Attention and BERT

Eshwar S R
Department of Computer Science and Automation
Indian Institute of Science
eshwarsr@iisc.ac.in

Abstract

In the recent times, attention based models are out performing LSTM and GRU based networks in a wide range of NLP tasks. Pre-trained models like BERT, RoBERTa etc have achieved state-of-the-art results in many NLP tasks. In this work, I would like to review the basic components of transformers and model architecture of google's BERT model.

1 Introduction

Recurrent neural networks, Long short-term memory networks, Gated Recurrent networks were performing very well on sequence based tasks, like language modelling, machine translation, for quite some time and they were the default networks for any sequence based problems. Elmo (1) was one of the famous model for generating contextual embeddings, which was very popular for quite some time, was based on LSTM networks. The pretrained model was used on many tasks like question answering, textual entailment etc, and it outperformed in many of those tasks.

Though recurrent networks perform well, they are sequential and take inputs one at a time. This limits us in parallelization during training. However, few factorization tricks (2) and conditional computation (3) tricks are tried out, the underlying constraint of sequential computation still holds.

Transformer (4) model goes away with this sequential computation constraint and uses self attention mechanisms to infer the global dependencies between inputs and the outputs. They are no more sequence based and hence can be parallelized, yielding better results with less training time. Transformer was the first transduction model relying entirely on self-attention to compute representations of its input and output without using sequence aligned RNNs or convolution.

There were some similar attempts made earlier to go away with the sequential dependencies, some of which are Extended Neural GPU (3), ByteNet (5) and ConvS2S (6), all of which use Convolution networks to compute hidden representations for all input and output positions.

BERT (7) and other recent models use transformers as basic blocks and are trained on huge corpus. The trained models are shared publicly. These models are then fine tuned as per the task to give state-of-the-art results.

2 Transformer Model Architecture

Transformers [1] follow the basic framework as that of encoder decoder model, where the encoder maps the sequence of symbols into a latent representation of some higher dimensions. Once the representation is found, the decoder uses it to generate a sequence of symbolic outputs in an auto-regressive fashion utilising the previously generated outputs.

Preprint. Under review.

2.1 Encoder and Decoder Stacks

Encoder: The encoder is typically made up of N=6 blocks stacked one over the other. Each block has two sublayers, first is the multi head self-attention layer followed by a fully connected feed forward network. Both of the sublayers are immediately followed by a layer normalization (8) which also take in input from the residual connections (9). To facilitate this residual connections all the blocks produce same dimensional output, say d_{model}.

Decoder: The decoder is also typically made up of N=6 blocks stacked one over the other. The sublayers of decoder are similar to that of encoder, but there is an additional layer on top of those two layers to perform multi-head attention over the output from input stack. One more subtle change is the masking of future words to avoid attention towards the unseen words.

2.2 Attention

Attention can be well described as a mapping from a query vector to an output vector utilizing a set of key value vectors. A similarity function is used to calculate similarity between the query vector and all the keys, then these similarities are used to calculate the weighted sum of corresponding values to get the final output vector.

2.2.1 Scaled Dot-Product Attention

The authors propose an idea called "Scaled Dot-product Attention" [A]. The query and keys are vectors of dimensions d_k, and the values are vectors of dimensions d_v. The idea is to compute dot products between query and all the keys, divide them by $\sqrt{d_k}$ and then apply softmax to obtain the normalized weights for summing the corresponding values. These computations are parallelized using matrix multiplications. There are two most commonly used attention functions. Additive attention (10) which computes the similarity using a feed forward network with single hidden layer. Multiplicative attention is identical to the author's algorithm, except for the scaling factor of $\sqrt{d_k}$.

2.2.2 Multi-Head Attention

Instead of performing a single attention, there are multiple heads computing same dimensional output vectors. After which these vectors are concatenated and projected again to get the final output [A]. Allowing such multiple heads leads the model to attend to different representation subspaces at different positions. There are 3 different places attention is used in transformers. Encoder Decoder Attention allows every position in decoder to attend to every position in encoder sequence. Encoder Attention allows each encoder position to attend to each position of the output from the previous encoder layer. Decoder Attention allows each position in the decoder to attend to all positions in the decoder up to and including that position. Appropriate masking is done to prevent the information leak.

2.3 Pointwise Feed forward networks

As illustrated in the figure [1], every block contains a fully connected feed forward network which is applied to each position separately and identically. This network consists of ReLU activation sandwiched between two linear layers. Each block has its own set of parameters and there is no sharing of parameters as such.

2.4 Positional Encoding

As transformer model does not take input in a sequence, the ordering information should be somehow well communicated to the model. There are two ways of doing this positional encoding. These positional encodings are added to the input embeddings at the bottom of encoder and decoder stacks. The positional embeddings can be either learned or fixed (6). Authors use sine and cosine functions with different frequencies. They also claim that there were no big differences by learning those positional embeddings.

3 BERT: Bidirectional Encoder Representations from Transformers

The core idea of BERT is to create a pre-trained language model, which can be fine tuned on tasks to achieve good results even with less data. There are two ways of using pre-trained language models. One such approach is called "feature based", like that of ELMo, which uses task-specific architectures that include the pre-trained representations as additional features. The other way is fine-tuning, like that of BERT, where all or subset of parameters are updated during the training of each task specific model.

Authors show that the fine-tuned $BERT_{LARGE}$ model produced state-of-the-art results in eleven NLP tasks.

3.1 Model Architecture

BERT is completely based on Transformers and its implementation is very much identical to that of the Transformers paper (4).

The authors published two models. Namely, $BERT_{BASE}$ and $BERT_{LARGE}$. $BERT_{BASE}$ model is similar in size of OpenAI GPT (11) another transformer based model, useful for comparison purposes. So, the $BERT_{BASE}$ model consists of 12 layers, each with 12 attention heads and embedding dimension of 768. Where as the $BERT_{LARGE}$ is a very huge model with 24 transformer layers, each with 16 attention heads and embedding dimension of 1024.

3.2 Input and output representation

The BERT model is flexible to take single sequence inputs as well as pair of sequences. BERT uses WordPiece (12) embedding with a vocabulary size of 30,000 tokens. There are three important special tokens one should be aware of. [CLS] token which is the first token in any sample fed into the model. The corresponding representation is used for training a classifier on top of BERT during fine tuning. [SEP] token, which is used to differentiate between the pair of sequences in the input. [PAD] token used for padding the sequences to have same length. Each input representation is sum of its embedding and the positional encoding.

3.3 Training

BERT uses bidirectional transformers and is trained on two tasks, namely, masked language modeling(MLM) and next sentence prediction(NSP).

In MLM, the authors mask 15% of all WordPiece tokens in each sequence at random. If the i-th token is chosen, authors replace the i-th token with (i) the [MASK] token 80% of the time (ii) a random token 10% of the time (iii) the unchanged i-th token 10% of the time. Then, model is trained to predict the original token with cross entropy loss.

In NSP, when choosing the sentences A and B for each pretraining example, 50% of the time B is the actual next sentence that follows A (labeled as IsNext), and 50% of the time it is a random sentence from the corpus (labeled as NotNext).

3.3.1 Dataset

The pre-training corpus authors used are the BooksCorpus (800M words) (13) and English Wikipedia (2,500M words).

In Wikipedia data, only text passages were considered leaving out all the other items like headers, lists, tables, html tags etc.

4 Experiments

I coded[1] up the transformers using pytorch and ran on IMDB Movie review dataset (14) which contains 25K train and 25K test samples. Out of the 25K training samples, I used 5K as validation set.

[1]Code for all the experiments can be found at: `https://github.com/EshwarSR/TransAndBert`

Table 1: Accuracies on various transformer models. H=heads, L=layers, FT=Fine tuned

Model	8H 6L	8H 12L	16H 6L	16H 12L	FT $BERT_{BASE}$
Train	0.97	0.97	0.94	0.99	0.99
Valid	0.77	0.77	0.77	0.78	0.94
Test	0.76	0.76	0.76	0.76	0.93

I tried with a total of 4 transformer architectures. A combination of 8 and 16 heads over 6 and 12 layers trained for 20 epochs. After that, I also fine-tuned the $BERT_{BASE}$ uncased model over 5 epochs to compare the performances.

5 Results

The accuracies are summarized in Table 1. All the 4 transformer models provide similar results, but the fine tuned $BERT_{BASE}$ model outperforms the other models with a great margin. Please see [B] for the accuracy plots.

6 Discussion

With attention models boosting the performance of many NLP tasks, it has raised a lot of curiosity among researchers related to the interpretability of attention mechanisms. Studies explaining and debugging the current system's decision (15) and distilling important traits of a dataset (16) are also carried out. Authors of "Is Attention Interpretable?" (17) feel attention weights are only noisy predictors of even intermediate components' importance, and should not be treated as justification for a decision. They claim that in order for a model to be interpretable, it must not only suggest explanations that make sense to people, but also ensure that those explanations accurately represent the true reasons for the model's decision. They conclude that while attention noisily predicts input components' overall importance to a model, it is by no means a fail-safe indicator.

The work "Attention is not explanation" (18) performs extensive experiments across a variety of NLP tasks (like text classification, natural language inference (NLI), and question answering) that aim to assess the degree to which attention weights provide meaningful "explanations" for predictions. That authors find that they largely do not. The authors also propose an approach to counterfactual attention weights called "adversarial attention".

The work "Attention is not not explanation" (19) is another work challenging the claims of the previously mentioned work (18). The authors propose four alternative tests to determine when/whether attention can be used as explanation: a simple uniform-weights baseline; a variance calibration based on multiple random seed runs; a diagnostic framework using frozen weights from pretrained models; and an end-to-end adversarial attention training protocol. Each of which allows for meaningful interpretation of attention mechanisms in RNN models.

Similar works are done in trying to understand the Geometry of BERT (20) too. The authors show that the BERT model does good job with respect to word senses. They shown how mistakes in word sense disambiguation may correspond to changes in internal geometric representation of word meaning. They run experiments to show that BERT has both syntactic (particularly, dependency relations) and semantic subspaces. They claim that the internal geometry of BERT may be broken down into multiple linear subspaces, with separate spaces for different syntactic and semantic information.

Irrespective of the interpretability of attention, the truth is that these transformer based models have created a new benchmark in the domain of NLP. In the present crisis of the Covid-19 pandemic, these models are helping fight spread of misinformation and in retrieval of right information too. Some of the related work are, "An Analysis of BERT FAQ Retrieval Models for COVID-19 Infobot"[2], model to analyse COVID-19 Content on Twitter using BERT (21) and CrisisBERT (22), a classifier of crisis events, such as natural disasters, terrorist attacks and pandemics.

[2] `https://openreview.net/forum?id=dGOeF3y_Weh`

References

[1] Peters, M., M. Neumann, M. Iyyer, et al. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics, New Orleans, Louisiana, 2018.

[2] Kuchaiev, O., B. Ginsburg. Factorization tricks for LSTM networks. *CoRR*, abs/1703.10722, 2017.

[3] Kaiser, L. u., S. Bengio. Can active memory replace attention? In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, R. Garnett, eds., *Advances in Neural Information Processing Systems 29*, pages 3781–3789. Curran Associates, Inc., 2016.

[4] Vaswani, A., N. Shazeer, N. Parmar, et al. Attention is all you need, 2017.

[5] Kalchbrenner, N., L. Espeholt, K. Simonyan, et al. Neural machine translation in linear time. *CoRR*, abs/1610.10099, 2016.

[6] Gehring, J., M. Auli, D. Grangier, et al. Convolutional sequence to sequence learning. *CoRR*, abs/1705.03122, 2017.

[7] Devlin, J., M.-W. Chang, K. Lee, et al. 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. Association for Computational Linguistics, Minneapolis, Minnesota, 2019.

[8] Ba, J. L., J. R. Kiros, G. E. Hinton. Layer normalization, 2016.

[9] He, K., X. Zhang, S. Ren, et al. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[10] Bahdanau, D., K. Cho, Y. Bengio. Neural machine translation by jointly learning to align and translate, 2014.

[11] Radford, A., K. Narasimhan, T. Salimans, et al. Improving language understanding with unsupervised learning. *Technical report, OpenAI*, 2018.

[12] Wu, Y., M. Schuster, Z. Chen, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016.

[13] Zhu, Y., R. Kiros, R. Zemel, et al. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books, 2015.

[14] Maas, A. L., R. E. Daly, P. T. Pham, et al. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150. Association for Computational Linguistics, Portland, Oregon, USA, 2011.

[15] Lee, J., J.-H. Shin, J.-S. Kim. Interactive visualization and manipulation of attention-based neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 121–126. Association for Computational Linguistics, Copenhagen, Denmark, 2017.

[16] Yang, F., A. Mukherjee, E. Dragut. Satirical news detection and analysis using attention mechanism and linguistic features. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1979–1989. Association for Computational Linguistics, Copenhagen, Denmark, 2017.

[17] Serrano, S., N. A. Smith. Is attention interpretable?, 2019.

[18] Jain, S., B. C. Wallace. Attention is not explanation, 2019.

[19] Wiegreffe, S., Y. Pinter. Attention is not not explanation, 2019.

[20] Coenen, A., E. Reif, A. Yuan, et al. Visualizing and measuring the geometry of BERT. *CoRR*, abs/1906.02715, 2019.

[21] Müller, M., M. Salathé, P. E. Kummervold. Covid-twitter-bert: A natural language processing model to analyse covid-19 content on twitter, 2020.

[22] Liu, J., T. Singhal, L. T. M. Blessing, et al. Crisisbert: a robust transformer for crisis classification and contextual crisis embedding, 2020.

Appendix

A Figures

The following images are taken from the original transformer paper (4).

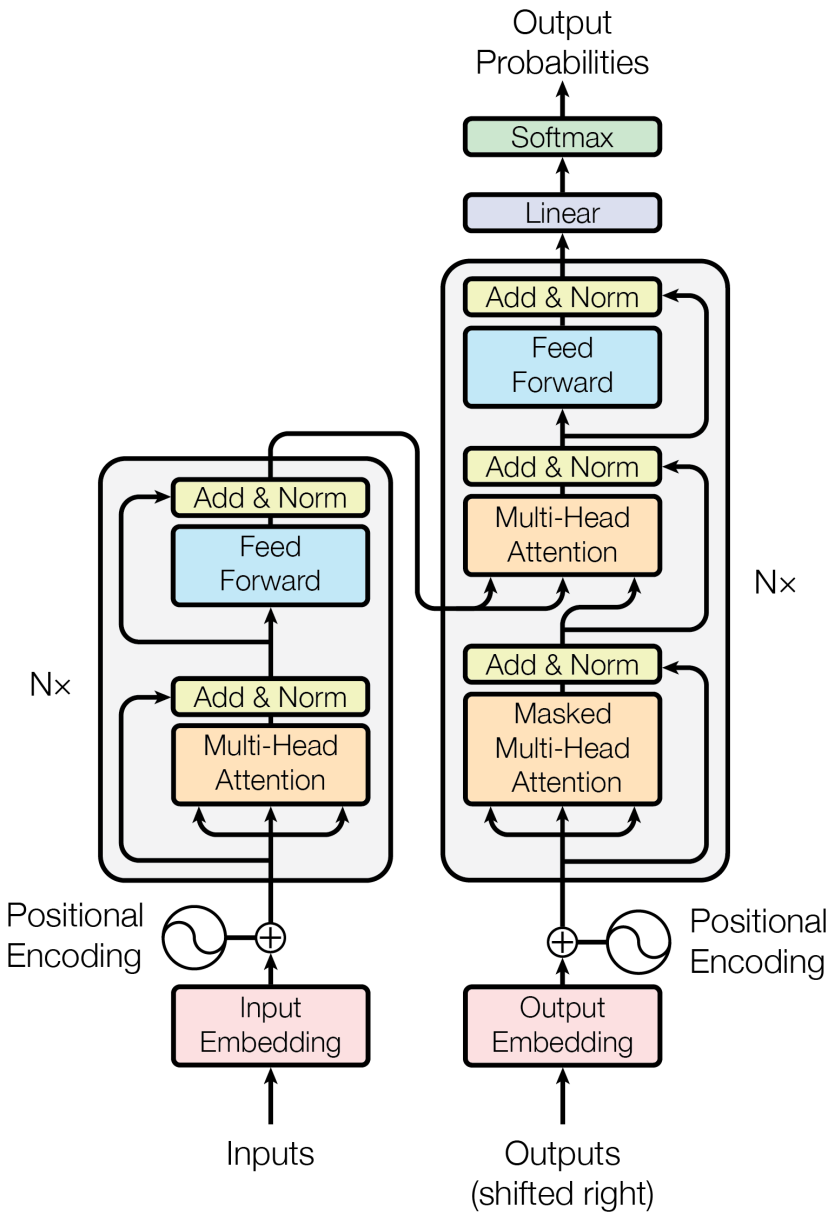


Figure 1: Overview of Transformer Architecture

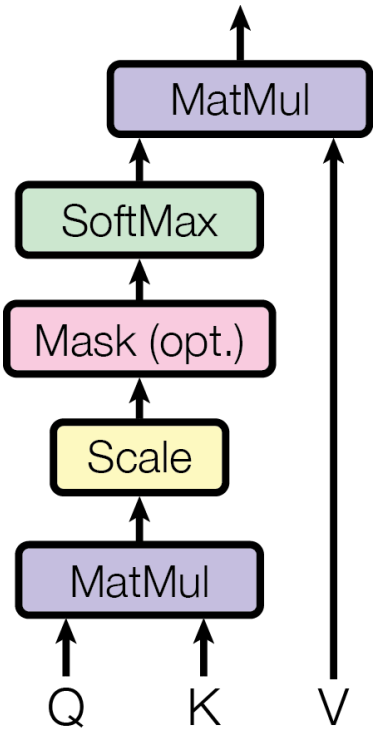


Figure 2: Scaled dot product

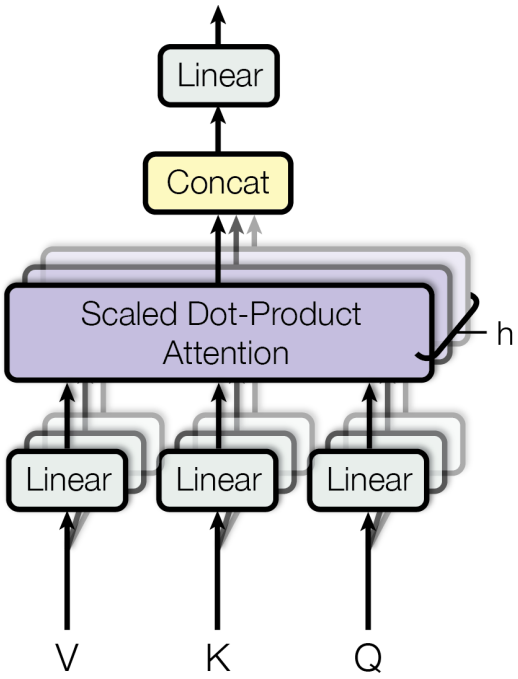
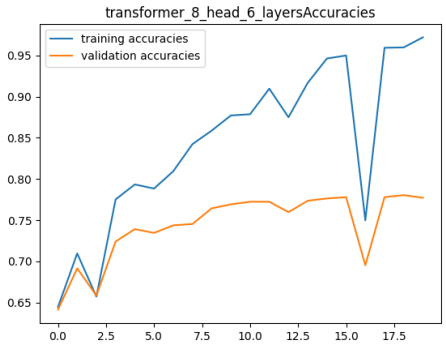


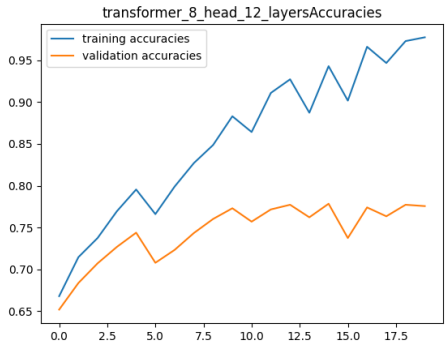
Figure 3: Multihead attention

B Plots

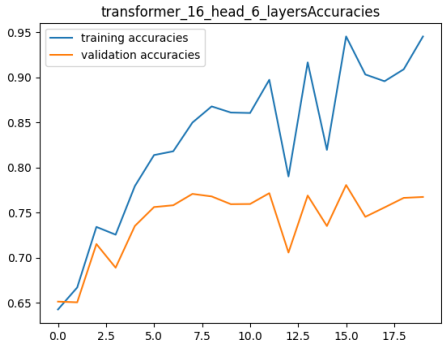
Below are the training and validation accuracy plots against each epoch for the models of Table 1.



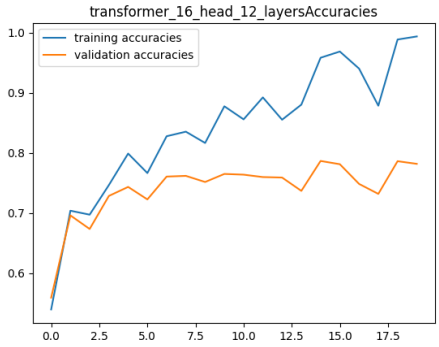
Accuracies of 8 heads 6 layered model



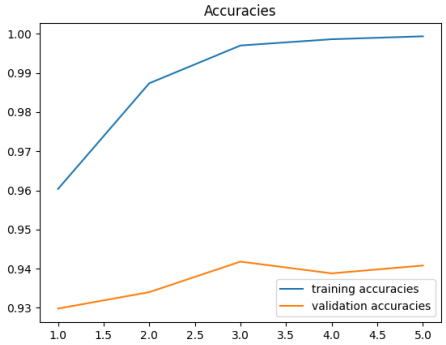
Accuracies of 8 heads 12 layered model



Accuracies of 16 heads 6 layered model



Accuracies of 16 heads 12 layered model



Accuracies of fine-tuned BERT