

Generative Pretrained Transformer

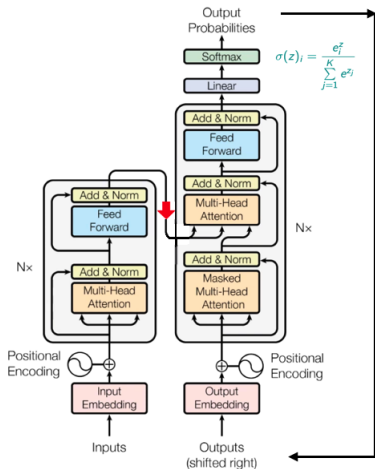
GPT & LLM

Jaideep Ganguly

August 31, 2024

Agenda

- 1 Encode, Positional encoding
- 2 Key, Query and Value \rightarrow Attention Filter
- 3 Multi Head Attention
- 4 Information preservation, normalisation
- 5 Decode, Masked Attention
- 6 Potential Innovations
- 7 Conclusion



Decoders are autoregressive models;
They are trained to predict the next token
after reading the preceding ones

Generalised Pretrained Transformer (GPT) - 2017

Attention Is All You Need

Ashish Vaswani*
Google Brain
avaswani@google.com

Noam Shazeer*
Google Brain
noam@google.com

Niki Parmar*
Google Research
nikip@google.com

Jakob Uszkoreit*
Google Research
usz@google.com

Llion Jones*
Google Research
llion@google.com

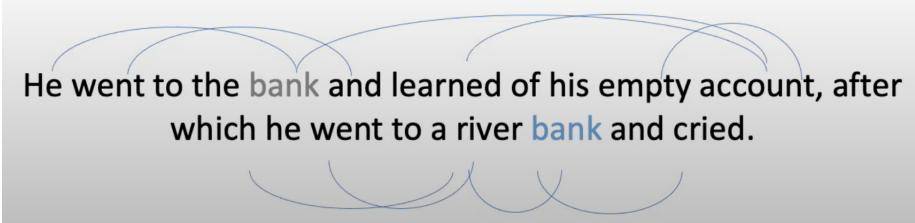
Aidan N. Gomez* †
University of Toronto
aidan@cs.toronto.edu

Lukasz Kaiser*
Google Brain
lukaszkaizer@google.com

Illia Polosukhin* ‡
illia.polosukhin@gmail.com

"Neural Machine Translation by Jointly Learning to Align and Translate" by Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio, 2014. Introduced concept of attention mechanism and laid the foundation for subsequent developments in NLP and DL, including the transformer architecture introduced in "Attention Is All You Need."

Attention



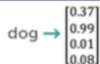
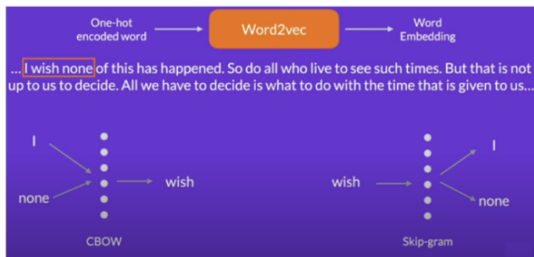
He went to the bank and learned of his empty account, after which he went to a river bank and cried.

- 1 Attention mechanism has an infinite reference window
- 2 In contrast, Recurring Neural Network (RNN) has a short reference window, Long Short Term Memory (LSTM) has a longer window. RNN does not work, LSTM has limited capability.

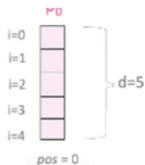
One Hot Encoding, Word Embedding, Positional Encoding

Linear & logistic regression models can deal with numerical data only, not text. In one-hot encoding, the string encoded variable is replaced with new variables of boolean type, e.g., a feature such as color can be encoded as:

red	green	blue
1	0	0
0	1	0
0	0	1

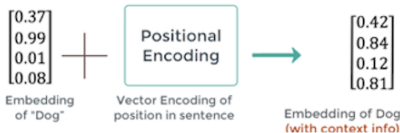


AJ's **dog** is a cutie → Position 2
 AJ looks like a **dog** → Position 5

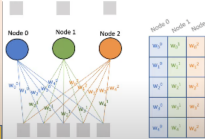
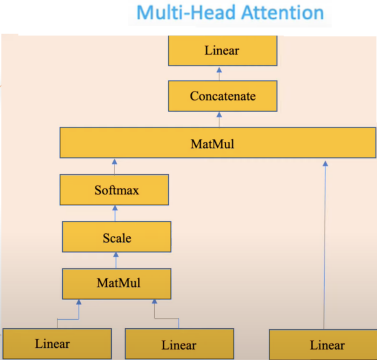
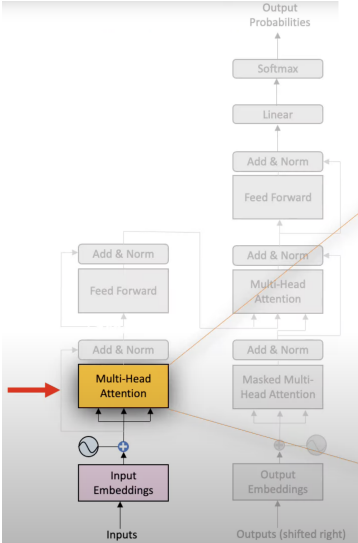


$$PE_{(pos, 2d)} = \sin\left(\frac{pos}{10000^{2d}}\right)$$

odd pos sin, even pos cos



Multi Head Attention



Linear Layer - No Activation Function

Attention: Query, Key, Value

Concept of:

- 1 Query (Q)
- 2 Key (K)
- 3 Value (V)

All constructed from the embedding.

I came from your other question **Self-attention original work?** The key/value/query formulation of attention is from the paper **Attention Is All You Need**.

31

How should one understand the queries, keys, and values

✓

+50

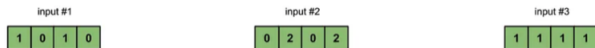
The key/value/query concepts come from retrieval systems. For example, when you type a query to search for some video on Youtube, the search engine will map your **query** against a set of **keys** (video title, description etc.) associated with candidate videos in the database, then present you the best matched videos (**values**).

Mimic the retrieval of a value v_i for a query q based on a key k_i in DB.

$$attention(q, k, v) = \sum_i similarity(q, k_i) \times v_i \text{ (weighted)}$$

Attention: Q, K, V and weights Q_w , K_w , V_w

Weights are initialised randomly using a random distribution. Example:



Because every input has a dimension of 4, each set of the weights must have a shape of 4×3 .
Weights are initialised randomly, it is done once before training.

Weights for key Weights for query Weights for value

$\begin{bmatrix} [0, 0, 1], \\ [1, 1, 0], \\ [0, 1, 0], \\ [1, 1, 0] \end{bmatrix}$ $\begin{bmatrix} [1, 0, 1], \\ [1, 0, 0], \\ [0, 0, 1], \\ [0, 1, 1] \end{bmatrix}$ $\begin{bmatrix} [0, 2, 0], \\ [0, 3, 0], \\ [1, 0, 3], \\ [1, 1, 0] \end{bmatrix}$

Key representation
for input 1:

$$\begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} \times \begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \end{bmatrix}$$

Key representation
for input 2:

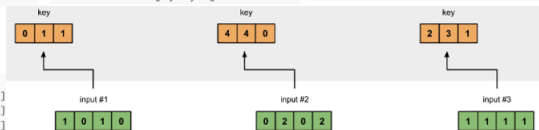
$$\begin{bmatrix} 0 \\ 2 \\ 0 \\ 2 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 4 & 4 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \end{bmatrix}$$

Key representation
for input 3:

$$\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \times \begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \end{bmatrix}$$

Key representation:
(Vectorise)

$$\begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} \times \begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \end{bmatrix}$$



Attention: Q, K, V and weights Q_w , K_w , V_w

Query representation:

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 2 & 0 \\ 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 2 & 2 \\ 2 & 1 & 3 \end{bmatrix}$$



Value representation:

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 2 & 0 \\ 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} 0 & 2 & 0 \\ 0 & 3 & 0 \\ 1 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 8 & 0 \\ 2 & 6 & 3 \end{bmatrix}$$

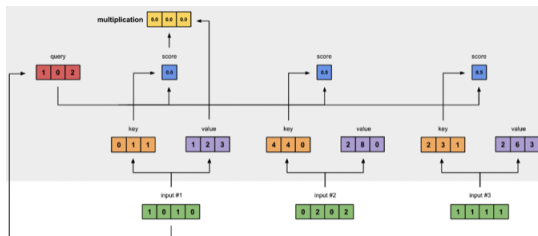


Attention: Q, K, V and weights Q_w , K_w , V_w

Attention scores - dot product between Input 1's query (red) with all keys (orange), including itself

```
[0, 4, 2]
[1, 0, 2] x [1, 4, 3] = [2, 4, 4]
[1, 0, 1]
```

$\text{softmax}([2, 4, 4]) = [0.0, 0.5, 0.5]$



Multiply Softmax attention scores for each input (blue) by its corresponding value (purple).

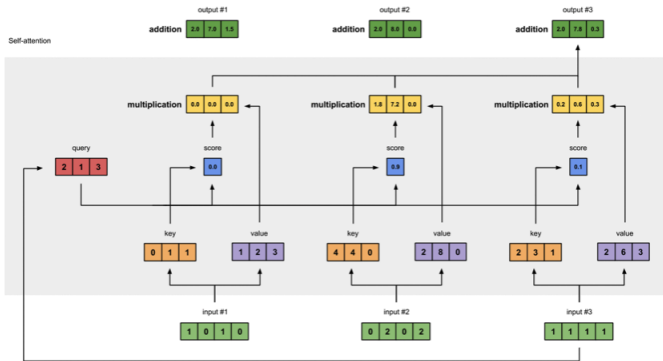
```
1: 0.0 * [1, 2, 3] = [0.0, 0.0, 0.0]
2: 0.5 * [2, 8, 0] = [1.0, 4.0, 0.0]
3: 0.5 * [2, 6, 3] = [1.0, 3.0, 1.5]
```

Attention: Q, K, V and weights Q_w , K_w , V_w

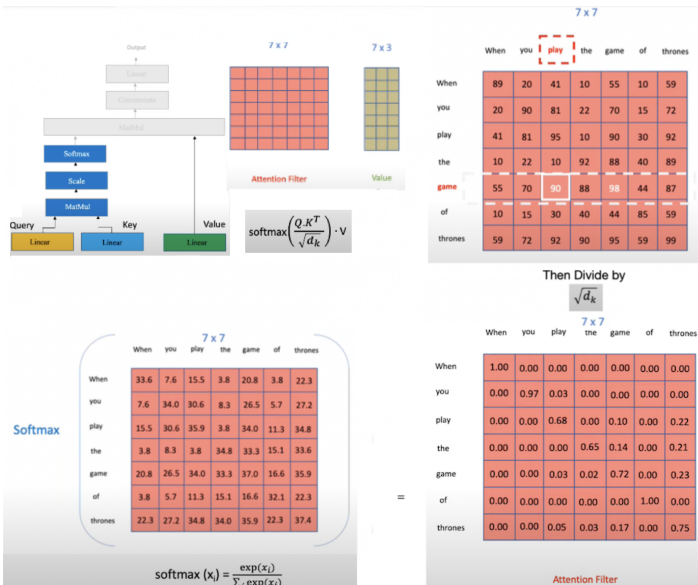
Sum weighted values to get Output 1

$$\begin{aligned} & [0.0, 0.0, 0.0] \\ + & [1.0, 4.0, 0.0] \\ + & [1.0, 3.0, 1.5] \\ \hline = & [2.0, 7.0, 1.5] \end{aligned}$$

Repeat for
Input 2 &
Input 3

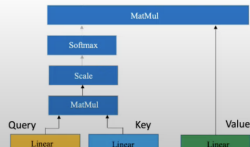
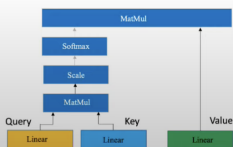
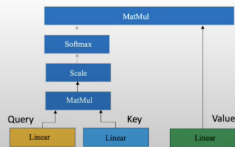
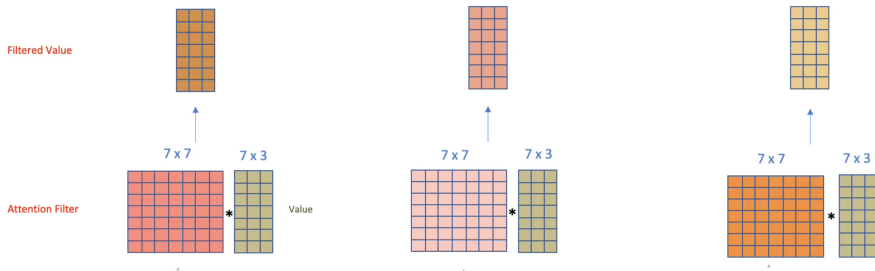


Attention Filter - filter out unnecessary information (noise)

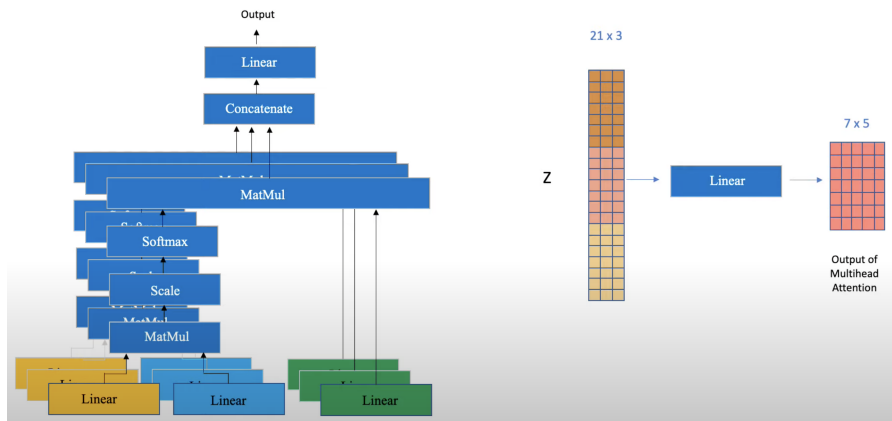


Multi Head Attention

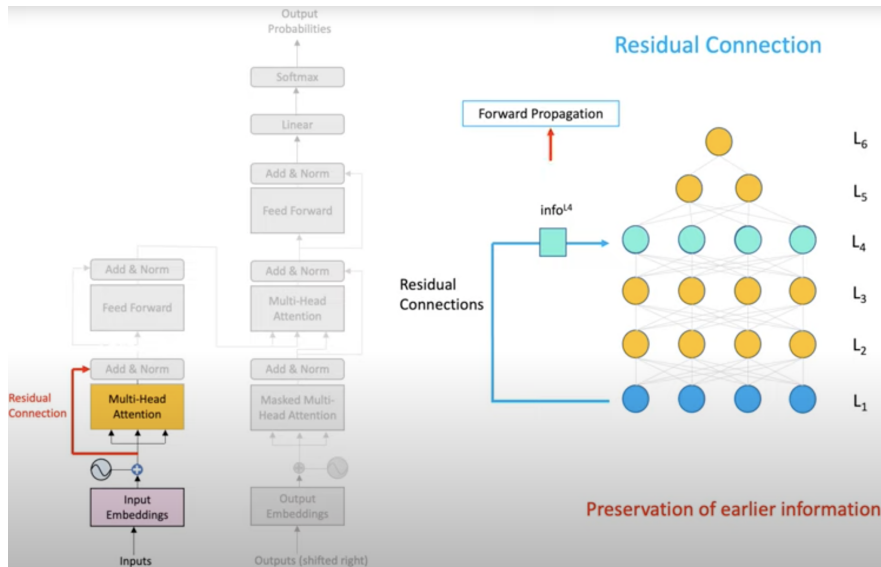
Multi-Head Attention



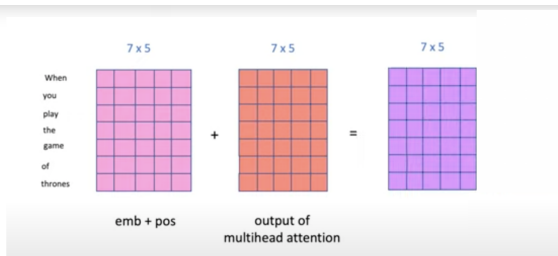
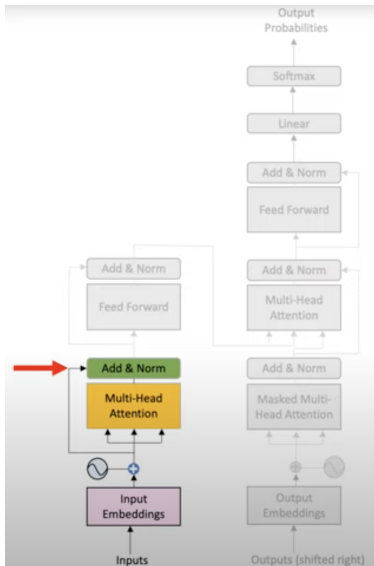
Multi Head Attention - Concatenation



Information Preservation



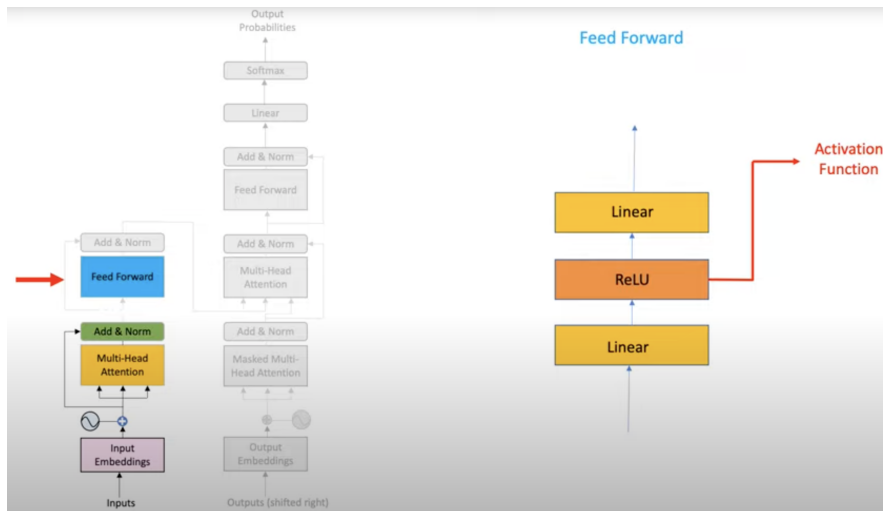
Add (Preserve Information) & Normalisation



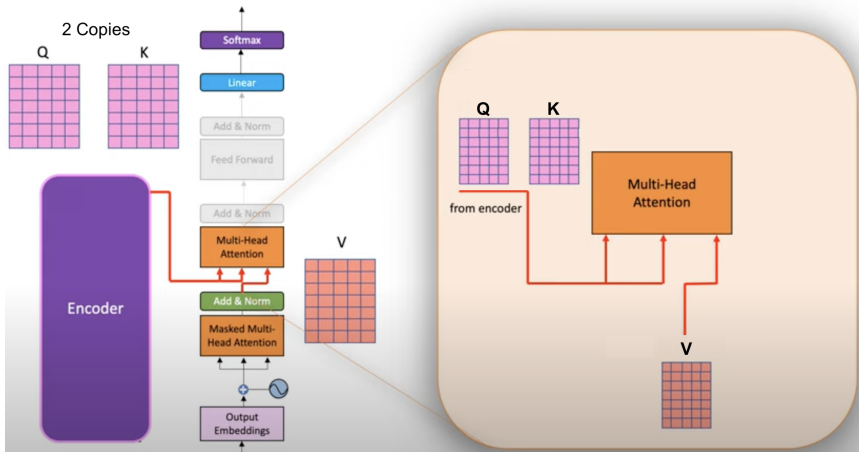
	7 x 3					mean (μ)	std (σ)
x_0 = When	0.98	1.28	0.41	0.27	0.41	0.67	0.44
x_1 = you	0.52	0.01	2.06	0.27	0.33	0.64	0.82
x_2 = play	2.22	0.27	0.10	0.41	2.06	1.01	1.04
x_3 = the	0.99	1.00	0.11	0.27	0.33	0.54	0.42
x_4 = game	0.52	0.01	0.33	2.06	0.52	0.69	0.79
x_5 = of	0.10	2.06	0.73	0.27	0.41	0.71	0.79
x_6 = thrones	0.33	0.01	0.13	0.27	1.28	0.40	0.51
	f_0	f_1	f_2	f_3	f_4		

$$x_0 = \frac{0.98 - 0.67}{\sqrt{0.44^2 + 0.0001}}$$

Feed Forward



Decoder Layer



Masking

<start> | I am no man <end>

<start> | am no man <end>

<start>	33.6	7.6	15.5	3.8	20.8	22.3
I	7.6	34.0	30.6	8.3	26.5	27.2
am	15.5	30.6	35.9	3.8	34.0	34.8
no	3.8	8.3	3.8	34.8	33.3	33.6
man	20.8	26.5	34.0	33.3	37.0	35.9
<end>	3.8	5.7	11.3	15.1	16.6	37.4

Attention Filter

<start> | I am no man <end>

<start>	0	-inf	-inf	-inf	-inf	-inf
I	0	0	-inf	-inf	-inf	-inf
am	0	0	0	-inf	-inf	-inf
no	0	0	0	0	-inf	-inf
man	0	0	0	0	0	-inf
<end>	0	0	0	0	0	0

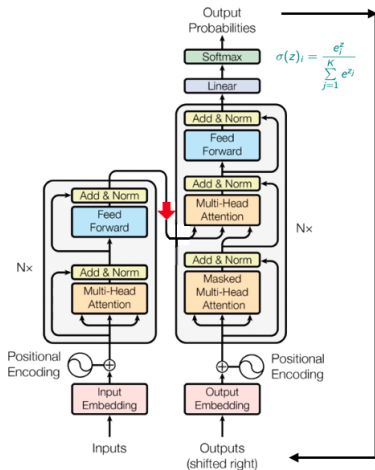
Mask Filter

<start> | I am no man <end>

<start>	1	0	0	0	0	0
I	0.01	0.99	0	0	0	0
am	0.001	0.004	0.995	0	0	0
no	0.003	0.004	0.003	0.99	0	0
man	0.003	0.003	0.04	0.02	0.93	0
<end>	0.001	0.001	0.001	0.001	0.001	0.995

Masked-Attention Filter

Decoder Output



Decoders are autoregressive models;
They are trained to predict the next token
after reading the preceding ones

Potential Innovations

- ① Softmax calculation for output probabilities are independent and is simplistic. In many cases, joint probabilities for bi-gram and tri-gram occurrences are more appropriate.
- ② The next token generated in the decode can be a "composite" token in response to the computed attention from the input.
- ③ Reduce "Cost to Serve (CTS)" through reduced parameters and better efficiency through quantisation.

Conclusion

- 1 GPT does not replicate human writing or speaking processes. Although they imitate human writing, any apparent cleverness primarily arises from our inclination to attribute human characteristics to non-human entities (anthropomorphization).
- 2 LLMs are essentially establishing statistical connections among vectors representing words and more extended grammatical structures. Each word within a sentence is linked to the subsequent word in the sequence with an associated probability.
- 3 This diverges significantly from human cognitive processes, where we employ word meanings to construct intricate and precise structures of "meanings" and definitions. For LLMs, definitions are confined to statistical interrelations among intricate vectors encompassing words, sentences, and more extensive grammatical constructs. LLMs cannot innovate, not just yet.
- 4 That said, LLMs are impressive and have massive use cases.

Questions?

Thank You!