大语言模型的原生位置无关缓存研究

摘要： 大语言模型（Large Language Model，LLM）的键值缓存（KV cache）基于前缀匹配，这个限制使得在检索增强生成（Retrieval-Augmented Generation，RAG）的场景下，模型推理十分低效。因为检索回来的多个文档以任意可能的顺序排列，排列在后面的文档的KV cache因其前缀不匹配而不能被复用，只能被重新计算。为了实现不受位置约束的KV cache复用，位置无关缓存（Position-Independent Caching，PIC）已被提出以加速大模型的推理过程；然而，现有方法往往导致显著的模型输出精度下降，限制了其实际应用。为解决这一问题，本文提出了原生PIC方案：重新引入Transformer的编码器（Encoder）到主流的仅解码器（Decoder-only）LLM中，并对Encoder进行训练以支持PIC。我们进一步开发了COMB：一个与现有推理框架（如vLLM和SGLang）集成的PIC缓存系统。实验结果表明，COMB将首字延迟降低了51-94%，吞吐量提升了3倍，同时保持准确度不下降。此外，在模型DeepSeek-V2-Lite-Chat上的准确度提升揭示了COMB对其他结构Decoder-only LLM的适用性。COMB系统的代码以及模型参数已经开源在下方链接：
https://github.com/shijuzhao/Comb

关键词： 大语言模型，位置无关缓存，编码器-解码器 Transformer

Abstract: The Key-Value (KV) cache of Large Language Models (LLMs) is prefix-based, making it highly inefficient for processing contexts retrieved in arbitrary order. Position-Independent Caching (PIC) has been proposed to enable KV reuse without positional constraints; however, existing approaches often incur substantial accuracy degradation, limiting their practical adoption. To address this issue, we propose native PIC by reintroducing the encoder to prevalent decoder-only LLMs and explicitly training it to support PIC. We further develop COMB, a PIC-aware caching system that integrates seamlessly with existing inference frameworks. Experimental results show that COMB reduces Time-to-First-Token (TTFT) by 51-94% and increases throughput by 3× with comparable accuracy. Furthermore, the quality improvement when using DeepSeek-V2-Lite-Chat demonstrates the applicability of COMB to other types of decoder-only LLMs. Our code is available at https://github.com/shijuzhao/Comb.

Keywords: LLM, PIC, encoder-decoder transformer

一、研究背景及意义

大语言模型（Large Language Model, LLM）在广泛的复杂任务（如自动软件工程、文档问答）中展现出强大的能力。它通过基于自然语言的提示词（即词元（Token）序列）提供了用户友好的交互界面。随着 LLM 能力的提升和易用性的增强，其使用模式已从简单的对话任务转变为更复杂的场景，如多轮规划、推理、工具使用和少样本学习（Few-shot Learning）。这种转变导致了跨请求的提示词中存在大量重复 Token，例如系统提示词、少样本学习示例和文档，这些内容的变化频率较低（即静态），而用户特定的指令则变化频繁（即动态）。

基于前缀的上下文缓存（前缀缓存）是一种高效计算的方法，它复用先前请求中重复 Token 的中间表示——键值（KV）向量，以减少计算量。前缀缓存将当前请求与先前请求进行匹配，以复用最长公共前缀的 KV 向量。前缀缓存仍是现有系统中的主流方法 [1] [2] [3] [4]，但它要求跨请求的前缀完全匹配，这限制了其在少样本学习和检索增强生成（Retrieval-Augmented Generation, RAG）等场景中的复用能力——在这些场景中，静态块（如文档）在跨请求时保持不变，但其前面会有不同的前缀。

为解决前缀缓存的局限性，位置无关缓存（Position-Independent Caching, PIC） [5] [6]实现了静态 Token 的 KV 向量的模块化复用，不受其前缀的影响。如图 1 所示，PIC 技术是一个类似于位置无关代码（Position-Independent Code）的**编译**和**链接**的两阶段框架。首先，**编译**阶段将各个静态块提交给 LLM，从位置零开始生成并存储其各自的 KV 向量。其次，**链接**阶段按需加载并拼接 KV 向量（以任意顺序/位置），并可选择性地进行重计算以恢复精度。PIC 显著增加了复用机会，但它偏离了标准的注意力机制，可能导致精度下降；因此，恢复精度成为其主要挑战。

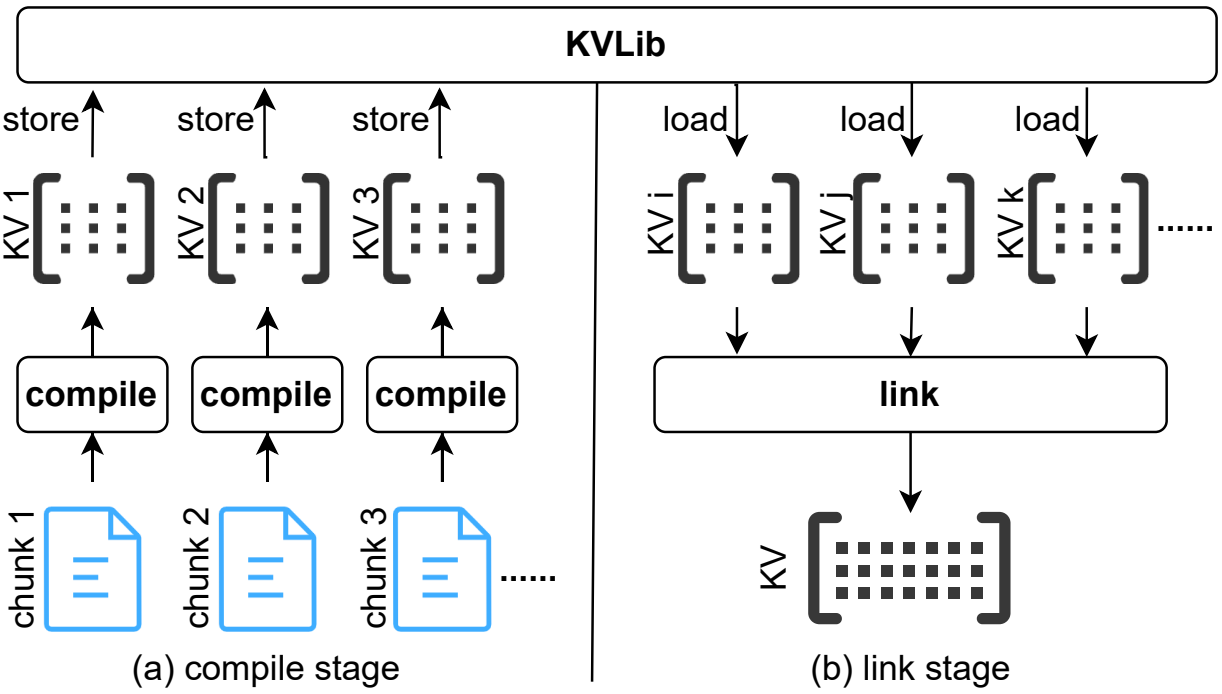


图 1 PIC 的编译链接过程。其中图右侧的$KV\ i, KV\ j, KV\ k$代表可以在任意位置复用的 KV 向量。

本文的后续部分组织如下：首先简述相关工作、介绍研究成果，接下来详细介绍研究方法（COMB 系统设计和编码器插件设计），最后展示实验结果、讨论未来工作。

二、相关工作

PIC 研究可分为两种范式：**无需训练的 PIC** 和**基于微调的 PIC**。

无需训练的 PIC [5] [6] [7] [8] [9] [10]通常在链接阶段重新计算一部分 Token 以恢复精度。例如，对每个静态块，CacheBlend [5]比较第一层的新计算出的 KV 向量和先前存储的 KV 向量，从中挑选出差异最大的$r\%$个（如 20%个）Token，在后续 Transformer 层的计算中重计算这$r\%$ Token 的 KV 向量，复用其它 Token 的 KV 向量。EPIC [6]发现新旧 KV 向量之间有差异主要是因为注意力锚点（Attention Sink）现象，它重计算每个静态块的前N个（如 64 个）Token 来移除 Attention Sink 对 KV 向量的影响。这些方法是非侵入性的（它们不训练模型或修改模型架构），可以作为插件部署，但存在精度降低的问题。这是因为未被重计算的 Token 的 KV 向量仍有差别，会影响 LLM 正确输出结果。

基于微调的 PIC [11] [12] [13]通过在编译阶段显式训练模型以感知 PIC 的使用，从而在链接阶段实现接近零的开销。例如，BlockAttention [12]将标准的因果注意力掩码（Causal Attention Mask）改成分块因果注意力掩码，并对新的掩码输入进行微调。KVLink [13]引入了特殊的链接 Token，并微调模型以识别多个独立的静态块。这些方法实现了高精度，但它们需要微调，因此会永久改变模型行为，可能导致灾难性遗忘和非 RAG 任务上的性能下降。

三、研究成果概述

为解决这些局限性并结合两种范式的优势，本文提出了 COMB，它重新将经典 Transformer 的编码器（Encoder）[14]引入主流的仅解码器（Decoder-only）LLM 中，并仅训练编码器以支持 PIC。该设计具有三个关键优势。首先，与无需训练的方法相比，COMB 通过集成 PIC 专用组件——编码器——并对其进行显式的 PIC 生成训练，从而实现了最高精度，使 PIC 成为模型的原生能力，类似于 DeepSeek 中的原生稀疏注意力 [15]。其次，与基于微调的方法相比，COMB 提供了最大的灵活性：编码器作为插件运行，其交叉注意力可以被完全移除而不影响标准解码流程。第三，与所有方法相比，尽管引入了额外的参数（来自编码器），但 COMB 实现了最低的首字延迟（Time-to-First-Token, TTFT），这得益于 COMB 编码器-解码器架构的高效性。

本研究实现了 COMB 的两个组件：模型组件和系统组件。对于模型组件，我们将编码器重新引入标准的仅解码器架构中，冻结解码器参数，并显式训练编码器以适应 PIC。输入块由编码器独立处理以生成其 KV 向量，并使用最大似然目标对这些块上的查询进行监督，以匹配真实输出。在架构上，编码器层与解码器层以梳状方式交错排列，且仅执行交叉注意力，这启发了 COMB 的命名。对于系统组件，我们在现有推理框架（包括 HuggingFace transformers 和 vLLM [3]）之上实现了 PIC 管理系统，包含约 5 千行 Python 代码。本文的训练代码、系统代码和模型权重已公开发布。

本文选取了两个主流开源模型来验证提出的设计方案：Llama-3.1-8B-Instruct [16]和 DeepSeek-V2-Lite-Chat [17]。Llama 系列模型在当前的 LLM 研究中最为常用；而 DeepSeek 模型采用了新颖的多头潜在注意力（Multi-head Latent Attention, MLA）结构。基于这两个模型的实验展现了 COMB 的普适性。LongBench 数据集 [18]上的实验结果表明，COMB 在所有方法中实现了最高的精度和最低的首字延迟。当发生位置无关缓存命中时，COMB 将首字延迟降低最高达 94%，吞吐量提升最高达 3 倍，同时准确度与前缀缓存相当或更优。重要的是，这些收益是在不永久修改底层仅解码器模型的情况下实现的：COMB 仍是一种插件机制，可根据需求启用或禁用，在未使用 PIC 时恢复为标准的前缀缓存。

四、原生位置无关缓存的设计

本章首先介绍 COMB 系统的工作流程，再介绍如何将 PIC 植入为 LLM 的原生能力。

4.1. COMB 系统设计

COMB 的输入包含一个问题（Question）以及可能没有或多个静态[1]上下文（Context），这些 Context 期望在任意位置被复用。我们让用户自行区分 Context 和问题，因为用户最了解自己的需求。如图 2 所示，① 对于多个 Context，COMB 通过哈希表查找其位置无关缓存（PICache[2]）是否存在。② 对于没有缓存的 Context，块处理器（Chunk Processor）将生成其 KV 缓存（Cache），③ 存储 KV 缓存，并 ④ 更新哈希表。随后，⑤ 这些 Context 的 PICache 被获取并 ⑥ 传输至 LLM 推理引擎（如 HuggingFace transformers、vLLM 或 SGLang）。⑦ 问题被直接传递给 LLM 推理引擎，并 ⑧ 与 PICache 一起得到 LLM 的应答。图中右下角的蓝色和橙色方块分别代表 LLM 的自注意力层和交叉注意力层，将在下一节详细介绍。值得注意的是，COMB 可以无缝集成到现有的分离式预填充（Prefill）-解码（Decode）服务范式中。块处理器可视为预填充节点，而 LLM 推理引擎可视为解码节点。

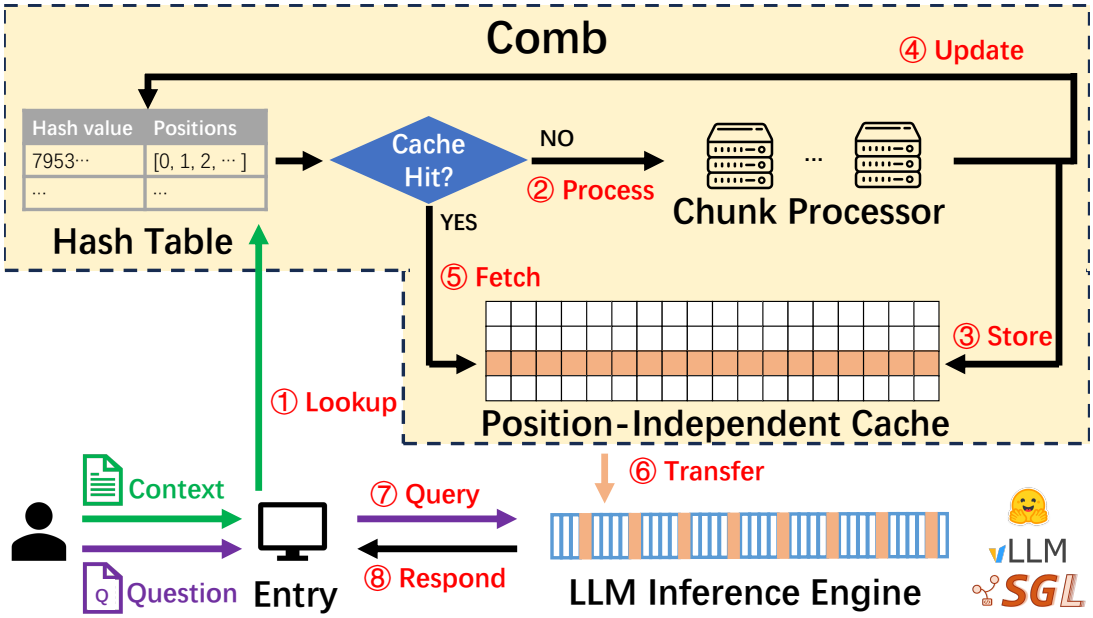


图 2 COMB 的工作流程。

4.2. 编码器插件设计

如图 3 所示，COMB 模型与经典的编码器-解码器 Transformer 类似，但在设计理念上有所不同。编码器用来生成静态块的 PICache，而解码器负责生成回答。在解码阶段，新生成的

[1] 静态指文本内容不会发生改变。

[2] 本文使用术语 PIC 指代位置无关缓存技术，使用术语 PICache 指代位置无关的 KV 向量。

Token 会被追加到解码器的输入之后，继续预测下一个 Token。解码器直接从现有的开源 LLM 得到，处理用户问题（Query）和生成的新 Token。编码器的结构与编码器类似，但层数更少，它的嵌入层（Embedding）[3]从解码器继承得到。例如图中展示了 8 层编码器层与 32 层解码器层交错排列。这里设置编码器的层数为 8 是为了平衡模型训练效率和输出精度：更多的层数会提升模型的能力，但受到硬件资源限制，我们必须减少参数量来避免图形处理器（Graphics Processing Unit, GPU）的内存过载（Out-of-Memory, OOM）。值得注意的是，解码器的所有参数在训练时会被冻结，所以在推理时如果没有编码器的输入，就不会执行交叉注意力层（Cross-Attention Layer），模型的输出就会和原来的仅解码器模型的输出相同。

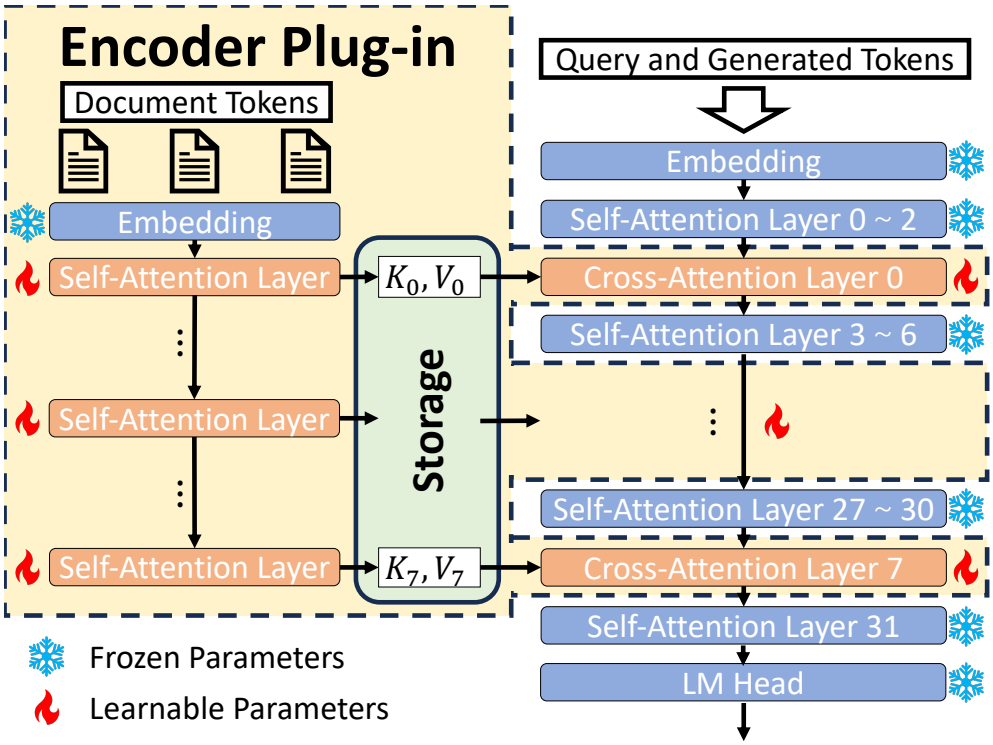


图 3 COMB 的模型结构。

这个模型主要由两个组件构成：自注意力层（Self-Attention Layer）和交叉注意力层。编码器和解码器的自注意力层与当前 LLM 的注意力层结构相同，其中编码器的自注意力层仅处理静态块，而解码器的自注意力层仅处理动态的用户问题（Query）。静态块和动态问题在交叉注意力层中交互。在交叉注意力层中，动态问题生成查询矩阵 Q，与解码器生成的 KV 向量执行注意力计算，以此查询静态块中所需要的信息。这里的 KV 向量就是需要被存储的 PICache，每一个静态块都有相应的 PICache，他们可以被任意拼接供解码器查询。静态块的位置可以由注意力掩码来表达：位置在静态块前面的 Token 不与它做交叉注意力，而位置在它后面的 Token 计算交叉注意力来查询信息。通过将静态块从解码器的输入中分离出来，COMB 模型实现了以存代算和可以在任意位置复用的 KV 缓存。

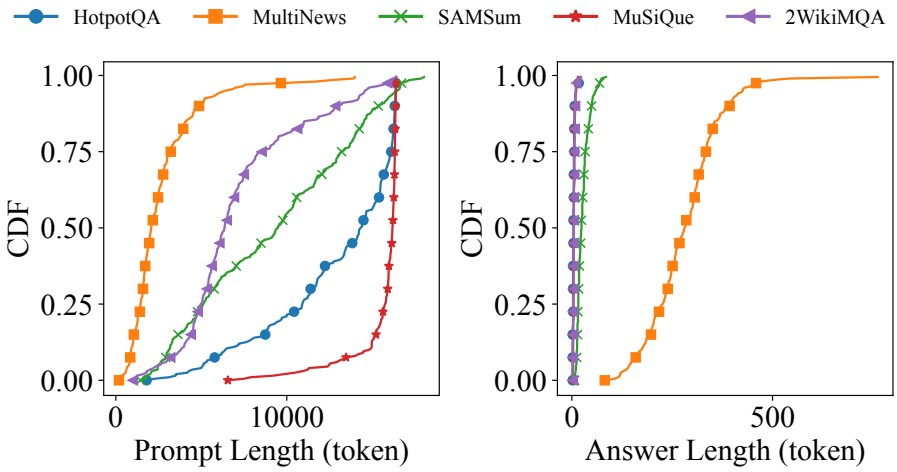
本研究选取 Llama-3.1-8B-Instruct 和 DeepSeek-V2-Lite-Chat 作为解码器进行了训练，训练得到的模型分别被称为 CombLlama 和 CombDeepseek。训练过程使用四张 NVIDIA A100-80GB GPU，并设置张量并行度为 4。训练数据集包括 SQuAD [19]、Natural-Instructions [20]、XSum [21]和 Super-Natural-Instructions [22]。具体的训练配置可以在开源代码中找到。

[3] 嵌入层将 Token 映射为高维空间中的向量，方便 LLM 理解。

五、位置无关缓存的性能

本章展示了对 COMB 和其他先进技术的性能评估。对比算法包括前缀缓存（Prefix caching）、CacheBlend [5]、EPIC [6]和 BlockAttention [12]。前缀缓存是标准的全量注意力机制；CacheBlend 和 EPIC 是无需训练的 PIC 方法；BlockAttention 是基于微调的 PIC 方法。CacheBlend 的重计算率被设置为 20%，而 EPIC 的重计算 Token 数被设置为 32。所有的实验都在一个拥有四张 NVIDIA A100-80GB GPU 的服务器上完成。服务器有 128 核 Intel(R) Xeon(R) Platinum 8358P CPU@2.6GHz 和 1 TB DRAM。所有算法使用的推理引擎均为 vLLM 0.12.0。更多实验配置可以参考开源代码。

实验使用的数据集包含 MuSiQue（长文本问答）、SAMSum（少样本学习）、MultiNews（多文档总结）、HotpotQA 和 2WikiMQA（多文档问答）。每个数据集都包含 200 个测试例，其输入长度分布和回答长度分布如图 4 所示。其中对问答类任务包括 MuSiQue、HotpotQA 和 2WikiMQA 使用 F1 分数 [18]（越高越好）来评估回答质量；对总结类任务包括 SAMSum 和 MultiNews 使用 Rouge-L 分数 [23]（越高越好）来评估。F1 分数通过 Token 的查准率和查全率测量模型回答与标准答案之间的相似度；Rouge-L 通过最长公共子序列来评估较长的模型回答与标准答案之间的相似度。本章使用首字延迟 [3]（越低越好）来评估所有算法的推理速度，KV 缓存复用的主要目的就是减少首字延迟。此外，吞吐量（每秒处理的 Token 数，越高越好）可以评估所有算法在高并发请求到来时的性能表现。



(b) 输入长度分布　　(a) 回答长度分布

图 4 不同数据集的长度分布。

5.1. PIC 方法的准确度

本节探讨当启用 PIC 时，COMB 是否保持了底层仅解码器模型的生成质量。图 5(b)和图 5(c)报告了 Llama-3.1-8B-Instruct 的 F1 和 Rouge-L 分数，图 6(b)和图 6(c)报告了 DeepSeek-V2-Lite-Chat 的相同指标。在所有数据集和两个模型系列中，COMB 在评估的基于 PIC 的方法中达到了较高的准确度。

本文不将绝对精度视为主要贡献，因为比较并非完全受控。具体而言，COMB 被显式训练以感知 PIC，而 EPIC 无需训练，其他对比算法则微调更少的参数、使用更少的步数。相反，这些结果证实了本文的核心主张：在 PIC 使用场景中，将 PIC 感知组件纳入模型架构并相应地进行训练，使模型能够恢复——并在某些情况下超越——标准基于前缀注意力的精度。COMB

在多个数据集以及两个架构截然不同的模型家族（Llama 和基于 MLA 的 DeepSeek）上的稳定表现，为 PIC 感知训练的有效性提供了经验证据。值得注意的是，CombDeepseek 比其底层解码器 DeepSeek-V2-Lite-Chat 拥有显著更高的精度。这是因为基础 DeepSeek 模型的输出质量较低；因此本研究在训练期间使用 Llama 生成的高质量数据作为监督，带来了显著的精度提升。这也说明了训练数据的质量对模型输出精度有很大影响。

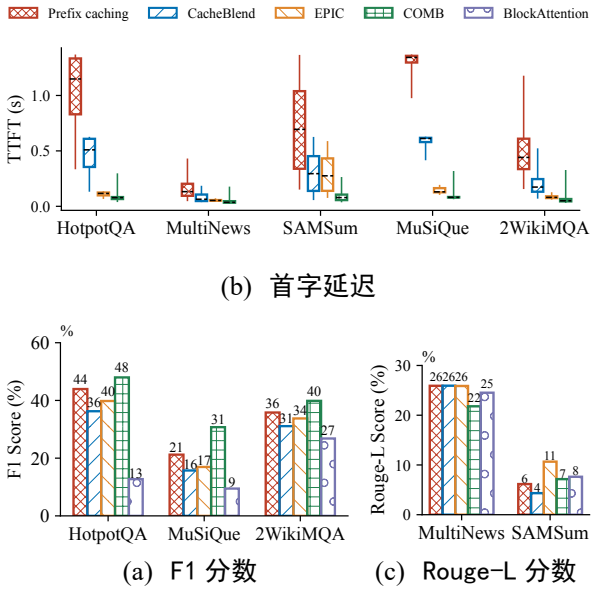


图 5 Llama-3.1-8B-Instruct 模型上的首字延迟和准确度对比。其中缺失 BlockAttention 的首字延迟是因为它没有被集成到 vLLM 中。

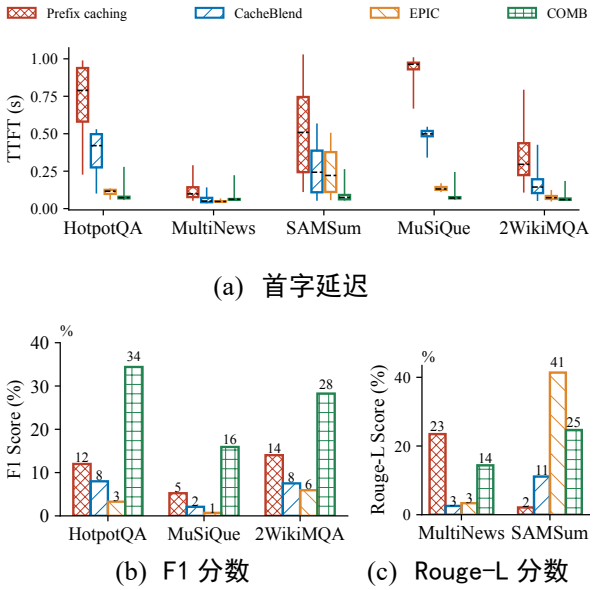


图 6 DeepSeek-V2-Lite-Chat 模型上的首字延迟和准确度对比。其中缺失 BlockAttention 的数据是因为它没有在该模型上训练过。

5.2 COMB 的非侵入性

本节强调 COMB 是非侵入性的。在生产环境中，编码器可根据需求启用或禁用。如果用户不想使用 PIC 功能，可以将所有文本放入解码器的输入，不改变模型结果。因为当编码器被禁用时，COMB 完全退化为底层仅解码器模型，恢复其原始的前缀缓存行为和精度，没有任何退化（正如图 7 中所示的前缀缓存）。

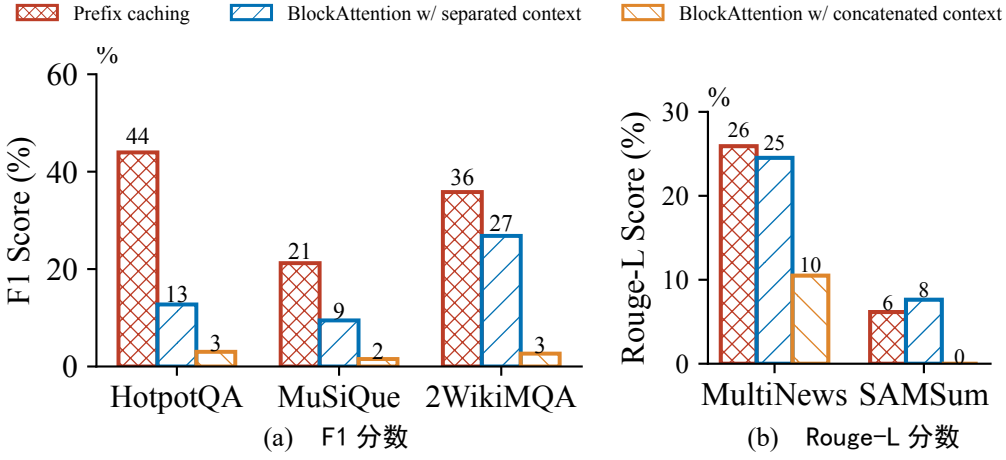


图 7 使用分块的上下文作为 BlockAttention 的输入和不使用之间的准确度对比。

相比之下，BlockAttention 等基于微调的方法本质上是侵入性的。当不需要 PIC 时，如果将多个文档直接输入此类模型，精度可能会崩溃。原因是 BlockAttention 被显式训练为将注意力限制在预定义的文档块内。如果用户不将输入分块并按其预期方式应用块注意力，模型的假设就会被违反，导致严重的性能下降（图 7(a)和图 7(b)，蓝色柱与橙色柱的对比）。

5.3. 首字延迟

接下来，本节分析 PICache 命中和不命中时的首字延迟。对于测试集中的每个样本，我们首先将其输入推理系统，并记录缓存未命中的首字延迟，如图 8 所示。图中没有 CacheBlend 和 EPIC 的结果是因为当缓存未命中时，它们退化为前缀缓存。当缓存未命中时，COMB 执行冷启动编译以生成新文档的 PICache。即使在这种情况下，首字延迟仍与仅解码器模型相当或更优。因为编码器的层数比解码器的少，且仅应用于静态文档的 Token，因此额外的预填充成本是适当的。

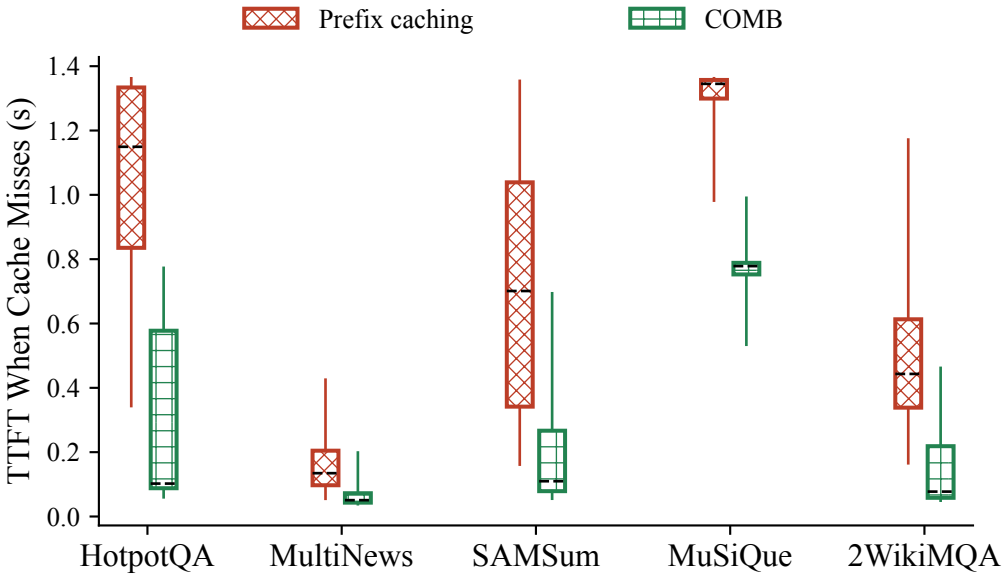


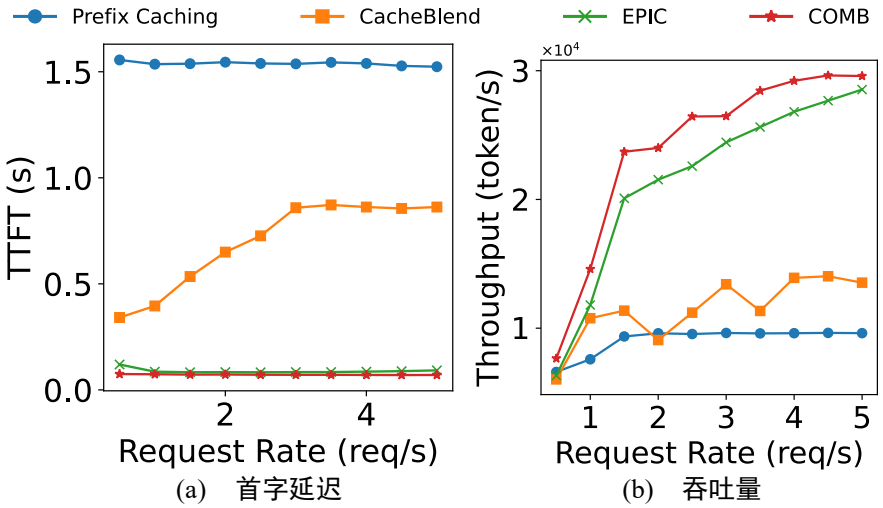
图 8 KV 缓存未命中时的首字延迟对比。使用的模型为 Llama-3.1-8B-Instruct。

然后，我们修改样本的系统指令以模拟另一个具有相同上下文但不同提示词的请求，从而记录缓存命中的首字延迟，如图 5(a)和图 6(a)所示。在缓存命中时的实验结果如预期一致，所有 PIC 方法都显著降低了首字延迟，因为大部分文档的预填充计算被分摊到跨请求中。在现有的 PIC 方法中，EPIC 实现了特别低的首字延迟，因为它在链接阶段仅重新计算每个块边界的 64 个 Token，产生最小的重计算开销。BlockAttention 等训练感知方法提供了准确度优势，但在实践中尚未集成到生产级推理引擎（如 vLLM）中，因此通常仅使用 HuggingFace transformers 后端进行评估。于是，BlockAttention 的首字延迟显著高于其他方法，未在图中展示。

COMB 由于轻量级编码器架构和较低的时间复杂度而实现了最低的首字延迟：每个文档的 PICache 被编译一次并在后续请求中复用，因此当缓存命中时，解码器只需通过交叉注意力将相对较短的 Query 序列与预计算的文档 PICache 耦合。与标准的前缀缓存相比，COMB 将首字延迟降低了 51-94% 。

5.4. 在线服务场景下的延迟和吞吐量

最后，本节评估了在线服务场景下，随着请求到达速率增大的性能变化。图 9(a)和图 9(b)分别报告了首字延迟和吞吐量，随着并发用户数量以及 PICache 所使用的 GPU 显存量的增加。在评估的负载范围内，COMB 保持了最低的首字延迟，同时实现了所有方法中最高的吞吐量。



(a) 首字延迟　(b) 吞吐量

图 9 随着请求到达速率增加的性能变化。

这些收益与跨请求的 PICache 复用密切相关。在标准仅解码器架构中，KV 内存随整个提示词长度（文档长度加 Query 长度）增大，迅速耗尽 GPU 显存并限制并发数。相比之下，COMB 仅在少部分编码器层中为静态文档存储编码器 KV 向量，仅为短得多的查询序列存储解码器 KV 向量。这种轻量级编码器设计减少了每请求占用的 KV 内存，释放了可用于批处理更多请求或容纳更多用户的显存容量。综合上述精度结果，这些发现表明，在真实服务场景中，原生 PIC 技术提供了有利的精度-延迟-吞吐量权衡。

六、讨论

如今我们已经进入了智能体（Agent）的时代。什么是 Agent，它与 LLM 有什么区别？一个普遍认可的说法是 Agent 可以根据长期目标制定短期动作、会调用工具。那么 Agent 一个很重要的能力就是检索。除了需要查阅参考文献来获得信息，Agent 也需要检索应该使用哪个工具更合适。只要涉及到检索，就需要 PIC，因为检索回来的东西在各种排列组合的可能下一定是乱序的。如果使用前缀缓存，基本上只有排在第一个的项目可以复用 KV cache，后面的 KV cache 全都不能复用，这非常低效。我们可以给 Agent 装上一个编码器，将所有检索到的乱序的东西丢进编码器，问题和模型的思考留在解码器，模型会将自己需要的信息输出作为自己的思维链（Chain of Thought）；当模型需要检索新的东西时，就把以前检索到的东西丢掉，把新检索到的东西扔进编码器。想象一下，如果解码器的 128K 上下文里只有问题和模型自己的思维链，没有嘈杂的参考文献的信息，那么这个 Agent 的能力将会有多强。

参考文献

[1]. Deepseek. Deepseek api introduces context caching on disk, cutting prices by an order of magnitude. https://api-docs.deepseek.com/news/news0802, 2024.

[2]. Google. Gemini context caching. https://ai.google.dev/gemini-api/docs/caching, 2026.

[3]. Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

[4]. Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. SGLang: Efficient execution of structured language model programs. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[5]. Jiayi Yao, Hanchen Li, Yuhan Liu, Siddhant Ray, Yihua Cheng, Qizheng Zhang, Kuntai Du, Shan Lu, and Junchen Jiang. Cacheblend: Fast large language model serving for rag with cached knowledge fusion. In *Proceedings of the Twentieth European Conference on Computer Systems*, EuroSys '25, page 94–109, New York, NY, USA, 2025.

[6]. Junhao Hu, Wenrui Huang, Haoyi Wang, Weidong Wang, Tiancheng Hu, Qin Zhang, Hao Feng, Xusheng Chen, Yizhou Shan, and Tao Xie. EPIC: efficient position-independent caching for serving large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, pages 24391–24402, 2025.

[7]. Shubham Agarwal, Sai Sundaresan, Subrata Mitra, Debabrata Mahapatra, Archit Gupta, Rounak Sharma, Nirmal Joshua Kapu, Tong Yu, and Shiv Saini. Cache-craft: Managing chunk-caches for efficient retrieval-augmented generation. *Proc. ACM Manag. Data*, 3(3), (SIGMOD), June 2025.

[8]. Zhisong Zhang, Yan Wang, Xinting Huang, Tianqing Fang, Hongming Zhang, Chenlong Deng, Shuaiyi Li, and Dong Yu. Attention entropy is a key factor: An analysis of parallel context encoding with full-attention-based pre-trained language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9840–9855, Vienna, Austria, July 2025.

[9]. Shiju Zhao, Junhao Hu, Rongxiao Huang, Jiaqi Zheng, and Guihai Chen. MPIC: Position-independent multi-modal context caching system for efficient mllm serving, 2025.

[10]. Jiahao Wang, Weiyu Xie, Mingxing Zhang, Boxing Zhang, Jianwei Dong, Yuening Zhu, Chen Lin, Jinqi Tang, Yaochen Han, Zhiyuan Ai, Xianglin Chen, Yongwei Wu, and Congfeng Jiang. From Prefix Cache to Fusion RAG Cache: Accelerating LLM Inference in Retrieval-Augmented Generation. *Proc. ACM Manag. Data*, 4(1), (SIGMOD), February 2026.

[11]. Songshuo Lu, Hua Wang, Yutian Rong, Zhi Chen, and Yaohua Tang. TurboRAG: Accelerating retrieval-augmented generation with precomputed KV caches for chunked text. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 6588–6601, Suzhou, China, November 2025.

[12]. Dongyang Ma, Yan Wang, and Tian Lan. Block-attention for efficient prefilling. In *The Thirteenth International Conference on Learning Representations*, 2025.

[13]. Jingbo Yang, Bairu Hou, Wei Wei, Yujia Bao, and Shiyu Chang. KVLink: Accelerating large language models via efficient KV cache reuse. In *The Thirty-ninth Annual Conference on*

Neural Information Processing Systems, 2025.

[14]. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, page 6000–6010, Red Hook, NY, USA, 2017.

[15]. Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, Yuxing Wei, Lean Wang, Zhiping Xiao, Yuqing Wang, Chong Ruan, Ming Zhang, Wenfeng Liang, and Wangding Zeng. Native sparse attention: Hardware-aligned and natively trainable sparse attention. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 23078–23097, Vienna, Austria, July 2025.

[16]. Meta. Introducing Llama 3.1: Our most capable models to date. https://ai.meta.com/blog/meta-llama-3-1/, 2024.

[17]. Deepseek. DeepSeek-V2-Lite-Chat. https://huggingface.co/deepseek-ai/DeepSeek-V2-Lite-Chat, 2024.

[18]. Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the Sixty-Second Annual Meeting of the Association for Computational Linguistics*, pages 3119–3137, 2024.

[19]. Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia, July 2018.

[20]. Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. Cross-task generalization via natural language crowdsourcing instructions. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3470–3487, Dublin, Ireland, 2022.

[21]. Shashi Narayan, Shay B. Cohen, and Mirella Lapata. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Brussels, Belgium, October-November 2018.

[22]. Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Atharva Naik, Arjun Ashok, Arut Selvan Dhanasekaran, Anjana Arunkumar, David Stap, Eshaan Pathak, Giannis Karamanolakis, Haizhi Lai, Ishan Purohit, Ishani Mondal, Jacob Anderson, Kirby Kuznia, Krima Doshi, Kuntal Kumar Pal, Maitreya Patel, Mehrad Moradshahi, Mihir Parmar, Mirali Purohit, Neeraj Varshney, Phani Rohitha Kaza, Pulkit Verma, Ravsehaj Singh Puri, Rushang Karia, Savan Doshi, Shailaja Keyur Sampat, Siddhartha Mishra, Sujan Reddy A, Sumanta Patro, Tanay Dixit, and Xudong Shen. Super-naturalinstructions: generalization via declarative instructions on 1600+ tasks. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5085–5109, Abu Dhabi, United Arab Emirates, December 2022.

[23]. Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.