UMEÅ UNIVERSITY
Department of Mathematics
and Mathematical Statistics

January 5, 2023

Attention Is All You Need

Statistical Learning with High Dimensional Data

Supvervisor
Armin Eftekhari

Christoffer Sjölund: chsj0028@student.umu.se
Hynek Kydlíček: kydlicek.hynek@gmail.com
Sam Adhami: saad0029@student.umu.se

Contents

1 Introduction

1.1 Summary of paper

"Attention is All You Need" is a research paper that presents the Transformer, a new type of neural network architecture for natural language processing tasks. The Transformer is based on the idea of self-attention, which allows the model to directly model relationships between input tokens without the need for recurrence or convolution.
The authors of the paper demonstrate that the Transformer outperforms previous state-of-the-art models on several benchmarks for machine translation and language modeling tasks. The model is able to achieve this performance while being simpler and more efficient than previous models, making it a practical choice for many NLP applications.
One key innovation of the Transformer is the use of multi-headed self-attention, which allows the model to attend to multiple input tokens simultaneously. This allows the model to capture complex dependencies between input tokens, which is important for many NLP tasks.
Overall, the "Attention is All You Need" paper has had a significant impact on the field of natural language processing, and the Transformer architecture has become widely used in a variety of NLP tasks.

2 Background

Before this paper was published, most successful approaches to natural language processing tasks, such as machine translation and language modeling, used recurrent neural networks (RNNs) or convolutional neural networks (CNNs). These models had some limitations, including difficulty in capturing long-range dependencies between input tokens and difficulty in parallelizing computations. The "Attention is All You Need" paper introduces a new type of neural network architecture called the Transformer, which is based on the idea of self-attention. Self-attention allows the Transformer to directly model relationships between input tokens, rather than relying on recurrence or convolution. This makes the Transformer simpler and more efficient than previous models, while also allowing it to capture complex dependencies between input tokens. The authors of the paper demonstrate that the Transformer outperforms previous state-of-the-art models on several benchmarks for machine translation and language modeling tasks.
In summary, the "Attention is All You Need" paper presents the Transformer, a new type of neural network that uses self-attention to model relationships between input tokens and is simpler and more efficient than previous models. The Transformer has had a significant impact on the field of natural language processing and is widely used in a variety of tasks.

3 Related works

The transformer layed the foundation to a lot of work within neural networks and deep learning where Bidirectional Encoder Representations from Transformers and generative pre-trained transformer are popular further works [5][2]. It is a neural machine translator and it is used to translate text from one language to another [9]. Self attention have gotten good results while reducing the computation times [3]. The transformer reduced the data needed and therefore the computational cost compared to previous encoder-decoder networks. It was later improved with weighted transformers [1]. It has also been adapted to speech interpretation and video frame interpolation [8][4]

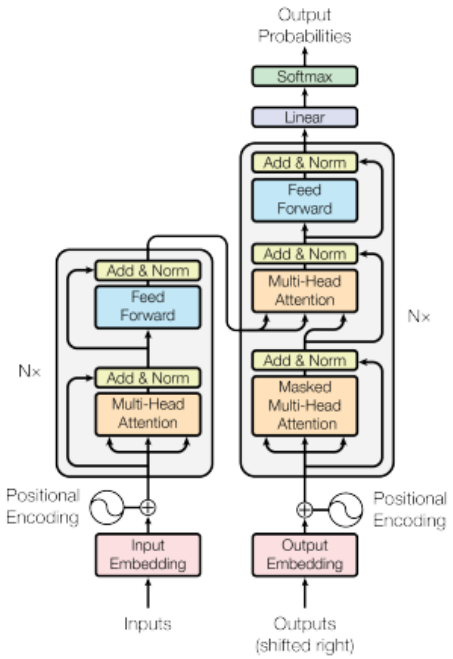

Figure 1: Model architecture [9]

4 Method

4.1 Overview of the Model

The transformer is a encoder-decoder architecture. This means that we first use an encoder to encode the input sequence into a hidden state and then we use a decoder to decode the hidden state into the output sequence. Let's no look at the encoder and decoder in more detail. The model architecture can be seen in Figure 1.

4.2 Encoder

The encoder is a stack of N identical layers. Each layer is a multi-head self-attention layer followed by a feed-forward layer. While feed-forward network is simple expand-reduce FF the multi-head self-attention layer is the most innovative feature of the whole model. Model also uses residual connections and layer normalization, which are there to improve stability of learning (Layer normalization) and to make model train faster (Residual connections as it improves gradient flow). It also employs dropout to prevent overfiting. Let's now explain

the key part of the model, the multi-head self-attention layer.

4.3 Multi-Head Self-Attention Layer

To simplify the notation we will assume non-batched input. Thus input to the block is tensor of shape (sequence_length, d_model) and output is also tensor of shape (sequence_length, d_model). First we create 3 tensors of shape (sequence_length, d_att) from input using Feed-Forward. These tensors are called query, key and value. Then we split them into H heads creating 3 tensors of shape (sequence_length, heads, d_att/heads). For every head we now calculate attention weights as follows:

$$\text{Attention}(Q_H, K_H, V_H) = \text{softmax}(\frac{Q_H {K_H}^T}{\sqrt{d\_att/heads}})V_H$$

We then concatenate the heads and apply another Feed-Forward layer to get the output of the block as follows:

$$\text{MultiHeadAttention}(Q, K, V) = \text{Feed-Forward}(\text{concat}(\text{Attention}(Q_1, K_1, V_1), \ldots, \text{Attention}(Q_H, K_H, V_H)))$$

The output of the block is then of shape (sequence_length, d_model) and we can use it as an input to the next layer.
The intuitive idea behind the attention layer is that we want to find out which parts of the input are the most important for the current part of the input. This is done by calculating the dot product of query and key vectors. We then keep the most important parts by multiplying the soft-amaxed dot product with values. Thus only important values for the other parts will have high impact.
As we can also see such the whole computation is just matrix multiplications and simple concatenation. This allows to do all in parallel and speed up the model unlike RNNs.

4.4 Decoder

Decoder is very similar to encoder. It is also a stack of N identical layers. Each layer is a multi-head self-attention layer followed by a multi-head attention layer and a feed-forward layer. We have already explained both multi-head self-attention and feed-forward layers. The multi-head attention layer is just a multi-head self-attention layer with different query, key and value tensors. The query tensor is the output of the previous layer, the key and value tensors are the output of the encoder. This allows the decoder to attend to the encoder output. The decoder also uses residual connections and layer normalization. It also employs dropout to prevent overfitting.

4.5 Positional Encoding

The positional encoding is a simple trick that allows the model to use relative position of the words. It is a tensor of shape (sequence_length, d_model) and it

Type	Layer Complex.	Seq. Operations	Max. Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Reccurent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(logk(n))$
Self-Attention (rest.)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

Table 1: Comparison of different layer types. (n - sequence length, d - d_model, k - kernel size, r - neighborhood restriction)

is added to the input of the encoder. The values of the tensor are calculated as follows:

$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d\_model})$$

$$PE_{(pos,2i+1)} = \cos(pos/10000^{2i/d\_model})$$

While the paper used this positional encoding the later models used different encodings. The most common one is the one used in BERT [5]. Instead of using sine and cosine it uses trained embedding for each position. The output is then again added to the input of the encoder. The approach however has small drawback as it then doesn't allow input to be longed than the number of positions in the embedding.

4.6 Masking

To be able to use batch inputs it is important to employ masking of input. Because not all sentences are the same lenght we have to add padding to sentences but we don't want this padding to affect the model. Thus we mask the padding so that the model doesn't see it. We also mask the future tokens in decoder self-attention so that the model doesn't cheat and look at the future tokens.

4.7 Why is Self-attention other approaches?

The authors compare the architecture with the other commonly used in 3 aspects. Overall complexity (Complexity per Layer), parallelization (Sequential Operations) and long range dependencies (Maximum Path Length). The table 1 above shows the results of the comparison. As can be seen transformers are very good at long term dependencies and are very well paralyzable. However there can be problem when we use too long sequences as the complexity per layer will be higher than for other approaches. The authors also considered modified version called restricted transformer where the self-attention is restricted to a neighborhood of size r. This allows to have lower complexity per layer but it also limits the model to only long term dependencies.

5 Adjustment for the WMT14 Task

5.1 WMT14 Translation Task

WMT14 Translation Task is a translation task. It proposes language pairs and the task is to translate the sentences from one language to another. The evaluation for this task is done by using BLEU score [6]
Since the paper used the WMT14 en-de we only trained the model on this pair.

5.2 Data Preprocessing

As proposed by the paper we used byte-pair tokenization [7]. The advantage of this tokenization is that it allows producing and reading out of vocabulary words. This is important as we want to be able to translate words that are not in the training set.

5.3 Training

We used Transformer with similar parameters as described in paper. This means we used vocabulary of size 30k, embedding size of 512, 3 layers, 4 heads and 2048 hidden units in the feed-forward layer.
As optimizer we used Adam with alternating learning rate as described in paper. We used beta 1 of 0.9, beta 2 of 0.98 and epsilon of 10^{-9}. We and warmup steps of 4000. We used batch size of 16 and no gradient clipping.
Due to the limited computation resources we used only 2M sentence pairs for training. We trained the model for 1 epoch.
We trained the model on google Colab using GPU(K80 GPU)) for about 8 hours.

6 Results

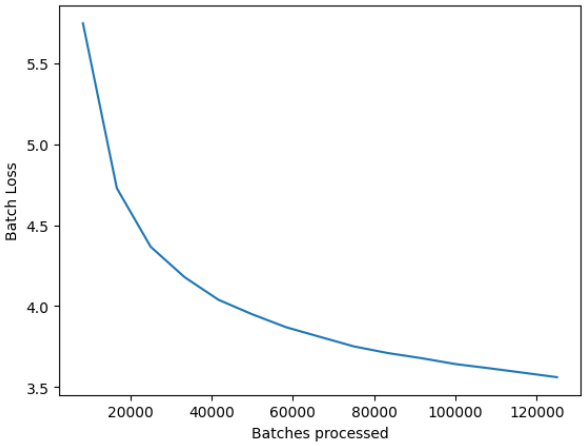

Figure 2: Training results

The training results can be seen at Figure 2. We used BLEU(ngrams=2) score to evaluate the model. Unfortunately we couldn't find the n_grams used by the paper so we used 2. We got BLEU score of 0.2 on the test set.

7 Discussion

The observed BLEU score is much worse than the paper which got 28.4. This shouldn't be surprising as we used only 2M sentence pairs with for training and the paper used 4.4M sentence pairs. The paper also employed model Ensemble method by average the last x checkpoints of models. In addition to this they also used beam search with beam size of 4. We didn't use any of these methods. The translation task is very complex and BLEU score is very unforgiving, thus to get better score we would need to train much bigger model for longer time.

References

[1] Karim Ahmed, Nitish Shirish Keskar, and Richard Socher. *Weighted Transformer Network for Machine Translation*. Nov. 6, 2017. arXiv: 1711.02132 [cs]. URL: http://arxiv.org/abs/1711.02132 (visited on 01/02/2023).

[2] Tom B. Brown et al. *Language Models Are Few-Shot Learners*. July 22, 2020. arXiv: 2005.14165 [cs]. URL: http://arxiv.org/abs/2005.14165 (visited on 01/02/2023).

[3] Jianpeng Cheng, Li Dong, and Mirella Lapata. *Long Short-Term Memory-Networks for Machine Reading*. Sept. 20, 2016. arXiv: 1601.06733 [cs]. URL: http://arxiv.org/abs/1601.06733 (visited on 01/02/2023).

[4] Myungsub Choi et al. "Channel Attention Is All You Need for Video Frame Interpolation". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34.07 (07 Apr. 3, 2020), pp. 10663–10671. ISSN: 2374-3468. DOI: 10.1609/aaai.v34i07.6693. URL: https://ojs.aaai.org/index.php/AAAI/article/view/6693 (visited on 01/02/2023).

[5] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. May 24, 2019. arXiv: 1810.04805 [cs]. URL: http://arxiv.org/abs/1810.04805 (visited on 01/02/2023).

[6] Kishore Papineni et al. "Bleu: A Method for Automatic Evaluation of Machine Translation". In: *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. ACL 2002. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, July 2002, pp. 311–318. DOI: 10.3115/1073083.1073135. URL: https://aclanthology.org/P02-1040 (visited on 01/02/2023).

[7] Rico Sennrich, Barry Haddow, and Alexandra Birch. *Neural Machine Translation of Rare Words with Subword Units*. June 10, 2016. arXiv: 1508.07909 [cs]. URL: http://arxiv.org/abs/1508.07909 (visited on 01/02/2023).

[8] Cem Subakan et al. *Attention Is All You Need in Speech Separation*. Mar. 8, 2021. arXiv: 2010.13154 [cs, eess]. URL: http://arxiv.org/abs/2010.13154 (visited on 01/02/2023).

[9] Ashish Vaswani et al. *Attention Is All You Need*. Dec. 5, 2017. arXiv: 1706.03762 [cs]. URL: http://arxiv.org/abs/1706.03762 (visited on 01/02/2023).