Report: Attention is All You Need

V. GÖLZ and L. FUENTES

February 27, 2023

Contents

1 Introduction

In this report we summarize the most important ideas from the paper [Vas+17] and also introduce our own simple implementation.

In this paper, which is called *Attention is All You Need*, the authors propose a novel neural-network architecture, called the *Transformer*. Today (2023), this architecture is still state-of-the-art (SOTA), meaning that it is used in many different areas in deep learning while yielding by far the best results compared to other techniques. Transformers are notably used in *natural language processing* (NLP), but also in *computer vision* (CV) and *speech processing*.

2 Historical Background

Before the introduction of transformer-based models, *recurrent-neural-networks* (RNNs) were the SOTA models. Over the years, different versions of these models were developed. These models marked a milestone in the history of NLP, since their performance was superiour to previous techniques. Still, RNNs had many problems, for example that they were not capable of retaining long-term information. They also had vanishing and exploding gradient problems, and, arguably even more important, they were not able to train in parallel because of their sequential structure. This detail made such networks very inefficient to train, even more so because modern computers and especially graphics processing units (GPUs) are optimized for matrix-vector products. An improvement for the memory problem were *long-short-term-memory networks* (LSTMs) and *gated recurrent units* (GRUs) which introduced a memory mechanism, leading to better results for long-term dependencies. Still, the main problems remained the same, namely the fact that they cannot be efficiently trained in parallel and that long-term dependencies still posed problems. The stage was now set for new innovations to come along.

In 2017 a research team at *Google Brain* lead by Ashwin Vaswani published the paper *Attention is All You Need* [Vas+17]. This publication was transformational in the field of *Deep Learning* and especially in NLP. Almost all SOTA architectures use this paper as a foundation, notably ChatGPT, DALL-E, BERT and more.

What made this paper so special was the introduction of the *Transformer*. This architecture solved many of the existing challenges with recurrence-based models. The main features of transformer models can be summarized as follows:

1. They are able to understand long-term dependencies without limits or bottlenecks.
2. They have less problems with gradient vanishing and explosion, because recurrent mechanisms are no longer used.
3. Training-costs can be reduced massively. Modern hardware (namely GPUs such as the ones from Nvidia) are heavily optimized for parallel matrix-vector calculations, strongly favoring the transformer architecture over RNNs.

Recently, Transformer models have drawn a lot of media attention. Since the introduction of ChatGPT in late 2022, many newspapers and authors ask themselves if this is in fact the beginning of a new AI era. ChatGPT in particular has seen over a 100 million clients using their service. This also started a technological race between giants such as Alphabet, Microsoft and Meta. While opinions on this topic differ, it clearly shows the impact of the paper.

3 The Attention-Mechanism

3.1 Intuition

We now dive deeper into the mathematical details and introduce the *attention mechanism*. For this purpose, we first explain the intuition behind attention. The idea is to introduce contextual information to the original input. The easiest example is *self-attention*. Here the input and out phrases are same. As visualized in figure 1, the algorithm finds other words in the same phrase that relate most to the given word.

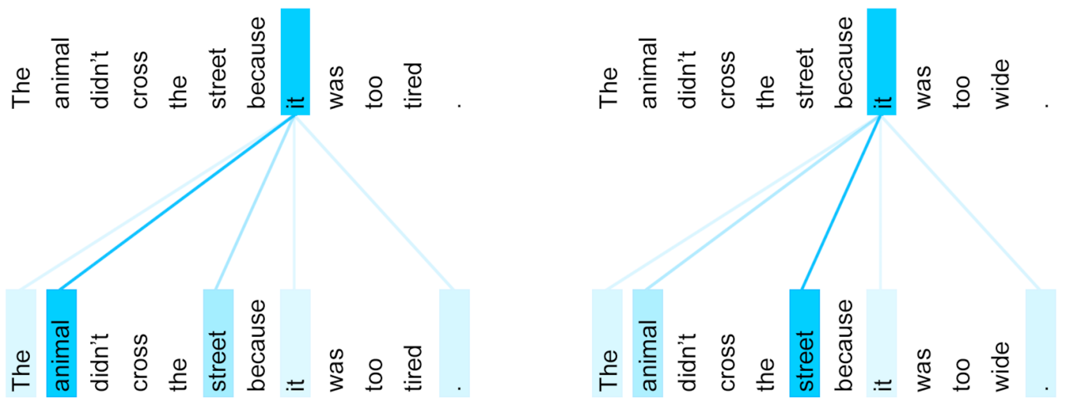
The animal didn't cross the street because it was too tired .
The animal didn't cross the street because it was too tired .
The animal didn't cross the street because it was too wide .
The animal didn't cross the street because it was too wide .

Figure 1: The encoder self-attention distribution for the word *it* from the 5th to the 6th layer of a Transformer trained on English to French translation (one of eight attention heads). Image source: ai.googleblog.com.

3.2 Mathematical Formulation

Mathematically the *attention mechanism* is modelled after a query q that is retrieving values v from a database. The database consists of key-value pairs (k_i, v_i). In a classic database, the query fits exactly one key, hence only one value is returned. The attention mechanism uses a more probabilistic approach. Let $q, k_i \in \mathbb{R}^{d_k}$ and $v_i \in \mathbb{R}^{d_v}$. Then the formula is given by

$$\text{attention}(q, k, v) = \sum_I \text{similarity}(q, k_i) \times v_i, \tag{1}$$

where we have

$$\text{similarity} : \mathbb{R}^{d_k \times d_k} \to [0, 1]. \tag{2}$$

Note that if the similarity function simply tests for equality, we are in the classic setting. Popular similarity functions for attention are

- $q^\intercal k$ (dot-product)
- $\frac{q^\intercal k}{\sqrt{d_k}}$ (scaled dot-product)
- $q^\intercal W k$ (weighted dot-product)
- $W^q q + W^k k$ (weighted sum)

where d_k is the dimension of the keys and $W, W^q, W^k \in \mathbb{R}^{d_k \times d_k}$ are weight matrices. In the paper [Vas+17] the scaled dot-product is used. Note also that this function can easily be run in parallel by concatenating several queries, keys and values to matrices. The attention function from the paper therefore is

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\intercal}{\sqrt{d_k}}\right) V. \tag{3}$$

The purpose of the division by $\sqrt{d_k}$ is to prevent the dot products from getting too large. Assume that q and k are independent random variables with mean 0 and variance 1. Then $q^\intercal k = \sum_{i=1}^{d_k} q_i k_i$ has mean 0 and variance d_k, hence the division scales back the variance. The function also uses *softmax* to normalize the factors that are multiplied with the values.

3.3 Multi-Head-Attention

In the paper, the authors found that it is useful to use several attention functions in parallel. Therefore, they linearly project the queries, keys and values h times with different learned projections. These projections are then passed through the attention function and concatenated, before once again being linearly projected to the desired output shape. The formula is

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h) W^O \tag{4}$$

with

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{5}$$

The projections are trainable parameter matrices $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$ and $W^O \in \mathbb{R}^{h \times d_v \times d_{\text{model}}}$. The idea is that by using several different projections of the same inputs, the model can pay attention to different patterns at the same time. This idea is similar to using several different kernels in CNNs in order to detect different patterns in images, thus yielding better results.

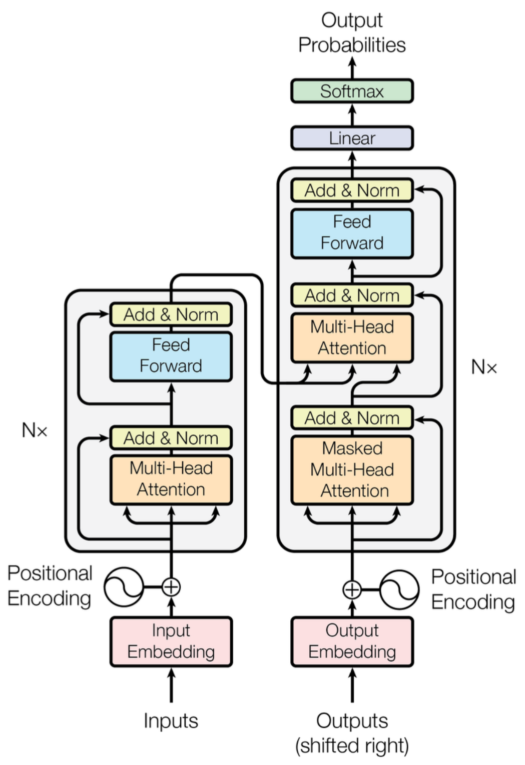

Figure 2: The Transformer model architecture. Main features are input embedding, positional encoding, the encoder block, output embedding and the decoder block. Source: [Vas+17].

4 General Architecture

The structure of the Transformer is similar to older architectures with recurrent layers, such as in [Cho+14]. It uses an *encoder-decoder* structure, where the main difference is the replacement of recurrent layers with attention layers. Because attention layers do not retain positional information, *positional encoding* (PE) is added to the input embedding. An overview of the model architecture can be found in Figure 2.

4.1 Encoder-Decoder

The Encoder aims to add information to the original input sentence, regarding the importance of words with respect to other words in the phrase. This block is composed of a stack of N identical layers, which we call *attention layers*. Each attention layer is composed of two sub-layers. The first one is a multi-head attention layer that employs self-attention, the second one is a simple fully connected feed-forward layer formed by two linear functions with a ReLU in between. Both sub-layers have a residual connection around it and layer normalization afterwards in order to evade vanishing gradient problems and speed up training. We repeat this procedure N times. The first time we consider words, on the second round, we analyse the importance of pairs of words, and so on. Note that the input and output shapes are equal.

The Decoder is used to predict the most probable word in the translated sentence. It has a similar structure compared to the encoder, consisting of a stack of M identical layers. Note that in the paper $N = M = 6$ is used. However, a decoder layer consists of three sub-layers. The first is a masked multi-head attention layer employing self-attention. This sublayer masks the next outputs to avoid illegal connections and makes predictions using only previous information. The second is a normal multi-head attention layer, where the keys and queries are taken from the encoder output, while the values come from the previous sub-layer. This is where the translation process happens. By using keys and queries from the encoder, the decoder extracts information from the original sentence to predict the next word. The last layer is a fully connected layer as it was the case in the encoder.

4.2 Positional Encoding

Positional encoding adds a representation of a word's position in a sentence to the embedding. In the paper, they use a method called *frequency-based positional embedding* by using the following equations:

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{model}})$$

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

This offers a unique encoding at each step, generalizes well on longer sequences and is deterministic. By applying the sine and cosine functions, we have bounded values inside [-1,1] which we will be used to spot the rate of change. The frequency is used in order to modify the wavelength in which words will be plotted. The output for each positional encoding step pos is a vector of dimension d_{model} which is concatenated to the input embedding.

5 Performance

In the initial paper, the performance was measured by several translation tasks such as EN-to-DE and EN-to-FR. The metric used to evaluate the performance was the BLEU-score[1]. Table 3 shows the performance of the architecture employed in the paper compared to other SOTA models. In Figure 3, we can already see that the Transformer performed better than all other models. Furthermore, it is important to note that the previous models have been highly optimized whereas the Transformer was essentially a prototype. Note also the huge reduction in training costs by an order of 10^3. That alone represents a significant research finding.

[1]The BLEU-score evaluates the quality of a translation by comparing it to reference translations made by expert humans. For more information, see wikipedia.org/wiki/BLEU.

Model	BLEU		Training Cost (in FLOPS * 10^{18})	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet (Kalchbrenner et al., 2016)	23.75			
Deep-Att + PosUnk (Zhou et al., 2016)		39.2		100
GNMT + RL (Wu et al., 2016)	24.6	39.92	23	140
ConvS2S (Gehring et al., 2017)	25.16	40.46	9.6	150
MoE (Shazeer et al., 2017)	26.03	40.56	20	120
GNMT + RL Ensemble (Wu et al., 2016)	26.30	41.16	180	1100
ConvS2S Ensemble (Gehring et al., 2017)	26.36	**41.29**	77	1200
Transformer (base model)	27.3	38.1	**3.3**	
Transformer (big)	**28.4**	**41.8**	23	

Figure 3: Performance evaluation from the paper. The results clearly show that the transformer model improves on standard language tasks over previous SOTA models. Note that at the same time, training costs were cut by a factor of 10^3.

6 Own Implementation

To see the preceding sections in practice, we implement our own model with a similar architecture in order to predict the sentiment of movie reviews. Note that this is a binary classification problem. The goal is to predict the labels 1 or 0 indicating whether the review is positive or negative, respectively. Since we are not generating new phrases or translations, we do not need a decoder in our architecture. Hence, we only use the first half of the Transformer architecture, followed by linear and softmax function.

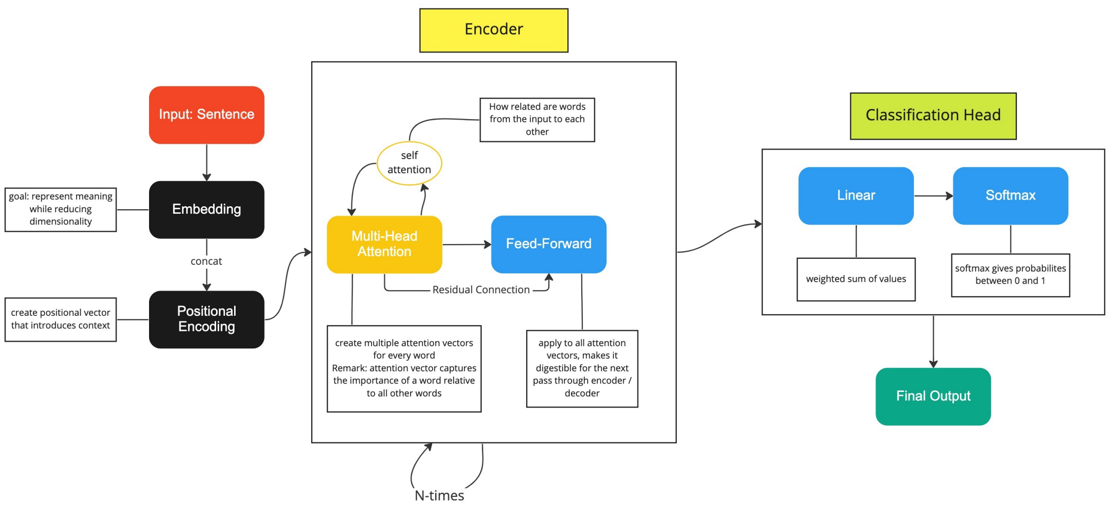
Encoder
Input: Sentence
goal: represent meaning while reducing dimensionality
Embedding
concat
create positional vector that introduces context
Positional Encoding
How related are words from the input to each other
self attention
Multi-Head Attention
Feed-Forward
Residual Connection
create multiple attention vectors for every word Remark: attention vector captures the importance of a word relative to all other words
apply to all attention vectors, makes it digestible for the next pass through encoder / decoder
N-times
Classification Head
Linear
Softmax
weighted sum of values
softmax gives probabilites between 0 and 1
Final Output

Figure 4: Architecture of our own implementation. The N encoder layers are followed by a classification head. This architecture was build to predict the sentiment of movie reviews.

Our implementation can be found here. We use python 3.10 and pytorch 1.13.1 for the network. The attention implementation has been taken from annotated-transformer.

Dataset. We use a dataset containing n=50.000 movie reviews from the web page IMDB and their respective labels (1 for "positive" and 0 for "negative"). We first take the processed reviews, merge values into single variables separated by whitespaces and obtain a list of words. Then, we build the vocabulary by counting, sorting and enumerating the different words in the dataset. We encode the words using the functions review.split and pad our sequences to have the same length in all reviews. We set seq_length to 256.

Architecture. We employ input embedding and positional encoding before passing the tensors into our *attention layers*, as introduced in 4.1. We use N layers. The tensors are then passed into the *classification head*. The head consists of two linear layers, where ReLU is used as activation function for the first layer. The $(\text{input}, \text{output})$-dimensions of the layers are $(d_{\text{model}} \times \text{seq\_length}, 128)$ and $(128, 2)$.

Training and Testing. For training, we use the *Adam* optimizer with a learning rate of 0.001, epochs between 5 and 7 and n=40.000 training samples. We use n=10.000 test samples.

Results. The results are shown in Table 1. Overall we can see that the model performed reasonably well, with an accuracy of up to 88%. One interesting finding is that an increased model size did not necessarily result in a better performance. This might be due to overfitting problems or an insufficient amount of data. In particular, increasing the number of epochs to more than 5 did not improve the accuracy. For larger models with more parameters, n=40.000 might not be enough to tune all weights properly.

epochs	**d_model**	**N**	**d_ff**	**h**	**Accuracy**
5	128	2	256	2	87.8%
7	128	2	256	2	87.3%
5	128	2	256	4	**88.4%**
7	128	4	256	4	88.0%
5	128	4	256	2	86.9%
5	128	4	256	4	83.4%
5	512	2	1024	4	84.4%

Table 1: Test results for different hyperparameter settings. Greater model size and more epochs did not results in a better performance.

7 Conclusion

In this work, we explained the main ideas from the paper *Attention is all you need*. We briefly introduced the historical background and set the paper into context. We then moved on to explain the most important features of the transformer architecture conceptually and mathematically. Afterwards, we presented the performance improvements of the network compared to other models. Lastly, we implemented a simple version of this model to perform a sentiment analysis on a dataset of movie reviews. We tested different hyperparameters and managed to achieve an accuracy of almost 90%, which is a good result for such a small model.

8 References

[Cho+14] Kyunghyun Cho et al. *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation*. 2014. DOI: 10.48550/ARXIV.1406.1078. URL: https://arxiv.org/abs/1406.1078.

[Dev+18] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2018. DOI: 10.48550/ARXIV.1810.04805. URL: https://arxiv.org/abs/1810.04805.

[Vas+17] Ashish Vaswani et al. *Attention Is All You Need*. 2017. DOI: 10.48550/ARXIV.1706.03762. URL: https://arxiv.org/abs/1706.03762.

[Vas+18] Ashish Vaswani et al. *Tensor2Tensor for Neural Machine Translation*. 2018. DOI: 10.48550/ARXIV.1803.07416. URL: https://arxiv.org/abs/1803.07416.

[Xie+21] Huiqiang Xie et al. "Deep Learning Enabled Semantic Communication Systems". In: *IEEE Transactions on Signal Processing* 69 (2021), pp. 2663–2675. DOI: 10.1109/tsp.2021.3071210. URL: https://doi.org/10.1109.