A Seminar Report on:

“Transformers”

Prepared by : <u>Krunal Rajeshbhai Rank</u>

Roll. No. : <u>U18CO081</u>

Class : B.Tech –IV (Computer Engineering) 7th Semester

Year : 2021-22

Guided by : <u>Shivangi Modi</u>

Department of Computer Engineering

Sardar Vallabhbhai National Institute of Technology,

Surat -395007 (Gujarat), India

Sardar Vallabhbhai National Institute of Technology,

Surat -395007 (Gujarat), India

<u>CERTIFICATE</u>

This is to certify that the seminar report entitled **<u>Transformers</u>** is prepared and presented by **<u>Krunal Rajeshbhai Rank</u>** bearing Roll No. : **<u>U18CO081</u>**, 4th Year of **B. Tech (Computer Engineering)** and his/her work is satisfactory.

GUIDE	**JURY**	**HOD**
(Shivangi Modi)		**COED**

Abstract

The most prominent sequence to sequence machine learning models involve usage of complex convolutional networks as well as stacking them on top of recurrent neural networks that allow the input data to be processed considering the time frame in which the input data has been recorded. It was only recently that this idea was challenged by yet another machine learning concept presented by the researchers from Google Brain and University of Toronto. They presented a proposal of processing the sequential data into two phases, namely, encoding and decoding, which gave rise to Transformers. This seminar report highlights the limitations of the existing techniques which gave birth to transformers as well as some of the salient features and advantages of using transformers instead of the existing technology. The report also presents one of the several use cases offered by transformers and how transformers are modified to achieve use case specific goals and results.

Keywords : Sequence to Sequence Machine Learning - Recurrent Neural Networks - Transformers - Speech to Text Recognition

Contents

List of Figures

Acronyms

ASR Automatic Speech Recognition. 3, 9, 16

BERT Bidirectional Encoder Representations from Transformers. iii, v, 2, 3, 9, 12–14, 16

DNA Deoxyribose Nucleic Acid. 2

EOS End of Sentence. 10

GPU Graphics Processing Unit. 1

GRU Gated Recurrent Units. 1, 3, 5, 16, 19

LSTM Long Short Term Memory. 1, 3, 6, 16, 19

MHA Multi Head Attention. 1, 11

MLM Masked Language Model. 12, 13

NLP Natural Language Processing. 12, 13, 16

NSP Next Sentence Prediction. 12, 13

RNA Ribo Nucleic Acid. 2

RNN Recurrent Neural Network. 1, 3, 4, 16, 19

SOS Start of Sentence. 10

Chapter 1

Introduction

Recurrent Neural Network, Long Short Term Memory and Gated Recurrent Units were firmly established as the state of the art approaches in sequence to sequence machine learning and transduction problem domain such as automatic language translation and language modeling. There were many efforts made to push the boundaries of the available architectures[7][8][9].

The major drawback of RNN is the sequential nature of processing input data of a single RNN unit. This preludes the RNN's parallelization within the training examples. Recent work regarding the factorisation techniques[10] and conditional computation[11] have achieved somewhat improvements as well an improvement in model performance but the drawback of sequential processing still remains the same.

The paper on Transformers[3] presents a proposal of Self Attention and Multi Head Attention that allows modelling of dependencies without keeping in mind their distance in the input or output sequences. Transformer came out to be a model architecture that abandoned recurrence and completely relied on attention mechanisms to derive global dependencies between input and output. This allowed parallelization and allowed to reach new state of the art results in various seqeuence to sequence learning tasks such as machine translation.

1.1 Applications

With the emergence of Transformers, several use cases have achieved significant results which were exceedingly better than the previous existing benchmarks. Transformers allow parallel processing of the input data both in the encoder as well as the decoder section that makes the training process harness the capabilities of the current generation Graphics Processing Unit.

Some of the areas where transformers are currently used are as follows:

- Machine Translation :- With an availability of a large amount of data in more than one languages, it is now possible to train computers to translate a sequence of phrases and sentences from one language to another with the help of transformers.
- Summarization :- Extractive Summarization, at the root, is an optimal subset problem, wherein, a subset of sentences from the given document need to be chosen that contains the maximum information.

- Speech to Text Recognition :- Speech to Text Recognition is one of the most important use cases when it comes to subtitle generation as well as recognizing valuable keywords from raw speech data. This has led to a rise in the number of Speech controlled Personal Assistants. Transformers play a crucial role in this domain.
- Named Entity Recognition :- This problem pertains to recognizing proper nouns and names present in a given sentence or a document that helps in optimizing specific search results in a search engine.
- Biological Sequence Analysis :- It is the process of subjecting a DNA, RNA or peptide sequence to many analytical techniques to understand its functions, features, structures or evolution in general.

This may only be some of the few most important use cases out of several others. Basically, transformer is a tool worth trying out for any observed sequential input data to extract the features and trends of the sequential data and put them into use.

1.2 Motivation

The motivation behind choosing Transformers as a research topic lies behind the fact that technology has kept changing for the better and mankind has reached to a point where it is finally able to achieve results significantly closer to the ones obtained by human brain.

Transformers have provided researchers and developers a new domain of Sequence to Sequence Machine Learning that can be explored even further than the current findings. It required a lot of patience and persistence to reach the current position in this particular field.

The way transformers handle sequential data is something worth studying in a detailed manner. This has been the source of motivation.

1.3 Objectives

The primary objective of this Seminar Report is to highlight the study and research based on the following topics:

- Limitations of existing Sequence to Sequence Machine Learning Techniques
- Self Attention
- Transformers and parallel processing

The secondary objective of this Seminar Report is to discuss a use case of Transformer as well as one of the improved transformers - Bidirectional Encoder Representations from Transformers.

1.4 Contribution

To achieve required objectives, a thorough study of existing literature explaining the current Sequence to Sequence Machine Learning techniques such as RNN, LSTM, GRU as well as study of the literature explaining Transformers, Automatic Speech Recognition and BERT has been conducted. The report contains information based on all the referred literature.

1.5 Organization of Report

Below are the descriptions of the contents presented in each of the chapters in this seminar report.

- Chapter 1 : Presents a brief introduction on the Report, the applications of Transformers, motivation and objectives of the Report.
- Chapter 2 : Presents Theoretical Background regarding the technology namely RNN, LSTM and Transformers.
- Chapter 3 : Presents one of the use cases of Transformers in the field of Automatic Speech Recognition and one of the improved transformers - BERT.
- Chapter 4 : Concluding Remarks based on the study conducted

Chapter 2

Theoretical Background

This chapter presents an overview of the technologies dealing with sequential data as a part of sequence to sequence machine learning. Sequential data is quite different from the other kinds of data, such as those obtained from images or those that have distinct features. Sequential data, also known as time series data, consists of values for the same features at different time intervals or timestamps. It is usually of variable length. The problem of predicting the upcoming values in sequential data can not be solved seamlessly using conventional methods. Recurrent Neural Network have been proposed to solve this problem.

2.1 Recurrent Neural Network

Sequence-to-Sequence data cannot be handled by just standard Neural Networks due to two main reasons: a. Input and output length can be different in different training examples. b. Features are not shared across different positions of text.

Recurrent Neural Network (RNN)[12] are an extension of a conventional feed-forward neural network for handling variable length sequence inputs. RNNs can handle time series because they have a recurrent hidden layer whose activation depends on activation of previous input data in the time series. A simple structure of RNN can be seen below. x, h and o in the figure below represents input state, hidden state and output states respectively. U, V and W are weights of the network.

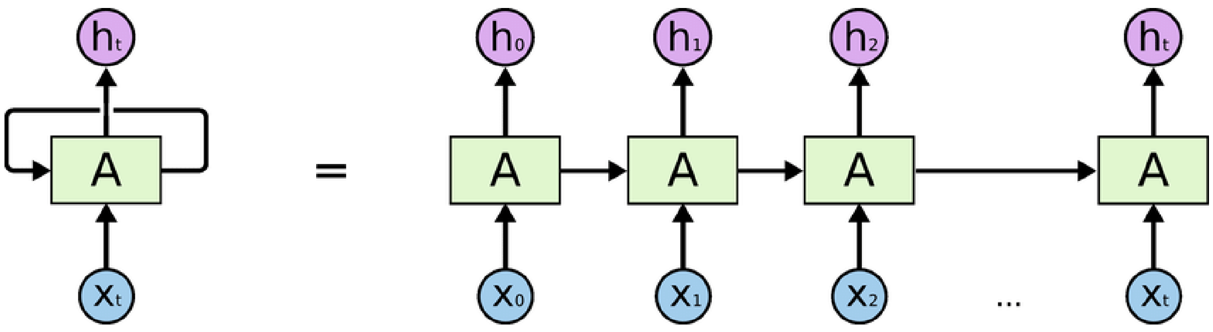

Figure 2.1: RNN Structure[1]

Provided an input sequence $x = (x_1, x_2, x_3, ...x_T)$, the RNN updates its recurrent hidden state h_t by

$$h_t = \begin{cases} 0, & \text{if } t = 0 \\ \phi(h_{t-1}, x_t), & \text{otherwise} \end{cases}$$

where ϕ is a nonlinear function like logistic sigmoid. Optionally, it may have an output $y = (y_1, y_2, ..., y_T)$ of any variable length. The update of the hidden state of RNN is implemented as:

$$h_t = g(Wx_t + Uh_{t-1})$$

However, RNNs may not capture long term dependencies because the gradients tend to either vanish or explode. The exploding gradient problem is generally solved by methods like gradient clipping, in which the gradient vector is clipped once it exceeds a certain threshold. However, solving vanishing gradients is much more difficult.

2.2 Gated Recurrent Networks

To solve the vanishing gradient problem, a variant of RNN having a more sophisticated activation function was introduced called Gated Recurrent Units (GRU)[13]. Each unit can capture dependencies of various time scales. It uses gates to modulate the flow of information inside the unit.

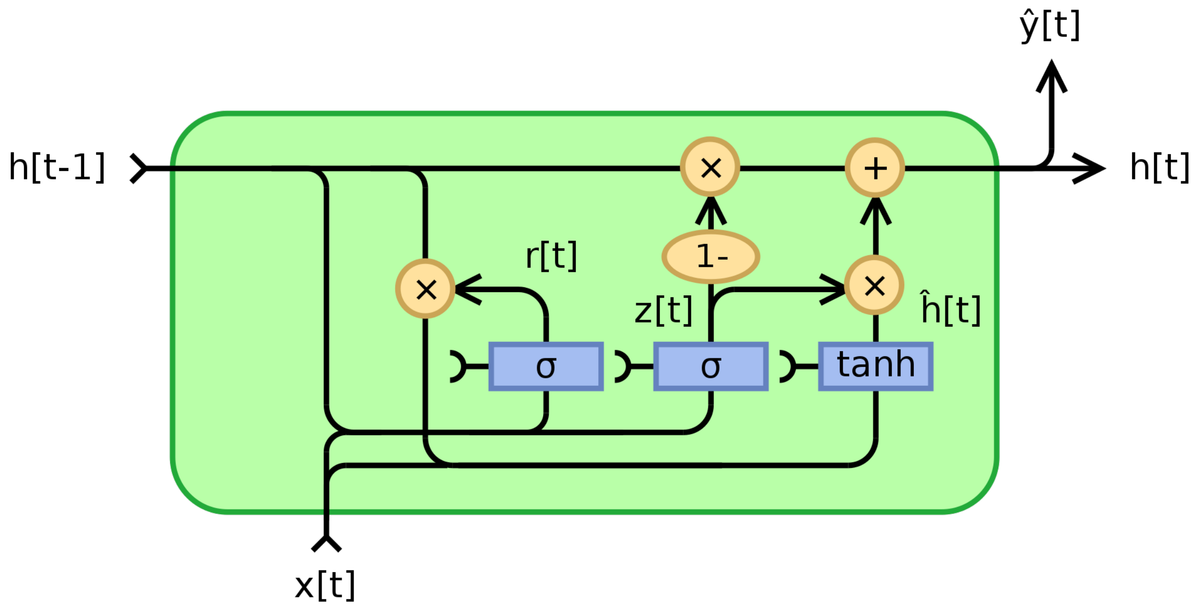

Figure 2.2: GRU Structure[2]

The activation function h_t of GRU at time t is given by:

$$h_t = (1 - z_t)h_{t-1} + z_t\tilde{h_t}$$

where z_t decides how much to update the activation function and $\tilde{h_t}$ is the candidate activation function. The update gate is calculated using the formula below:

$$z_t = \sigma(W_z x_t + U_z h_{t-1})$$

and the candidate activation is calculated similarly using:

$$\tilde{h_t} = tanh(Wx_t + U(r_t \cdot h_{t-1}))$$

where $\cdot$ indicates element-wise multiplication and r_t is the reset gate. Similarly we can calculate the reset gate using the formula:

$$r_t = \sigma(W_r x_t + U_r h_{t-1})$$

2.3 Long Short Term Memory

To handle the vanishing gradient problem, another variant of RNN was introduced called LSTM[13]. Each LSTM unit maintains a memory c_t at time t. The output or the activation function of the LSTM h_t is calculated using:

$$h_t = o_t tanh(c_t)$$

where o_t is the output gate which is calculated using

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + V_o c_t)$$

where σ is logistic sigmoid function. V_o is a diagonal matrix.

The memory cell c_t is updated by partially forgetting existing memory and partially adding a new memory:

$$c_t = f_i c_{t-1} + i_t \tilde{c_t}$$

where $\tilde{c_t}$ is new memory content, f_i is forget gate, i_t is input gate.

The forget gate manages the extent to which existing memory should be forgotten and the input gate manages the degree to which new memory content is added to memory cell. These gates are calculated using:

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + V_f c_{t-1})$$

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + V_i c_{t-1})$$

Simple RNNs just overwrite the content every time whereas LSTM can decide whether to keep memory or not. In the case where LSTM detects an important feature from input, it captures the information and use them later when needed.

2.4 Transformers : Attention is all you need

RNNs and LSTMs have certain limitations:

- They are slow
- They are sequential in nature

Hence they don't utilize modern day parallel processing power of GPUs. Transformer[3] is a network based solely on attention mechanisms to draw global dependencies between input and output. They allow parallelization of input sequence and take less time to train and hence they have made a significant impact on the development of sequence-to-sequence architectures.

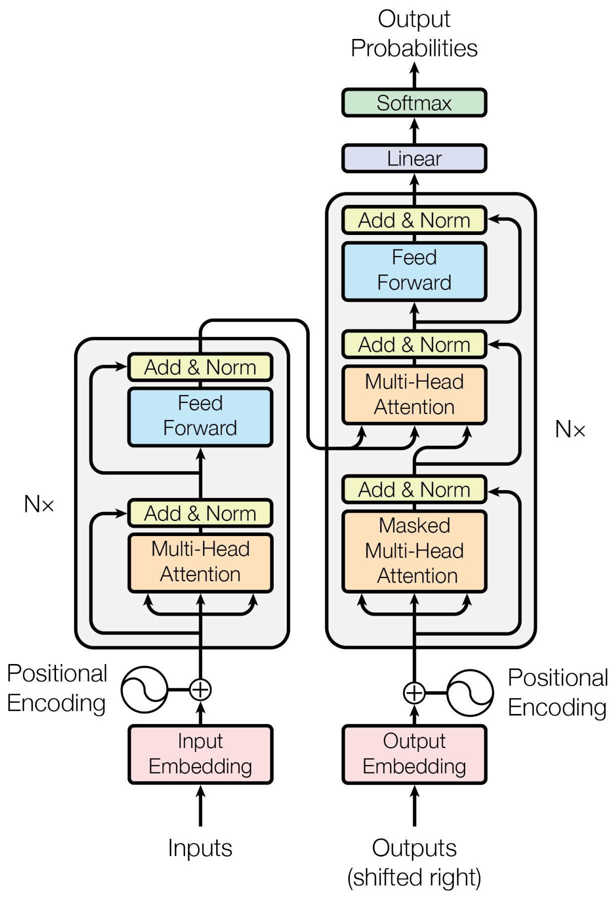

Figure 2.3: Transformer Network[3]

2.4.1 Model Architecture

The input to the transformer is a vector containing input embeddings. Then we perform positional encoding to the input embeddings which takes the positions into account and generates an output vector containing positional information.
Transformer contains the following blocks:

- Encoder: Encoder consists of a stack of N identical layers([3] suggests N = 6. Each layer has 2 sub-layers:

  1. Multi head attention layer
  2. Feed forward neural network

  Each sub-layer is passed through layer normalization

- Decoder: Decoder is also composed of a stack of N identical layers. In addition to the two sub-layers similar to encoder block, decoder inserts a new sublayer which performs multi-head attention over the output of the encoder stack. The self-attention layer in the decoder block is modified such that the subsequent positions after a position are masked. This ensures that the predictions for every position depend only on the positions before it.

2.4.2 Attention

The Attention function for the transformer takes input in the form of queries and keys having dimension d_k and values of dimension d_v. The queries, keys and values are packed into matrices Q, K and V and output attention function is calculated as follows:

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

Instead of using a single attention function, we can linearly project queries, keys and values h times and then perform attention function in parallel yielding d_v dimensional output values which are concatenated and projected again as shown here:

$$MultiHead(Q, K, V) = Concat(head_1, head_2, ...head_h)W^O$$

$$\text{where, } head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

Transformers can replace recurrent layers most commonly used in encoder-decoder architectures with multi-headed self attention.

Chapter 3

Literature Review

This chapter presents one of the use cases of Transformers that is present in the industry currently as well as one of the improved versions of Transformers - Bidirectional Encoder Representations from Transformers.

Automatic Speech Recognition requires us to process raw audio data which is in the form of waves and detect phrases and sentences out of this provided audio data.

Later, the idea of Bidirectional Encoder Representations from Transformers is presented as well as how BERT can be used for another one of the use cases, Text Summarization.

3.1 ASR using Transformers

Encoder - Decoder based Sequence to Sequence models have proved to be state of the art models in Automatic Speech Recognition for a long time. One of such techniques is described in this subsection.

The model architecture proposed in the referred paper is described by the following diagram :

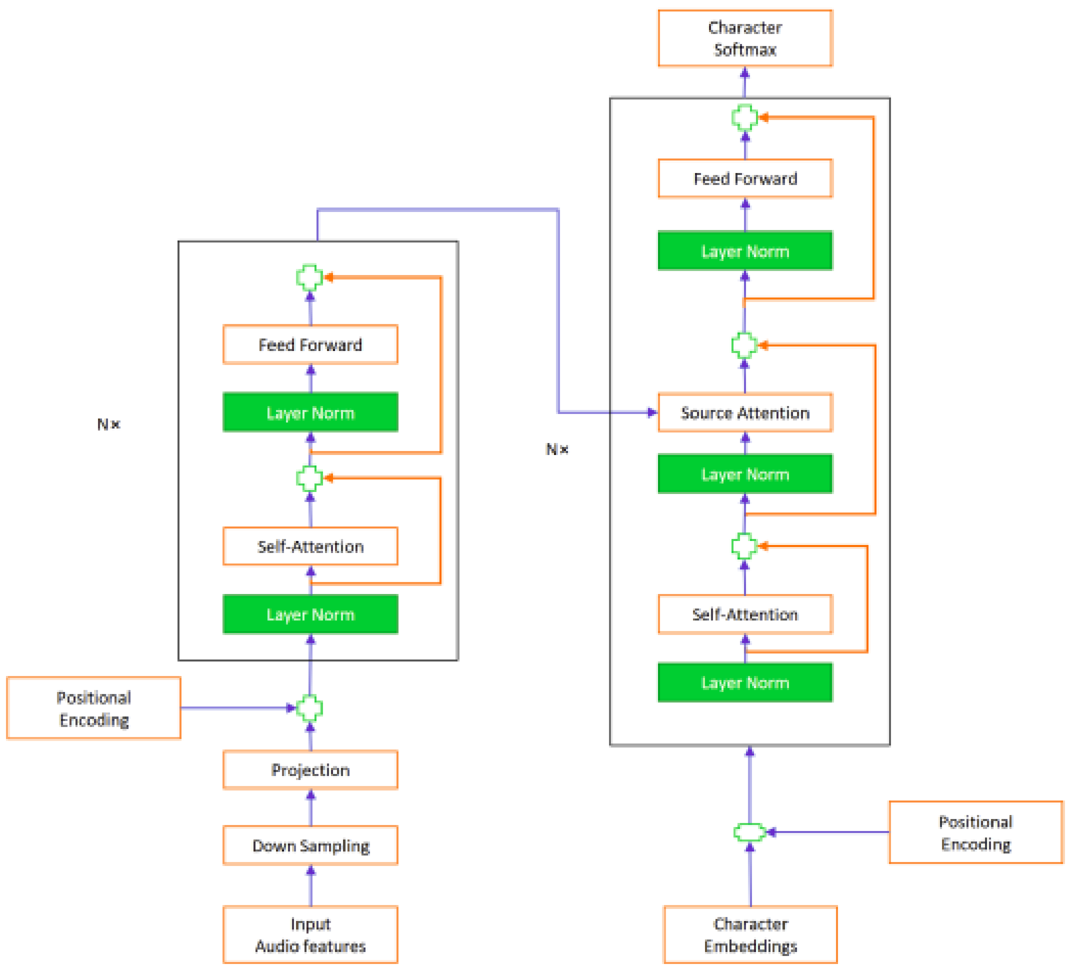

Figure 3.1: An Encoder - Decoder Transformer used to map Acoustic features from audio data to character level transcriptions[4]

3.1.1 Preprocessing Audio Data

The audio data is preprocessed first by converting raw waves into spectrograms that show frequency modifications over time. This can be done by using Short Time Fourier Transform[14] to change audio data domain to time-frequency.

Short Time Fourier Transform is actually obtained as a Complex number. However, we only require the magnitude of the value. The obtained data is then normalised using mean and standard deviation. It is then padded to obtain an audio worth of 10 second duration.

3.1.2 Tokenizing Output Characters

Output is processed character by character. 34 output characters are used as classes that include alphabetic characters, punctuation marks, unknown character markers and a Start of Sentence (SOS) and End of Sentence (EOS) marker. The Output characters are then vectorized using the same set of characters.

3.1.3 Speech Embedding Layer

The speech embedding layer [15] is used to reduce dimensionality of the input data by using 1 Dimensional convolutional layers. This is identical to Down Sampling of given audio data.

An embedding layer is used to create embeddings out of input vectors.

3.1.4 Encoder Layer

The encoder layer first normalises the created speech embeddings and then uses a Multi Head Attention (MHA) Layer. The Q,K,V vectors of the MHA layer are the normalised speech embeddings.

The output of the MHA layer is then added to the residual input layer and normalised [16] similar to the one explained in the Residual Network layer.

It is then passed through a feed forward network that is composed of a linear neural network and again added with the residual input and normalised.

The normalised layers are also made available with dropout layers to prevent overfitting of speech embeddings.

3.1.5 Token Embeddings

Token Embeddings[17] are first created using Output character vectors. These token embeddings are an addition of token vector embeddings and positional embeddings. These embeddings serve as an input for the Decoder.

3.1.6 Decoder

Decoder masks the input Token embeddings and then passes it to the Decoder MHA Layer.

Its output is then fed to the Source Attention MHA layer.

The K,V vectors of Source Attention MHA Layer are the outputs from the encoder. The Q vector is the output of Decoder MHA Layer.

The MHA Layer output is then added with residual input and normalised.

It is then fed to a Feed Forward layer and again added with residual input and normalised.

3.1.7 Calculating Loss

The model is trained using the preprocessed data, using Adam's optimizer [18] and a Learning Rate Schedule. The Decoder output is then passed to the character softmax and its loss is calculated using Categorical Cross Entropy.

3.1.8 Conclusion

Mapping Acoustics directly to characters is actually a challenging task. The above procedure successfully demonstrated how this challenging problem can be approached.

3.2 Bidirectional Encoder Representations from Transformers

3.2.1 Introduction

Bidirectional Encoder Representations from Transformers is a language representation model that is trained on unlabeled text by jointly conditioning on both left and right context in all layers.According to the authors[5], BERT is conceptually simple and empirically powerful, obtaining a state of the art result on Natural Language Processing tasks.

There are two strategies employed to apply language representations to downstream tasks:

- Feature Based Approach :- It uses task specific architectures that include pre-trained representations as additional features. Such an approach is explained as a part of ELMo.
- Fine Tuning Approach :- It uses minimal task specific parameters and focuses on downstream tasks by simply tuning all pre-trained parameters. One such example for this approach is the Generative Pretrained Transformers by OpenAI.

As far as BERT is concerned, there are two steps in the framework:

- Pre-training
- Fine-tuning

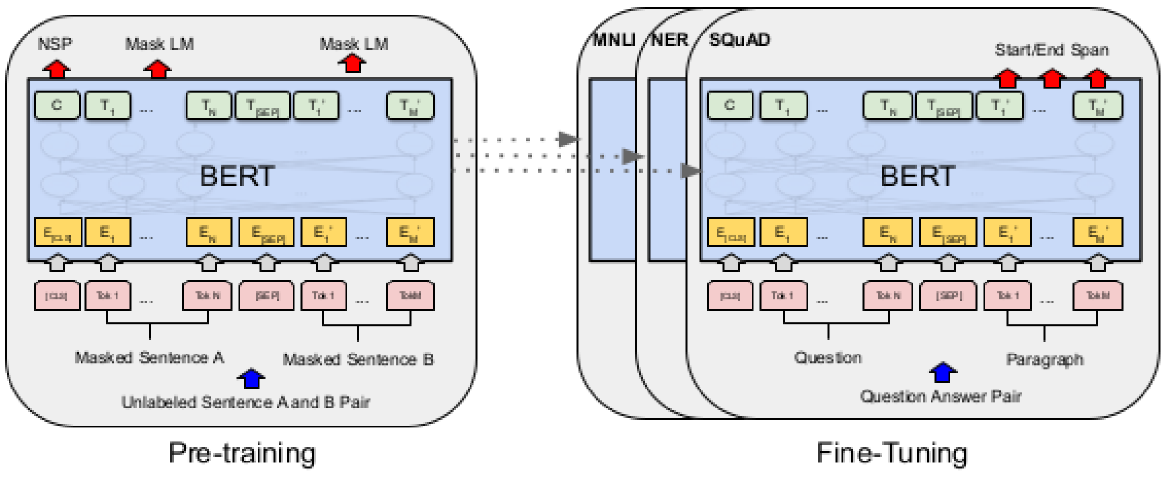

Figure 3.2: BERT[5]

3.2.2 Pre-training BERT

Pre-training BERT has two tasks:

1. Masked Language Model
2. Next Sentence Prediction

Masked Language Model

A standard conditional language model can only be trained on left-to-right or right-to-left context or a shallow concatenation of both of these techniques.

In order to train the deep bidirectional representations, some random tokens in the input are masked and those tokens are predicted. 15% of the tokens are masked in each sequence. Now this may lead to a problem in fine-tuning process where in the [MASK] token is not observed in fine-tuning process. Hence, out of the 15% masked tokens, 80% of these tokens are replaced with [MASK] token, 10% of these tokens are replaced with any random wordpiece and the rest are left unchanged. The other unmasked tokens are then used to predict the masked tokens using a cross entropy loss.

Next Sentence Prediction

Some of the NLP based tasks such as Question Answering and Natural Language Inference are based on understanding the relationship between sentences, something which is not captured in language representations. In order to train a model that understands sentence relationships, a prediction task wherein the model is expected to predict whether the given sentences are actually consecutive sentences or not. Specifically, two sentences are chosen at random by the generator with a 50% chance that the sentence B is the next sentence for chosen sentence A. These two sentences are separated by [SEP] token. This task is somewhat similar to representation learning objectives[19].

3.2.3 Fine-tuning BERT

Fine-tuning BERT is simpler as compared to pre-training process. For every task, the input and output sequences are plugged into BERT and parameters are fine-tuned end-to-end. In pre-training stage, sentences A and B can be analogous to:

- Sentence pairs in paraphrasing
- Hypothesis Premise Pairs
- Question Passage Pairs

Compared to pre-training, fine tuning is relatively inexpensive and can be done pretty much faster.

3.2.4 Conclusion

Recent improvements with the help of transfer learning with language models have demonstrated that rich, unsupervised pre-training is an important part of many language understanding systems. The major contribution of BERT is further general- izing these findings to deep bidirectional architec- tures, which allows the same pre-trained model to suc- cessfully tackle numerous NLP tasks.

3.3 Extractive Summarization with BERT

Summarization strategies are typically categorized as extractive, abstractive or mixed.

- Extractive strategies select the top K sentences that best represent the maximum information of the article.[6]
- Abstractive summaries seek to reproduce the key points of the article in new words. [20]
- Mixed strategies either produce an abstractive summary after identifying an extractive intermediate state or they can choose which approach to use based on the particulars of the text.

Extractive strategies are set up as binary classification problems where the goal is to identify the article sentences belonging to the summary. Extractive summarization is a challenging task that became practical only quite recently. The progress in this domain was due to the super embeddings offered by BERT. BERT can parse meaning from all the nuances of language and steps such as stop word removal, stemming and lower-case transformations are purposely ignored.

3.3.1 Input and Output

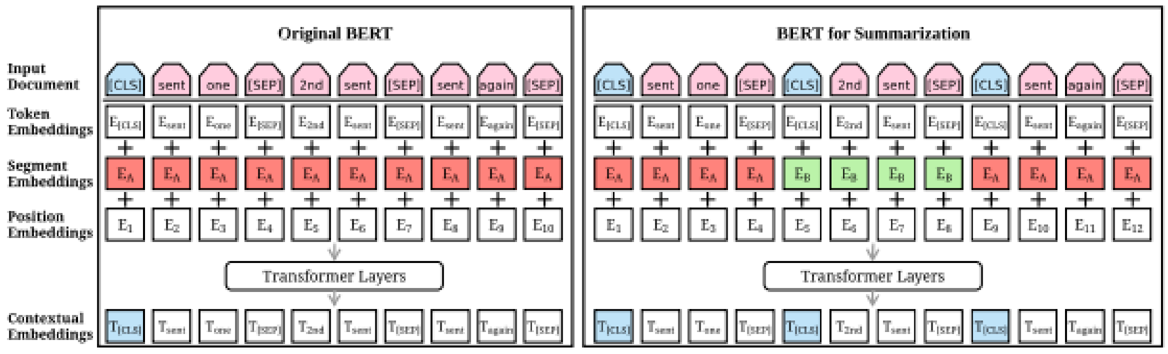

Figure 3.3: Original BERT architecture and modified BERT architecture for summarization (BERTSUM)[6]

The way in which BERTSUM is represented is quite different from the original BERT. The sentences are separated using [CLS] token and features from the previous sentence are collected and processed by BERTSUM to create features for the current sentence. Interval Embedding is also used to demarcate sentences at odd and even positions so that document representations are learned hierarchically where lower Transformer layers represent adjacent sentences, while higher layers, in combination with self-attention, represent multi-sentence discourse.

BERTSUM outputs a list of scores that show representiveness of sentences towards document. Hence, the higher the score is, the more prominent the sentence is.

3.3.2 Summarization Encoder

Summarisation encoder represent individual sentences, by adding external [CLS] tokens at the beginning of each sentence. It collects features for the sentence preceding it.

Interval segment is used to distinguish multiple sentences within the document. For a particular sentence $sent_i$ we assign segment embedding E_A or E_B depending on whether i is odd or even.For example, for document $[sent_1, sent_2, sent_3, sent_4, sent_5]$, we would assign embeddings $[E_A, E_B, E_A, E_B, E_A]$. In this approach, document representations are learnt in a hierarchical manner, with lower Transformer layers representing neighbouring phrases and higher levels representing multi-sentence discourse when combined with self-attention. The original BERT model has a maximum length of 512 position embeddings; we circumvent this constraint by adding additional position embeddings that are randomly started and fine tuned with other encoder settings.

3.3.3 Fine tuning BERT for Summarization

Although BERT has been used to fine-tune a variety of NLP tasks, it is not as simple to apply to summarization. Because BERT is a masked-language model, the output masked vectors are limited to tokens rather than phrases, The majority of extractive summarization methods work with sentence-level representations. In BERT, segmentation embeddings represent various sentences; however, they only apply to sentence pair inputs, but in summarization, multi-sentential inputs must be encoded and manipulated. So we modify our BERT architecture as shown above in the figure in order to summarise (which we call BERTSUM).

3.3.4 Conclusion

BERTSUM Transformer was able to achieve a score of 43.25 ROUGE-1, 20.24 ROUGE-2 and 39.63 ROUGE-L on CNN/DailyMail dataset, which was found to be better than LEAD(an extractive baseline which uses the first-3 sentences of the document as a summary).

Chapter 4

Conclusion

4.1 Conclusion

I was able to thoroughly study the concepts of Recurrent Neural Network and its limitations. The limitations of RNN are filled up by slightly modified versions of RNN - Gated Recurrent Units and Long Short Term Memory. These networks require sequential processing and thereby weren't able to utilise the modern day GPU Parallel Processing, which was overcome with the advent of another new architecture, Transformers. As I was able to study the above concepts, I also learnt a bit more on one of the use cases of Transformers, Automatic Speech Recognition and one of the improved versions of Transformers, Bidirectional Encoder Representations from Transformers.

4.2 Future Prospects

With the advent of modified transformers like BERT, certain NLP tasks have taken a different turn. In case of summarization, the challenge has shifted from extractive summarization to abstractive summarization that completely changes the problem from choosing the given sentences to creating sentences and phrasings automatically from the provided input text document. Even more advanced models are under research that have claimed to outperform BERT.

Bibliography

[1] Moumita Tora, Jianhui Chen, and J.J. Little. Classification of puck possession events in ice hockey. *Proceedings / CVPR, IEEE Computer Society Conference on Computer Vision and Pattern Recognition. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 07 2017.

[2] Adnan Riaz, Muhammad Nabeel, Mehak Khan, and Huma Jamil. Sbag: A hybrid deep learning model for large scale traffic speed prediction. *International Journal of Advanced Computer Science and Applications*, 11:287–291, 01 2020.

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[4] Ngoc-Quan Pham, Thai-Son Nguyen, Jan Niehues, Markus Müller, and Alex Waibel. Very deep self-attention networks for end-to-end speech recognition. *CoRR*, abs/1904.13377, 2019.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[6] Yang Liu and Mirella Lapata. Text summarization with pretrained encoders, 2019.

[7] Yonghui Wu, Mike Schuster, Zhifeng Chen, and Quoc V. Le. Google's neural machine translation system: Bridging the gap between human and machine translation, 2016.

[8] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation, 2015.

[9] Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. Exploring the limits of language modeling, 2016.

[10] Oleksii Kuchaiev and Boris Ginsburg. Factorization tricks for lstm networks, 2018.

[11] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer, 2017.

[12] Alex Sherstinsky. Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *Physica D: Nonlinear Phenomena*, 404:132306, Mar 2020.

[13] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling, 2014.

[14] Leigh Alsteris and Kuldip Paliwal. Asr on speech reconstructed from short-time fourier phase spectra. 10 2004.

[15] Yu-An Chung and James Glass. Speech2vec: A sequence-to-sequence framework for learning word embeddings from speech, 2018.

[16] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016.

[17] Lifu Tu, Kevin Gimpel, and Karen Livescu. Learning to embed words in context for syntactic tasks, 2017.

[18] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[19] Yacine Jernite, Samuel R. Bowman, and David Sontag. Discourse-based objectives for fast unsupervised sentence representation learning, 2017.

[20] Ramesh Nallapati, Bowen Zhou, Cicero Nogueira dos santos, Caglar Gulcehre, and Bing Xiang. Abstractive text summarization using sequence-to-sequence rnns and beyond, 2016.

Acknowledgement

The opportunity to thoroughly study the concept of Transformers under the domain of Natural Language Processing was certainly worthwhile, presenting gratitude towards my mentor, Ms. Shivangi Modi, who guided me on the basics of Recurrent Neural Network, Gated Recurrent Units, Long Short Term Memory and Transformers. Her contribution towards helping me in creating an ideal Seminar Report as well as presenting my findings is unparalleled.

I would also like to thank Prof. Udai P. Rao for providing with all the required guidelines and necessities to prepare a well curated seminar report and his constant supervision.

I would also like to thank Prof. Mukesh Zaveri, HOD, Computer Engineering Department for supervising the course.