Attention is All You Need!

Week #5 - 17 Oct ‘24

NLP Lab

NLP Lab Project : Build your own GPT

1. Text Processing and Bag of Words Model
2. N-grams and TF-IDF
3. Theory of Word Embeddings : Word2Vec, GloVe
4. RNNs: LSTM, GRU; Attention Mechanisms
5. Transformer Model, Advanced Transformers: GPT, BERT
6. Fine-Tuning, SFT, Continued Pre-training
7. Custom Language Model using Prompt Engineering, RAG

1. Transformer Model

1.1 Transformer Architecture

- Encoder & Decoder Stacks
- Attention
    - Scaled Dot-Product Attention
    - Multi-Head Attention
- Point-wise Feed-forward Networks
- Embeddings and Softmax
- Positional Encoding

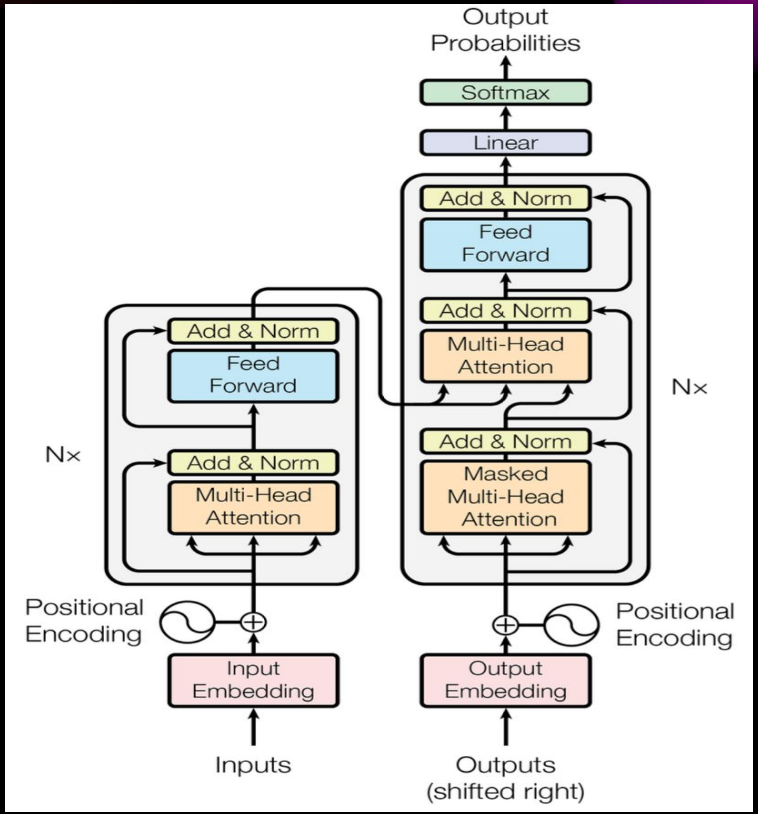


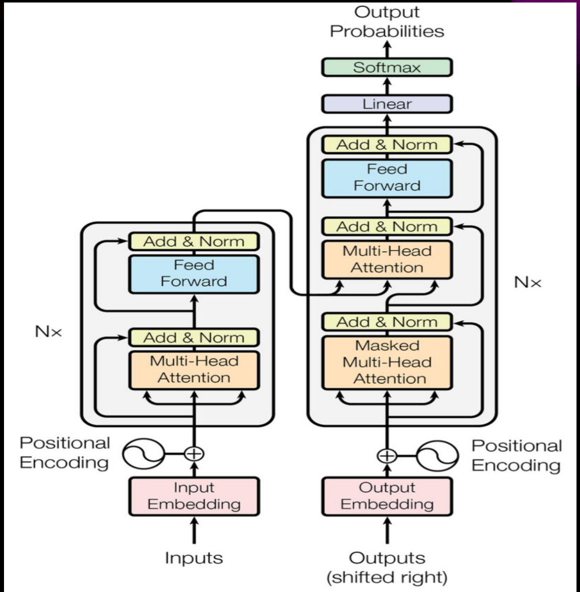
Image Source : A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need”,

1.2 Encoder & Decoder Stacks

Encoder :

- Composed of a stack of N = 6 identical layers.
- Each layer has two sub-layers :
  - multi-head self-attention mechanism
  - position- wise fully connected feed-forward network
- Residual Connection is applied around each of the two sub-layers.

LayerNorm(x + Sublayer(x))



Decoder :

- Composed of a stack of N = 6 identical layers.
- In addition to the sub-layers in each encoder layer, the decoder inserts a third sub-layer, which performs multi-head attention over the output of the encoder stack.
- The self-attention sub-layer in the decoder stack is modified (combined with fact that the output embeddings are offset by one position) to prevent positions from attending to subsequent positions.

1.3 Scaled Dot-Product Attention

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

- Dot-product attention is identical to this, except for the scaling factor of $1/\sqrt{d_k}$.
- The authors suspected that for large values of d_k, the dot products grow large in magnitude, pushing the softmax function into regions where it has extremely small gradients 4. To counteract this effect, we scale the dot products by $1/\sqrt{d_k}$.

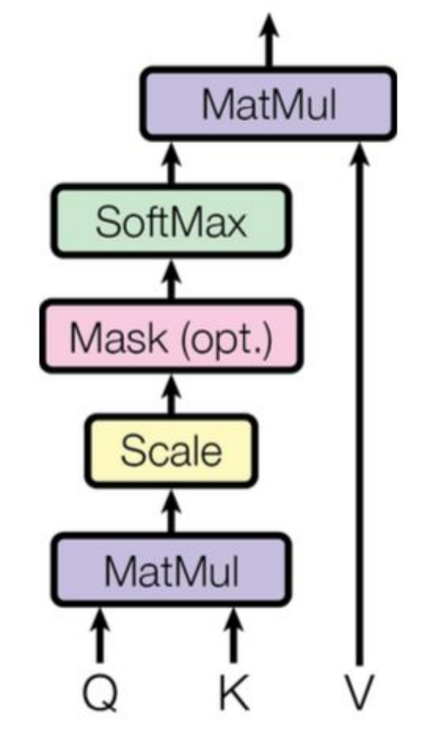
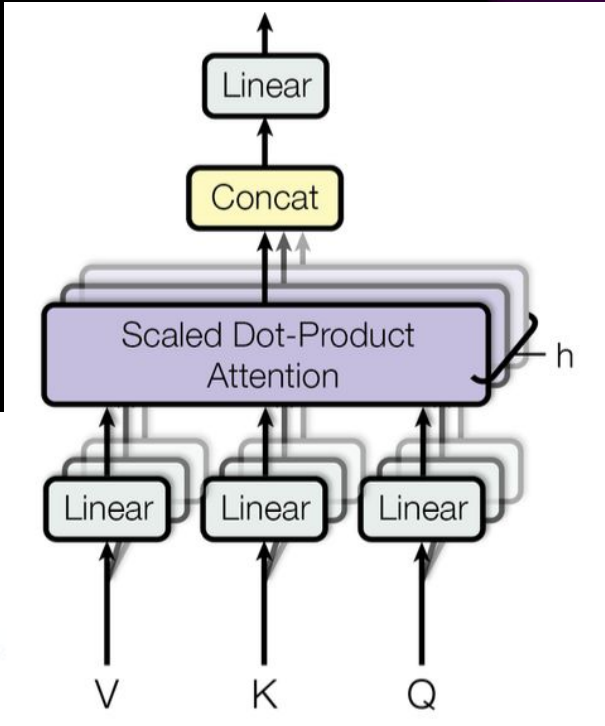


Image Source : A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need",

1.4 Multi-Head Attention

- Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions.
-



$$\mathrm{MultiHead}(Q,K,V) = \mathrm{Concat}(\mathrm{head}_1,\ldots,\mathrm{head}_\mathrm{h})W^O$$
$$\text{where } \mathrm{head}_\mathrm{i} = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Where the projections are 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 $W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$.

Image Source : A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need”,

Applications of Attention in our Model

- Encoder - Decoder attention layers : Queries from the decoder later; memory keys and values from the output of the encoder.

- Encoder contains self attention layers : All the keys, values and queries come from the same place (the output of the previous layer)

- Decoder contains self attention layers : Allows each position in the decoder to attend all positions in the decoder up and including that position

1.5 Point-wise Feed-Forward Networks

- In addition to attention sub layers, each of the layers in the encoder and decoder contains a fully connected FFN.

- Consists of 2 linear transformations with a ReLU activation in between.

$$\mathrm{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

1.6 Embeddings & Softmax

- Learned embeddings is used to convert the input and the output tokens to vectors.

- The usual learned linear transformation and softmax function are used to convert the decoder output to predicted next-token probabilities.

- Same weight matrix is shared between the two embedding layers and the pre-softmax linear transformation

1.7 Positional Encoding

- Since the model contains no recurrence and no convolution, we need to inject some information about relative or absolute position of the tokens in the sequence.

- That is why we add positional encodings to the input embeddings at the bottom of the encoder and the decoder stacks.

- The original paper used sine and cosine functions of different frequencies.

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

1.8 Why Self-Attention

- **Computational Complexity:** O(1) per layer; efficient for short sequences.
- **Parallelization:** High parallelization allows simultaneous processing.
- **Long-Range Dependencies:** Short paths facilitate easier learning of dependencies.
- **Model Interpretability:** Provides clear attention distributions for better insights.

1.9 Advantages of Transformer over other Sequence models

- **Parallelization :** Transformers process entire sequences simultaneously due to self-attention
- **Memory Efficiency :** Transformers use positional encodings to retain order without requiring explicit memory structures like RNNs/LSTMs, reducing the memory overhead.
- Transformers generalize better with attention mechanisms and multi-head architecture, learning more nuanced relationships across the entire dataset.
- Transformers scale better with longer sequences due to direct relationships formed via attention heads, which maintain information from all parts of the sequence.

Test you knowledge!

-