Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 03, March-2021

Decoding Deep Learning Approach to Question Answering

Shivam Marathe
Information Technology
COEP
Pune,India

Mohan Ambekar
Information Technology
COEP
Pune, India

Utkarshbhanu Andurkar
Information Technology
COEP
Pune, India

Abstract **- We have focused on various deep learning methods for Question answering task of information retrieval, a famous problem in NLP. This paper is structured in such a way that we discuss all the concepts relating to various deep learning models i.e. RNN, BERT and T5 with examples. It is easy to relate to each model and it's functioning. As a benefit of it we can select the particular model by focusing on the needs and resources. We have discussed results of these advanced models as well.**

Keywords—NLP, BERT, T5, Attention, Transfer Learning, Question Answering (QA), RNN, Teacher Forcing, Transformer

I. INTRODUCTION

Improved performance and reduced human efforts are leading to increase in popularity of deep learning techniques for natural language processing tasks. In this paper, we will review use of some deep learning models to one such NLP task which is "Question Answering". Particularly we will focus on comparison of BERT and T5(11B) model for context-based QA. We will also touch on to several concepts to understand working of these models.

II. TRADITIONAL MODEL

A. RNN (seq2seq)

RNNs can process sequential data, so they will be one of the primary deep learning techniques used for Question Answering task. Baseline model used in RNN approach is seq2seq. Here RNN encoder first processes the context followed by the special separator and then the question. Special separator is used to signal the model to start decoding with decoder's initial state being final state of encoder. Decoder then produces the answer sequence followed by the stop symbol that indicates the end of processing. Seq2seq gets trained using cross entropy method on decoder. During training correct answer sequence is provided while during validating/testing, processing is done till question only.

B. Advantages and Disadvantages

Only advantage of RNN for QA task is, its' machine learning pipeline is smaller than BERT/T5. But the sequential nature of RNNs makes it more difficult to fully take advantage of modern fast computing devices such as TPUs and GPUs, which excel at parallel and not sequential processing, thereby making learning over RNN slow compared to T5/BERT [1]. Long-term information has to sequentially travel through all cells before getting to the present processing cell. This means it can be easily corrupted by being multiplied many times by small numbers (less than 1) during backpropagation. This is the cause of vanishing gradients. LSTM and GRU offer partial solution to this problem. But in case of longer text sequences like in QA, even LSTM/GRU are of little significance [2].
RNNs become very ineffective when the gap between the relevant information and the point where it is needed becomes very large. This is due to the fact that the information is passed at each step and the longer the chain is, more probable is the loss of information along the chain. But this is exactly the case for Question Answering, as model has to process through all of the context and question to produce answer, therefore the chances of information loss are high in case of QA-task [3].

III. RELATED CONCEPTS

Before we talk more about BERT and T5 model, let us learn some underlying concepts used in them. We will explain these concepts abstractly and will provide references for enthusiastic readers.

A. Attention

Attention models, or attention mechanisms, are input processing techniques for neural networks that allows the network to focus on specific aspects of a complex input, one at a time until the entire dataset is categorized. The goal is to break down complicated tasks into smaller areas of attention that are processed sequentially. Similar to how the human mind solves a new problem by dividing it into simpler tasks and solving them one by one. Attention models require continuous reinforcement or backpropagation training to be effective.
In broad strokes, attention is expressed as a function that maps a query and a set of key value pairs to an output. One in which the query, keys, values, and final output are all vectors. The output is then calculated as a weighted sum of the values, with the weight assigned to each value expressed by a compatibility function of the query with the corresponding key value.
The most work so far with attention mechanisms has focused on Neural Machine Translation (NMT). Traditional automated translation systems rely on massive libraries of data labelled with complex functions mapping each word's statistical properties.
Using attention mechanisms in NMT is a much simpler approach. Here, the meaning of a sentence is mapped into a fixed-length vector, which then generates a translation based on that vector as a whole. The goal isn't to translate the general, "high level" overall sentiment. Besides drastically

improving accuracy, this attention-driven learning approach is much easier to construct and faster to train.
The context vector turned out to be a bottleneck for the seq2seq type of model. It made it challenging for the models to deal with long sentences. Attention allows the model to focus on the relevant parts of the input sequence as needed. Encoder passes all the hidden states to the decoder. Ability to amplify the signal from the relevant part of the input sequence makes attention models produce better results than models without attention.

1) Self Attention :
   A layer that helps the encoder look at other words in the input sentence as it encodes a specific word.
   Say the following sentence is an input sentence we want to translate:
   "The animal didn't cross the street because it was too tired"
   What does "it" in this sentence refer to? Is it referring to the street or to the animal? It's a simple question to a human, but not as simple to an algorithm.
   When the model is processing the word "it", self-attention allows it to associate "it" with "animal".
   As the model processes each word (each position in the input sequence), self attention allows it to look at other positions in the input sequence for clues that can help lead to a better encoding for this word.

2) Multihead Attention:
   It is the layer used in many models for focusing on relevant parts of the input sentence.
   For eg. in the above example of self attention though the word 'it' will associate with other words in the sentence, the effect of other words on its attention output will be suppressed by itself.
   By using parallel heads initialized differently, we get output of attention on 'it' as a proper combination and there won't be any suppression effect.
   Here we use multiple blocks of self attention.
   Layers of attention mechanism used in deep learning models:
   a) Encoder Decoder attention layer - Queries are taken from Decoder.
   b) Causal attention layer - Self attention without using future reference
   c) Bidirectional self attention - Self attention using all words from right and left side.

B. Teacher Forcing

Use of ground truth during training ie passing *previous target word* as next input to the decoder is known as Teacher forcing. Eg. Suppose we want to translate "How are the results" to its' German equivalent "Wie sind die Ergebnisse". Decoder produces this translation step by step ie one word at a time. At each step previously predicted word is used as input to decoder to produce next word. Suppose following is predicted translation without teacher forcing- "Wie geht zu Hause?". We can see, second predicted word "geht" is wrong. As we have used this word as input to the decoder in third step the next word "zu" is also wrong. Now we can understand the problem – Use of incorrect prediction as input for next step is detrimental because it may produce incorrect results. Teacher forcing helps in this situation. In teacher forcing technique, at each step, although we use predicted word for evaluation (calculating error), we use ground truth (in our example "sind" at third step) during training. Teacher forcing helps model to reach convergence faster during training. Without teacher forcing errors will accumulate producing sequence of wrong predictions. Limitation of this technique is Exposure Bias problem [4]. As during inference, we don't have ground truth available, we won't be able to apply teacher forcing. There will be discrepancy between training and inference which may lead to poor model performance.

C. Transfer Learning

Transfer learning is a training technique in NLP, where model is trained on data rich tasks, which is subsequently trained on downstream task to produce predictions on downstream task e.g., suppose we want to train a model for neural machine translation (NMT), so we use transfer learning and train our model first on combination of many different tasks like Question Answering, STSB, MNLI to learn model weights. Now there are two ways (approaches) to proceed

1) Feature based - Here, we use learned model weights to extract features (like word embeddings) which are used to train another model with NMT data.

2) Fine tuning – In this approach, we use learned weights with same model and train it using NMT data.

The tasks for which data set size is comparatively small gains most from transfer learning. Model learns many natural language features which can not be learned using only downstream task. Transfer learning also reduces training time and improves prediction. Transfer learning is used during training in both BERT and T5.

D. Transformer Architecture

The Transformer is a novel architecture that aims to solve sequence-to-sequence tasks while handling long-range dependencies with ease. The Transformer was proposed in the paper "Attention Is All You Need" [5]. The transformer is the first transduction model relying entirely on self-attention to compute representations of its input and output without using sequence-aligned RNNs or convolution. Here, "transduction" means the conversion of input sequences into output sequences. The idea behind Transformer is to handle the dependencies between input and output with attention and recurrence completely.

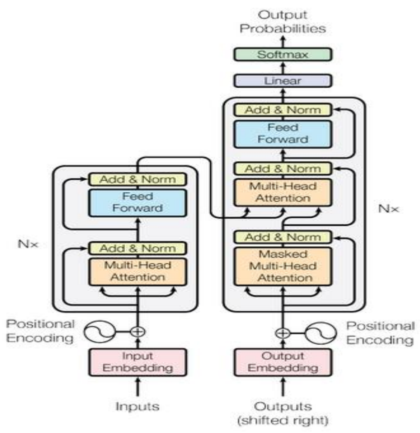

Fig 1: Transformer Architecture [6]

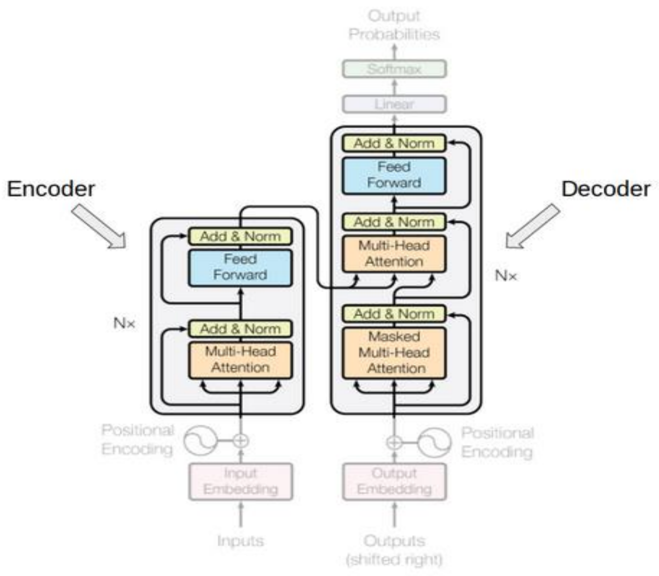

Fig 2: Encoder decoder structure

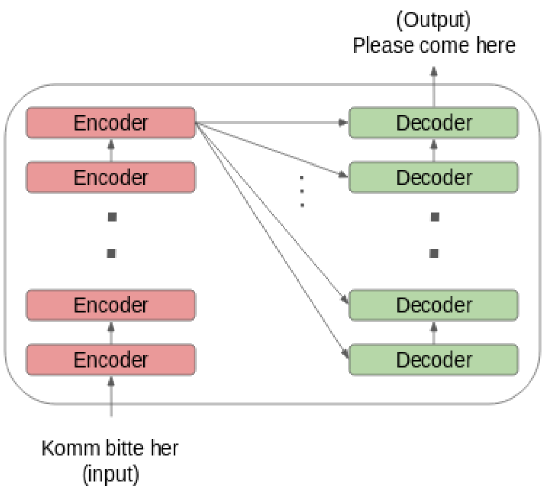

Fig 3: Encoder decoder stack

In the architecture of transformer encoder and decoder are two important parts. The Encoder block has 1 layer of a Multi-Head Attention followed by another layer of Feed Forward Neural Network. The decoder, on the other hand, has an extra Masked Multi-Head Attention. The encoder and decoder blocks are actually multiple identical encoders and decoders stacked on top of each other. Both the encoder stack and the decoder stack have the same number of units. The number of encoder and decoder units used is a *hyperparameter*. In the paper [5], 6 encoders and decoders have been used.

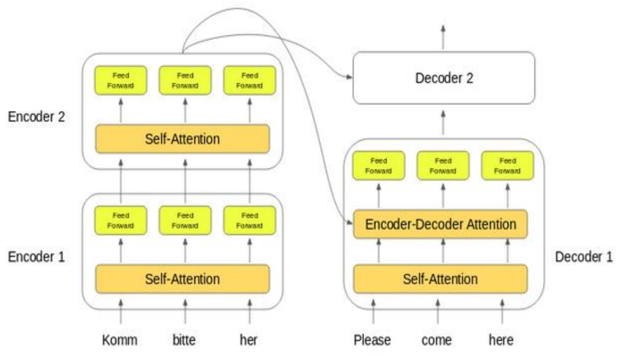

Fig 4: Translation using transformer

The word embeddings of the input sequence are passed to the first encoder, these are then transformed and propagated to the next encoder. The output from the last encoder in the encoder-stack is passed to all the decoders in the decoder-stack. In addition to the self-attention and feed-forward layers, the decoders also have one more layer of Encoder-Decoder Attention. This helps the decoder focus on the appropriate parts of the input sequence.

For calculation of self Attention three vectors are created from each of the encoder's input vectors: Query Vector, Key Vector, Value Vector. These vectors are trained and updated during the training process. Next, self attention for every word in the input sequence is calculated. To calculate the self-attention for the first word, scores for all the words in the phrase with respect to the first word are calculated. This score determines the importance of other words when we are encoding a certain word in an input sequence. The score for the first word is calculated by taking the dot product of the Query vector with the keys vectors of all the words. Then, these scores are divided by square root of the dimension of the key vector. Next, these scores are normalized using the softmax activation function. These normalized scores are then multiplied by the value vectors and sum up the resultant vectors to arrive at the final vector, this is the output of the self-attention layer. It is then passed on to the feed-forward network as input. So final vector is the self-attention vector for the first word of the input sequence. We can get the vectors for the rest of the words in the input sequence in the same fashion.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 03, March-2021

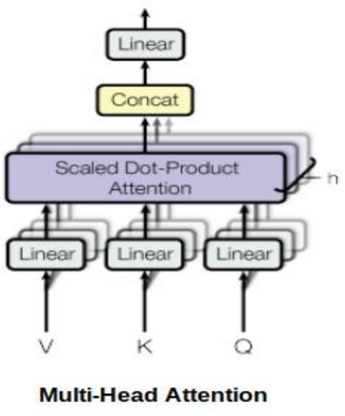

Fig 5

Self-attention is computed not once but multiple times in the Transformer’s architecture, in parallel and independently. It is therefore referred to as Multi-head Attention. Next, the outputs are concatenated and linearly transformed. Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions.

IV. BERT MODEL

A. Architecture

Earlier only unidirectional approach was used where tokens were used to attend previous tokens for self attention but for applying a fine-tuning approach, we need to incorporate context from both the directions. BERT uses a masked language model for this task. There are two main models. BERT$_{BASE}$ (L=12, H=768, A=12, Total Parameters=110M) and BERT $_{LARGE}$ (L=24, H=1024, A=16, Total Parameters=340M). L is for layers, H is for hidden units, A for self attention heads.

It can handle one sentence as well as multiple sentences as input. Sentences need not be in meaningful manner. The first token of every sequence is always a special classification token ([CLS]). Sentence can be separated by ([SEP]) token

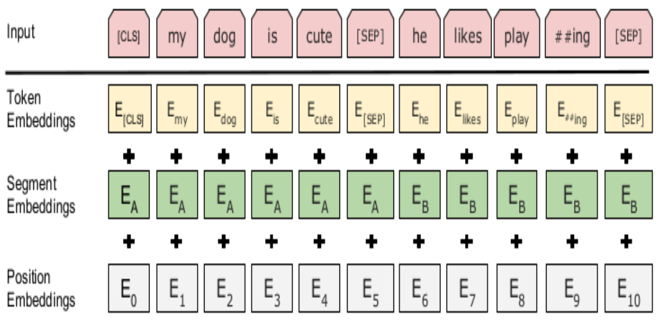
Input
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} E_{dog} E_{is} E_{cute} $E_{[SEP]}$ E_{he} E_{likes} E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Segment Embeddings
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Position Embeddings
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

Fig6: BERT input representation [7]

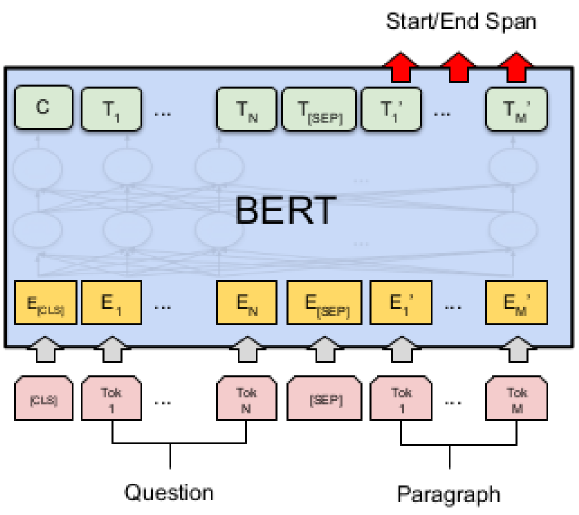

Fig 7: BERT for QA task [7]

B. Pretraining (Objective)

Model is trained on unlabeled data over different pre-training tasks. Each downstream task is then fine tuned with labelled data and initialized parameters taken from pre-training. It is pre-trained on two supervised tasks. We use bidirectional conditioning for pre-training where we mask

a) 80% of the token with [M A S K] token
b) 10% of the token with random token
c) 10% of the token unchanged.

Next sentence prediction is like a binary classification problem whether next sentence follows the input sentence or not. For eg. A: Raju lives in Delhi. B: He is a data scientist. Here output can be ‘yes B follows A’. NSP is beneficial for QA.

Datasets used for pre-training are BooksCorpus (800M words) and English Wikipedia (2,500M words).

C. Fine Tuning

All we need to fine tune BERT for QA task is to replace fully connected output layer with fresh set of output layers that can basically output answer to the question that we want. Then we can perform supervised training using QA dataset. It won’t take long since its only output parameters that are learned from scratch. Rest of the parameters are slightly fine tuned, leading to faster training.

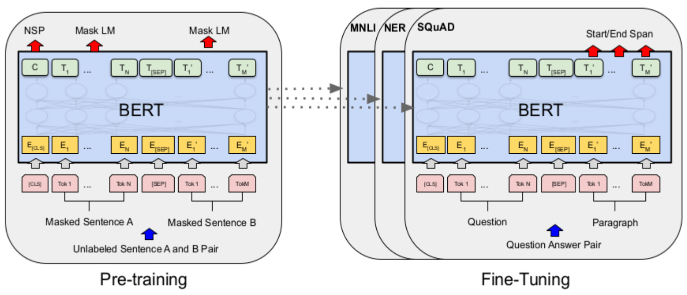

Fig 8: Pretraining and fine tuning in BERT [7]

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 03, March-2021

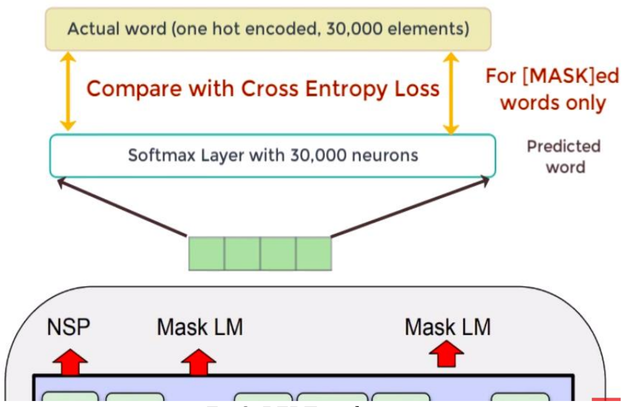
Actual word (one hot encoded, 30,000 elements)
Compare with Cross Entropy Loss
For [MASK]ed words only
Softmax Layer with 30,000 neurons
Predicted word
NSP
Mask LM
Mask LM

Fig 9: BERT predictions

D. Results

Model	SQuAD EM	SQuAD F1
Previous best (On test set)	90.1	95.5
BERT LARGE	80.0	83.1
Human Performance	82.30	91.22

Table 1: Results are reported on SQuAD validation set [7].

V. T5(11B) MODEL

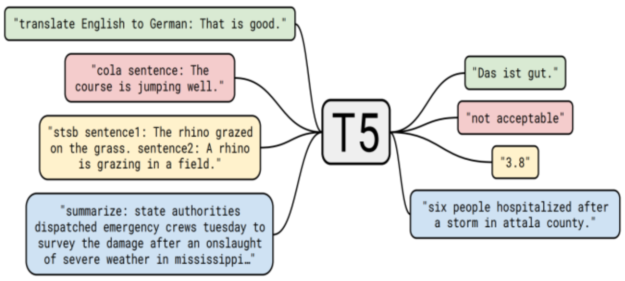
"translate English to German: That is good."
"cola sentence: The course is jumping well."
"stsb sentence1: The rhino grazed on the grass. sentence2: A rhino is grazing in a field."
"summarize: state authorities dispatched emergency crews tuesday to survey the damage after an onslaught of severe weather in mississippi…"
T5
"Das ist gut."
"not acceptable"
"3.8"
"six people hospitalized after a storm in attala county."

Figure 10: Applications of T5 [8]

Researchers at “Google” worked on “Text to text framework” for all NLP tasks [8]. T5(11B) is one of the attention-based models they invented (largest in term of scale), which achieved state of the art result on nearly every NLP task. Following subsections will get you through T5(11B)’s architecture, pretraining, fine tuning and results on QA task specifically.

A. *Architecture*

Unlike BERT which has decoder only architecture, T5(11B) has a 24 layer encoder and decoder architecture with 128 headed multihead attention. The “key” and “value” matrices of all attention mechanisms used have an inner dimentionality of d_{kv} = 128, output dimentionality of feed forward neural network d_{ff} = 63536, all other sublayers and embeddings have a dimentionality of d_{model} = 1024. It is a model with about 11 billion parameters.

B. *Pretraining (Objective)*

T5 uses span correction objective (Section 3.3.4 [8]). Specifically, it uses a mean span length of 3 and corrupt 15% of the original sequence. T5(11B) is pretrained on a multi-task mixture of unsupervised and supervised tasks before fine tuning. C4 dataset with standard example-proportional mixing (Section 3.5.2 [8]) with artificial dataset size set to 133,000,000 examples is used for pretraining. It is pretrained for 1 million steps on a batch size of 2^{11} sequences of length 512, corresponding to about 1 trillion pre-training tokens with inverse square root learning rate schedule.

C. Fine Tuning and Testing

T5 is fine tuned on individual downstream tasks for respective application. Here we consider fine tuning for “Question Answering” task.

T5 is fine tuned on Stanford Question Answering Dataset (SQuAD) with a batch size of 8 length-512 sequences with constant learning rate and dropout regularisation.

T5 model is always trained using maximum likelihood, i.e. using teacher forcing and a cross entropy loss. For optimization, AdaFactor [9] and at test time greedy decoding (choosing highest-probability logit at every timestep) is used.

T5(11B)’s performance is reported on the SQuAD validation set [8].

D. Results

Research in machine learning domain suggests that general methods that can leverage additional computation ultimately win against methods that rely on human expertise. Recent results suggests that this might also be true in case of NLP i.e. Model can benefit from it’s scale. T5(11B) benefits from its’ scale.

As T5 can handle large dataset like C4, it can learn many language related features that other models can’t because of their limited scale.

For SQuAD dataset, T5(11B) outperformed the previous state-of-the-art [10] by over one point on the exact match(EM) score and over 0.7 points on F1 score [Table: 2].

Model	SQuAD EM	SQuAD F1
Previous best (On test set)	90.1	95.5
T5(11B) (On validation Set)	91.26	96.22
Human Performance	82.30	91.22

Table 2: Results are reported on SQuAD validation set [8]

VI. CONCLUSION

In this paper, we have decoded some deep learning models, which are used for information retrieval tasks based on given context. Question Answering task is purely based on method of giving input and expected output to the selected neural network. Transformer architecture offers significant improvements over RNN for this task. There seems a tradeoff between resources used and accuracy achieved in case of BERT and T5. While use of resources by BERT is less than T5 so is its’ performance. More research and innovation are needed to cope up with need of handling ever increasing text data.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 03, March-2021

REFERENCES

[1] Jakob Uszkoreit (2017, August 31), "Transformer: A Novel Neural Network Architecture for Language Understanding", https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html

[2] Eugenio Culurciello (2018, April 13), "The Fall of RNN/LSTM", https://towardsdatascience.com/the-fall-of-rnn-lstm-2d1594c74ce0

[3] Giuliano Giacaglia (2019, March 11), "How transformers work", https://towardsdatascience.com/transformers-141e32e69591

[4] Tianxing He, Jingzhao Zhang, Zhming Zhou, James Glass, (2020), "Quantifying Exposure Bias for Open-ended Language Generation", arXiv: 1905.10617

[5] Ashish Vaswani and Noam Shazeer and Niki Parmar and Jakob Uszkoreit and Llion Jones and Aidan N. Gomez and Lukasz Kaiser and Illia Polosukhin, (2017), "Attention Is All You Need", arXiv: 1706.03762

[6] Jay Alammar (2018, June 27), "The Illustrated Transformer", The Illustrated Transformer – Jay Alammar – Visualizing machine learning one concept at a time. (jalammar.github.io)

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova , (2018), "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", arXiv: 1810.04805

[8] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J. Liu, Karishma Malkan, Noah Fiedel, and Monica Dinculescu, (2020) "Exploring Transfer Learning with T5: the Text-To-Text Transfer Transformer", arXiv: 1910.10683

[9] Noam Shazeer and Mitchell Stern, (2018) "Adafactor: Adaptive Learning Rates with Sublinear Memory Cost", arXiv: 1804.04235

[10] Zhenzhong Lan and Mingda Chen and Sebastian Goodman and Kevin Gimpel and Piyush Sharma and Radu Soricut, (2020) , "ALBERT: A Lite BERT for Self-supervised Learning of Language Representations", arXiv: 1909.11942