Spring25 CS598YP

23.2: CacheBlend

Yongjoo Park

University of Illinois Urbana-Champaign

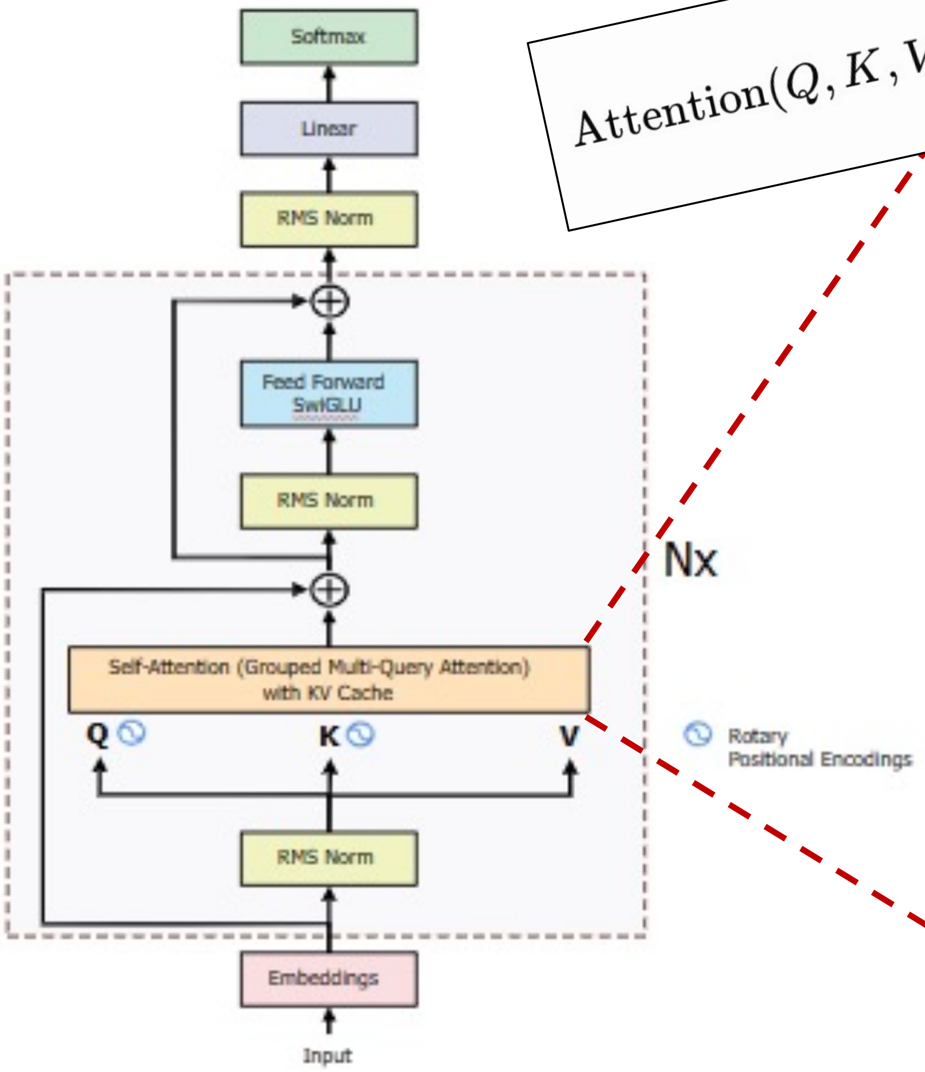
Outline

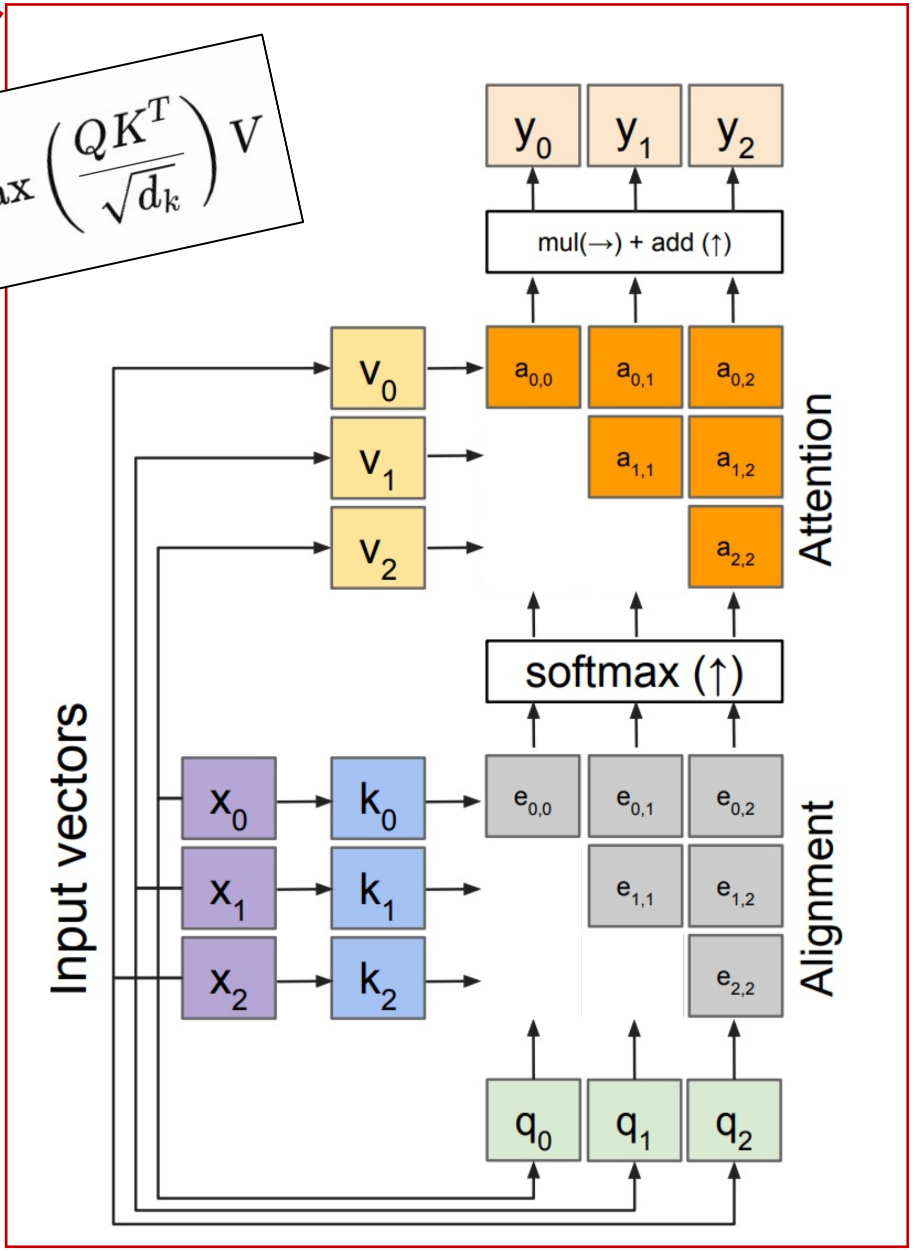
- *Recap: **KV Cache***
- *Rotary Position Embedding*
- *CacheBlend*

Attention and KV Cache

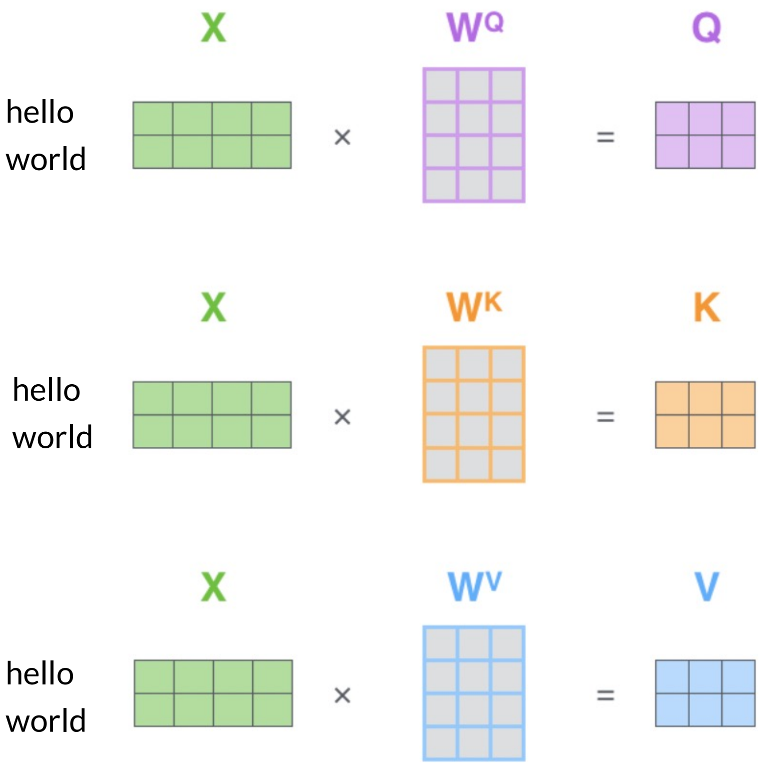
Attention zoom-ed in

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

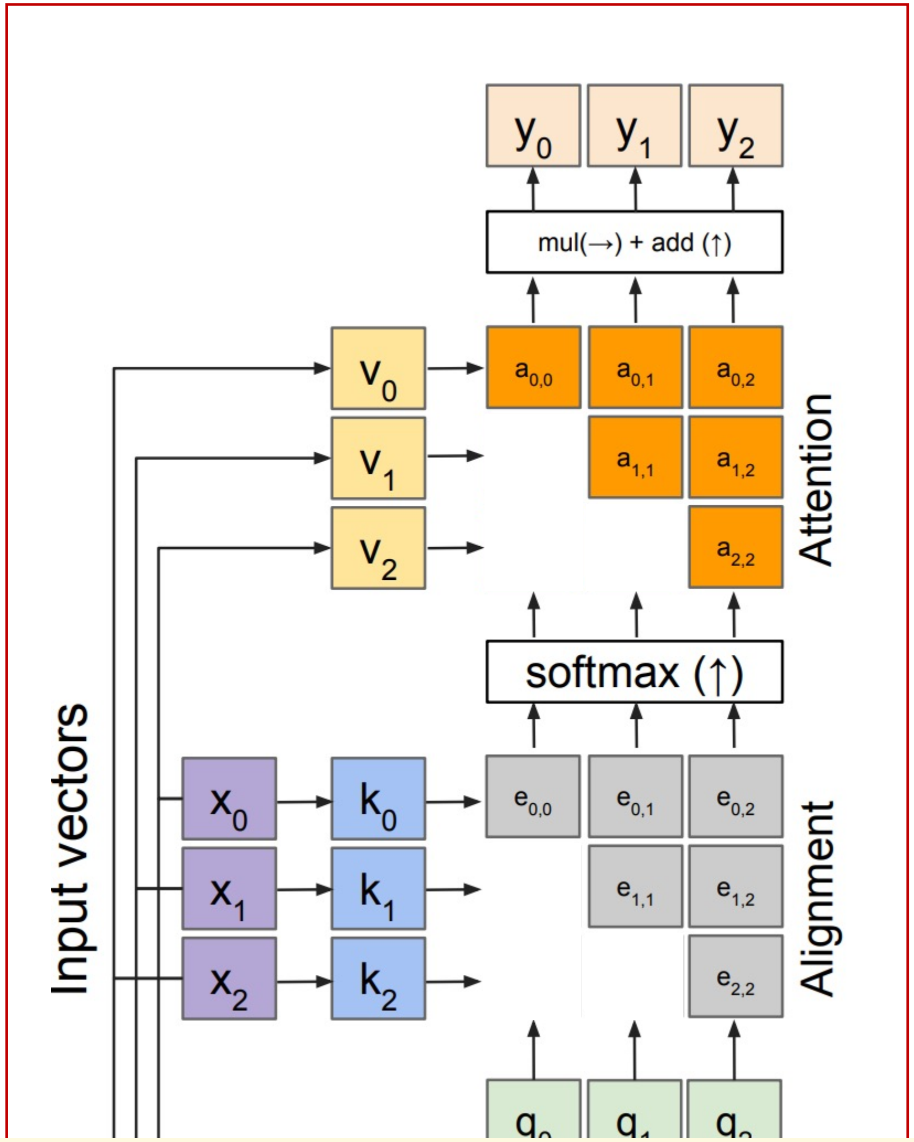




Getting K, Q, V is expensive



Dimension: 4,096 for Llama3-8B



We can re-use K and V for previous tokens -> ***KV Cache***

RoPE: Rotary Position Embedding

[Submitted on 20 Apr 2021 (v1), last revised 8 Nov 2023 (this version, v5)]

RoFormer: Enhanced Transformer with Rotary Position Embedding

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, Yunfeng Liu

Position encoding recently has shown effective in the transformer architecture. It enables valuable supervision for dependency modeling between elements at different positions of the sequence. In this paper, we first investigate various methods to integrate positional information into the learning process of transformer-based language models. Then, we propose a novel method named Rotary Position Embedding(RoPE) to effectively leverage the positional information. Specifically, the proposed RoPE encodes the absolute position with a rotation matrix and meanwhile incorporates the explicit relative position dependency in self-attention formulation. Notably, RoPE enables valuable properties, including the flexibility of sequence length, decaying inter-token dependency with increasing relative distances, and the capability of equipping the linear self-attention with relative position encoding. Finally, we evaluate the enhanced transformer with rotary position embedding, also called RoFormer, on various long text classification benchmark datasets. Our experiments show that it consistently overcomes its alternatives. Furthermore, we provide a theoretical analysis to explain some experimental results. RoFormer is already integrated into Huggingface: \url{this https URL}.

Comments: fixed some typos

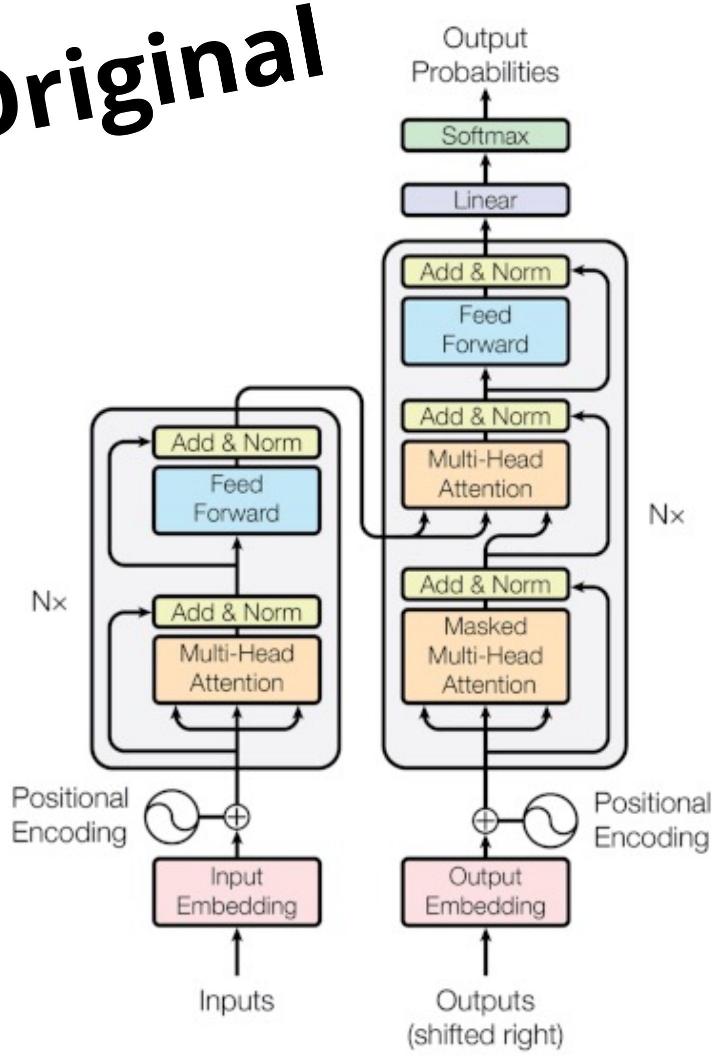
Subjects: **Computation and Language (cs.CL)**; Artificial Intelligence (cs.AI); Machine Learning (cs.LG)

Cite as: arXiv:2104.09864 **[cs.CL]**

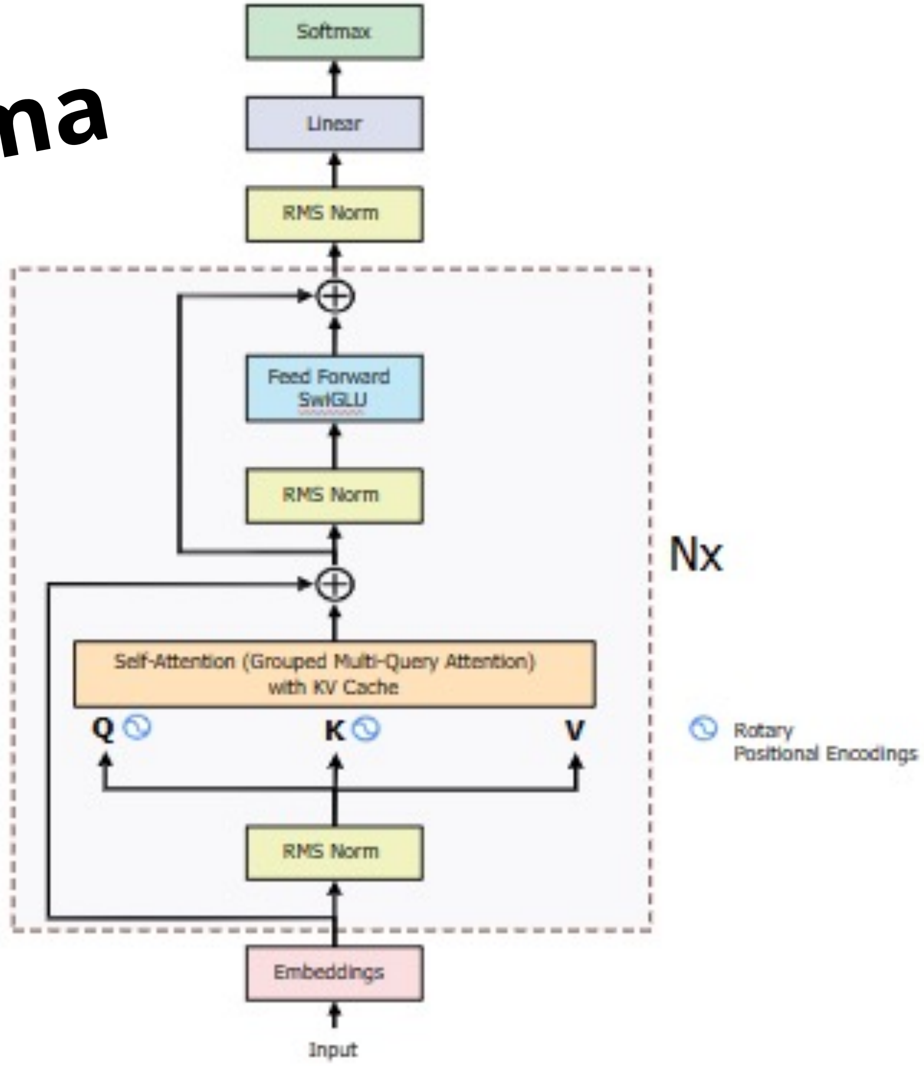
(or arXiv:2104.09864v5 **[cs.CL]** for this version)

Today: Positional encoding ***inside*** transformer block

Original

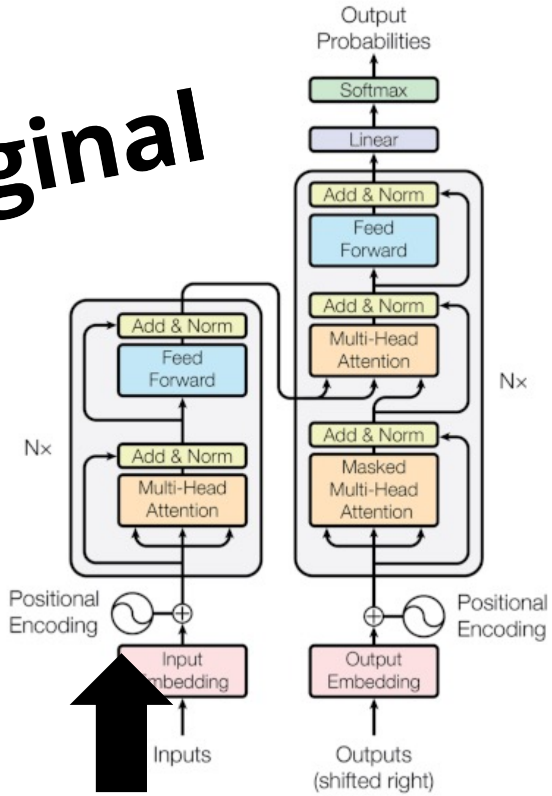


Llama

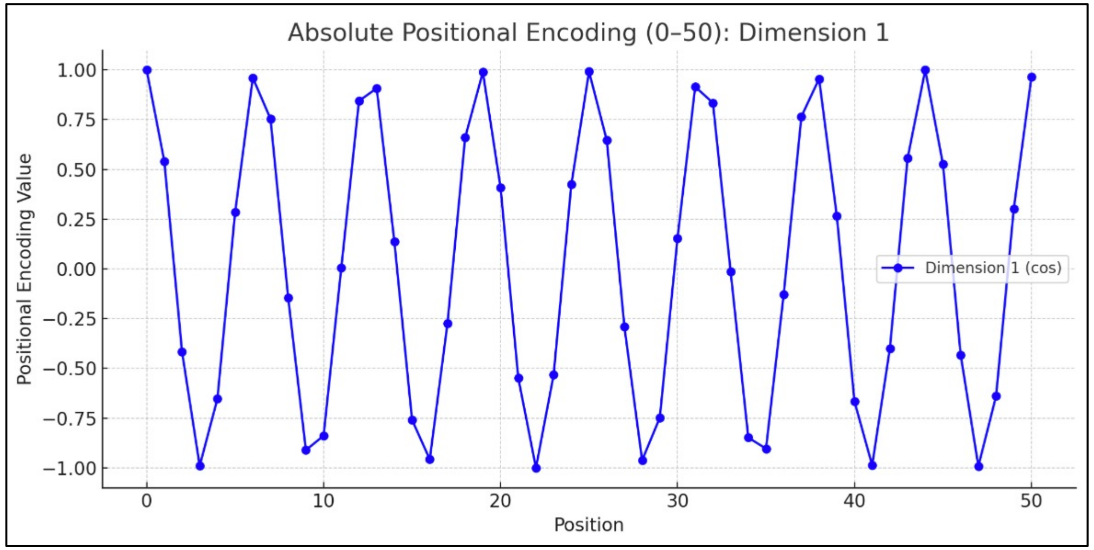


Absolute positional encoding

Original



p(i) for dimension 1



This is an (inaccurate) formula in the paper correction: k -> i (on the right-hand side)

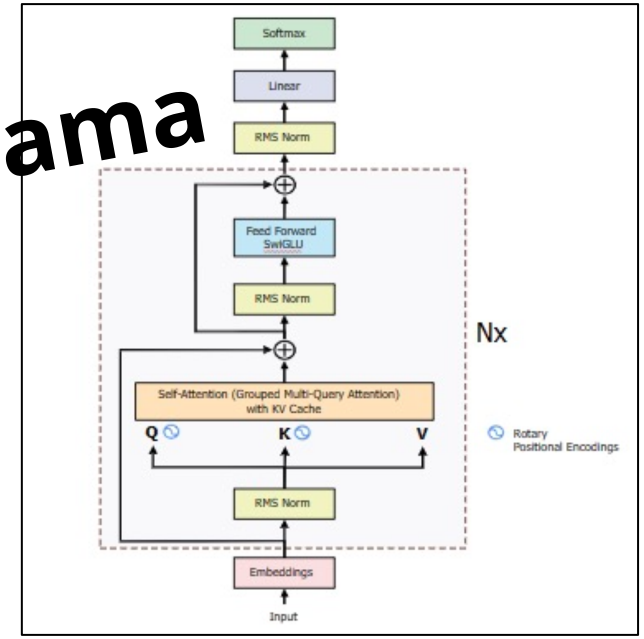
These numbers are added

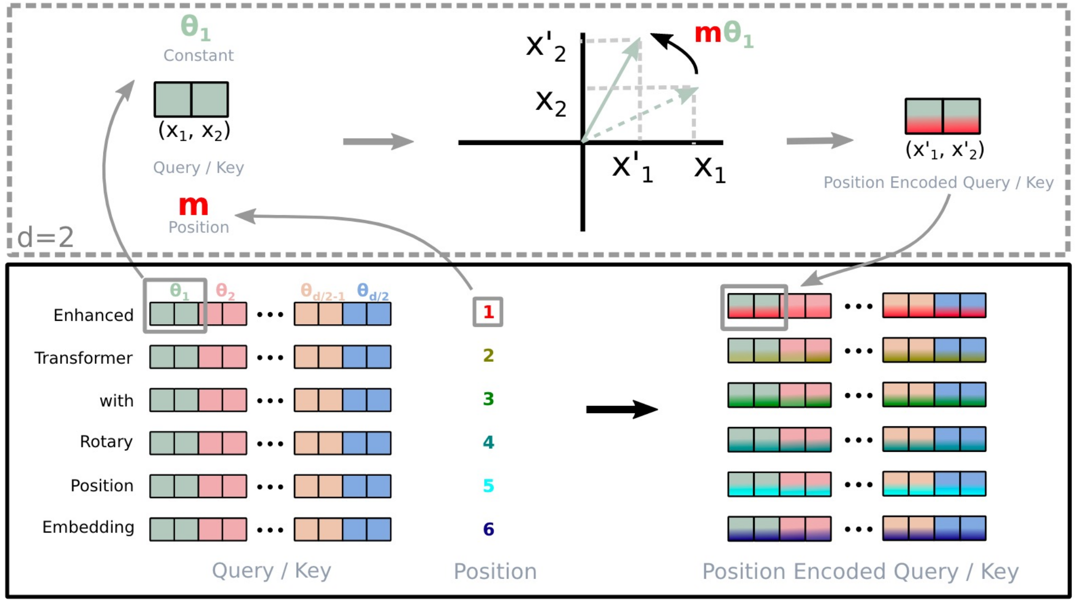
$$\begin{cases} \boldsymbol{p}_{i,2t} &= \sin(k/10000^{2t/d}) \\ \boldsymbol{p}_{i,2t+1} &= \cos(k/10000^{2t/d}) \end{cases} \tag{4}$$

in which $\boldsymbol{p}_{i,2t}$ is the $2t^{th}$ element of the d-dimensional vector $\boldsymbol{p}_i$. In the next section, we show that our proposed RoPE is related to this intuition from the sinusoidal function perspective. However, instead of directly adding the position to the context representation, RoPE proposes to incorporate the relative position information by multiplying with the sinusoidal functions.

RoPE: Rotary position embedding

Llama

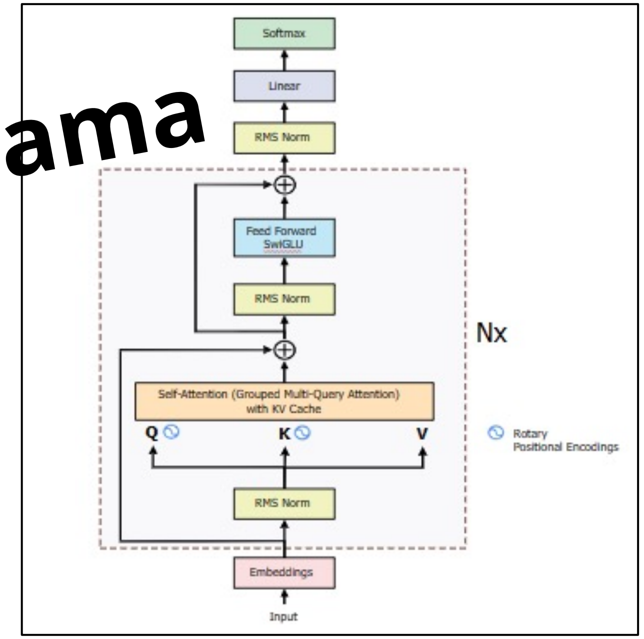


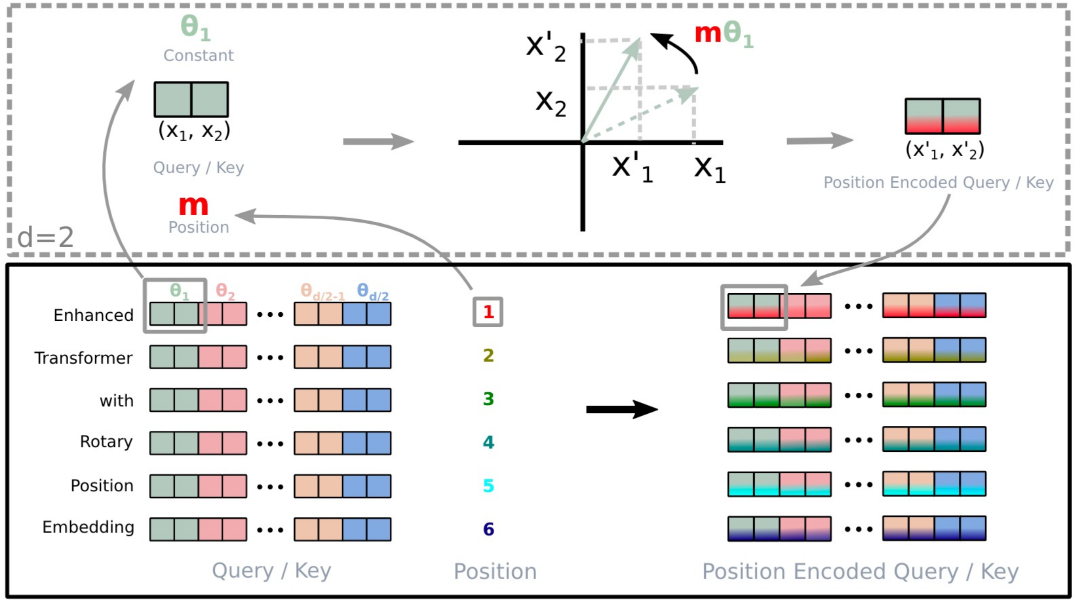


We perturb Q and K by **rotating** vectors (its angle *proportional to* the position *m*)

RoPE: Rotary position embedding

Llama





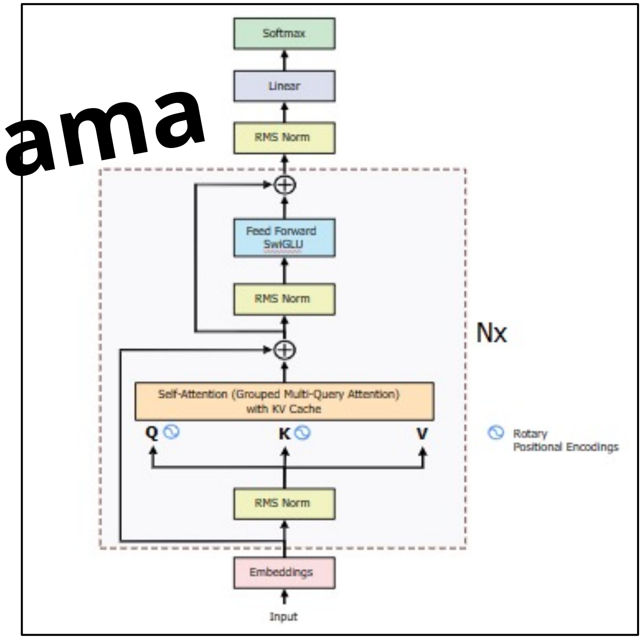
$$f_{\{q,k\}}(\boldsymbol{x}_m, m) = \boldsymbol{R}^d_{\Theta,m} \boldsymbol{W}_{\{q,k\}} \boldsymbol{x}_m$$

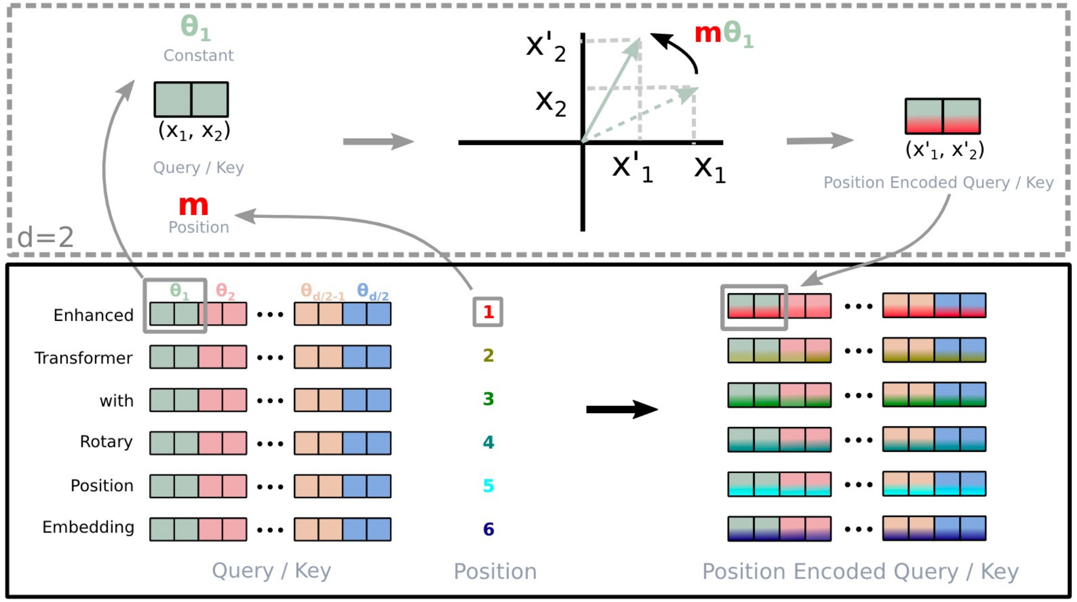
$$\boldsymbol{R}^d_{\Theta,m} = \begin{pmatrix} \cos m\theta_1 & -\sin m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ \sin m\theta_1 & \cos m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cos m\theta_2 & -\sin m\theta_2 & \cdots & 0 & 0 \\ 0 & 0 & \sin m\theta_2 & \cos m\theta_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \cos m\theta_{d/2} & -\sin m\theta_{d/2} \\ 0 & 0 & 0 & 0 & \cdots & \sin m\theta_{d/2} & \cos m\theta_{d/2} \end{pmatrix}$$

matrix with pre-defined parameters $\Theta = \{\theta_i = 10000^{-2(i-1)/d}, i \in [1, 2, ..., d/2]\}$. A graphic

RoPE: Rotary position embedding

Llama





$$f_{\{q,k\}}(\boldsymbol{x}_m, m) = \boldsymbol{R}^d_{\Theta,m} \boldsymbol{W}_{\{q,k\}} \boldsymbol{x}_m$$

$$\boldsymbol{R}^d_{\Theta,m} = \begin{pmatrix} \cos m\theta_1 & -\sin m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ \sin m\theta_1 & \cos m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cos m\theta_2 & -\sin m\theta_2 & \cdots & 0 & 0 \\ 0 & 0 & \sin m\theta_2 & \cos m\theta_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \cos m\theta_{d/2} & -\sin m\theta_{d/2} \\ 0 & 0 & 0 & 0 & \cdots & \sin m\theta_{d/2} & \cos m\theta_{d/2} \end{pmatrix}$$

matrix with pre-defined parameters $\Theta = \{\theta_i = 10000^{-2(i-1)/d}, i \in [1, 2, ..., d/2]\}$. A graphic

inner product is an angular distance

$$x_1 \cdot x_2 = |x_1||x_2| \cos\theta$$

RoPE: only relative distance matters

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

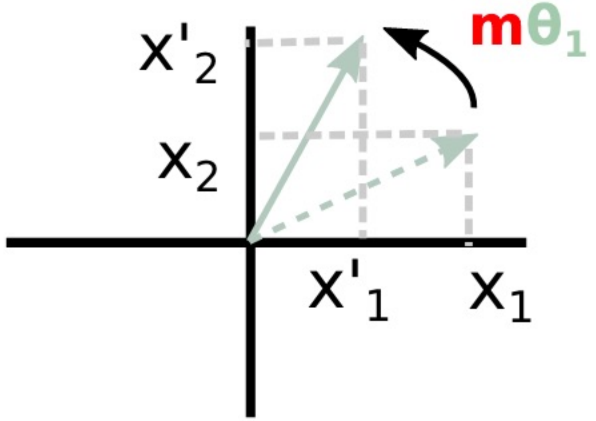
Attention is inner product

$$f_{\{q,k\}}(\boldsymbol{x}_m, m) = \boldsymbol{R}^d_{\Theta,m}\boldsymbol{W}_{\{q,k\}}\boldsymbol{x}_m$$

$$\boldsymbol{R}^d_{\Theta,m} = \begin{pmatrix} \cos m\theta_1 & -\sin m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ \sin m\theta_1 & \cos m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cos m\theta_2 & -\sin m\theta_2 & \cdots & 0 & 0 \\ 0 & 0 & \sin m\theta_2 & \cos m\theta_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \cos m\theta_{d/2} & -\sin m\theta_{d/2} \\ 0 & 0 & 0 & 0 & \cdots & \sin m\theta_{d/2} & \cos m\theta_{d/2} \end{pmatrix}$$

matrix with pre-defined parameters $\Theta = \{\theta_i = 10000^{-2(i-1)/d}, i \in [1, 2, ..., d/2]\}$. A graphic

RoPE is rotation

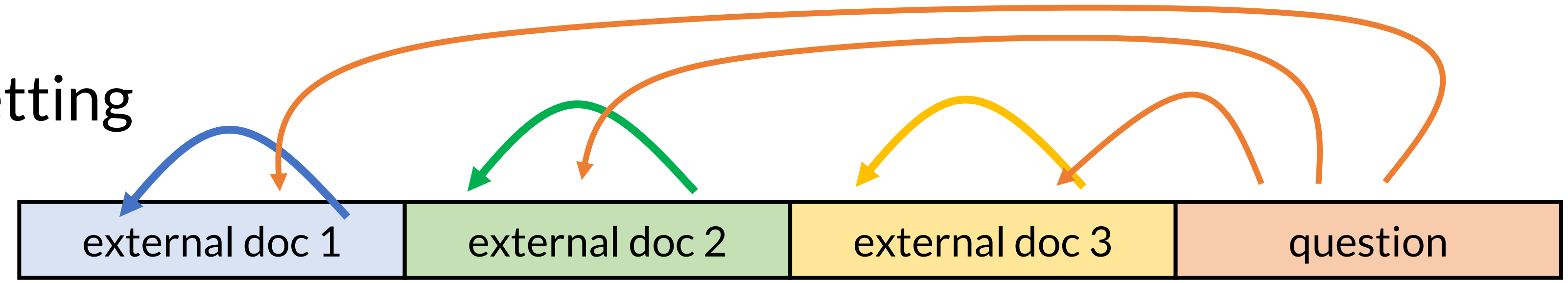


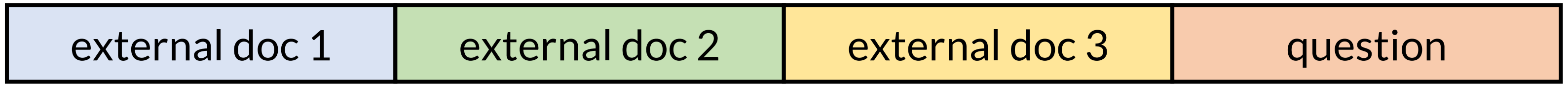
$$x_1 \cdot x_2 = |x_1||x_2|\cos\theta$$

Inner product is rotation

RoPE allows KV-cache re-use

Suppose RAG setting

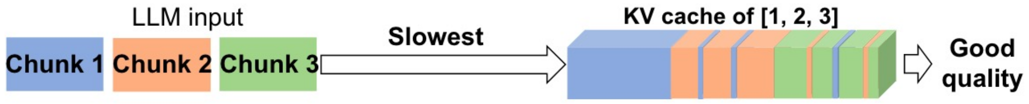




KV cache can be re-used after ***re-encoding*** *positions*

CacheBlend

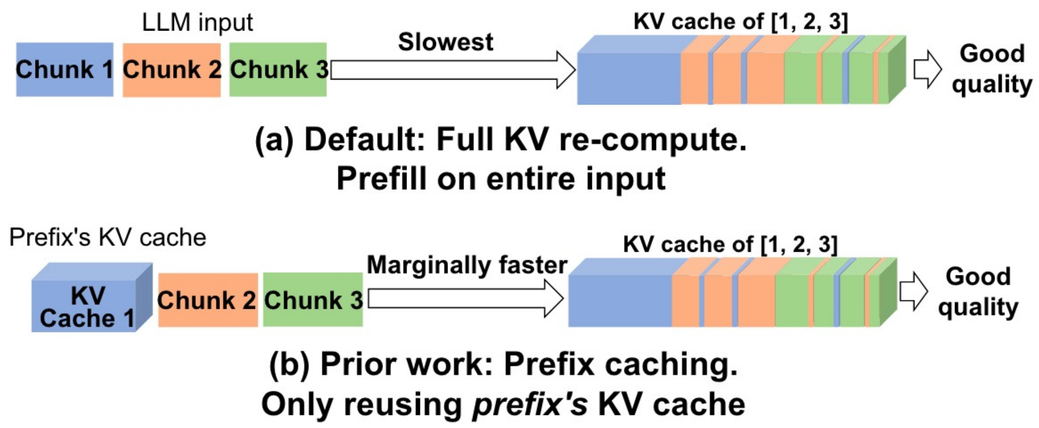
Different forms of KV cache re-use



(a) Default: Full KV re-compute. Prefill on entire input

for a new prompt

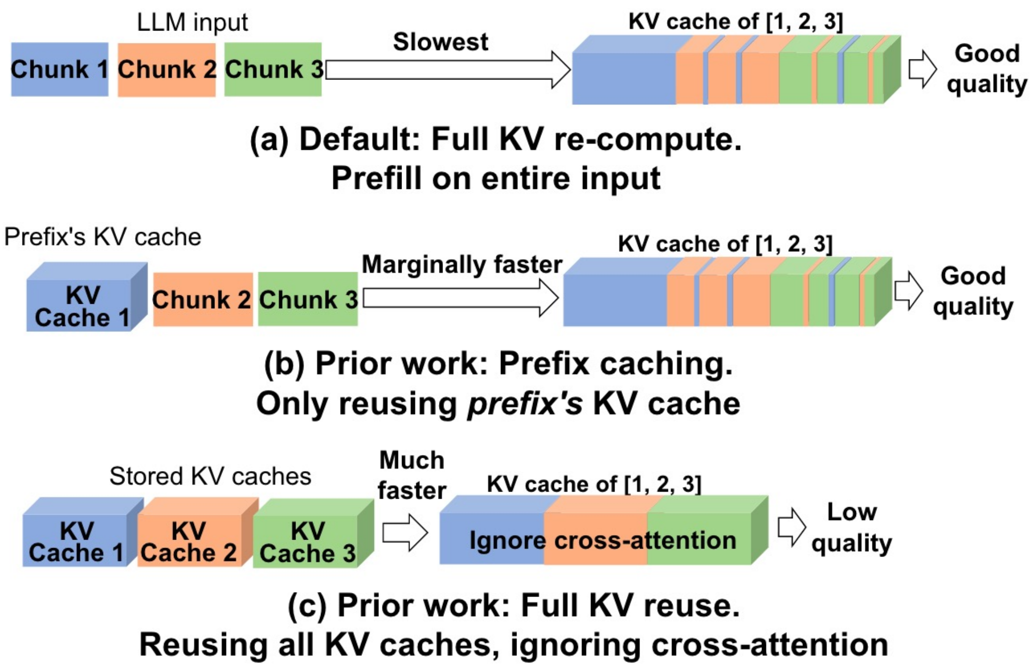
Different forms of KV cache re-use



transformers library may do this

used by vllm

Different forms of KV cache re-use

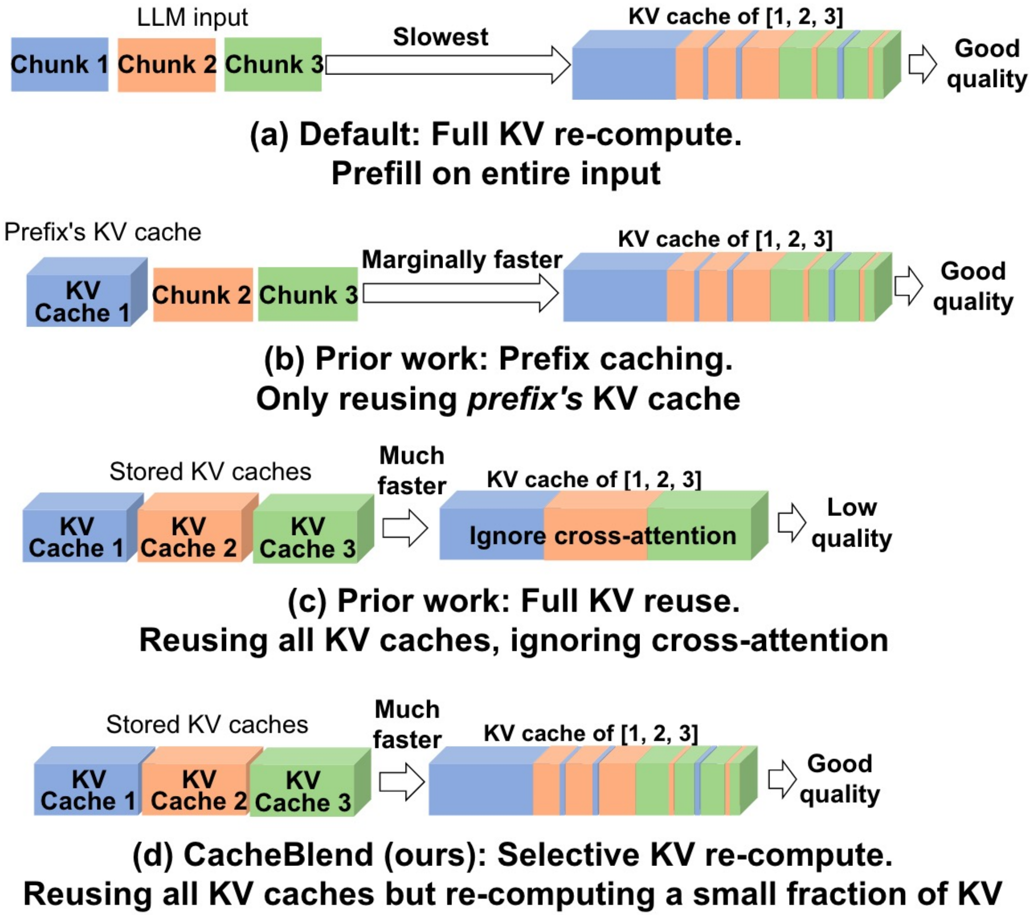


transformers library may do this

used by vllm

prompt cache

Different forms of KV cache re-use



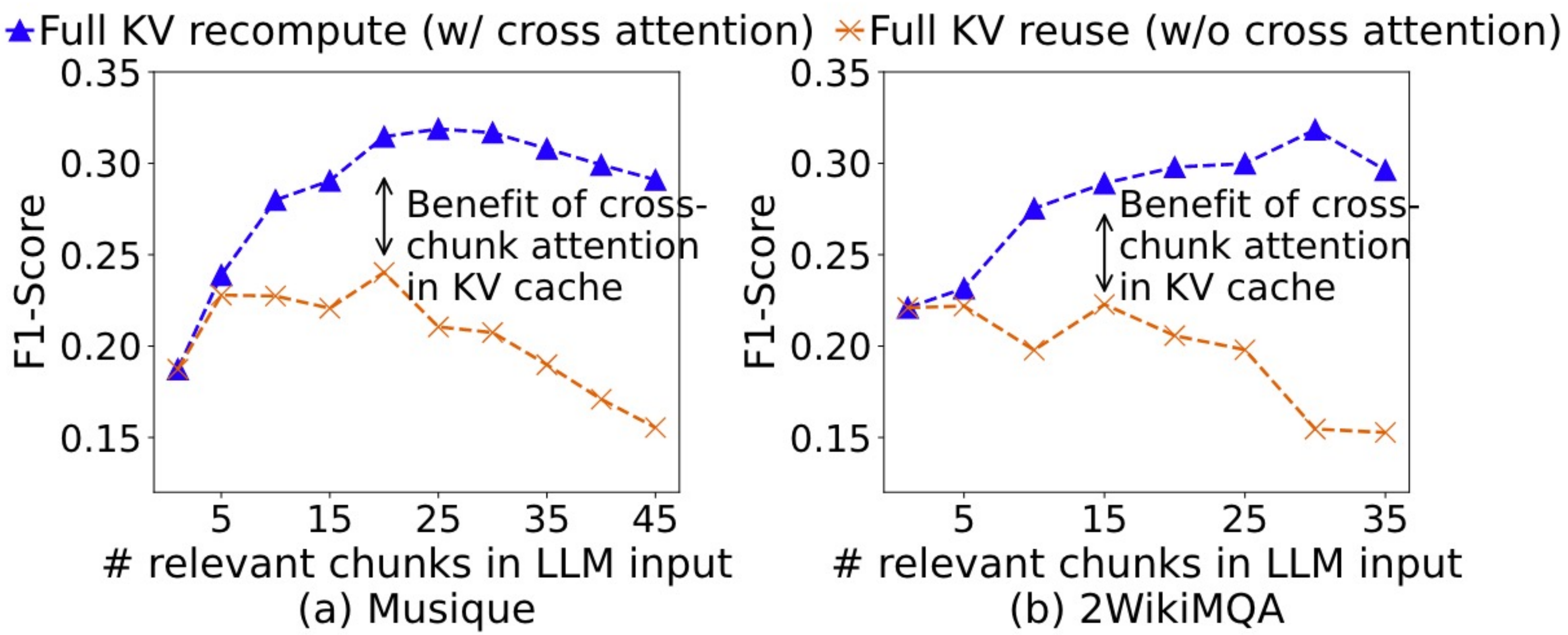
transformers library may do this

used by vllm

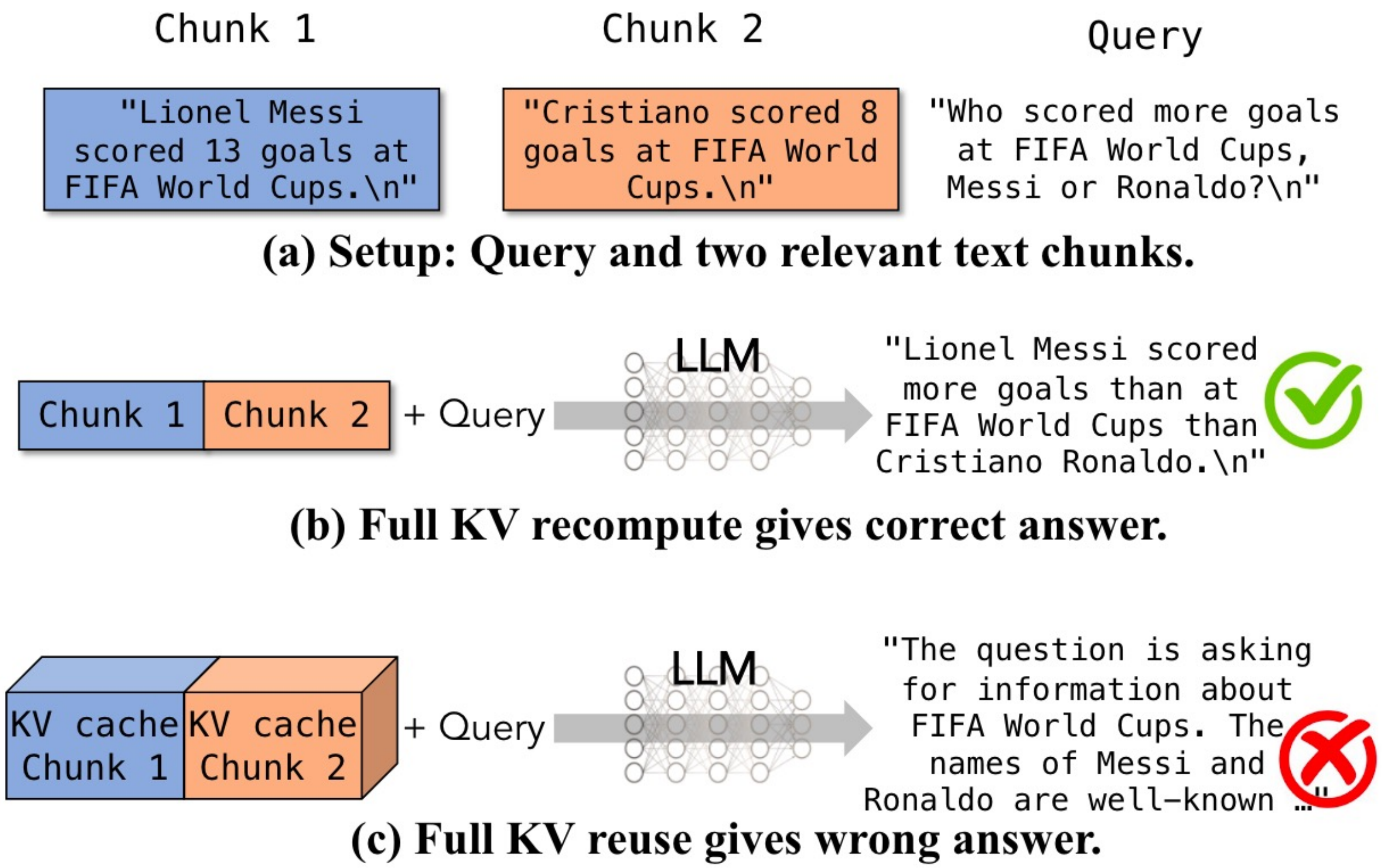
prompt cache

CacheBlend

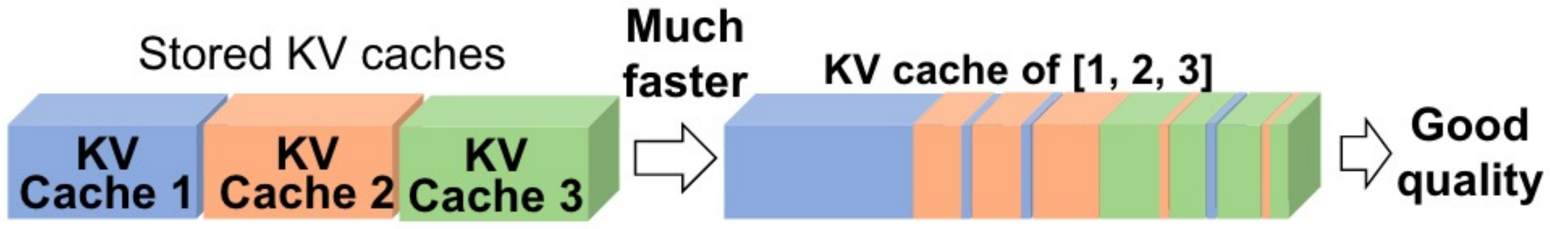
Cross attention is important: benchmark results



Cross attention is important: example



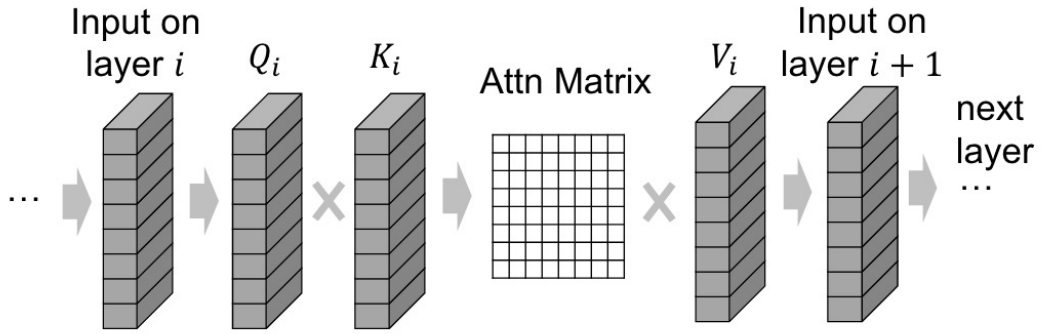
Which tokens to recompute?



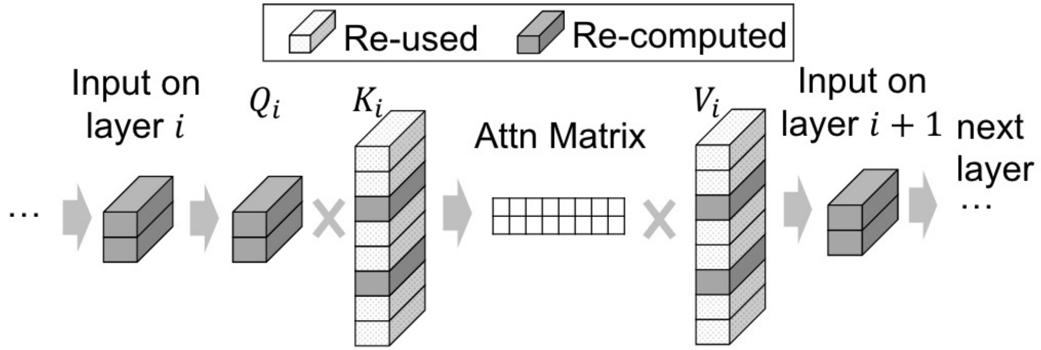
(d) CacheBlend (ours): Selective KV re-compute.
Reusing all KV caches but re-computing a small fraction of KV

KV deviation: We define the *KV deviation* of a KV cache KV on layer i of token j as the absolute difference between $KV_i[j]$ and $KV_i^{\text{full}}[j]$, denoted as $\Delta_{\text{kv}}(KV_i, KV_i^{\text{full}})[j]$. It measures how much different the given KV is on a particular token and layer compared to the full-prefilled KV cache. We will later use the KV deviation to identify which tokens' KV has higher deviation and thus need to be updated.

Insight 2. *Tokens with the highest KV deviations on one layer are likely to have the highest KV deviations on the next layer.*



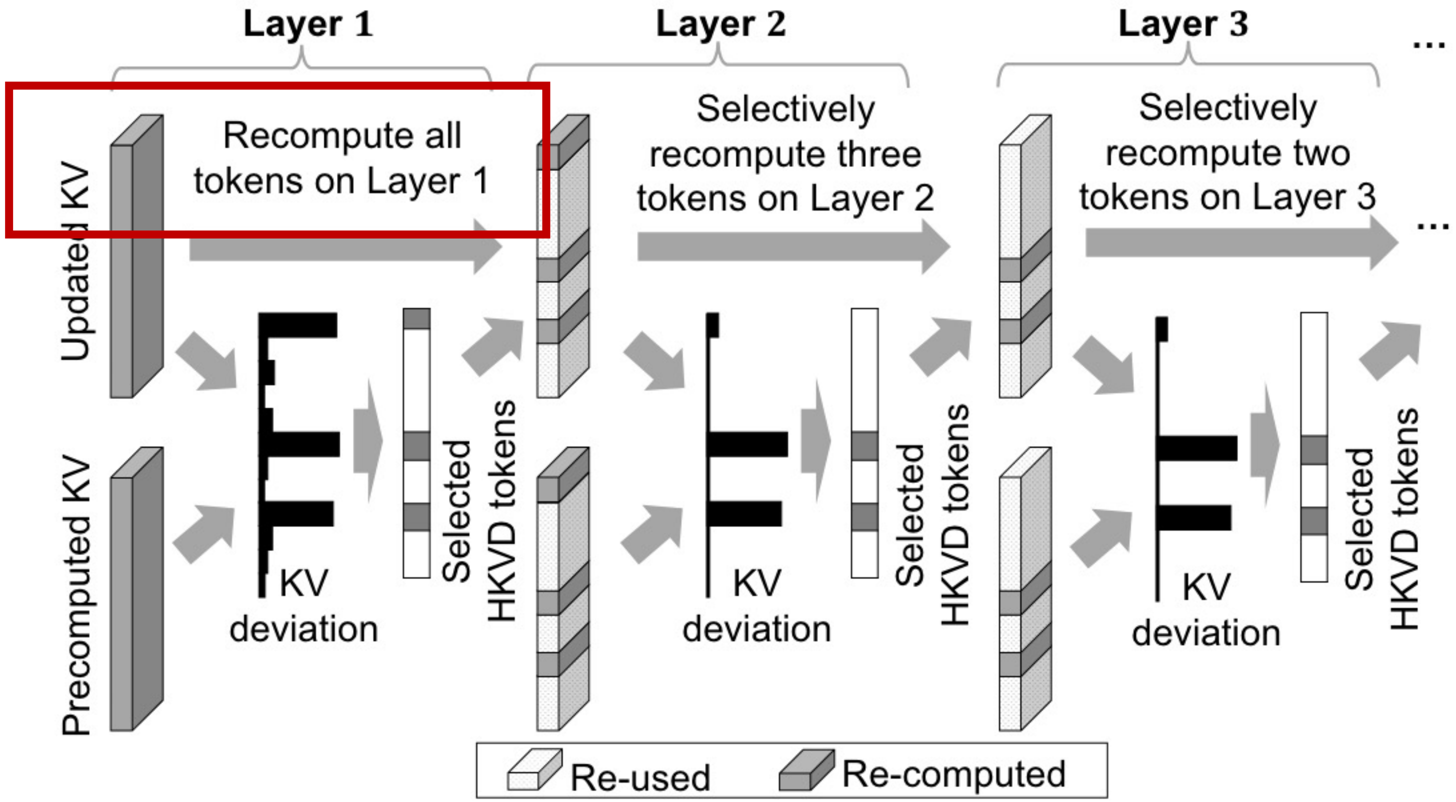
(a) Full KV recompute for reference



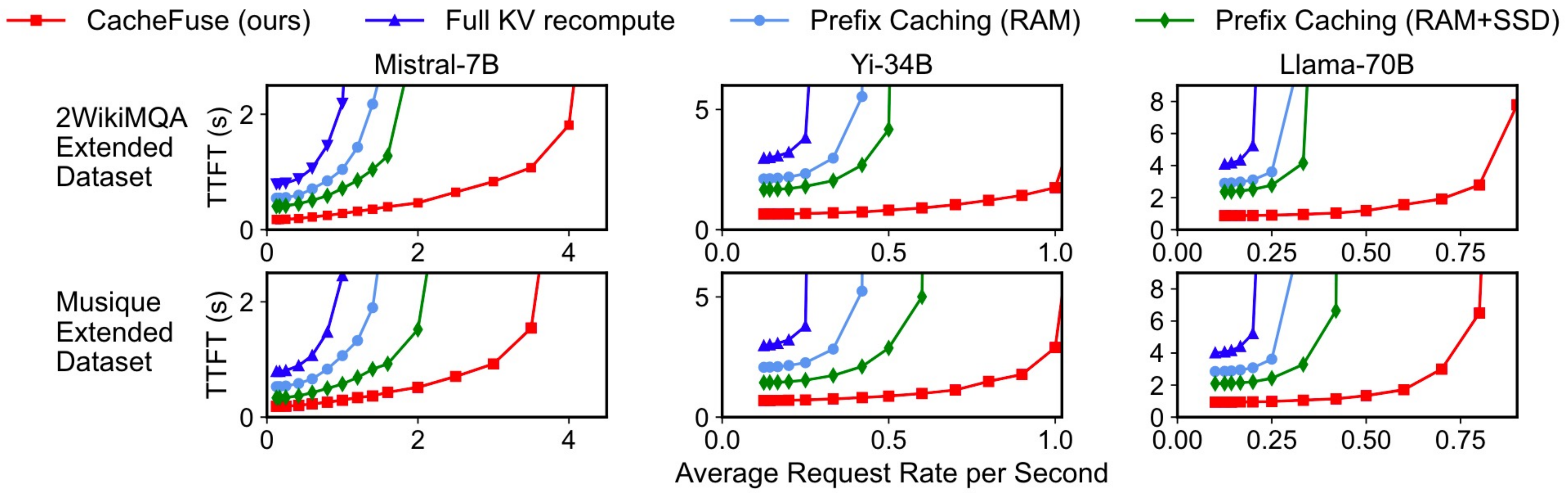
(b) Selective KV recompute on two selected tokens

Insight 2. *Tokens with the highest KV deviations on one layer are likely to have the highest KV deviations on the next layer.*

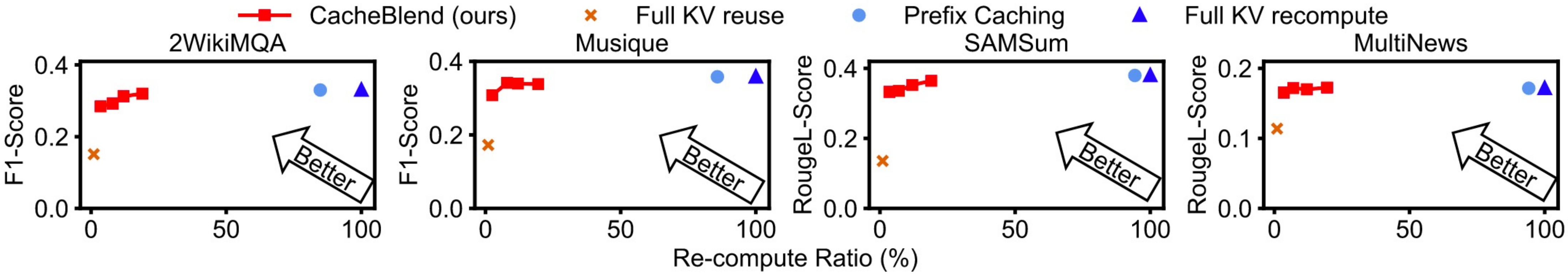
KV deviation is estimated on Layer 1



Time to First Token (TTFT) is lower/better



Retains quality



Summary

- Rotary Position Embedding allows **block-wise** KV cache re-use
- Simple approach (Prompt Cache) lowers accuracy
- CacheBlend ***selectively re-computes*** cross-attention
- Preview: Block Attention will overcome the same problem w/ **fine-tuning**

Questions?