LLM Edge

after reviewing the fundamentals this note dives deep into some of the more specific knowledge gaps regarding LLMs

Think before you speak: Training Language Models With Pause Tokens

Sachin Goyal[*†]
Machine Learning Department
Carnegie Mellon University
sachingo@andrew.cmu.edu

Ziwei Ji
Google Research, NY
ziweiji@google.com

Ankit Singh Rawat
Google Research, NY
ankitsrawat@google.com

Aditya Krishna Menon
Google Research, NY
adityakmenon@google.com

Sanjiv Kumar
Google Research, NY
sanjivk@google.com

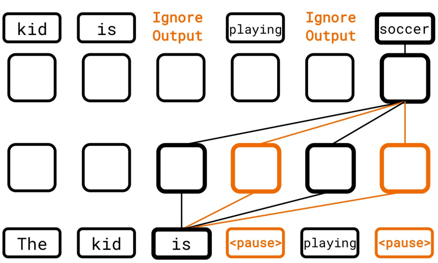
Vaishnavh Nagarajan[†]
Google Research, NY
vaishnavh@google.com

as for each $(K+1)^{th}$ token the model uses K previous vectors, paper argues for some challenging tokens, K+M vectors might be needed.

even though this works, it's not known why :))

by appending dummy tokens to the input sequence, the generation of tokens is Delayed.

during pre-training <pause> inserted at random and the associated output is ignored for calculating the loss function.



(b) Pause-pretraining

the pause tokens don't give the model more "time" to think, but rather more computation to work with. pause tokens manipulate the model to do more computation than it normally does.

Note, that more tokens don't cause more activations or using more "nodes", The computational flow in the paper means that more tokens provide more context or hidden vectors (token embeddings)

Meet in the Middle: A New Pre-training Paradigm

Anh Nguyen* Nikos Karampatziakis* Weizhu Chen

Microsoft Azure AI

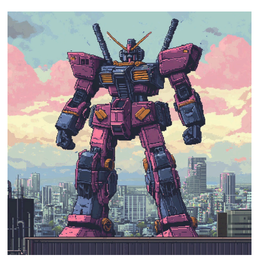
regarding the unidirectional limitation of autoregressive models (referred on my notes about hallucination), this paper proposes a forward and a backward models that agree on a token in the middle. This is great for infill tasks where you don't generate text from scratch.

bidirectional models while better for infilling tasks, are less capable of in-context learning.

training loss:

$$\sum_{x\in S}\sum_{i=1}^{|x|} -\log\left(\overrightarrow{p}(x_i|x_{<i})\right) - \log\left(\overleftarrow{p}(x_i|x_{>i})\right) + \overbrace{\beta D^{TV}_{i,x}\left(\overrightarrow{p}\,\|\,\overleftarrow{p}\right)}^{\text{regularizer to agree on same tokens}}.$$

during inference, both the forward and backward models generate sequences until they meet in the middle. In the best case they produce the same tokens. To avoid false positive (both agreeing on "the") n-gram is used and not just a single token.



Infinite Context Window

how does Gemini manage a one million CW?

Leave No Context Behind:
Efficient Infinite Context Transformers with Infini-attention

Tsendsuren Munkhdalai, Manaal Faruqui and Siddharth Gopal
Google
tsendsuren@google.com

introduces memory of past context similar to how RNNs work.

1. We introduce a practical and yet powerful attention mechanism – Infini-attention with long-term compressive memory and local causal attention for efficiently modeling both long and short-range contextual dependencies.
2. Infini-attention introduces minimal change to the standard scaled dot-product attention and supports plug-and-play continual pre-training and long-context adaptation by design.
3. Our approach enables Transformer LLMs to scale to infinitely long context with a bounded memory and compute resource by processing extremely long inputs in a streaming fashion.

an overview of how self-attention works

$$K = XW_K,\ V = XW_V \text{ and } Q = XW_Q.$$

$W_V \in \mathbb{R}^{d_{model} \times d_{value}}$ and $W_Q \in \mathbb{R}^{d_{model} \times d_k}$

ion context is calculated as a weighted

$$A_{dot} = \text{softmax}\left(\frac{QK^T}{\sqrt{d_{model}}}\right)V.$$

$$A_{mem} = \frac{\sigma(Q)M_{s-1}}{\sigma(Q)z_{s-1}}.$$

a piece of memory (A_{mem}) is retrieved from the memory matrix M. σ is a non-lin activation

once a memory is retrieved, the new memory is updated with new K and V. if the new info already exists, no update is required

$$M_s \leftarrow M_{s-1} + \sigma(K)^T\left(V - \frac{\sigma(K)M_{s-1}}{\sigma(K)z_{s-1}}\right).$$

the retrieved memory and the new A_{dot} are aggregated through a learnable β weight.

$$A = sigmoid(\beta) \odot A_{mem} + (1 - sigmoid(\beta)) \odot A_{dot}.$$

Why store K&V in the memory?

The K are important to finding the relevant V to a Q. Without a K, the model would not be able to retrieve.

there is nothing directly learnable about the memory matrix.

Recap

the previous K and Vs are stored in a M matrix like **[K|V]**

now each Q, is compared to the Ks by dot product (or something) and through the resulting attention weights, related Vs are returned. This is actually a weighted sum of previous Vs.

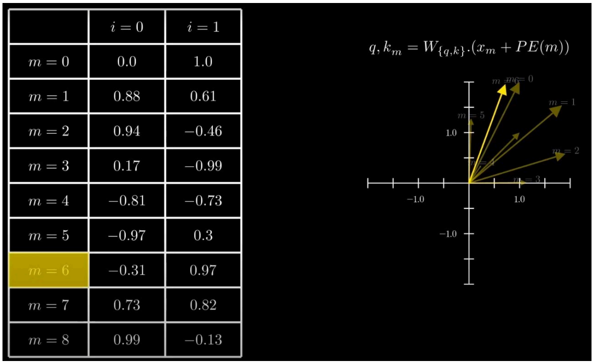
This Could be key to Gemini's 1M Context Window!

so the CW isn't 1M in reality, but through a memory function, it effectively stores the previous 1M tokens.

RoPE

Rotary Positional Encoding

So what is wrong with the original sinusoidal PE?
It's Choatic! The model has to memorize all the PEs rather than figure out a pattern.



	$i=0$	$i=1$
$m=0$	0.0	1.0
$m=1$	0.88	0.61
$m=2$	0.94	−0.46
$m=3$	0.17	−0.99
$m=4$	−0.81	−0.73
$m=5$	−0.97	0.3
$m=6$	−0.31	0.97
$m=7$	0.73	0.82
$m=8$	0.99	−0.13

ROFORMER: ENHANCED TRANSFORMER WITH ROTARY POSITION EMBEDDING

Jianlin Su
Zhuiyi Technology Co., Ltd.
Shenzhen
bojonesu@wezhuiyi.com

Yu Lu
Zhuiyi Technology Co., Ltd.
Shenzhen
julianlu@wezhuiyi.com

Shengfeng Pan
Zhuiyi Technology Co., Ltd.
Shenzhen
nickpan@wezhuiyi.com

Ahmed Murtadha
Zhuiyi Technology Co., Ltd.
Shenzhen
mengjiayi@wezhuiyi.com

Bo Wen
Zhuiyi Technology Co., Ltd.
Shenzhen
brucewen@wezhuiyi.com

Yunfeng Liu
Zhuiyi Technology Co., Ltd.
Shenzhen
glenliu@wezhuiyi.com

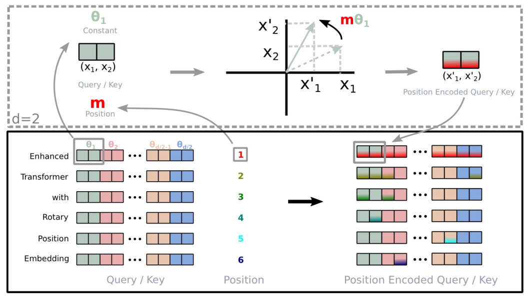


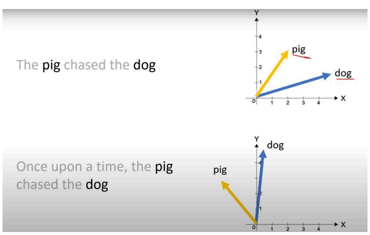
Figure 1: Implementation of Rotary Position Embedding(RoPE).

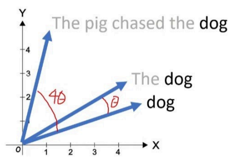
Sinusoidal is an absolute PE, meaning that position 1 vs. 87 have different PE relations than 2 vs. 88.

With relative Positional embeddings the PE between two tokens of relatively same positions is the same.

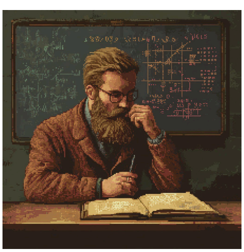
The RoPE method uses rotations to encode the position of each token.

Rotary Positional Embeddings





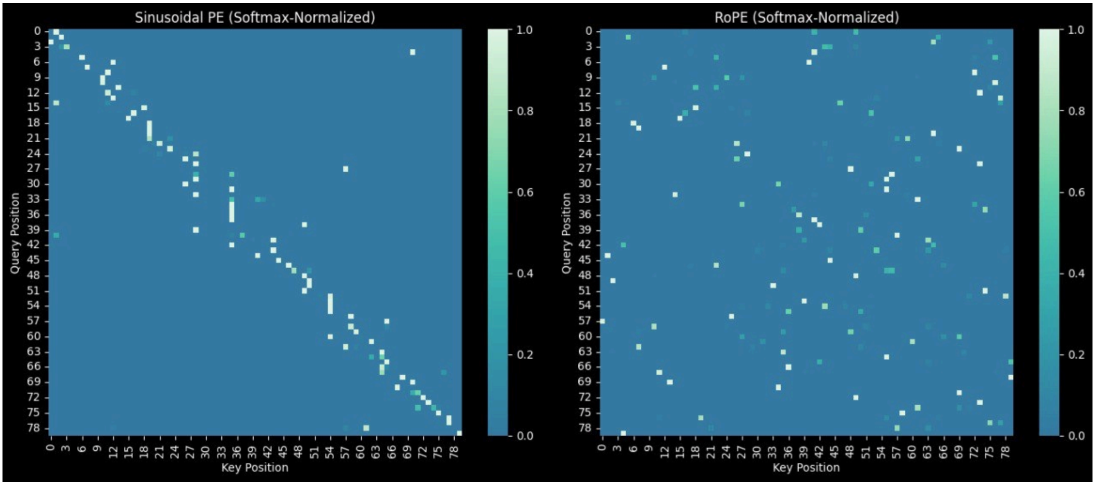
Also, the relativity between tokens of same distance is preserved.



the formula breaks the dims to chunks of two dimensions

$$\begin{pmatrix} \cos m\theta_1 & -\sin m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ \sin m\theta_1 & \cos m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cos m\theta_2 & -\sin m\theta_2 & \cdots & 0 & 0 \\ 0 & 0 & \sin m\theta_2 & \cos m\theta_2 & \cdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \ddots & \cos m\theta_{d/2} & -\sin m\theta_{d/2} \\ \vdots & \vdots & \vdots & \vdots & & \sin m\theta_{d/2} & \cos m\theta_{d/2} \end{pmatrix}$$

Sparse matrix and Computationally efficient let's see the practical difference of RoPE VS Sinusoidal. The code is in my repo of "ml_retreat".



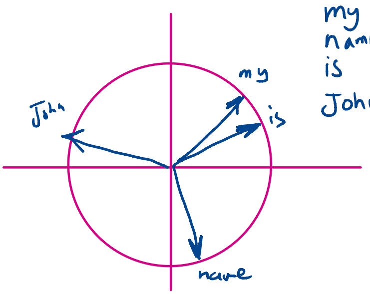
now we can see clearly that sinusoidal puts more emphasis on nearby tokens while RoPE captures long-ranged dependencies. **why?** The Sinusoidal, being an absolute PE method assigns a PE score to tokens based on their positions. It puts more emphasis on closer tokens because it understands local context where the absolute positional difference is large enough. RoPE has the main idea to capture how far tokens are from each other rather than their specific absolute positions.

There's a big difference in how they're applied

Sins: Token Emb + Pos Emb

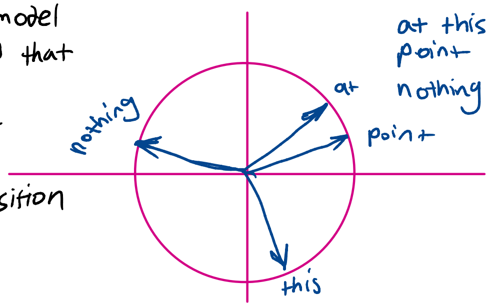
RoPE: Rotated Q. Rotated K

1. RoPE doesn't directly modify token embs
2. because of that, the V is not modified
3. RoPE is based on a Rotation Transformation on the Q, K. So regardless of position, the positional Encoding does not decay. The transformation also does not change size of the token embeddings. Unlike Sinusoidal, RoPE doesn't mess with the embeddings.



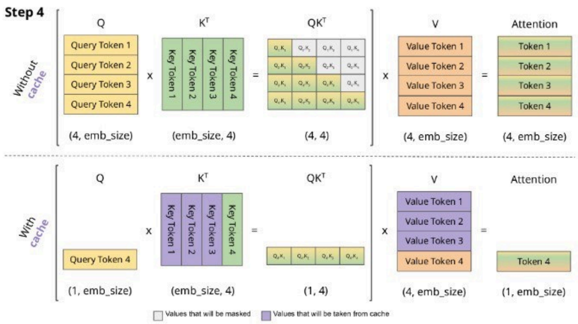
my name is John

the model learns that each degree relates to a position



at this point nothing

KV Cache



Comparison of scaled dot-product attention with and without KV caching. emb_size means embedding size.
Image created by the author.

during the process of calculating attention scores, many operations are repeated. KV cache stores some intermediate values and eliminate the need to perform every calculations again.

Optimizations

$K = W_K \cdot X$ is optimized as the K of previous tokens is stored and concatenated to new K.

$$K_{new} = W_K \cdot X_{new}$$

$$K = \text{concat}(K_{cach}, K_{new})$$

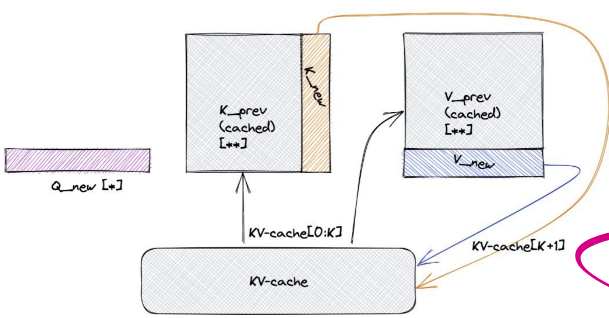
$V = W_V \cdot X$ is optimized the same way

$$V_{new} = W_V \cdot X_{new}$$

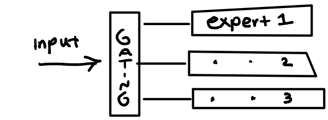
V is V_{cach} and V_{new} concated

Scaled dot Product is performed for the new token only.

$$\text{Attention Score} = Q_{new} \times K^T_{cach+new}$$



Mixture of Experts (MoE)



we have multiple smaller models each expert in a set of tasks

Also the Gating function choosing the expert for each input

* it's not an entirely novel idea

Mixtral of Experts

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, William El Sayed

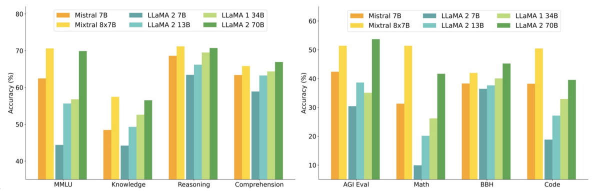
$$\sum_{i=0}^{n-1} G(x)_i \cdot E_i(x).$$

the output is a weighted sum of all expert outputs. if $G(x)$ is sparse, only one or few experts are needed for computation.

$G(x)$ takes input, applies a simple classification via a FF layer, select the TopK logits and applies Softmax to normalize.

$$G(x) := \text{Softmax}(\text{TopK}(x \cdot W_g)),$$

normal feedforward layer



it can outperform vanilla models of bigger size.

the assigment to experts is per token not input meaning that a sequence could be handed over multiple experts.

Also, the authors didn't notice any patterns of expert assignments on a topic.

This wraps LLM Edge