为什么LLM推理加速有KV Cache而没有Q Cache?

喜欢卷卷的瓦力 瓦力算法学研所 2024年07月21日 09:00 上海



→◇ 技术总结专栏 ◇>-



瓦力算法学研所

我们是一个致力于分享人工智能、机器学习和数据科学方面理论与应用知识的公众号。我... 117篇原创内容

公众号

本篇介绍为什么LLM推理加速有KV Cache而没有Q Cache。

简单来说,LLM在**decoding阶段的每次推理只会用到当前的Q**,这次用的Q下次不会用到,所以不用Cache Q;

但是每次都要用到当前和过去所有的KV,这次用到的KV下次马上就要再用一次,所以Cache KV可以加速推理。

下面说明原因:

观察Attention公式,这个K和Q怎么看都很对称,为什么只Cache K而不Cache Q?

$$egin{aligned} & \operatorname{Attention}(K,Q,V) = \operatorname{softmax}(rac{QK^{ op}}{\sqrt{d_k}})V, \ & K,Q \in \mathbb{R}^{T imes d_k}, V \in \mathbb{R}^{T imes d_y} \end{aligned}$$

把KQV写成分块的形式,像这样:

$$K = egin{bmatrix} k_1 \ k_2 \ dots \ k_T \end{bmatrix}, Q = egin{bmatrix} q_1 \ q_2 \ dots \ q_T \end{bmatrix}, V = egin{bmatrix} v_1 \ v_2 \ dots \ v_T \end{bmatrix} \ k_i, q_i \in \mathbb{R}^{1 imes d_k}, v_i \in \mathbb{R}^{1 imes d_u}, i = 1, 2, \ldots, T \end{cases}$$

然后Q和K转置的矩阵乘就变成了这样:

$$QK^\top = \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_T \end{bmatrix} \begin{bmatrix} k_1^\top \ k_2^\top \dots \ k_T^\top \end{bmatrix} = \begin{bmatrix} q_1 \cdot k_1 & q_1 \cdot k_2 & \cdots & q_1 \cdot k_T \\ q_2 \cdot k_1 & q_2 \cdot k_2 & \cdots & q_2 \cdot k_T \\ \vdots & \vdots & \ddots & \vdots \\ q_T \cdot k_1 & q_T \cdot k_2 & \cdots & q_T \cdot k_T \end{bmatrix}$$

直到这一步,K和Q看上去都很对称。轮换一下K和Q对结果没有本质影响。

V的引入破坏了这一对称性。忽略 dk 系数,第i行的softmax简写成 Si ,attention操作的结果变成了这样:

$$\mathcal{S}(QK^{ op})V = egin{bmatrix} \mathcal{S}_{1}(q_{1} \cdot k_{1}) & \mathcal{S}_{1}(q_{1} \cdot k_{2}) & \cdots & \mathcal{S}_{1}(q_{1} \cdot k_{T}) \ \mathcal{S}_{2}(q_{2} \cdot k_{1}) & \mathcal{S}_{2}(q_{2} \cdot k_{2}) & \cdots & \mathcal{S}_{2}(q_{2} \cdot k_{T}) \ dots & dots & \ddots & dots \ \mathcal{S}_{T}(q_{T} \cdot k_{1}) & \mathcal{S}_{T}(q_{T} \cdot k_{2}) & \cdots & \mathcal{S}_{T}(q_{T} \cdot k_{T}) \end{bmatrix} egin{bmatrix} v_{1} \ v_{2} \ dots \ v_{2} \ dots \ v_{T} \end{bmatrix} \\ = egin{bmatrix} \mathcal{S}_{1}(q_{1} \cdot k_{1})v_{1} + \mathcal{S}_{1}(q_{1} \cdot k_{2})v_{2} + \cdots + \mathcal{S}_{1}(q_{1} \cdot k_{T})v_{T} \ \mathcal{S}_{2}(q_{2} \cdot k_{1})v_{1} + \mathcal{S}_{2}(q_{2} \cdot k_{2})v_{2} + \cdots + \mathcal{S}_{2}(q_{2} \cdot k_{T})v_{T} \ dots \ \mathcal{S}_{T}(q_{T} \cdot k_{1})v_{1} + \mathcal{S}_{T}(q_{T} \cdot k_{2})v_{2} + \cdots + \mathcal{S}_{T}(q_{T} \cdot k_{T})v_{T} \ \mathcal{S}_{T}(q_{T} \cdot k_{T})v_{T} \ \mathcal{S}_{T}(q_{T} \cdot k_{1})v_{1} + \mathcal{S}_{T}(q_{T} \cdot k_{2})v_{2} + \cdots + \mathcal{S}_{T}(q_{T} \cdot k_{T})v_{T} \ \mathcal{S}_{T}(q_{T} \cdot k_$$

这是没有Causal Mask (因果掩码) 的情况。加入Causal Mask会变成这样:

$$\begin{split} \mathcal{S}(QK^{\top})V &= \begin{bmatrix} \mathcal{S}_{1}(q_{1} \cdot k_{1}) & 0 & \cdots & 0 \\ \mathcal{S}_{2}(q_{2} \cdot k_{1}) & \mathcal{S}_{2}(q_{2} \cdot k_{2}) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{S}_{T}(q_{T} \cdot k_{1}) & \mathcal{S}_{T}(q_{T} \cdot k_{2}) & \cdots & \mathcal{S}_{T}(q_{T} \cdot k_{T}) \end{bmatrix} \begin{bmatrix} v_{1} \\ v_{2} \\ \vdots \\ v_{T} \end{bmatrix} \\ &= \begin{bmatrix} \mathcal{S}_{1}(q_{1} \cdot k_{1})v_{1} \\ \mathcal{S}_{2}(q_{2} \cdot k_{1})v_{1} + \mathcal{S}_{2}(q_{2} \cdot k_{2})v_{2} \\ \vdots \\ \mathcal{S}_{T}(q_{T} \cdot k_{1})v_{1} + \mathcal{S}_{T}(q_{T} \cdot k_{2})v_{2} + \cdots + \mathcal{S}_{T}(q_{T} \cdot k_{T})v_{T} \end{bmatrix} \end{split}$$

可以写一下结果的通项, 没有Causal Mask:

$$[\mathcal{S}(QK^{ op})V]_t = \sum_{j=1}^T \mathcal{S}_t(q_t \cdot k_j)v_j$$

有Causal Mask:

$$[\mathcal{S}(QK^{ op})V]_t = \sum_{j=1}^t \mathcal{S}_t(q_t \cdot k_j)v_j$$

无论有没有Causal Mask, Q和K在结果中都是不对称的。

在序列的t位置,Q只有当前位置的 qtq_t 参与了计算,而K和V多个位置参与了计算,所以需要KV Cache,而不需要Q Cache。

在没有Causal Mask时,计算t位置的Attention需要未来的KV,这在实际进行自回归推理时无法得到;加上Causal Mask之后,只需要1,2,...,t位置的KV就可以进行推理。

想要获取技术资料的同学欢迎关注公众号, 进群一起交流~

参考文献

[1] 为什么加速LLM推断有KV Cache而没有Q Cache? - 知乎 (https://www.zhihu.com/question/653658936/answer/3545520807)



添加瓦力微信

算法交流群·面试群大咖分享·学习打卡

● 公众号·瓦力算法学研所



瓦力算法学研所

我们是一个致力于分享人工智能、机器学习和数据科学方面理论与应用知识的公众号。我... 117篇原创内容

公众号

学术理论解析 53

学术理论解析 · 目录

上一篇

下一篇

如何预估训练或推理大模型时所需要的显存? 解析不同参数下大模型显存量化方法 为什么多模态大语言模型最近用BLIP2中Q-Former结构的变少了?