Improvements in Natural Language Processing with the advent of Bidirectional Encoder Representations from Transfomers

Rajdipa Chowdhury

5 June, 2019

1 Introduction

Natural Language Processing (NLP) is the study of interactions between computer and Natural Languages. In particular, it is concerned with how computers process and analyze large amounts of natural language data. Over the last few years, deep learning has been implemented in Natural Language Processing. Deep Learning is the subset of Machine Learning concerned with algorithms inspired by the structure and function of the brain. In this paper, we look at a new deep learning language representation model called Bidirectional Encoder Representations from Transfomers (BERT) which was developed by Google AI language and the improvements it has brought to the field of Natural Language Processing. BERT has a novel technique to introduce deep bidirectionaity while training which have led to some ground breaking results in NLP.

This paper is organized as follows. Section 2 explores some NLP models and provides necessary background knowledge to acquaint the reader with BERT. In Section 3, We look at the specifics of BERT and its performance in several NLP tasks. Finally, Section 4 provides a conclusion.

2 Back ground

2.1 Recurrent Neural network

Humans understand things from context. The way we process information is not isolated. To understand a new thing we try to connect it with our previous knowledge.

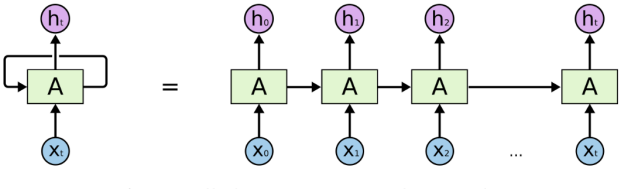

Figure 1: An unrolled RNN. Here , A represents a neural network ;x_t and h_t represents the input and output at time t respectively. The left most block shows a RNN uses a loop to pass information. The blocks on the right breaks down how the loop passes information from one time step to the next.

With vanilla neural networks we cannot have context. What ever the model predicted in the previous time step will not be considered in the next time step. This is a major shortcoming. For instance, we have the task of classifying what type of event is happening at every time step in a movie. Vanilla neural network considers each event as isolated event, it doesnot take into account the relationship of scenes at different time step.

With RNN, we can introduce context. The output that was predicted in the previous time step is fed to the neural network at the next time step. Figure 1 illustrates the schematics of this process.

The Block A in Figure 1 is the the repeating Module of a standard RNN. This repreating module will usually have a very simple structure, like the hyperbolic tan function.

Although we have introduced context with RNN, the standard RNN doesnot perform very well when it comes to larger context. For example, if we are trying to predict the next word in the sentence: "The clouds are in the ... ". The standard RNN will be able to correctly complete this sentence since it does not need a lot of context but if we have a scenario where we need a larger context, the standard RNN will not be able to do a good job. For instance, if we want to predict the word at the end of this paragraph, "I grew up in France, away from my parents. It was hard to be away from my parents at a really young age ... I speak fluent _____", a standard RNN will be unable to connect the information. Bengio, et al (1994) further discusses the fundamental reasons why a standard RNN would not work well on long term dependencies.

2.2 LSTM

Long Short Term Memory networks (LSTMs) are a special kind of RNN specifically designed to handle long term dependencies. The structure of

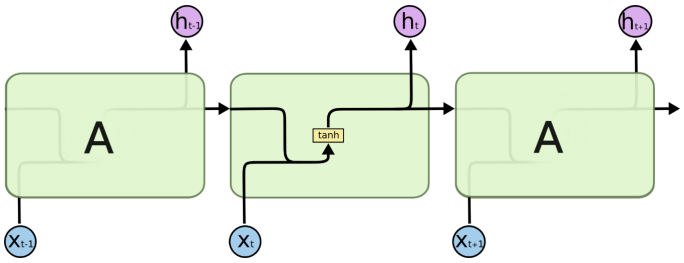

Figure 2: Repeating module of a RNN. In a standard RNN, Repeating module contains a single layer. Here, the single layer is the hyperbolic tan function. All the repeating module has this function.

LSTM is represented in Figure 3.

The very top horiontal line in figure 3 is called the cell state. Information flows in the cell state with very minor linear interactions. Information is added and removed from the cell state with the help of gates. In LSTMs, gates contain a sigmoid neural network and a point wise multiplication operation.LSTM has three such gates.

Figure 4, shows a more detailed look of the layes in a repeating module of LSTM. The first step in the LSTM is to decide what to forget. $f_t = \sigma(W_f \cdot [h_{t-1}, x_t]+b_f)$ is the function representing what to forget.Here, W_f, b_f, h_{t-1}, x_t are the weights, biases, output from the previous module, and the input to the current module respectively.[x,y] represents concatenation of x and y. The next step is to decide what to store in the cell state. This is done in two parts: The sigmoid input gate decides what values to update (represented by $i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$ and a hyperbolic tan layer creates new candidates values, $\widetilde{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$, which will be added to the cell state. The new cell state will be given by

$$C_t = f_t * C_{t-1} + i * \widetilde{C}_t$$

The current output h_t is given by

$$h_t = o_t * tanh(C_t)$$

Here $o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o$

In general LSTM are implemented in natural language processing by using an encoder-decoder model. The core idea is that encoder takes the sequence of input words, converts it to some intermediate representation, then passes that representation to the decoder which produces the output sequence. If our task is to predict sensible senses, the Encoder-decoder models are trained

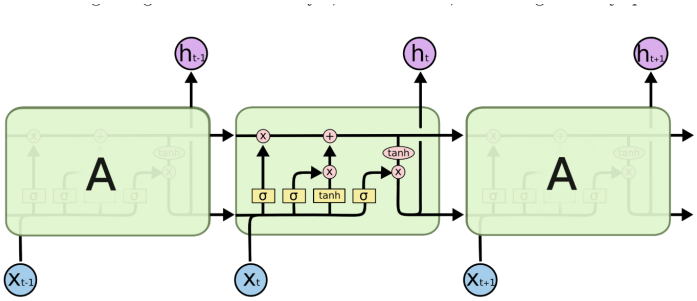

Figure 3: Repeating module of LSTM. Unlike RNN, LSTM contains 4 interacting layers. The yellow blocks represent a neural network, the red circle represents a pointwise operation and the straight arrow represents a vector transfer. When two lines are meged, the two vectors get contcatenated and when a line is split into two branches, the vector is copied to both the branches.

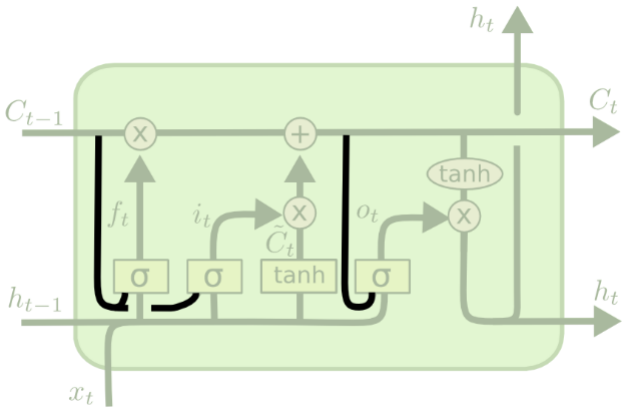

Figure 4: detailed look of the layers in the repeating module of LSTM

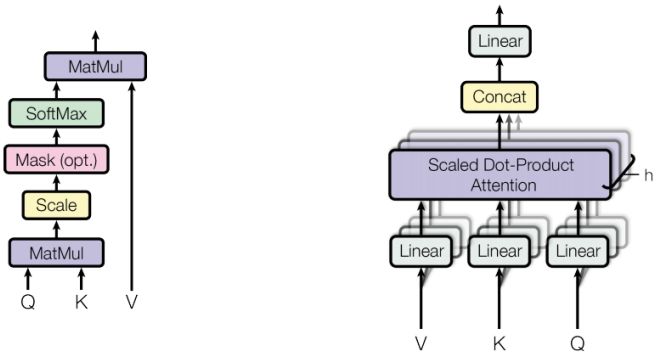

Figure 5: (left) Scaled dot-product attention. (right) Multi head attention

to maximise the likelihood generating the correct output sequence. At each step, the decoder is rewarded for predicting the next word correctly and penalized for making mistakes.

2.3 Attention Mechanism

We know the movtivation behind deep learning is the human Brain. Encoder-decoder LSTM models works in a way very different than the human brain. For instance, If we wanted to translate a really lomg sentance :Encoder-decoder LSTM models would have to read the whole sentence, memorize it and then generate the translation while a human would read part of the sentence translate it and then move on to the next part. In fact, sentences with words greater than 40, the translation has a deteriorating BLEU score (Papineni, et al 2001).

Attention fixes this problem by looking at the relevant words while generating the translation. Attention gives the decoder access to all the encoder's hidden states and it allows the decoder pick and choose what information to use and ignore by looking at the weights of the hidden state.

"An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key" (Vaswani et al 2017). This mapping is not fixed and can have different forms. We are going to discuss the types of attention relevant for understanding BERT.

2.3.1 Scaled Dot-Product Attention

Scaled Dot-Product Attention is given by the following equation:

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

Here, Q,K,V represents the queries, keys and values of dimension d_k. The softmax funtion is given by $\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{L} e^{z_j}}$ for i = 1,...,K and z = $(z_1, .., z_K) \in \mathbb{R}^K$.

Figure 5 shows the structure of Scaled Dot-Product Attention. The basic attention mechanism is simply a dot product between the query and the key. The size of the dot product tends to grow with the dimensionality of the query and key vectors thus it is rescaled by dividing by $\sqrt{d_k}$ to prevent huge values. For each query, the attention score of each value is the dot product between the query and the corresponding key.

2.3.2 Multi-Head Attention

If we only computed single attention, it would be difficult to capture various aspects of the input. For example, in the sentence "I like ice cream more than cake", we want to capture that the sentence compares two things while also retaining the two things that are being compared. To learn diverse representations ,the Multi-Head Attention applies different linear transformations to the values, keys, and queries for each "head" of attention. The Multi-Head Attention block just applies multiple blocks (scaled dot product attention) in parallel, concatenates their outputs, then applies one single linear transformation. Figure 5 shows the structure of muti head attention. Mutli head attention is represented by the following equation.

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

Here, $head_i = Attention(Q\mathrm{W}_Q^i, K\mathrm{W}_K^i, V\mathrm{W}_V^i)$. $\mathrm{W}_Q^i \in \mathbb{R}^{d_{model} \times d_k}, \mathrm{W}_K^i \in \mathbb{R}^{d_{model} \times d_k}, \mathrm{W}_V^i \in \mathbb{R}^{d_{model} \times d_v} and W^0 \in \mathbb{R}^{hd_v \times d_{model}}$ are parameter matrices

2.4 Transformer

Transformer is a novel architecture introduced in the paper "Attention Is All You Need" by Ashish Vaswani at Google Brain. Like LSTMs, Transformers uses encoders and decoders but unlike LSTMs, it doesnot imply any form of recurrent neural network.It implements the attention mechanism with out the use of any RNN.

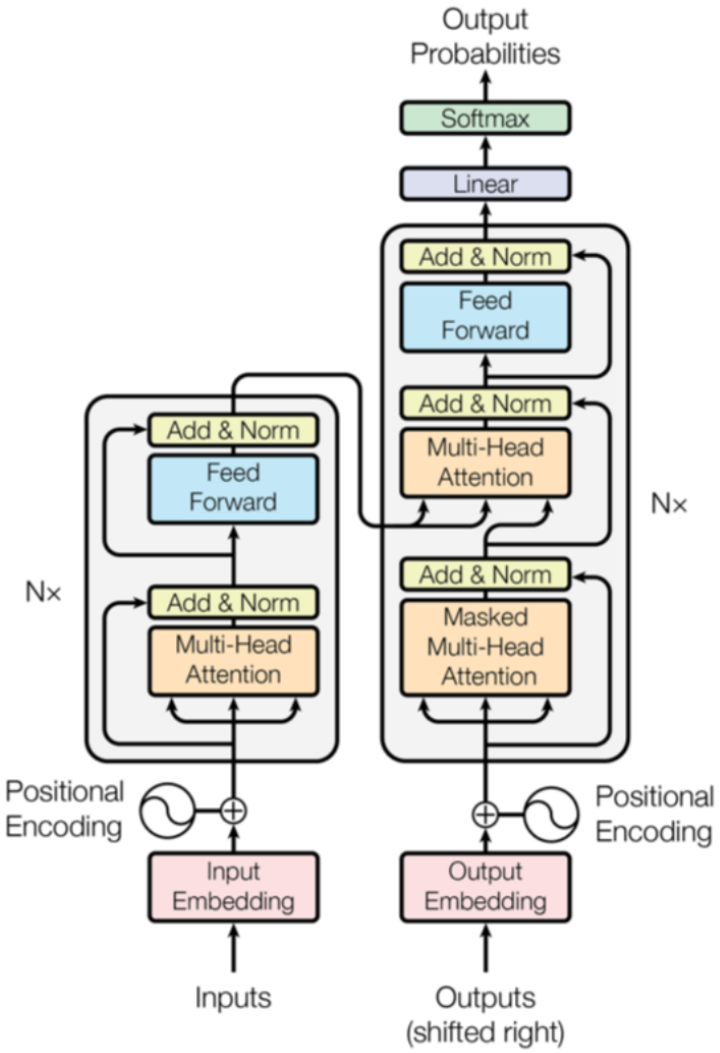

Figure 6: The architecture of a transformer. The left side is the encoder and the right side is the decoder.

Figure 6 illustrates the structure of a transformer. The Encoder is on he left and the Decoder is on the right. Both Encoder and Decoder are composed of modules that can be stacked on top of each other multiple times, which is described by N_x in the figure. We see that the modules consist mainly of Multi-Head Attention and Feed Forward layers. Feed Forward refers to a vanilla neural network. An important aspect for Transformers is positional encoding. Since we have no recurrent networks, the multi-head attention network cannot naturally make use of the position of the words in the input sequence.Without positional encodings, the output of the multi-head attention network would be the same for the sentences "I like ice cream more than cake" and "I like cake more than ice cream". Thus, we need to somehow give every word/part in our sequence a relative position since a sequence depends on the order of its elements. These positions are added to the embedded representation (n-dimensional vector) of each word. The positional encoding for a Transformer is given by:

$$PE_{(pos,2i)} = sin(pos/10000^{\frac{2i}{d_{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{\frac{2i}{d_{model}}})$$

Here, *pos* is the position and i is the dimension. Thus each dimension of the positional encoding corresponds to a sinusoid. This function was chosen because for a fixed off set k, PE_{pos+k} can be represented as a linear function of PE_{pos}. Thus, it would allow the model to easily learn to attend by relative positions.

The encoder block basically does a bunch of matrix multiplications, followed by a couple of element transformations. The decoder is similar to the encoder but has a masked multi-head attention block in addition. The reason we need the masked multi-head attention block is to mask input to the decoder from future time steps. When we train the Transformer, we want to process all the sentences at the same time. When we do this, if we give the decoder access to the entire target sentence, the model can just repeat the target sentence.

3 BERT

BERT is a new state of the art language representation model conceived by researchers at Google AI Language(Devlin el at 2019). BERT's techincal novelty is applying bidirectional training of a Transformer. The problem with present pre-trained language representational models are that the model architectures need to be changed drastically based on a specific NLP task.

BERT attempts to solve this problem by using bidirectional training to generalize one model architecture for a broad set of NLP tasks.

3.1 Architecture

The model architecture of BERT is a multi-layer bidirectional Transformer encoder which is described in section 2.6. L represents the number of transformer blocks, H represents the hidden size and A represents the number of attention heads. There are two versions of BERT: $BERT_{BASE}$ has L=12, H=768, A=12 and Total Parameters=110 million and $BERT_{LARGE}$ has L=24, H=1024,A=16 and Total Parameters=340million. The transformers used in BERT uses bidirectional attention.

3.2 Pre-training

Pre-training in Deep learning refers to training a model on a particular task (eg. Object recognition) and applying this model to on another domain (eg. Bird sub categorization).

While pre-training, BERT was trained in two tasks simultaneously: Masked Language Model(MLM) and Next Sentence Prediction. MLM helps BERT to avoid unidirectionality.We are going to further discuss on the two tasks.

3.2.1 Masked Language Model(MLM)

The masked language model randomly masks some of the tokens from the input, and the task is to predict the original vocabulary id of the masked words based only on it's context. 15% of the words that were fed in as input were masked. But not all the words were masked in the same way. We will go over the example illustrated in Devlin el at (2019). Suppose we have a sentence "My dog is hairy". Let's assume 'hairy' was among the tokens chosen to be replaced. The tokens were replaced in the following way: 80% were replaced by the '<MASK>' token. eg. "My dog is <MASK>". 10% were replaced by a random token. eg. "My dog is apple". 10 % were left intact. eg. "My dog is hairy". A variety in masking was implemented to make the model more general. This makes the model converge slower but the effectivenessss of transfer learning is considerably improved. Unlike uni directional language model pre-training, the MLM objective enables the representation to fuse the left and the right context, which allows us to pretrain a deep bidirectional Transformer

3.2.2 Next Sentence Prediction

This task involves giving the model two sentences and asking it to predict if the second sentence follows the first in a corpus or not.Specifically, when choosing the sentences A and B for each pretraining example, 50 % of the time B is the actual next sentence that follows A and 50 % of the time B is a random sentence from the corpus. This task was added to make sure the model new how to relate two different sentences to perform downstream tasks like question answering. The masked model language did not capture this aspect.

The pre-training corpus was built from BookCorpus (800M words) and English Wikipedia (2,500M words).For Wikipedia only the text passages were extracted and lists, tables, and headers were ignored. Tokens were tokenized using 37,000 WordPiece tokens.

3.3 fine tuning

Fine tuning is a process to take a model that has already been trained for a given task, and make it perform a second task in another domain. Fine tuning usually replaces the output layer. For BERT, fine tuning is very straight forward since pretraining makes the model very general for tasks in other domain. For each task, we simply plug in the task specific inputs and outputs into BERT and fine tune all the parameters end-to-end.

3.4 Results

BERT has achieved State of The Art(SOTA) results in 11 varied NLP tasks. We are going to discuss some of them.

3.4.1 GLUE

The General Language Understanding Evaluation (GLUE) benchmark (summarized in Wang et al., 2018) is a collection of diverse natural language understanding tasks. GLUE includes the following benchmarks:

- **MNLI** : Multi-Genre Natural Language Inference is a large-scale, crowd-sourced entailment classification task. Given a pair of sentences, the goal is to predict whether the second sentence is an entailment, contradiction, or neutral with respect to the first one.
- **QQP** Quora Question Pairs is a binary classification task where the goal is to determine if two questions asked on Quora are semantically equivalent.

System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	**Average**
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.9	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	88.1	91.3	45.4	80.0	82.3	56.0	75.2
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.1	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**91.1**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**81.9**

Figure 7: : GLUE Test results (Devlin el at 2019) . The number below each task denotes the number of training examples

- **QNLI** Question Natural Language Inference is a Question Answering Dataset which has been converted to a binary classification task. The positive examples are (question, sentence) pairs which do contain the correct answer, and the negative examples are (question, sentence) from the same paragraph which do not contain the answer.
- **SST-2** The Stanford Sentiment Treebank is a binary single-sentence classification task consisting of sentences extracted from movie reviews with human annotations of their sentiment.
- **CoLA** The Corpus of Linguistic Acceptability is a binary single-sentence classification task, where the goal is to predict whether an English sentence is linguistically 'acceptable' or not
- **STS-B** The Semantic Textual Similarity Benchmark is a collection of sentence pairs drawn from news headlines and other sources. They were annotated with a score from 1 to 5 denoting how similar the two sentences are in terms of semantic meaning.
- **MRPC** Microsoft Research Paraphrase Corpus consists of sentence pairs automatically extracted from online news sources, with human annotations for whether the sentences in the pair are semantically equivalent
- **RTE** Recognizing Textual Entailment is a binary entailment task similar to MNLI, but with much less training data.

Figure 7 summarizes the GLUE scores for various models. We see, $BERT_{base}$ not only beat OpenAI GPT on all tasks achieving SOTA, but it improved SOTA by an impressive 5% on average. $BERT_{large}$ beat $BERT_{base}$ for all tasks as well.

System	Dev		Test	
	EM	F1	EM	F1
Leaderboard (Oct 8th, 2018)				
Human	-	-	82.3	91.2
#1 Ensemble - nlnet	-	-	86.0	91.7
#2 Ensemble - QANet	-	-	84.5	90.5
#1 Single - nlnet	-	-	83.5	90.1
#2 Single - QANet	-	-	82.5	89.3
Published				
BiDAF+ELMo (Single)	-	85.8	-	-
R.M. Reader (Single)	78.9	86.3	79.5	86.6
R.M. Reader (Ensemble)	81.2	87.9	82.3	88.5
Ours				
$\text{BERT}_{\text{BASE}}$ (Single)	80.8	88.5	-	-
$\text{BERT}_{\text{LARGE}}$ (Single)	84.1	90.9	-	-
$\text{BERT}_{\text{LARGE}}$ (Ensemble)	85.8	91.8	-	-
$\text{BERT}_{\text{LARGE}}$ (Sgl.+TriviaQA)	**84.2**	**91.1**	**85.1**	**91.8**
$\text{BERT}_{\text{LARGE}}$ (Ens.+TriviaQA)	**86.2**	**92.2**	**87.4**	**93.2**

Figure 8: SQUAD results (Devlin el at 2019).

3.4.2 SQUAD

The Standford Question Answering Dataset (SQuAD) is a collection of 100k crowd sourced question/answer pairs. Given a question and a paragraph from Wikipedia containing the answer, the task is to predict the answer text span in the paragraph. Figure 8 illustrates the results of the SQUAD test. A big improvement in performance was achieved by BERT large. The model that achieved the highest score was an ensemble of BERT large models, augmenting the dataset with TriviaQA (Joshi et al 2017).

3.4.3 SWAG

The Situations With Adversarial Generations (SWAG) dataset contains 113k sentence-pair completion examples that evaluate grounded commonsense inference.Given a sentence, the task is to choose the most plausible continuation among four choices.

Figure 9 summaries the results of SWAG tasks. BERT large improved SOTA by a staggering 27.1% which is remarkable!

System	Dev	Test
ESIM+GloVe	51.9	52.7
ESIM+ELMo	59.1	59.2
$\text{BERT}_{\text{BASE}}$	81.6	-
$\text{BERT}_{\text{LARGE}}$	**86.6**	**86.3**
Human (expert)†	-	85.0
Human (5 annotations)†	-	88.0

Figure 9: BERT achieved SOTA in SWAG (Devlin el at 2019). $Human^{\top}$ performance is measured with 100 samples

4 Conclusion

We see that in all the tests BERT base beat all the other models except BERT large. BERT large beat BERT base in all tests. These results suggest that the architectural modifications to include bidirectionality and the increase in parameters from BERT base to BERT large played a role in considerably improving the model's natural language understanding. Additionally, pre-training BERT (described in section 3.2) made fine tuning a very straight forward task. Thus, BERT was able to deliver a more generalized architecture for downstream tasks.

5 References

- Bengio, Y., et al. "Learning Long-Term Dependencies with Gradient Descent Is Difficult." IEEE Transactions on Neural Networks, vol. 5, no. 2, 1994, pp. 157-166., doi:10.1109/72.279181
- Devlin, Jacob. BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding. 24 May 2019
- Joshi, Mandar, et al. "TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension." Proceedings of the

55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2017, doi:10.18653/v1/p17-1147.

- Papineni, Kishore, et al. "Bleu" Proceedings of the 40th Annual Meeting on Association for Computational Linguistics - ACL 02, 2001, doi:10.3115/1073083.1073135.
- Vaswani, Ashish. Attention Is All You Need. 12 June 2017
- Wang, Alex. "Glue: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding." 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP, 2018, pp. 353-355.
- "Understanding LSTM Networks" Understanding LSTM Networks – Colah's Blog, colah.github.io/posts/2015-08-Understanding-LSTMs/