

赞同 5

分享

2024华为：DARE一种在工业级推荐系统部署LLM的解码加速框架



SmartMindAI

专注搜索、广告、推荐、大模型和人工智能最新技术，欢迎关注我

已关注

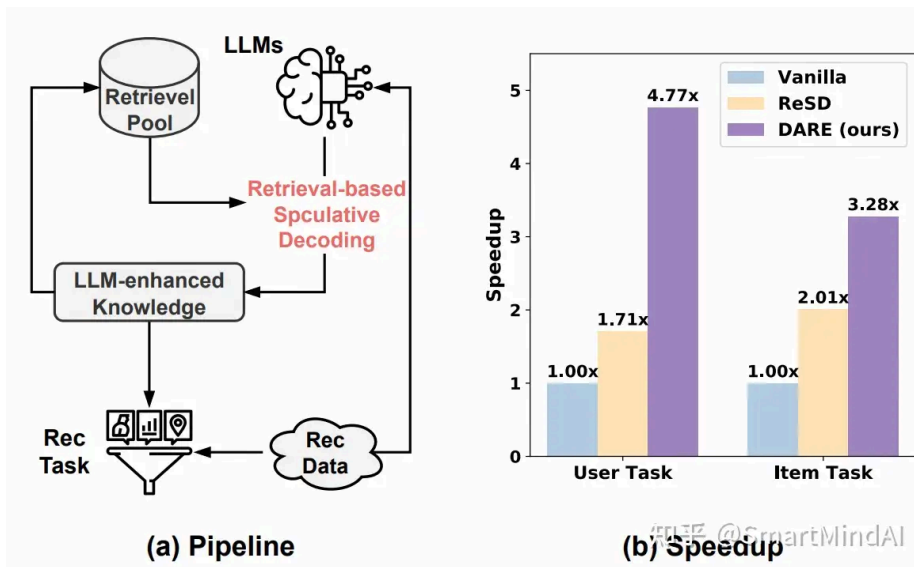
5 人赞同了该文章

收起

Introduction

自2020年以来，大型语言模型⁺（LLMs）凭借其广泛的知识 and 强大的推理能力，在各个领域引发了革命。在推荐系统⁺（RSs）中，LLMs与RSs的集成成为研究热点。

基于LLMs的推荐系统（RS）知识生成具有特定属性，适合基于检索的推测解码。首先，推荐知识生成是一个持续过程。随着用户行为变化和新内容的引入，我们需要不断生成新知识，同时拥有大量旧知识。其次，新旧知识之间往往有可复用的内容。例如，新老用户配置文件⁺可能因用户偏好连续性而重叠。因此，我们可以利用旧知识作为检索池来提取草案文本，然后使用LLMs进行验证，加速新知识生成，如图(a)所示。



然而，直接应用这种方法在实践中效果不佳。首先，RS中的大量内容和用户导致检索效率低下，影响加速效果。构建包含所有用户和内容知识的检索池会非常庞大，极大延长检索时间。因此，保持较小且相似的检索池至关重要，以确保低检索时间和高接受率的草案令牌。其次，RS对LLMs生成的文本具有较高容忍度。下游推荐任务可以通过语义接近但不完全相同的文本达到相似效果，这为推测解码提供了进一步加速的空间。

基于上述见解，本文提出了一种LLM推荐解码加速框架（DARE）。我们为基于检索的推测解码推荐引入了两个关键增强措施：

定制化检索池：提高检索效率。我们引入基于合作和属性的检索池构建方案，通过二进制路由器为用户和内容分配合适的检索池。这些个性化的紧凑检索池保持知识相似性，确保低检索时间和高接受率的草案令牌。

发散。

Preliminary Findings

Speculative Decoding for Recommendation

主流的基于Transformer的LLM通常采用自回归解码。对于输入的令牌序列 x_1, \dots, x_t ，语言模型 \mathcal{M} 生成下一个词如下： $x_{t+1} \sim q_{t+1} = \mathcal{M}(x_{\leq t})$

其中 q_{t+1} 表示来自 \mathcal{M} 的条件概率分布 q_{t+1} 是从 q_{t+1} 中采样的下一个令牌。之后 \mathcal{M} 遵循相同的过程生成下一个令牌。尽管生成质量令人满意，但自回归解码仅在解码步骤中生成一个令牌，使其效率低下且耗时。

为此，提出了推测解码。推测解码在每个解码步骤中生成一系列令牌。这是一种先拟后验证的解码范式，在每个解码步骤中，首先高效地拟合多个未来令牌，然后并行验证所有这些令牌与目标语言模型。拟合策略有很多，例如，使用小型语言模型，从数据库检索⁺。我们的工作主要集中在基于检索的拟合模型上，这些模型从给定的检索池中检索拟合，因为小型语言模型可能缺乏推荐知识，而推荐知识生成可以自然地提供合适的检索池。这里，我们以一个例子为例，深入探讨其两个子步骤-----拟合和验证：

拟合阶段负责高效地拟合多个未来令牌。正式地，给定输入序列 $\{x_1, \dots, x_t\}$ ，使用一个拟合模型 $\tilde{\mathcal{M}}$ （例如，一个从数据库检索相关文本的检索器）生成下一个 K 个拟合令牌： $\tilde{x}_1, \dots, \tilde{x}_K = \text{Draft}(x_{\leq t}, \tilde{\mathcal{M}})$

其中， $\tilde{x}_i, i = 1, \dots, K$ 表示由 $\tilde{\mathcal{M}}$ 生成的被挑选的令牌 $\text{Draft}(\cdot)$ 代表挑选策略。验证阶段利用目标LLM并行验证所有这些被挑选的令牌。在输入序列 $\{x_1, \dots, x_t\}$ 和被挑选的序列 $\{\tilde{x}_1, \dots, \tilde{x}_K\}$ 的情况下，目标LLM \mathcal{M} 同时计算了 $K + 1$ 个概率分布，这些分布分别对应于原始序列和每个被挑选的序列： $q_i = \mathcal{M}(x_{\leq t}, \tilde{x}_{<i}), i = 1, \dots, K + 1$

然后，每个草稿元素 \tilde{x}_i 依据特定准则 q_i 进行逐个验证。通常，采用贪婪验证策略进行基于检索的推断解码： $\tilde{x}_i = \arg \max q_i$ ，

仅当 \tilde{x}_i 满足公式（GR）中的标准时，将其作为最终输出选择，即 $x_{t+i} = \tilde{x}_i$ 。如果在位置 c 处拟合的令牌 \tilde{x}_c 未能通过验证⁺，它将由目标LLM的分布 q_c 进行校正，即 $x_{t+c} \leftarrow \arg \max q_c$

位置 c 之后的所有拟合令牌都将被剔除，以确保质量与目标LLM的标准一致。

推荐知识生成的特性使其非常适合应用检索式推测解码。检索式推测解码需要与当前生成的文本重叠的检索池。随着用户行为的演变和新内容的加入，我们需要不断为新内容和用户的新行为生成新的知识，从而产生一个连续更新的关于用户过去行为和现有内容的旧知识流。此外，旧知识与新知识之间存在明显的共通之处。例如，旧用户配置文件和新用户配置文件的部分可能会重叠。因此，我们可以利用旧知识构建检索池，并利用检索式推测解码加速新知识的生成。

Finding 1: Retrieval Inefficiency

根据上述方法，我们对MovieLens-10M数据集进行初步实验，遵循KAR的设置，首先使用LLMs生成推荐知识，然后将知识适应到下游任务。这里，我们利用vicuna-7b-v1.3[¹]来根据用户行为生成细化的用户偏好。为了实现基于检索的推测性解码，我们模拟用户的流式行为，并将其划分为多个段落。为了简化，用户的历史行为 $\{x_1, \dots, x_n\}$ 被分为两个段落：旧历史 x_1, \dots, x_m 和新历史 x_{n-m}, \dots, x_n ，其中 $(\frac{n}{2} < m < n)$ 。然后，vicuna-7b-v1.3基于自回归解码为所有旧历史生成知识，并在此基础上构建检索池。在为新历史生成知识时，我们采用推测性解码，从检索池中检索草稿，并由vicuna-7b-v1.3验证。

在上述条件下，我们探索了在构建检索池时，不同数量的旧知识样本（范围从10到用户数量的最大值）时，构建检索池时的令牌生成速度（令牌生成速度）和用于检索相关文本的总时间比例（检索时间比例）变化情况，结果见图。在这张图中，生成速度在检索池中的知识条目数量增加时先上升后下降。当检索池由所有用户的所有旧知识构建时，检索时间比例超过20%，导致生成速度从峰

因此，大检索池并不总是有利的。虽然大检索池可以提供更丰富的相关内容，但也带来了检索效率的降低，从而影响加速效果。构建一个既能同时保持低检索时间，又能包含与生成文本相似内容的最优检索池是至关重要的。

Finding 2: Diversity Tolerance

此外，我们还研究了LLM生成文本多样性的下游任务影响。类似于之前的实验，我们使用vicuna模型的7b版本v1.3来生成MovieLens-10M上的用户喜好知识。然而，在生成过程中，我们从最有可能的前 k 个词中采样，以创建近似但多样的文本。然后，我们参考相关方法，将BERT的知识编码适应到推荐系统中的CTR预测任务。具体来说，我们为数据集中的所有用户生成四组不同的用户偏好知识。然后，我们将这些知识应用于两个著名的CTR模型，即DIN模型和DCNv2模型，以展示它们在AUC指标和Logloss指标方面的性能。在表中，“无增强”指的是结果未进行知识增强，“知识1”到“知识4”表示在不同采样下生成的知识增强结果。

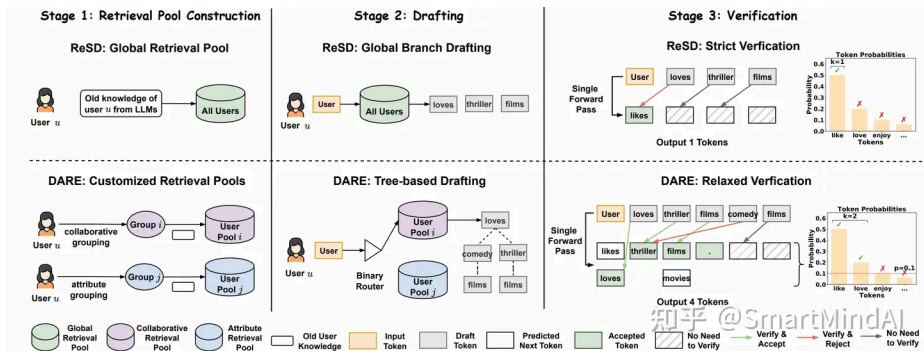
表中的结果显示，推荐任务对由LLM生成的知识文本的多样性具有很高的容忍度。相较于未进行增强的模型，知识增强能够带来显著提升，范围在1.5%到2%之间。然而，应用于下游任务的多样知识文本（知识1-4）之间的性能差异不足0.1%，这表明推荐任务对LLM生成文本的多样性不敏感。

过去，基于检索的推测性解码通常采用贪心验证，只接受概率最高的令牌以确保文本与自回归解码的一致性。然而，这种严格的验证限制了草稿令牌接受率。鉴于RS中的下游任务可以接受多样性的LLM文本，我们可以考虑放宽验证标准，允许推测性解码接受更多的草稿令牌，生成更多样化的文本内容，从而进一步加速这一进程。

Methodology

Overview

根据上述发现，我们为基于检索的推测解码在推荐知识生成中设计了两种改进。定制检索池涉及创建针对相似内容或用户的较小检索池，从而实现低检索时间。放松验证放宽了贪婪验证的条件，只接受最高概率的令牌，将其扩展为包括前 k 个最有可能的令牌，从而提高了草稿令牌的接受率。



我们提出的DARE的工作流程，包括三个阶段：定制检索池构建、基于树的起草和放松验证。在生成文本之前，定制检索池构建阶段利用先前生成的推荐知识，以二叉树的形式构建个性化检索池。我们首先将用户和内容分为不同的组，然后为每个组构建一个检索池。随后的阶段，基于树的起草，当为特定用户或内容生成新知识时，从指定的检索池中检索相关内容。这个过程产生了一个伪序列，来自二叉树分支，封装了多个潜在的后续文本，并通过目标LLM同时验证，附带有相应的注意力标记。在放松验证阶段，我们接受前 k 个最高概率令牌，其概率超过阈值 p 。这允许接受更多的草稿令牌，并在生成过程中防止发散，进一步提升了生成效率。

Customized Retrieval Pool Construction

自定义检索池需要适度的容量和内部知识相似性，这需要对用户和内容进行分区。为不同的组分配不同的检索池。为了保持知识相似性，我们可以整合协作信号来将相似的用户和内容分组。一种常见方法是根据推荐模型训练的用户-内容交互中提取的嵌入对用户或内容进行聚类。然而，新引入的用户和内容可能缺乏或根本没有交互记录，这使得获得可靠的嵌入变得具有挑战性。考虑到具有相似属性的用户或内容可能更加相似，它们的属性可以作为构建相似组的基础。因此，我们设计了

组。首先，使用推荐模型（例如，LightGCN）对用户-内容交互进行训练，随后提供相关嵌入，例如ID嵌入和属性嵌入。鉴于RSs不断训练推荐模型，我们可以重用这些嵌入。然后，应用聚类算法（例如K均值）对这些嵌入进行聚类，以获得不同的用户或内容组。同一组内的用户或内容表现出相似性，从而确保由LLMs生成的知识更加一致。这反过来增加了检索相关文本的可能性。

基于属性的检索池通过相似属性对缺乏大量交互记录的内容或用户进行分区。尽管它们缺乏交互记录以获得训练良好的嵌入，但它们具有内在属性。具有相似属性的内容或用户更有可能表现出更高的相似性，从而导致由LLMs生成的知识更加一致。因此，具有类似属性的内容或用户，如类别，可以被放在同一个组中。如果基于通用属性如类别的组的大小超过了一定阈值，我们可以通过额外的属性，例如子类别，进一步细分它们，这些属性由人工设计的规则或决策树选择。

接下来，对于每个组中的内容或用户，如果之前有LLMs生成的知识，将使用这些知识构建一个检索池，形式为前缀树⁺。前缀树是一种广泛用于高效检索和存储的数据结构，它有效地处理了每个节点作为单个字符或单词的前缀匹配。每个组都维护自己的前缀树，其中每个节点代表一个令牌，从根节点到叶子节点的路径构成一个分支。这些使用之前生成的知识构建的分支都是永久分支，不会被删除。在生成新知识时，当前提示和新生成的内容也与后续生成相关。因此，我们动态地将提示和新内容添加到前缀树作为临时分支，用于每次知识生成。这些添加可能不一定能加速其他知识生成的加速，分支在生成完成后将被删除。

在构建基于协作的和基于属性的检索池后，设计了一个二进制路由器来为需要知识生成的用户和内容选择合适的检索池。它非常灵活，可以支持各种选择方案。基于我们的动机，默认方案是为具有广泛交互历史的内容和用户选择基于协作的检索池，而新用户和内容，具有少量或没有交互，被分配到基于属性的检索池。

Tree-based Drafting

在为用户或内容生成新知识之前，我们根据二进制路由标识识别对应的检索池 D 。在知识生成过程中，我们首先根据当前输入文本从 D 中检索出相关的子树，这代表了多个潜在的后续序列。接下来，生成伪序列、注意力掩码矩阵和位置ID，以促进目标LLM（树注意力）的并行验证。具体来说，假设当前输入序列是 $\{x_1, \dots, x_t\}$ ，则从输入序列的最后 n 个令牌中选取作为前缀，从 D 中提取出子树 T_t 如下，从输入序列的最后 n 个标记中提取一个子树 T_t 。接下来，为该子树生成伪序列、注意力掩码矩阵和位置ID，以便目标LLM进行并行验证（树注意力）。然后选择输入序列的最后 n 个标记作为前缀：

$$T_t = \text{Retrieve}(D, \{x_{t-n}, \dots, x_t\}, K)$$

在其中 $\text{Retrieve}(\cdot)$ 表示从字典树中通过前缀检索子树 K 是草稿关键词的最大长度。子树 T_t 同样是一个前缀树，每条分支代表一个潜在的后续草稿序列。较短的前缀会产生大量内容，但可能不太相关，而较长的前缀能确保高度相关性，但可能无法检索到任何内容。因此，我们将在检索过程中动态调整 n 的值。最初，使用相对较大的 n ，即较长的前缀，以确保相关性。如果检索到的关键词数量大大低于最大长度 K ，则减少 n ，以进一步尝试检索过程，直到获得大量关键词。相反，如果检索到的关键词数量超过 K ，则选择频率最高的关键词作为草稿关键词。

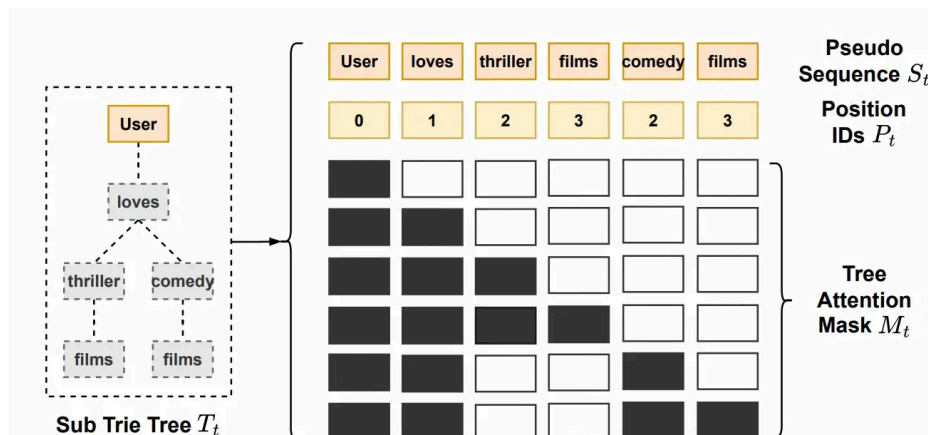


Figure 4: Tree attention. 知乎 @SmartMindAI

力机制⁺，同时并行验证多个潜在的草稿序列，如图1所示。这个机制通过深度优先搜索算法，为令牌树 T_t 构建了一个伪序列： $S_t = \{\tilde{x}_1, \dots, \tilde{x}_K\}$

请注意 S_t 的长度可能不会达到最大长度 K ；我们使用 K 是为了简化说明。同时，它调整了注意力掩码 M_t 和位置ID P_t 的设置，确保令牌树中的每个节点只能看到当前分支上的前一个节点，从而避免不同分支的草稿序列相互干扰。这样确保来自不同分支的草稿序列不会相互影响。

Relaxed Verification

在这个阶段，目标LLM将输入原始输入序列 $\{x_1, \dots, x_t\}$ ，伪序列 $S_t = \{\tilde{x}_1, \dots, \tilde{x}_K\}$ 树注意力mask M_t ，以及在草稿阶段获得的位置ID P_t 。然后执行单一前向传播，以并行验证所有草稿序列，即在每个位置生成条件概率，这确保了每个位置的条件概率被准确地计算出来：

$$q_i = \mathcal{M}(x_{\leq t}, \tilde{x}_{< i}, P_t, M_t), i = 1, \dots, K + 1$$

其中 q_i 表示词汇中所有词的概率分布 \mathcal{M} 为目标语言模型。在严格的验证中，我们从第一个位置开始，检查当前位置 i 上的标记 \tilde{x}_i 是否等于 q_i 中概率最高的标记。如果匹配，则接受标记 \tilde{x}_i ；否则，拒绝它。同样地，如果标记树 T_t 中的某个前驱节点被拒绝，那么它的所有后续节点都将被跳过。然后我们继续验证下一个可行的分支，最终接受最长的验证路径。

我们发现推荐任务对语言模型生成的知识文本具有很高的多样性容忍度，因此可以放宽严格的验证，以进一步提升生成效率。因此，我们将验证标准从概率最高的标记扩展到概率最高的 k 个词，即 $\tilde{x}_i \in \text{TopK}(q_i, k)$,

在 q_i 中选取概率最高的 k 个tokens的函数 $\text{TopK}(\cdot)$ 的地方。然而，我们通过在第 3 节的实验发现，仅放松这个约束可能导致生成的文本变得离散或发散，生成的文本长度远超使用自回归解码生成的文本。这可能是因为在概率最高的 k 个token中，例如，实际概率 $q_i(e)$ 仍然非常小。因此，我们对实际概率设定了一个概率阈值 p ，并得到：

$$\begin{cases} \tilde{x}_i \in \text{TopK}(q_i, k), \\ q_i(\tilde{x}_i) > p, \end{cases}$$

在分布 q_i 中 \tilde{x}_i 的概率由 $q_i(\tilde{x}_i)$ 表示，只有当 \tilde{x}_i 满足等式中的两个条件时，才会被接受到 q_i 分布中。这种放松验证提升了草稿令牌的接受概率，通过将最高概率放松到前 k 个概率值，有效地通过设定概率阈值 p 防止了发散生成。

Experiment

我们的实验在两个公共数据集上进行，分别是MovieLens-10M（简称ML-10M）和Amazon-Books。此外，我们模拟了流行为，并将用户的过往行为分为两个部分，旧历史和新历史。所有内容都随机分为两组，大小相等：一组作为现有内容，另一组作为新引入的内容。为了构建检索池，我们首先使用LLMs为用户的历史记录和现有内容生成旧知识，采用自回归解码。然后，在新历史和新内容上进行加速和下游任务的实验。

Overall Performance

加速性能和下游性能是我们需要研究的两个关键方面。首先，我们在四个基于LLM的推荐框架下，将DARE与简单的基于检索的推测解码（ReSD）在两个任务（用户和物品知识生成）中进行比较。接下来，我们利用DARE和ReSD为所有新用户历史和物品生成知识，并根据不同框架的设计将这些知识适配到DIN和DCNv2中。

Frame-work	Speedup Method	ML-10M								Amazon-Books							
		Speedup Performance				Downstream Performance				Speedup Performance				Downstream Performance			
		User Task		Item Task		DIN		DCNv2		User Task		Item Task		DIN		DCNv2	
		Gen. Speed	Speedup	Gen. Speed	Speedup	AUC	LL	AUC	LL	Gen. Speed	Speedup	Gen. Speed	Speedup	AUC	LL	AUC	LL
	base	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/
KAR	ReSD	94.8	2.53x	82.8	2.21x	0.8351	0.3469	0.8319	0.3500	64.3	1.71x	75.2	2.01x	0.8360	0.4962	0.8308	0.5000
	DARE	171.3	4.58x	107.3	2.87x	0.8349	0.3474	0.8318	0.3500	178.9	4.77x	123.0	3.28x	0.8358	0.4965	0.8306	0.4996
TRAWL	ReSD	70.9	1.90x	81.4	2.18x	0.8338	0.3485	0.8314	0.3506	82.2	2.19x	76.9	2.05x	0.8311	0.4997	0.8301	0.5005
	DARE	164.9	4.41x	100.5	2.69x	0.8336	0.3485	0.8314	0.3506	144.6	3.86x	120.2	3.21x	0.8311	0.4998	0.8300	0.5005
ONCE	ReSD	68.9	1.84x	66.3	1.77x	0.8321	0.3511	0.8283	0.3537	71.7	1.91x	60.1	1.60x	0.8337	0.4952	0.8289	0.5016
	DARE	154.9	4.14x	80.2	2.14x	0.8319	0.3511	0.8286	0.3529	184.5	4.92x	100.1	2.67x	0.8332	0.4956	0.8285	0.5017
RLMRec	ReSD	62.0	1.66x	85.2	2.28x	0.8301	0.3516	0.8281	0.3534	61.0	1.63x	55.3	1.50x	0.8325	0.4954	0.8285	0.5064
	DARE	152.8	4.09x	113.7	3.04x	0.8301	0.3515	0.8282	0.3537	150.5	4.01x	116.8	3.11x	0.8380	0.4903	0.8327	0.4962

(i) DARE在不同框架和任务中，一直在加速方面优于ReSD。例如，在Amazon-Books上的KAR用户知识生成任务中，DARE的加速比为**4.77x**，而ReSD为**1.71x**，提升了178%。这表明DARE的两项优化显著提升了推荐知识生成的速度。

(ii) 用户知识生成的加速效果比物品知识生成更显著，DARE在用户端的提升更大。DARE在用户端的加速比为3.86x-4.92x，而在物品端为2.14x-3.28x。这可能是由于用户偏好的连续性，导致新旧用户知识之间的相似性更高。

从下游性能来看，我们得出以下结论：


(i) LLM生成的知识显著提升了下游任务的性能，提升幅度因框架和数据集而异。例如，在ML-10M上，KAR生成的知识使DIN的AUC提升了2.3%。

(ii) 在不同框架、数据集和CTR模型中，DARE和ReSD生成的知识在下游任务中的性能差异很小。这表明DARE在显著加速知识生成的同时，能够保持下游任务的性能。

原文《A Decoding Acceleration Framework for Industrial Deployable LLM-based Recommender Systems》

发布于 2024-09-11 14:46 · IP 属地北京


LLM 华为 推荐系统



理性发言，友善互动

1 条评论

默认 最新

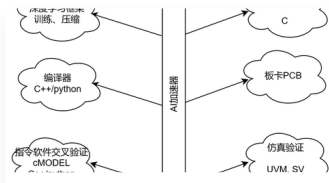
**猫的薛定谔**

盲猜就知道是交大诺亚那帮人

09-11 · 四川

回复 喜欢

推荐阅读



AI加速的前前后后

moon 发表于AI加速

众多的AI加速架构，它们都有什么区别？

因为（xx）两个不同架构项目的关系，不断有朋友问我，怎么看AI芯片和通用计算芯片在架构上的差异和优缺点？其实这个话题有点“旧”了，这几年中，既有学术界专家的比较正式的文章回答，也...

周永权 发表于GPU架构...



从MLPerf谈起：如何引领AI加速器的下一波浪潮

OneFl... 发表于前沿技术



深度学习的芯片加速器

黄浴