From Seq2Seq to BERT

Huali Z

ABSTRACT

In the end of 2018, BERT[5] caused a stir in machine learning with 7.7% point improvement on GLUE(General Langugage Understanding Evaluation) score and 5.1 point improvement on SQuAD(Stanford Question Answering Dataset) v2.0. Increasing numbers of pre-training models have been published based on BERT. Records of model scores on SQuAD, GLUE and MultiNLI(Multi-Genre Natural Language Inference) has been refreshed every year. The nitty-gritty of BERT is Transformer[16] and bi-directional training. For Transformer, Attention is all you need [16]. It is only 6 years of time since Seq2Seq[15] model was published in 2014. How the models evolved to consume less computer power, to enable parallel computing and to get better scores? This paper illustrates a short history of the NLP models evolution step by step. From Seq2Seq model to BERT, from intermediate fixed size vector to Transformer. It also connects the dots of lingoes like Seq2Seq, fixed size vector, attention, soft alignment, attention, Transformer, pre-training, fine-tuning, and BERT.

1 INTRODUCTION

Staring with Seq2Seq which solved the problem that Deep Neural Networks can only be applied to problem who's input and targets can be sensibly encoded with vectors of fixed dimensionality[15], this paper states a short evolution history of NLP models from 2014 to 2020 when variants of BERT models are published.
Seq2Seq model proposed a structure with a fixed size vector between input and output layer. This structure fixed the dimensionality problem of deep neural networks by the intermediate vector. When it comes to long sentences, this fixed size vector can be a bottleneck for performance.
Attention model introduced a encoder and decoder structure with soft alignment. After the attention model, different types of attention models are innovated. Soft attention works well on getting context information, however it is expensive. Hard attention is less expensive but requires more complicated techniques. Global attention takes into all hidden states, where as local attention only takes in a subset of hidden states. Self attention is using positions of a sequence to produce encoding a word in the same sequence. Attention can also be categorized by attention score calculation methods. Dot production attention and concat attention are most used methods. Vaswani's research has shown that dot production attention is much faster and more space efficient.
RNNs in attention models can be an obstacle for parallelism and resource efficiency. Transformer solves the problem by relying on self attention completely without RNNs. Along with self attention, Transformer also combines the techniques like scaled-dot attention, multi-head attention, position encoding and masking to achieve the best result.
Based on Transformer, BERT was innovated for pre-training. The ground breaking feature of BERT is bidirectional training of Transformer. It allows BERT to understand context better and enables information flow between layers. Variants of BERT has been published after BERT standing under the spotlight. Models like SpanBERT and SesameBERT improves the attention span to get better understanding of context. ALBERT, DistilBERT, TinyBERT and MobileBERT improves cost efficiency while keeps same performance. SciBERT and BioBERT gives pre-trained model for domain specific NLP tasks.

2 SEQ2SEQ

Deep neural networks has shown its power on parallel computation and back-propagation. At the same time, it has limitation on the dimensionality on input and outputs. The input and output has to be encoded with fixed size vectors. In real life, for many problems the best expression of input and output vector sizes are not known ahead of the time. For example, sequence problem like voice recognition.
Seq2Seq model[15] was created for language modeling. It is normally used for translation, summary extraction and question answer bots . The model contains two parts: encoder and decoder. Encoder maps the input sequence into a **fixed size vector** and decoder outputs a sequence from the vector. Both the encoder and decoder are multi-layered LSTM. The goal for the LSTM is to estimates the conditional probability $p(y_1, ..., y_{T'}|x_1, ..., x_T)$ [15] where the input sequence length T is different from the output sequence length $T^{'}$. The mathematical representation of this model is:

$$p(y_1, ..., y_{T'}|x_1, ..., x_T) = \prod_{i=1}^{T'} p(y_t|v, y_1, ..., y_{t-1}) \qquad (1)$$

In the equation, v is a fixed sized vector which produced by the last hidden state of input LSTM. This fixed size vector is being used as the initial input of the output LSTM. Each p(y_t | v, y_1, ... , y_t-1) is represented with a softmax all over the words in the volcabulary. [15]
There are 3 features which are not included in the math representation.

(1) Two different LSTM is used for input sequence and output sequence respectively. It helps on the parallelism while training the model with different languages with small computational cost.
(2) A deep LSTM with 4 layers is used because deep LSTM works better than shallow LSTM in this case.
(3) Reversing the words of input sequence introduced short term dependencies which improves the model's performance on long sentences.

3 ATTENTION

The fixed size vector in Seq2Seq model enables modeling without knowing input and output sequence size. However, when it comes to long sentences, it is a bottleneck for improving performance. Because all information is compressed into the fixed size vector, the beginning part of the information is forgotten. Bahdanau et all[2]

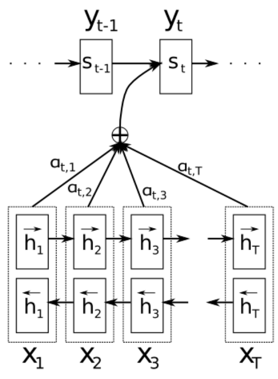
y_{t-1}
y_t
s_{t-1}
s_t
$a_{t,1}$
$a_{t,2}$
$a_{t,3}$
$a_{t,T}$
$\overrightarrow{h}_1$ $\overrightarrow{h}_2$ $\overrightarrow{h}_3$ $\overrightarrow{h}_T$
$\overleftarrow{h}_1$ $\overleftarrow{h}_2$ $\overleftarrow{h}_3$ $\overleftarrow{h}_T$
X_1 X_2 X_3 X_T

Figure 1: Attention architecture [2]

proposed a new mechanism which abort the fixed size vector and introduced a new architecture of encoder and decoder.

As Figure 1 illustrates, the encoder converts input sequence x into annotations h by a bi-directional RNN. These annotations are sum weighted into a context vector c. Using the context vector c, along with the previous hidden state s_{i-1} and previous output y_{i-1}, the decoder computes out the output y_t. Using bi-directional RNN enables the encoder to calculates hidden states in both directions. So for each word x_j the annotation is a concatenation $h_j = [\overrightarrow{h_{jT}} : \overleftarrow{h_{jT}}]$. The equation of this model can be written as:

$$p(y_i|y_1, ..., y_{i-1}, X) = g(y_{i-1}, s_i, c_i) \tag{2}$$

where:

$$s_i = f(s_{i-1}, y_{i-1}, c_i) \tag{3}$$

$$c_i = \sum_{j=1}^{T_x} a_{ij} h_j \tag{4}$$

$$a_{ij} = \frac{\exp e_{ij}}{\sum_{k=1}^{T_x} \exp e_{ik}} \tag{5}$$

$$e_{ij} = a(s_{i-1}, h_j) \tag{6}$$

s_i in equation 3 is the hidden state of the RNN at time i. c_i in equation 4 is the context vector which is a weighted sum of the annotation h_j. The annotations sequence $(h_1, ..., h_{T_x})$ is mapped from the input sequence. Each annotation h_i contains the information of all the inputs with a strong focus on the words around i-th word. The weight a_{ij} in equation 5 is calculated by an alignment model in equation 6. The alignment model scores how well the inputs around position j and the output at position i match. It is trained and optimized along with the whole translation model jointly[2]. From the equations, we can see that the next output y_t and hidden state s_t is depended on the previous hidden state s_{t-1} and annotation h_j. This design implements the **attention** mechanism in the decoder. With attention, the fixed size vector is no longer needed in this whole model. Instead, a **soft alignment** between the input and output is learnt. This improves the model's performance with long sentences significantly.

3.1 Attention Family

Depends on how the alignment score is calculated, attention mechanism can be categorized to dot-product[11], scaled dot-product[16], concat[2], general and others. In a broader categories attention mechanisms, there is global attention[11], local attention[11], hard attention[11], soft attention[2] and self attention[4].

3.1.1 hard vs soft. In a soft attention model, the alignment vector is learnt by an alignment model. See Equation 6. The alignment model directly computes a soft alignment, which allows the gradient of the cost function to be back-propagated through. This gradient can be used to train the alignment model as well as the whole translation model jointly[2]. For example, in a English to French translation task, soft alignment allows the model to translate the word 'the' to 'le' and 'la' dynamically based on the context word. The drawback of this model is when input sequence is long, the computation is expensive.
In hard attention, the alignment vector a_t is learnt with a select part of the input sequence. It is less expensive than the soft attention model but it is non-differentiable and requires more complicated techniques such as variance reduction or reinforcement learning to train[11].

3.1.2 global vs local. The idea of global attetntion model is to consider all the hidden states of the encoder when deriving the context vector c_t[11]. As Figure 2 shows, all the hidden states are used to calculate the alignment vector a_t at time t. Then the global context vector c_t is generated by doing weighted average on each a_t.
Using all the hidden states can be expensive and can potentially render it impractical to translate longer sequences, e.g. paragraphs or documents[11]. Local attention chose to pay attention to only a subset of the hidden states to get the weight vector a_t. Figure 3 illustrates that the alignment vector a_t is got from part of the input hidden states. These input hidden states are within a window [p_t-D, p_t+D]. p_t is aligned position and D is a selected value. The contet vector c_t is a weighted averaged value over a_t. Unlike the global attention model, a_t now is a fixed size vector. To avoid the non-differentiable alignment problem in the hard attention model. This local attention model predicts the aligned position p_t using equation 7 where the model parameters w_p and v_p will be learnt for predicting p_t.

$$p_t = S \times Sigmoid(v_p^T tanh(W_p h_t)) \tag{7}$$

3.1.3 self attention. Self attention is also called intra-attention[4]. It is a mechanism using 'attentioned' positions of a sequence to produce encoding of a word in the same sequence. Figure 4 shows how the much attention is paid in a machine reading task using self attention. The model processes text incrementally while learning which past tokens in the memory and to what extent they relate to the current token being processed[4].

3.1.4 alignments score calculation methods. There are variant ways of calculating alignment score. Score here means how well a source sequence hidden state and a target sequence hidden state is aligned. Equation 5 is a good example of alignment score calculation method. Bahdanau's research employed a feed forward network which is the

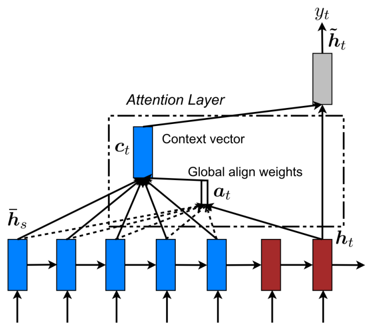

Figure 2: Global attention[11]

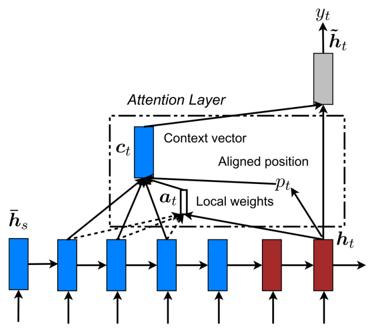

Figure 3: Local attention in machine reading task[11]

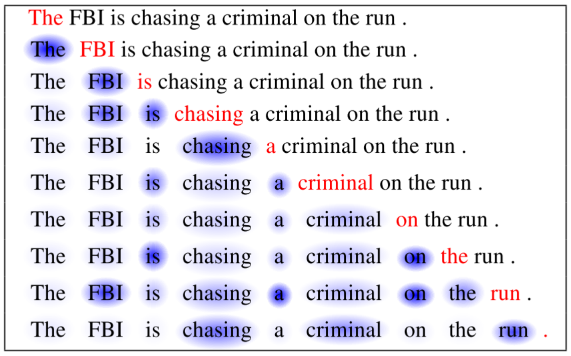
The FBI is chasing a criminal on the run .
The FBI is chasing a criminal on the run .
The FBI is chasing a criminal on the run .
The FBI is chasing a criminal on the run .
The FBI is chasing a criminal on the run .
The FBI is chasing a criminal on the run .
The FBI is chasing a criminal on the run .
The FBI is chasing a criminal on the run .
The FBI is chasing a criminal on the run .
The FBI is chasing a criminal on the run .

Figure 4: self attention[4]

a in equation 6 to derive the alignment score. This way of getting the alignment score is referenced as concat in Luong's research[11] and additive addition in Vaswani's paper[16]. Beside this method, Luong's research also explored 'dot production' $score(h_t, h_s) = h_t^T \bar{h_s}$ [11] , 'general' $score(h_t, h_s) = h_t^T W_a \bar{h_s}$ [11] and 'location based' function $a_{t,i} = softmax(W_a h_t)$ [11]. The research shows that the concat method is no better than the others. The dot production attention and concat attention are most used methods for getting alignment scores. Dot production attention is much faster and

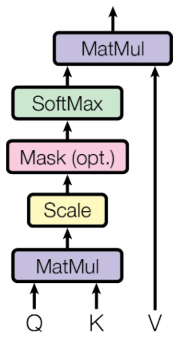

Figure 5: Scaled dot attention[16]

more space efficient in practice, since it can be implemented using highly optimized matrix multiplication code[16]. Vaswani's research introduced a new method called 'scaled dot production' $score(h_t, h_s) = \frac{h_t^T \bar{h_s}}{\sqrt{n}}$ which is similar to 'dot production' but with the source dimension as a scaling factor. Adding this scaling factor helps the model dealing with the large values of the source dimension.

4 TRANSFORMER

Using RNNs in sequential modeling was no doubt a state of art approach before the Transformer architecture. RNNs can capture the coherence in the sequence with hidden states. However, this inherently sequential nature precludes parallelization within training examples, which becomes critical at longer sequence lengths, as memory constrains limit batching across examples[16]. Vaswani's work proposed a Transformer model which is relying on self attention mechanism entirely without RNNs.

4.1 Transformer components

4.1.1 Transformer's self scaled-dot attention. The attention mechanism firstly creates three vectors: Q(query), K(key), and V(value) for each word embedding. These vectors are created by multiplying the input embedding with three metrics w_q, w_k, and w_v. It then calculates the score for each word in the input sentence, divides the the scores by the square root of the K dimension, softmax the dividended score, and finally multiples V vector with the sfotmaxed score. The whole process is shown as in Equation 8[16] and Figure 5[16]. Using scaled dot function helps on getting more stabled gradients and softmax function makes sure the scores at each input sequence position will add up to 1.

$$Attention(Q, K, V) = Softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{8}$$

4.1.2 Multi-head attention. As Figure 6 shows, multi-head attention linearly projects keys, values and queries h times. For each projected version of K, V and Q pairs, attention layer applied on. Each attention layer produces output values in parallel for each version of projected K, V and Q. These outputs are concatenated

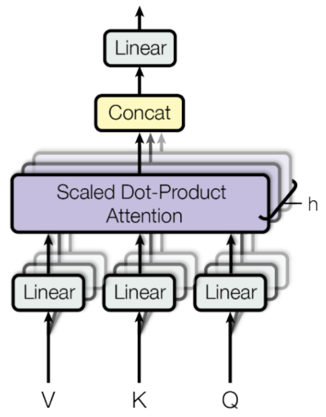

Figure 6: multi-head attention[16]

and linear projected to get the final value. By doing this multi-head attention, the model increase its ability to focus on different parts of input sequence by taking in information from different representations jointly. The equation of this multi-head attention mechanism can be written as equation 9[16].

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_h)W^o \tag{9}$$

where

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \tag{10}$$

4.1.3 position encoding. Removing the RNNs from the model discarded the hidden states which can indicates the inherent position information. To bring in the position information, Transformer injects positional encoding in the input embedings for both encoder and decoder. It adds the absolute or relative position information into the projected K, V and Q vectors.

4.1.4 masking. The self attention layer in decoder is different from the one in encoder. Masking is applied in the decoder self attention layer to ensure the decoder prediction only takes in words in earlier positions.

4.2 Transformer architecture

The architecture of Transformer is illustrated in Figure7. Like Seq2Seq model and other attention models, Transformer also has encoder and decoder structure. Both encoder and decode is composed by 6 identical layers. 6 is a chosen empirical number. All the encoder layers contain two sub-layers: multi-head self attention layer and feed forward network layer. After positional encoding, the input sequence flows into the multi-attention layer so the model can encode a word with its surrounding words. The output of the multi-attention layer is fed into a feed forward neural network which is applied to each position. The decoder has similar structure as the encoder but with one more multi-head attention layer to take in the output from the encoder. The output of the decoder will pass through a linear and a softmax layer so the output vector can be mapped to a word.

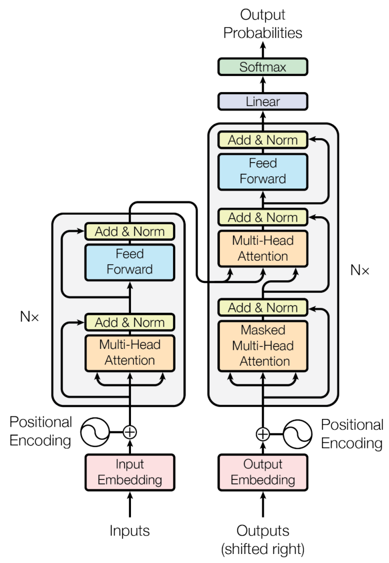

Figure 7: Transformer[16]

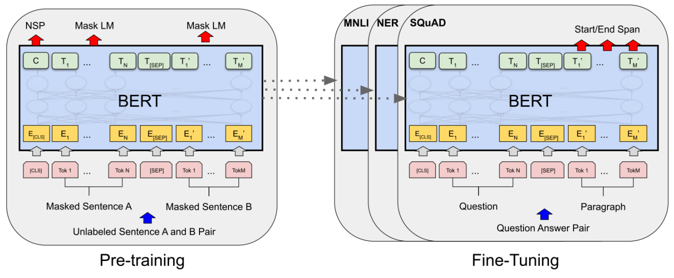

Figure 8: BERT framework [5]

5 BERT

BERT, which stands for Bidirectional Encoder Representations from Transformers[5], is a pre-training transformer encoder which improves fine-tuning by bidirectional training. When applying the pre-trained sequence to a specific downstream task, there are two main strategies can be used: feature-based and fine-tuning. Feature-based strategy is using the pre-trained representations as an extra feature. Fine-tuning is simply fine-tuning the task specific parameters. The pre-training models before BERT are unidirectional models which restricts the performance of context understanding. BERT introduced a state of art MLM(masked language model) along with "next sentence prediction" to enable bidirectional representation. The masked language model randomly masks some of the tokens from the input, and the objective is to predict the original vocabulary id of the masked word based only on its context[5].

5.1 Optimized BERT

5.1.1 Improved training procedure. After evaluating BERT parameters with datasets, Liu et all, found that BERT is significantly undertrained. RoBERTa(Robustly optimized BERT approach) [10] is created for providing a better way for training BERT models. This approach includes (1) training the model longer, with bigger batches, over more data; (2) removing the next sentence prediction objective; (3) training on longer sequences; and (4) dynamically changing the masking pattern applied to the training data[10]. With the approaches, RoBERTa get improved performance when comparing with $BERT_{LARGE}$.

5.1.2 Improved attention span. SpanBERT[7] is an extended BERT with randomly masked spans and SBO(span boundary objective). Comparing with the masked tokens in vanilla BERT, masked span provides a better context representation. SBO enables the model learns to predict the entire masked span from the observed tokens at its boundary without relying on the individual token representations within it[7].
SesameBERT[13] enables information flow across BERT stacked layers through Squeeze and Excitation. It also enriches local information by captureing neighboring context via Gaussian blurring[13].

5.1.3 Improved cost efficiency. As pre-training model size grows for better performance in fine-tuning, memory limitation of available hardware is an obstacle for getting a better scaled pre-trained model. The existing model parallelization and clever memory management method can mitigate the memory shortage issue but not communication overhead. ALBERT(A lite BERT)[8] is born for all the problems. ALBERT provides a BERT model with less parameters so it consumes lower memory and runs in a faster speed. There are two parameter reduction strategies used in ALBERT. The first one is a factorized embedding parameterization. The second one is cross-layer parameter sharing [8].
To alleviate the resource limitation, DistilBERT [12] introduce a triple loss combining language modeling, distillation and cosine-distance losses[12]. It reduces BERT model size by 40% and speeds up the model 60% while keeping 97% of the model's understanding ability. Similar modified BERT models are MobileBERT [14] and TinyBERT[6]. MobileBERT's goal is building a lightweight pre-trained model for different downstream tasks. TinyBERT is 7.5 times smaller and 9.4 faster with a novel Transformer distillation method and a two-stage learning method.

5.1.4 Domain oriented improvement. On specific domains, lack of high quality and larges scaled data is limiting BERT performance. Domain-specific pre-trained BERT models are published for providing generic but specific language representation models. For example, SciBERT[3] and BioBERT[9] is trained on large multi-domain scientific publication corpora and biomedical corpora respectively.

6 CONCLUSION

Exponential number of research is done with variants of BERTs in two years of time. While reading the papers, understating concepts like Attention helps to know what a model is based on. This paper introduces a short history of NLP models: Seq2Seq, Attention, Transformer and BERT. All these models are aiming at getting better performance and being less time and memory expensive at the same time.

REFERENCES

[1] Jay Alammar. 2019. *The illustrated Transformer.* http://jalammar.github.io/illustrated-transformer/
[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE. (2015). arXiv:1409.0473v7
[3] Iz Beltagy, Kyle Lo, and Arman Cohan. 2020. SCIBERT: A pretrained language model for scientific text. *EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Proceedings of the Conference* (2020), 3615–3620. arXiv:arXiv:1903.10676v3
[4] Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long Short-Term Memory-Networks for Machine Reading. (jan 2016). arXiv:1601.06733 http://arxiv.org/abs/1601.06733
[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. (oct 2018). arXiv:1810.04805 http://arxiv.org/abs/1810.04805
[6] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2019. Tinybert: Distilling bert for natural language understanding. *arXiv* (2019). arXiv:arXiv:1909.10351v5
[7] Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2019. SpanBERT: Improving Pre-training by Representing and Predicting Spans. (jul 2019). arXiv:1907.10529 http://arxiv.org/abs/1907.10529
[8] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. (sep 2019). arXiv:1909.11942 http://arxiv.org/abs/1909.11942
[9] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2020. BioBERT: A pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics* 36, 4 (2020), 1234–1240. https://doi.org/10.1093/bioinformatics/btz682 arXiv:1901.08746
[10] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. (jul 2019). arXiv:1907.11692 http://arxiv.org/abs/1907.11692
[11] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. (aug 2015). arXiv:1508.04025 http://arxiv.org/abs/1508.04025
[12] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. (oct 2019). arXiv:1910.01108 http://arxiv.org/abs/1910.01108
[13] Ta-Chun Su and Hsiang-Chih Cheng. 2019. SesameBERT: Attention for Anywhere. (oct 2019). arXiv:1910.03176 http://arxiv.org/abs/1910.03176
[14] Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. 2020. MobileBERT: a Compact Task-Agnostic BERT for Resource-Limited Devices. (apr 2020). arXiv:2004.02984 http://arxiv.org/abs/2004.02984
[15] Ilya Sutskever Google, Oriol Vinyals Google, and Quoc V Le Google. 2014. *Sequence to Sequence Learning with Neural Networks.* Technical Report.
[16] Ashish Vaswani, Google Brain, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. *Attention Is All You Need.* Technical Report. arXiv:1706.03762v5
[17] Lilian Weng. 2019. *Attention? Attention!* https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html