Intro to transformers

Monday, 5. August 2024 09:57

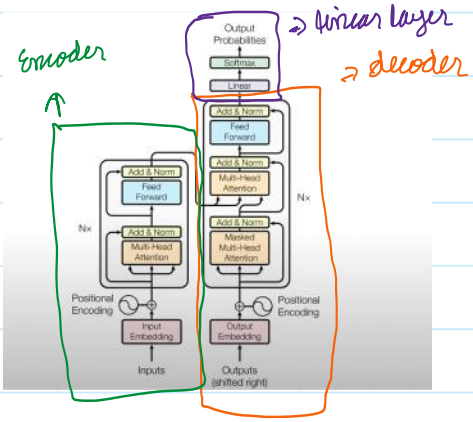
⇒ problems with RNN:

① Slow computation for long sequence.

② vanishing / exploading gradient problem.

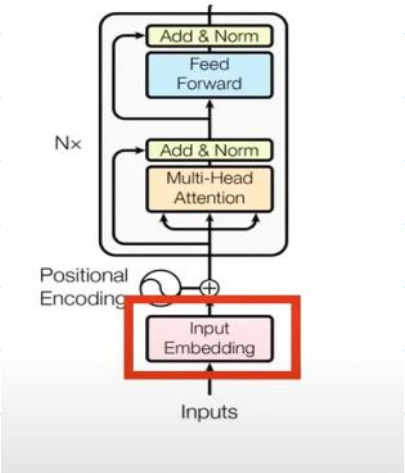
③ Difficulty in accessing information long time ago.

* To overcomes these problems we use transformer



① Encoder:

① Input Embedding:



conversion of the words (tokens) into dense vectors of fixed size that can be processed by the model.

In summary, input embeddings are a way to transform discrete tokens into a continuous space where the model can perform mathematical operations and learn patterns.

Monday, 5. August 2024 10:38

⇒ Breakdown of Embedding

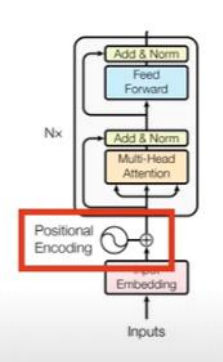
① **Tokenization**: Splitting of text into small subwords

e.g.:- "This is a cat" ⇒ ["This", "is", "a", "cat"]

② **Embedding layers**:- mapping of a vector into continuous vector space using Embedding layers.

e.g.:- "I" = [0.1, -0.5, 0.9, ...]

② **positional encoding**:



position Encodding are added to give model information about position of each token. This helps to capture sequential nature of data.

⇒ model should treat words which appear similar to each other same as compared to words which are different from each other
i.e in vector space similar words will be close to each other and different words will be far away from each other

05 August 2024 10:46

Encoding formula :

$$PE(pos, 2i) = \sin\left(\frac{Pos}{10000^{2i/d}}\right)$$

$$PE(pos, 2i+1) = \cos\left(\frac{Pos}{10000^{2i/d}}\right)$$

d = dimensionality of positional encoding
i = dimention index
Pos = position in sequence.

⇒ The Sine and cosine functions ensure that each position has a unique encoding and difference b/w the positions can be easily computed. This allows model to represent both short-long & long range positional information.

⇒ The positional encoding is then added to vector embeddings and the result provides us the both contextual and positional meaning of a word in a sentence.

★ Sum is elementwise.

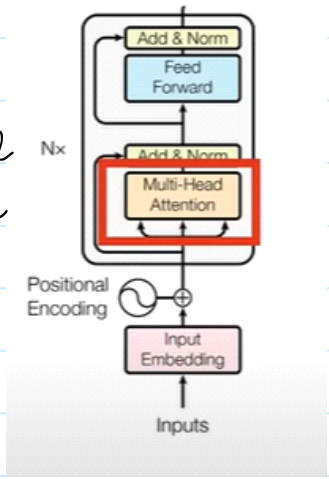
★ In summary :- positional encoding helps the transformer model leverage the order of tokens in sequences which is crucial for tasks involving language understanding and generation

+4

05 August 2024 10:46

③ Multi-head attention :-

To understand multi-headed attention first we will go through self-attention and then multi-headed attention.



① Self-attention

It helps the model to understand the relationships and dependencies between tokens regardless of their position in the sequence.

It is given by :-

$$\text{Attention}(Q, K, V) = \text{Softmax}\left[\frac{Q \cdot K^T}{\sqrt{d_k}}\right] V$$

* Q, K, V is the same matrix representing the input

Score matrix showing how intense is the relation

Self-Attention allows the model to relate words to each other.

In this simple case we consider the sequence length **seq** = 6 and $\mathbf{d_{model}} = \mathbf{d_k} = 512$.

The matrices **Q**, **K** and **V** are just the input sentence.

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

Softmax $\left(\frac{\mathbf{Q}\ (6, 512) \times \mathbf{K^T}\ (512, 6)}{\sqrt{512}}\right) =$

	YOUR	CAT	IS	A	LOVELY	CAT	Σ
YOUR	0.268	0.119	0.134	0.148	0.179	0.152	1
CAT	0.124	0.278	0.201	0.128	0.154	0.115	1
IS	0.147	0.132	0.262	0.097	0.218	0.145	1
A	0.210	0.128	0.206	0.212	0.119	0.125	1
LOVELY	0.146	0.158	0.152	0.143	0.227	0.174	1
CAT	0.195	0.114	0.203	0.103	0.157	0.229	1

(6, 6)

* all values are random.

* for simplicity I considered only one head, which makes $d_{model} = d_k$.

05 August 2024 10:46

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

	YOUR	CAT	IS	A	LOVELY	CAT
YOUR	0.268	0.119	0.134	0.148	0.179	0.152
CAT	0.124	0.278	0.201	0.128	0.154	0.115
IS	0.147	0.132	0.262	0.097	0.218	0.145
A	0.210	0.128	0.206	0.212	0.119	0.125
LOVELY	0.146	0.158	0.152	0.143	0.227	0.174
CAT	0.195	0.114	0.203	0.103	0.157	0.229

(6, 6)

X

V

(6, 512)

512

6 = Attention

(6, 512)

Each row in this matrix captures not only the meaning (given by the embedding) or the position in the sentence (represented by the positional encodings) but also each word's interaction with other words.

↓ Same shape as initial matrix

properties of self attention

① permutation invariant

② Requires no parameters

③ values along diagonal are expected to be maximum

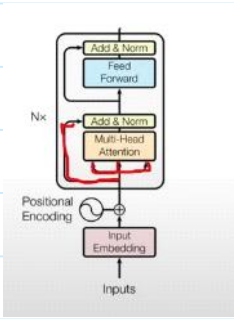
④ if we dont want some positions to interact we can set those values to $-\infty$ (masking) so that soft-max of those values is zero.

☆ In summary, the resulting matrix will have the same shape as input matrix but values in it will represent the dependencies and relationships within the sequences.

05 August 2024 10:46

Multi-head attention :->

The extention of self attention which allows model to focus on different parts of the sequence simultaniously it allows model to apply vairty of relationships and patterns by appling multiple self attention mechanisms in parallel.



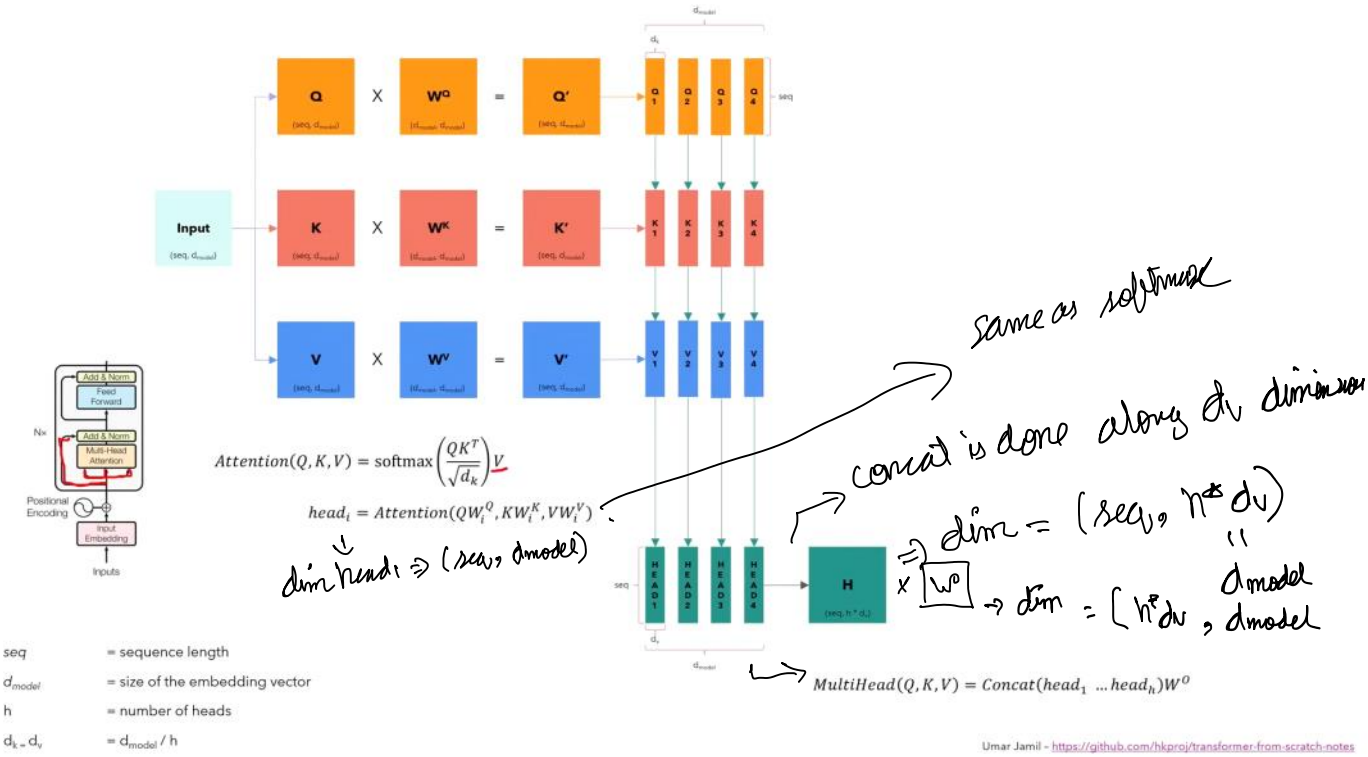
Working: Y same input values are produced 3 are sent to multi-head attention (Q, K, V) and one to add & norm layer.

The matricies of Q, K, V are multiplied by parameter matrices W_Q, W_K, W_V

* dimension of $Q, K, V = (seq, d_{model})$
" " $W_Q, W_K, W_V = (d_{model}, d_{model})$

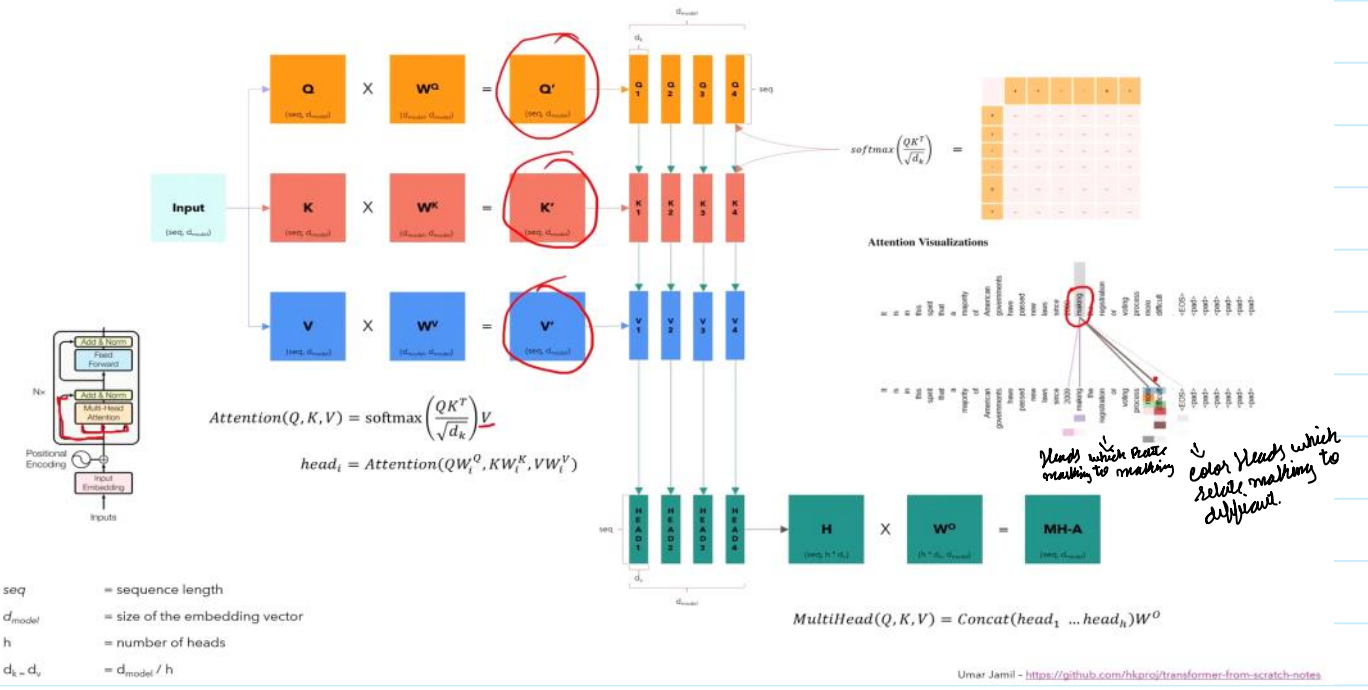
After multiplication we get Q', K', V', these are splitted into smaller parts d_k ($d_k = d_{model}/h$), after splitting the head; is calculated which is shown below

05 August 2024 10:46



After concatenation H is multiplied by W^O ($h*d_v, d_{model}$) and resulting matrix has same dimensions as initial input (seq, dmodel).

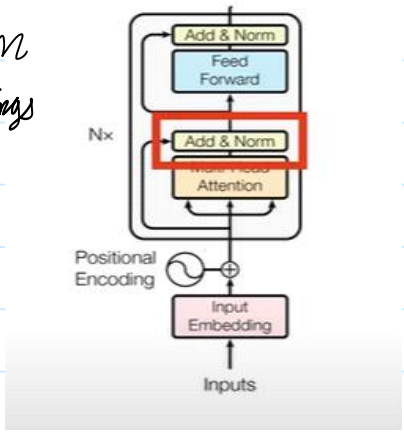
* Attention is calculated after splitting only as:-



05 August 2024 10:46

4) Add and norm:

The inputs from multi-head attention layers are combined with input embeddings and then normalized to facilitate gradient flow and convergence.



⇒ Structure :-

a) Residual connection (add)

The inputs to multi head attention are combined with outputs of multi-head attention, By this vanishing gradient problem can be overcomed.

b) Layer normalization (Norm):

layer normalization is done which helps to stabalize and speed up the process of training Ensuring that the mean will be zero and variance will be one.

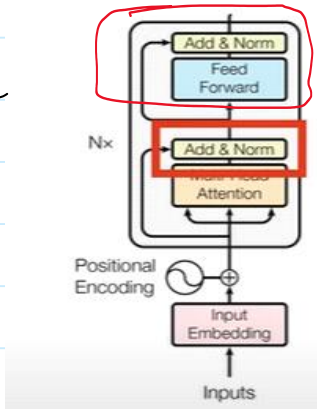
$$\text{i.e} \quad \text{layernorm}(x) = \frac{x - \text{mean}(x)}{\sqrt{\text{var}(x) + \epsilon}} \cdot \gamma + \beta$$

γ, β are learnable parameters.

In summary, the add and norm block helps the transformer train effectively.

05 August 2024 10:46

(5) feed forward layer :-



These are fully connected layers which are connected to the add & norm layer.
This layer captures the complex mechanism of transformer.

components :-

(1) first linear layer :- projects input to higher dimensions the dimension of this is larger than input & output typically.

(2) Activations :- non-linear activations introduce non-linearity in network.

(3) Second layer :- brings the intermediate representation back to original dimentionality.

⇒ In summary the feed forward network applies transformation to each position in the sequence independently. It consists of two fully connected layer and a non-linear function. which helps the model to learn non-linear relationships.

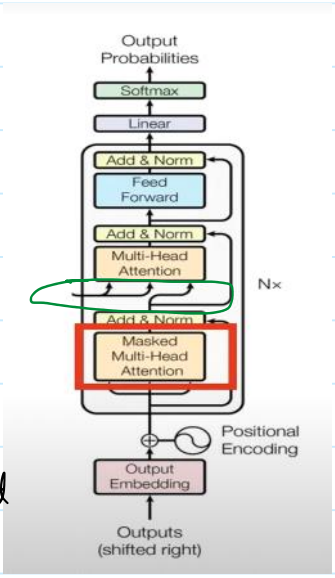
(6) add & norm :-

Similar to the one before (in (4)) it does the same steps and has same objective.

05 August 2024 10:46

Decoder :-

In decoder the ① output embeddings ② positional Encoding is exactly same as in encoder



☆ The green mark on image shows that the key & value comes from encoder and the Query comes from the masked multi head attention.

③ Masked - multi-head attention layer :-

By the help of this the attention mechanism is applied to only those words in the sequence which have been said the future words are set to $-\infty$ so softmax of them will be zero as shown in the figure.

Our goal is to make the model causal: it means the output at a certain position can only depend on the words on the previous positions. The model **must not** be able to see future words.

	YOUR	CAT	IS	A	LOVELY	CAT
YOUR	0.268	~~0.[illegible]~~	~~0.[illegible]~~	~~0.[illegible]~~	~~0.[illegible]~~	~~0.[illegible]~~
CAT	0.124	0.278	~~0.[illegible]~~	~~0.[illegible]~~	~~0.[illegible]~~	~~0.[illegible]~~
IS	0.147	0.132	0.262	~~0.[illegible]~~	~~0.[illegible]~~	~~0.[illegible]~~
A	0.210	0.128	0.206	0.212	~~0.[illegible]~~	~~0.[illegible]~~
LOVELY	0.146	0.158	0.152	0.143	0.227	~~0.[illegible]~~
CAT	0.195	0.114	0.203	0.103	0.157	0.229

→ Before applying softmax these values are set to $-\infty$.

11

05 August 2024 10:46

Structure :-

a) **Self-attention :-** Queries (Q), Keys (K), values (V) are used to compute attention scores and generate output vectors.

b) **masking :-** This is the key difference between encoder & decoder multi-head attention. It ensures that prediction for token at position i does not depend on the tokens which are present at position greater than i.

It is done before compiling the softmax function

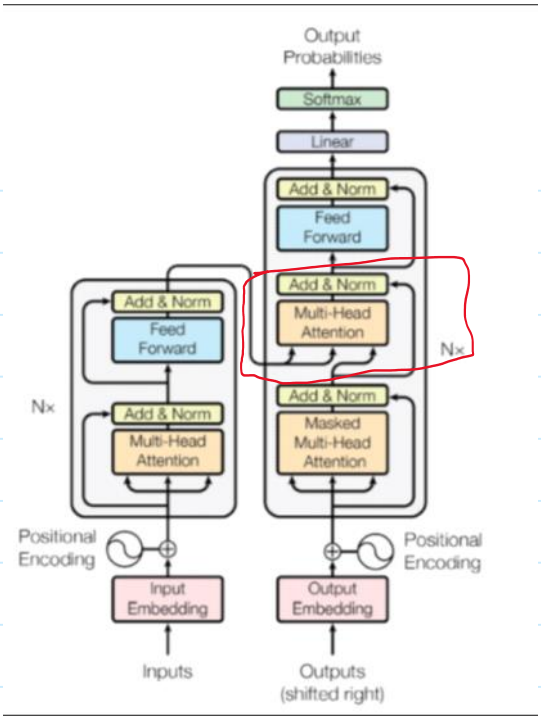
* e.g.: if we have a sequence of length n then we will get mask matrix of n×n and the value at positions where $j \leq i$ is set to $-\infty$ and softmax is calculated

c) **multi-headed attention :-**

Just like encoder the masked multi-head attention uses multiple heads to allow the model to capture different aspects of the dependencies b/w tokens.

In summary, the masked multi-head attention layer in the transformers decoder is designed to ensure that each token in the sequence generation process can only attend to previous tokens & not on future ones.

05 August 2024 10:46



(10) multi-head attention

In this case the multi head attention layer receives output of encoder and output of masked multi head attention. This attention is commonly known as "encoder-decoder attention" or "cross-attention".

⇒ This layer helps the decoder by makes use of the context provided by encoder. It enables the decoder to focus on relevant parts of the input sequence while generating each token in output sequence.

It allows decoder to attend to the encoders output using the decoders current state

structure :-

a) inputs :- Q ⇒ from masked multi head attention layer of decoder (represents current state)

K, V ⇒ from encoder. (represents contextual information)

b) attention mechanism :- multi-head mechanism is now applied similarly as it is applied in encoder.

05 August 2024 10:46

given by

$$\text{context}_i = \text{Softmax}\left[\frac{Q K^T}{\sqrt{d_k}}\right] \cdot V$$

⇒ In summary, the multi-head attention layer in decoder uses keys and values from encoder and queries from decoder (masked multihead attention). It allows decoder to incorporate and focus on relevant information from encoders output - facilitating better sequence generation by integrating input context with decodes current state.

(11) feed forward :-

It operates similar to the feed forward of the encoder it helps in capturing complex relationships and further refining each positions representation

* Same as in encoder (5)

⇒ Training of transformer

(1) Data preparation :-

a) tokenization :-> //

b) padding and truncation :-

↓ to ensure all that all sequences in batch have same length

↳ cut down sequences that exceed max length to fit computation constraints ↓ can lead to loss of information

05 August 2024 10:46

② Model initialization :- weights of model are initialized

③ forward pass :-

a) input embeddings → tokens to embedding layers

b) add positional encoding → add positional encoding

c) pass through encoder
→ self attention → compute attention score
→ feed forward → linear transformation

d) pass through decoder
→ masked self attention → self attention with mask to prevent leakage
→ encoder - decoder attention → //
→ FF layers → //

e) final linear & softmax layer → //

④ Loss computation :- loss is calculated (CE loss)

⑤ Backward pass :- Backward prop is used to compute gradients of the loss respect to each model parameter

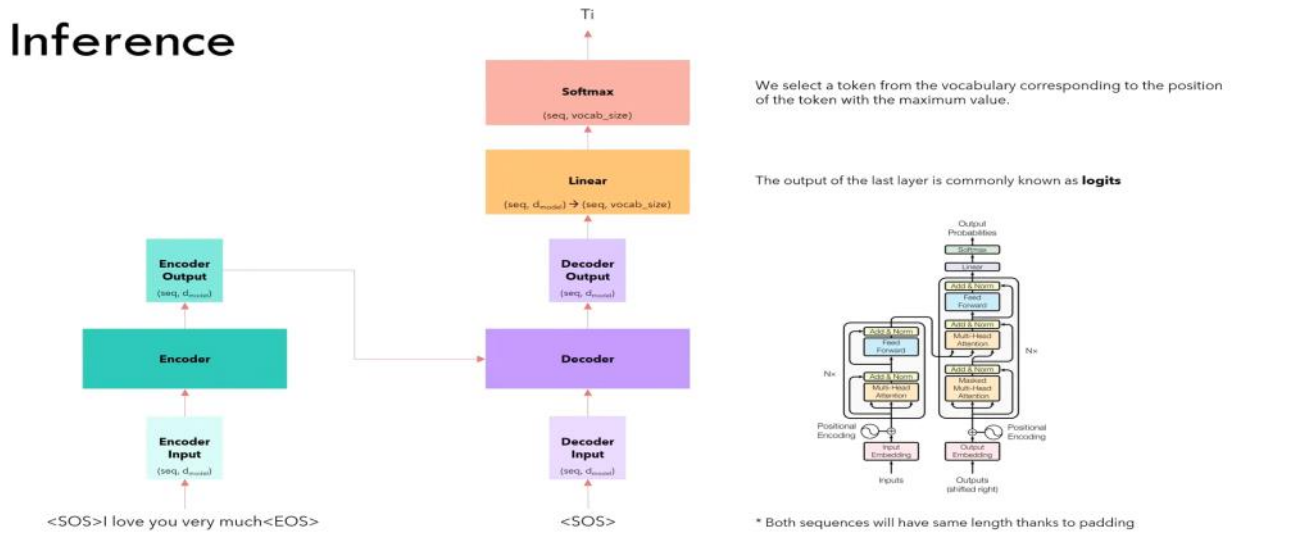
*** All of this happens in one time stamp as we give the matrix to the model so the output is also a matrix & loss and update is also calculated accordingly

15

05 August 2024 10:46

⇒ model inference :-

= lets say I want to translate "I love you very much" into italian.



lets belive we have created a model which translates the english into italian. and when we want to translate a sentence (I love you), we pass the sentance to the Encoder and the output of the Encoder becomes the key & values.

At the same time SOS (start of sentence) is passed to decoder which is our Query. The keys and value from encoder with Query get into multi head attention (* the decoder input is padded to match the dimensions). The multi head attention passes the values to feed forward and then to linear liner which passes it to the softmax. now the max value of softmax layer should be the translation of word "I" in italian

16

05 August 2024 10:46

Similarly in second time stamp the encoder part is already there so the output of the first time stamp is feeded as input to decoder and the same process takes place

⇒ same thing happens at further time stamps.

---

* Encoder and decoder just help us to convert the text value into numbers / vectors / embeddings the mapping of these numbers is done by linear and softmax layer at the end.

  mapping in this case can be translation of a word into italian. etc.

* while training the weights of multi-head attention, masked multi head attention layer are also calculated

3blue 1brown

05 August 2024 10:46

⇒ Addition to above notes

* When we do word embeddings we translate a word into a vector. The similar words will also be translated to vectors and the distance b/w those vectors in the vector space will be less.

* In that space difference between a words like women & man will almost be same as difference b/w words queen & king.

(???)

* Unembedding matrix is used to get the value/logit at the end of the transformer.

* Temperature value is a parameter which is used in the softmax function

high value gives kind of uniform distribution

lower value of temperature allows higher values in softmax to dominate the values

⇒ $e^{x_3/T}$

if T=0 the softmax always goes with the word which has high probability or in other words the probability of that word is equal to 1

⇒ if we choose high T value it has a risk that the meaning of the words selected will not make sense

Transformer explainer website

05 August 2024 10:46

⇒ Three key components of transformer:-

① Embedding

② Transformer block

- Attention mechanism
- MLP

③ output probabilities.

① Embeddings :-

Most important part of transformer is to convert words into the numbers which a model can understant. This is done by Embeddings. Embeddings is carried out in 4 steps

① Tokenization :-

The breakdown of sentence into small words is called tokenization. These tokens are defined before training the model (GPT-2 uses 50,257 unique tokens).

② Token Embeddings :->

These small tokens are represented into the vectors of e.g.:- dimentionality of 768. These dimensions depend

on the model to assign the semantic meaning to each token. These embedding matrix are stored into the shape of (50, 257, 768). containing 39 mil parameters.

③ positional Encoding:

The positional Encoding is added to token Embedding to keep the positional information of the tokens. different models use different positional Encoding.

② Transformer block:-

This is the core of the transformer block consisting of multi-head attention & MLP blocks stacked on top of each other to get the good visual representation of the data.

Multi-head self attention

It allows the model to capture and focus on relevents parts of the data. It consists of the following steps:-

① Query, key and value matrix

each token Embedding vector is transformed into three vectors Query (Q), key (K), value (V). The vectors are obtained by multiplying input Embeddings to learned weight matrices of Q, K, V.

06 August 2024 11:42

matrices for Q, K, and V. Here's a web search analogy to help us build some intuition behind these matrices:

- Query (Q) is the search text you type in the search engine bar. This is the token you want to *"find more information about"*.
- Key (K) is the title of each web page in the search result window. It represents the possible tokens the query can attend to.
- Value (V) is the actual content of web pages shown. Once we matched the appropriate search term (Query) with the relevant results (Key), we want to get the content (Value) of the most relevant pages.

② Masked self attention :⇒

It allows the model to generate data by allowing to focus on relevant parts of data and preventing access to future tokens.

Attention score :- It is the dot product of Query and Key matrix which gives a square matrix reflecting relationships between all input tokens.

Masking :- mask is applied to the upper triangle of attention matrix to prevent access to future tokens. by setting these value to $-\infty$ by which softmax is 0. By this model learns how to predict next token without peeking into future.

Softmax :- After masking this is applied such that each row sums upto one and indicates the relevent of other token to the left of it.

③ output :-

The masked self attention score are multiplied by the value matrix to get final output of self attention

mechanism.

MLP :-
These are used to enhance models representational capacity. These consists of two linear transformation with GELU activation.

④ output probabilities :-

The output of the transformer block is passed into final linear layer. This layer projects the final representation into 50,257 dimensional space. where every vocabulary has corresponding value called logit. The softmax is applied to convert logits into output probabilies which sums upto one.

⇒ temperature :- The logits of the model are divided by temperature

if temp > 1 : the probabilities are softer which ↑ randomness ↑ creativity of model

if temp < 1 : makes model confident and sharp leading to more predictable output.

if temp = 1 : has no effect on model.

06 August 2024 11:42

⇒ Advance architectural features of transformer

① layer normalization :=>

helps stabilize training process and improves the convergence of the model.

② Dropout :-

prevents overfitting

③ Residual connections :-

avoids vanishing gradient problem

⇒ just testing the paperlike on the tablet and I think it is very good?

Yo yo yohi Kashif Hof _x xx4=5 Are YOU?

1. Bridge course
2. MALE
3. MDNE
4. NLP (impt)
5. AML (" ")
6. DRL (attend classes)
7. interdisciplinary

/