -1 to 1 [21].

Forget Gate is used to consider which of the values are to be disposed. It used the sigmoid function to decide based on the input and he previous state whether to keep something or discard it. The output is 0 or 1 [21]. The output gate decides the output. This is done using the sigmoid function as the activation function to decide which of the values to go through to the

output based on the input value and the value stored in the memory cell. This value is between 0 and 1. Tanh function is then used to add weight to the approved values and the output is found by finding the product of the two functions. [22]

## III. Transformers

The concept of Transformers is necessary in order to understand the concept of BERT. Transformers is an architecture in NLP which solves issues pertaining to seq-to-seq problems by employing self-attention mechanism [23].

The architecture of a standard transformer learning model consists of an encoder and a decoder. The input sequence is fed into the encoder. It then proceeds to apply multiple layer of self-attention and then proceeds to feed those vectors into the feed forward neural network. The output of the feedforward neural network is contextualized representation of each token. [23]

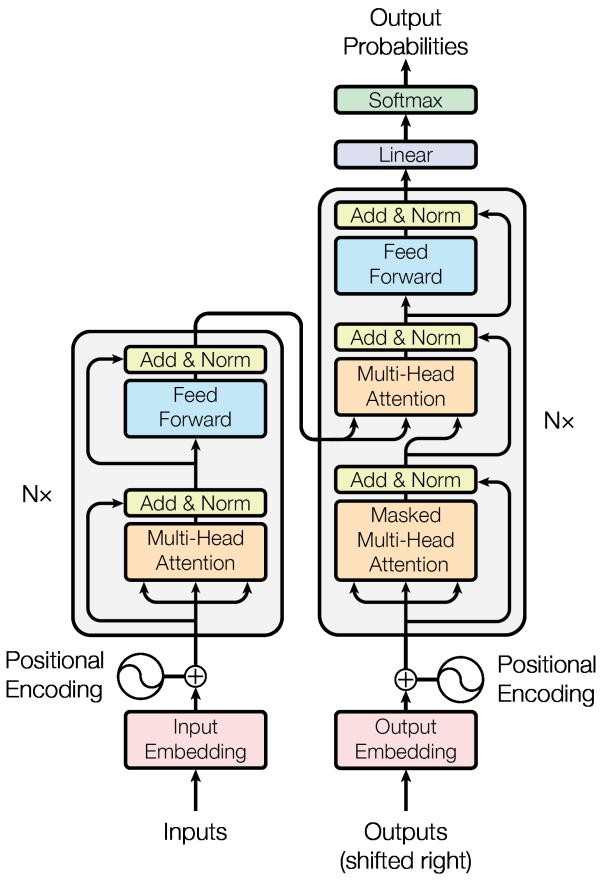


Figure 3: TRANSFORMER ARCHITECTURE [23]

The decoder then decodes the contextualized tokens by taking in the output embeddings and masking the multi-headed attentions layers. Different words are masked, which is like a fill-in-the-blank situation. This, together with the ouput from the encoder is then fed to another layer of attention and then passed through the feedforward neural network of the decoder block and the outputs are obtained by passing that through a linear activation function and then a softmax activation function.

## IV. BERT

Bidirectional Encoder Representations for Transformers is one of the most powerful language generational models as well as other NLP tasks. BERT puts the context in any given scenario by considering the words that precede the current word and the one that comes after it. Therefore, it is unlike conventional models which has the capacity to only look at the words in a specific order and only takes it input sequentially whereas BERT has the capacity to consider words from both sides.

The working mechanism of BERT is based on the transformer architecture explained above. It uses only the encoder part of the transformers and then stacks multiple of this encoder blocks end to end to create what we know as BERT [2].

Prior to providing BERT with input, a percentage of the words are swapped for a mask token which the model tries to predict the missing words based on the other given words. This requires placing a classification layer after the output of the encoder, embedding the output into a vocabulary dimension, and then using the softmax activation function to predict the probability of every single word in the vocabulary. The loss function of BERT converges slowly because it only considers the correctness of the masked values rather than the unmasked ones. [24]

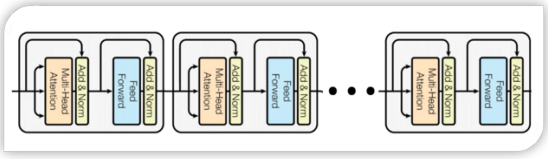

Figure 4: BERT ARCHITECTURE [2]

BERT makes use of transfer learning in order to work well. The concept of transfer learning is to be able to 'transfer' the knowledge gained or learned by a model while it was trained for a specific task to another similar task or domain [16]. Transfer learning usually comes with models that are pre-trained on specific datasets. The concept of transfer learning is to leverage the knowledge gained to improve the performance of the same model on a different dataset solving a different problem. Transfer learning performs well when there are limited data to work with and vastly reduces training time [2].

The application of transfer learning in this algorithm BERT is by fine-tuning the BERT model to our own dataset. We fine-tuned our data for the BERT model which means we trained the pre-trained data further on our dataset so that it adapts to the context of our dataset and performs accordingly by understanding the context [17]. Besides, training a huge model like BERT from scratch would take a very long time and huge computational requirements to train.

*C. Conditions and Assumptions*

Some conditions that we maintained throughout our experiments were that we used the same dataset throughout for the three different algorithms. We also maintained the same preprocessing for all the three models that we implemented.

The concept of pre-processing data is to shape our data in a form or style that is understandable by our model and is thus analyzable. In both of our datasets, we had text reviews which contained special characters, conjunctions and words which would not add anything meaningful or significant to our sentiment extraction such as words like 'and', 'to', 'in', etc. These are called stop words. There were also punctuations that added nothing of significance to the model. We used the NLTK library to remove these stop words and punctuations [15]. We also used the NLTK lemmatization, which means to group together different forms of a word to consider as the same word for ease of use, NLTK stemming, which is to reduce words to its base form or stem, and NLTK tokenization which breaks up the long texts into smaller portions or sections called tokens [15].

*D. Formal Complexity of Simulation Analysis*

The complexity of RNN depends mostly on the time-steps that it has as well as the hidden layers or units for each individual time-step. The computational complexity of RNN is directly dependent or proportional to the proportion of the hidden units so a larger number of hidden units would result in a an increased computational complexity. The computational complexity of RNN also increases with the parameters included within every hidden unit so increase in these parameters such as different weights and biases could make the model more computationally complex. The time complexity of RNN is $O(T*N^2)$ where N is the number of hidden layers and T is the number of time steps. The reason for the complexity of N raised to the power of 2 is due to the matrix multiplications.

The memory complexity of RNN depends solely on the number of time steps. For each time step, RNN contains a hidden state which is used to store certain information that are set by the users from the previous time step. Therefore, the more the number of time steps, the more the size of memory that is required.

The complexity of LSTM is more than that of RNN. The computations for the input-to-hidden layer in LSTM contains operations that are computed element by element and also involves matrix multiplication. Similar computations are also done during the hidden layer to hidden layer computations. The time complexity of LSTM is also $O(T*N^2)$ where N is the number of hidden layers and T is the number of time steps. Similarly to the RNN complexity, the complexity of LSTM increases with increase in number of parameters, number of hidden units or layers as well as the number of time steps.

The memory complexity of the LSTM model depends on the number of parameters, and the storage required for storing the hidden states and cell states in its memory cell for each time step. It also depends on the dimension of the input given to the model.

The complexity of BERT is proportional to the length of the string or sequence that has the highest length as the computation for BERT is parallel so the one with the longest sequence will be the one that the complexity will eventually depend on. The complexity of BERT also

depends on the size of the attention head. For the feedforward neural network, the hidden layers in it also contribute to the complexity of the model. Batch size also contributes to the complexity of the BERT model.

The memory complexity of the BERT model depends primarily on the many different parameters set by the users and the memory that is required in storing the values for the different passes, both forward and backward. It also depends on the maximum length of the string or sequence. Besides, storing of data such as loss and gradients for the backpropagation also requires additional memory.

## V. COMPUTATIONAL EXPERIMENTS

### *A. Experiments*

We ran the three models RNN, LSTM, and BERT on the two datasets that we had chosen in order to get results to analyze and compare whether the results obtained from BERT are comparatively better than the ones from RNN and LSTM.

### *B. Evaluation Metrics*

The evaluation metrics used for our experiments are Accuracy, Precision and Recall. Accuracy is the percentage of accurate predictions. Precision is the ratio $t_p/(t_p + f_p)$ where $t_p$ is the instances of true positives and $f_p$ the instances of false positives. The precision is a classifier's ability to not label a sample as positive when it is negative. Recall is the ratio $t_p/(t_p+f_n)$ where $t_p$ is the instance of true positives and $f_n$ the instance of false negatives. The recall is the classifier's ability to correctly find all the positive samples [25].

### *C. Implementation Details*

Our experiments were implemented using python in Google Colab. We also used the GPU for parallel processing while implementing BERT. We used the libraries tensorflow to create our models RNN and LSTM [26], NLTK for the preprocessing [15], Pretrained BERT model to implement BERT [2], Statistical Analysis using Sci-kit Learn [25] and NumPy [27] and plotting using Plotly [28].

### *D. Experimental Results*

The experimental results for our experiment on the first dataset – the Play Store App Review Dataset [5] is summarised in Table I, II and III.

Table I: ACCURACY FOR DATASET [5]

| Model | Experimental Accuracy |
|---|---|
| RNN | 62.2% |
| LSTM | 67.4% |
| BERT | 73.5% |

It is clear from Table I that BERT outperformed both RNN and LSTM in terms of accuracy. The accuracy of BERT could have been higher had the length of the reviews been longer because long strings of text are exactly where BERT shines as a model. Overall, the performance of BERT could be considered to be acceptable.

Table II: PRECISION FOR DATASET [5]

| Labels | RNN | LSTM | BERT | Instances |
|---|---|---|---|---|
| 0 | 0.68 | 0.69 | 0.75 | 457 |
| 1 | 0.23 | 0.25 | 0.32 | 144 |
| 2 | 0.64 | 0.69 | 0.74 | 399 |

It can be seen in Table II that BERT has better precision compared to both RNN and LSTM. However, it could be noticed that the precision for the neutral class 1 is low, but this could be attributable to the low number of overall instances for this class thus showing a class imbalance for the neutral class hence the trend.

Table III: RECALL FOR DATASET [5]

| Labels | RNN | LSTM | BERT | Instances |
|---|---|---|---|---|
| 0 | 0.69 | 0.81 | 0.84 | 457 |
| 1 | 0.14 | 0.01 | 0.06 | 144 |
| 2 | 0.71 | 0.78 | 0.85 | 399 |

A very similar trend to the one seen for precision can be observed for recall from Table III.

The experimental results for our experiment on the second dataset – the IMDB Movie Review Dataset [6] is summarised in Table IV, V and VI.

Table IV: ACCURACY FOR DATASET [6]

| Model | Experimental Accuracy |
|---|---|
| RNN | 76.3% |
| LSTM | 82.7% |
| BERT | 84.4% |

It can be seen in Table IV that BERT outperformed both RNN and LSTM in terms of accuracy. The accuracy of BERT is not very far off from LSTM in this context. Overall, the performance of BERT could be said to be quite good.

Table V: PRECISION FOR DATASET [6]

| Labels | RNN | LSTM | BERT | Instances |
|---|---|---|---|---|
| 0 | 0.82 | 0.88 | 0.91 | 530 |
| 1 | 0.71 | 0.78 | 0.78 | 470 |

It can be seen in Table V that BERT has better precision compared to both RNN and LSTM. However, the difference between LSTM and BERT is quite nominal here.

Table VI: RECALL FOR DATASET [6]

| Labels | RNN | LSTM | BERT | Instances |
|---|---|---|---|---|
| 0 | 0.71 | 0.78 | 0.79 | 530 |
| 1 | 0.82 | 0.88 | 0.91 | 470 |

It can be seen from Table VI that although BERT outperformed RNN, the results of LSTM and BERT are comparably similar, being marginally better. This goes to show that BERT performs reasonably well across different domains and comparatively better or at least similar to the previous models RNN and LSTM as well as other key models.

*E. Discussions*

As it can be seen, BERT performed well in both the datasets as per the thesis which we set out to prove. BERT produced computationally sound and accurate results which proves that BERT could overcome a lot of the issues that RNN and LSTM faced in the past regarding sentiment analysis.

## VI. CONCLUSION

*A. Summary*

This paper sought to compare the performance of BERT with RNN and LSTM on a key subset of Natural Language Processing – Sentiment Analysis. We ran the three models on two different datasets to consider the usability of BERT across different domains. We performed the same preprocessing using NLTK for both the models to maintain a constant environment for the models to work on. We demonstrated that BERT performs better than both the conventional sequential models RNN and LSTM across both the datasets.

*B. Future Research*

For future works, we plan on exploring different class configurations. We have worked with 2 and 3 class configurations for this paper, but we intend to test out multi-class configurations. We also intend to apply BERT on different other domains and apply it on different datasets in order to explore the possibility of using BERT on those domains.

*C. Open Problems*

Although BERT performs with good accuracy on sentiment analysis tasks, it still has some issues that are yet to be addressed or fixed. BERT is a pretty large model which takes up a lot of storage. Dealing with class imbalances in the dataset is also a problem which haunts BERT. Another issue that still persists with BERT is that it is yet to considered how to scale up for larger datasets.

## VII. REFERENCES


[1] G. Vinodhini and R. M. Chandrasekaran, "Sentiment analysis and opinion mining: a survey." *International Journal 2.6*, pp. 282-292, 2012.

[2] J. Devlin, et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”, *arXiv e-prints*, 2018.

[3] Larry R. Medsker, and L. C. Jain, "Recurrent neural networks." *Design and Applications*, pp. 64-67 p.2, 2001.
[4] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory", *Neural Computation*, vol. 9, no. 8, pp. 1735-1780, 1997.
[5] Prakar Rathi, Amar Sharma, Meet Vardoriya, "Google Play Store Reviews", *Kaggle,* Available: https://www.kaggle.com/datasets/prakharrathi25/google-play-store-reviews, 2020.
[6] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts, "Learning Word Vectors for Sentiment Analysis", *The 49th Annual Meeting of the Association for Computational Linguistics*, 2011.
[7] Radford, A., Jeffrey Wu, R. Child, David Luan, Dario Amodei and Ilya Sutskever. "Language Models are Unsupervised Multitask Learners.", 2019.
[8] Liu, Yiheng, et al. "Summary of chatgpt/gpt-4 research and perspective towards the future of large language models." *arXiv preprint arXiv:2304.01852*, 2023.
[9] Fabio Duarte, "Number of ChatGPT Users (2023)", *Exploding Topics,* 16-May-2023. [Online]. Available: https://explodingtopics.com/blog/chatgpt-users#:~:text=users%20are%20American-,How%20Many%20ChatGPT%20Users%20Are%20There%3F,February%202023%20to%20April%202023). [Accessed: 03-Jul-2023].
[10] Howard, J. and Ruder, S., "Universal Language Model Fine-tuning for Text Classification", *arXiv e-prints*, 2018.
[11] Shoeybi, M., Patwary, M., Puri, R., LeGresley, P., Casper, J., and Catan- zaro, B., "Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism", *arXiv e-prints*, 2019.
[12] C. Rosset, "Turing-NLG: A 17-billion-parameter language model by Microsoft," *Microsoft Research*, 13-Feb-2020. [Online]. Available: https://www.microsoft.com/en-us/research/blog/turing-nlg-a-17- billion-parameter-language-model-by-microsoft/. [Accessed: 03-Jul-2023].
[13] Wang, Xinyu, et al. "A depression detection model based on sentiment analysis in micro-blog social network." *Trends and Applications in Knowledge Discovery and Data Mining: PAKDD 2013 International Workshops: DMApps, DANTH, QIMIE, BDM, CDA, CloudSD, Gold Coast, QLD, Australia,* April 14-17, 2013, Revised Selected Papers 17. Springer Berlin Heidelberg, 2013.
[14] Vanaja, Satuluri, and Meena Belwal. "Aspect-level sentiment analysis on e-commerce data." *International Conference on Inventive Research in Computing Applications (ICIRCA),* IEEE, 2018.
[15] Bird, Steven, Edward Loper and Ewan Klein, Natural Language Processing with Python, O'Reilly Media Inc, 2009.
[16] Pan, S. J., & Yang, Q., "A survey on transfer learning" *IEEE Transactions on knowledge and data engineering*, 22(10), pp. 1345-1359, 2009.
[17] P. Joshi, "Transfer Learning NLP: Fine Tune Bert For Text Classification," *Analytics Vidhya,* 26-Jul-2020. [Online]. Available: https://www.analyticsvidhya.com/blog/2020/07/transfer-learning-for-nlp-fine-tuning-bert-for-text-classification/. [Accessed: 03-Jul-2023].
[18] R. DiPietro, Gregory D. Hager, 'Handbook of Medical Image Computing and Computer Assisted Intervention". Ch 21, pp. 503-519, 2020.
[19] Javaid Nabi, "Recurrent Neural Networks (RNNs)", *Towards Data Science*, 11-Jul-2019 [Online]. Available: https://towardsdatascience.com/recurrent-neural-networks-rnns-3f06d7653a85 [Accessed: 03-Jul-2023].
[20] A. Chugh, "Deep Learning | Introduction to Long Short Term Memory", *GeeksForGeeks*, 31-May-2023. [Online] Available: https://www.geeksforgeeks.org/deep-learning-introduction-to-long-short-term-memory/. [Accessed: 03-Jul-2023]
[21] A. Mittal, "Understanding RNN & LSTM", *Medium*, 12-Oct-2019. [Online]. Available: https://aditi-mittal.medium.com/understanding-rnn-and-lstm-f7cdf6dfc14e. [Accessed: 03-Jul-2023]

[22] Sherstinsky, A. “Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network” *Physica D: Nonlinear Phenomena, 404*, p.132306, 2020.

[23] Vaswani, A., “Attention Is All You Need”, *arXiv e-prints*, 2017.

[24] Rani Horev, “BERT Explained: State of the art language model for NLP”, *Towards Data Science*, 10-Nov-2018. [Online]. Available: https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270. [Accessed: 03-Jul-2023]

[25] L. Buitinck et al. “API design for machine learning software: experiences from the scikit-learn project,” *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 2013, pp. 108–122.

[26] Abadi, Martin et al., “Tensorflow: A system for large-scale machine learning”, *In 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pp. 265–283, 2016

[27] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, “Array programming with NumPy,” *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. [Online]. Available: https://doi.org/10.1038/s41586- 020-2649- 2

[28] Inc., P.T., “Collaborative data science”, 2015. [Available: https://plot.ly.]

[29] Mickel Hoang, Oskar Alija Bihorac, and Jacobo Rouces. “Aspect-Based Sentiment Analysis using BERT”, *Proceedings of the 22nd Nordic Conference on Computational Linguistics*, pp 187–196, 2019.

[30] L.Zhang, S.Wang, and B.Liu, "Deep learning for sentiment analysis: A survey," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 8, no. 4, p. e1253, 2018.

[31] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, “Distributed representations of words and phrases and their composi- tionality,” in Advances in neural information processing systems, 2013, pp. 3111–3119.

[32] D. Zimbra, A. Abbasi, D. Zeng, and H. Chen, “The state-of-the-art in twitter sentiment analysis: a review and benchmark evaluation,” *ACM Transactions on Management Information Systems (TMIS)*, vol. 9, no. 2, p. 5, 2018.