通过个性化实现长期用户兴趣的高效序列推荐

Qiang Zhang, Hanchao Yu, Ivan Ji, Chen Yuan, Yi Zhang, Chihuang Liu, Xiaolong Wang, Christopher E. Lambert, Ren Chen, Chen Kovacs, Xinzhu Bei, Renqin Cai, Rui Li, Lizhu Zhang, Xiangjun Fan, Qunshu Zhang 和 Benyu Zhang*Meta* 推荐系统 *(MRS)*，美国门洛帕克 qiangzhang, hanchaoyu, ivanji, chenyuan, yzha, chihuang,xlwang, chrislambert, renchen, ckovacs, xzbei, renqingcai, ruili, lizhu, maxfan,qunshuzhang,byzhang@meta.com

arXiv:2601.03479v1 [cs.IR] 7 Jan 2026

摘要——**近年来，序列建模、生成式推荐器和大语言模型在推荐领域取得了成功。尽管扩展定律已在序列模型中得到验证，但由于 Transformer 模型的计算复杂度呈非线性（二次方）增长，在推荐等实际应用中，其计算能力利用效率并不理想。为提高序列模型的效率，我们提出了一种全新的序列推荐方法，利用个性化技术提升效率和性能。我们的方法将较长的用户交互历史压缩为可学习令牌，再与近期交互相结合以生成推荐。该方法在保持较高推荐准确率的同时，显著降低了计算成本。我们的方法可应用于现有的基于 Transformer 的推荐模型，例如 HSTU 和 HLLM。在多个序列模型上开展的大量实验表明，该方法具有良好的通用性和有效性。源代码见 https://github.com/facebookresearch/PerSRec。**

索引词—**推荐、大语言模型、个性化、生成式推荐器、序列**

I. 引言

在线服务的快速增长带来了用户生成数据的爆发式增长，使推荐系统越来越难以有效捕捉用户的长期兴趣。传统的序列推荐模型在建模用户行为方面已展现出良好的效果 [38]，但在处理较长的用户交互历史时，往往会面临计算效率低下的问题。在可扩展性和效率至关重要的实际应用中，这一问题尤为突出。

为应对这一挑战，现有工作通常采用两阶段方法：先从较长的历史记录中进行采样或聚类，生成更短的序列，再基于该短序列执行序列推荐[3], [4], [7], [21], [27]。然而，由于采样/聚类过程与序列推荐模型相互脱节，这些方法可能导致性能不够理想。

近期大语言模型（GPT [2], Llama[10], Gemini [29], Claude [1], Deepseek [20], 通义千问 [36]）的进展表明，它们在包括推荐 [6]在内的各种自然语言处理任务中具有巨大潜力。然而，基于 Transformer 的大语言模型的二次复杂度使得它们在序列推荐任务中的计算成本高昂。为缓解这一问题，人们提出了许多方法，例如提高注意力操作的效率 [22], [31], [34]，或将较长的输入序列压缩为可学习令牌 [5], [8], [12], [25]。

基于这些进展，我们提出一种新颖的方法，利用个性化技术提升序列推荐的效率和性能。我们的方法将较长的用户交互历史压缩为可学习的令牌，再将其与近期交互结合以生成推荐。该方法在保持较高推荐准确率的同时，显著降低了计算成本。我们在多个序列模型上验证了该方法的有效性，并在一个大规模数据集上评估了其性能。结果表明，我们的方法优于当前最先进的序列推荐模型，同时实现了显著的计算开销节省。我们工作的主要贡献如下：

- 借助个性化专家实现高效序列推荐：据我们所知，这是首个将长用户交互历史压缩为可学习令牌的框架，在保持高推荐准确率的同时降低计算成本。
- 可扩展性与计算开销节省：与传统序列推荐模型相比，我们的方法显著节省了计算开销，因此适用于实际应用。
- 泛化能力：所提方法已在多种最先进的模型架构中得到验证，例如 HSTU [38] 和 HLLM [6]。

与此代码相关的代码将会开源。

II. 相关工作

A. 序列推荐

自 GRU4Rec [13] 将循环神经网络（RNN）引入推荐系统以来，推荐的序列建模一直是一个活跃的研究课题。这项工作仅将正向互动事件保留在序列中。作为循环模型，循环神经网络难以扩展，这与 Transformer 架构 [30] 形成对比。SASRec [16] 是首个尝试将 Transformer 引入推荐的工作

的研究：它使用交叉熵损失来预测序列中的下一个正样本物品，并从物品集合中随机采样负样本。受 SASRec 启发，研究者在推荐领域探索了多种 Transformer 变体，例如 BERT4Rec [28], 和 S3Rec [41]。最近，HSTU [38], 重新审视了序列推荐任务，并将其在生成式推荐器（GR）框架下重新表述为序列转换任务。结果显示，该模型在扩展至 1.5 万亿参数时仍具有竞争力的性能，并且在推荐领域呈现出与“扩展定律”相似的趋势。沿着这一方向，研究者还探索了多行为生成式推荐器 [23],以及从大型生成式推荐器中进行知识蒸馏 [35]等新课题。

B. 用于推荐的大语言模型

大语言模型（LLM）在通用人工智能（AGI）方面取得了显著进展。GPT4 和 GPT4o [2], [15] 等模型在需要人类水平智能的任务中展现出强大的能力，并涌现出新的能力 [32]。推荐需要对内容和用户进行深入理解，同时具备推理能力，因此被视为大语言模型的重要应用之一 [33]。人们已探索多种范式，以利用大语言模型进行推荐。

用于物品/用户表示的大语言模型。大语言模型的主要功能是自然语言理解，已有多项研究提出利用大语言模型在语言空间中学习物品/用户嵌入。在 NoteLLM 和 NoteLLM2 [39],[40], 中，大语言模型通过提示引导，在 < 物品、物品> 对上进行微调，学习目标是相关物品对的词元嵌入之间的对比损失。输入的物品表示可以是根据内容生成的文本摘要，也可以是多模态 [24]。NoteLLM 仅学习物品嵌入；相比之下，HLLM [6] 提出联合学习物品和用户表示，并采用两个堆叠的大语言模型：物品大语言模型用于提取物品嵌入，用户大语言模型用于理解用户参与序列。用户大语言模型不在文本域中理解用户，而是以物品嵌入序列为输入来表示用户交互历史，并通过下一物品嵌入预测任务或逐点排序等判别任务进行训练。

大语言模型作为推荐器。作为一种通用逼近器，人们已对在成对排序阶段使用大语言模型进行了多种探索。 [14], 中的大量实验表明，在零样本设置下，大语言模型可以直接用作排序器。LlamaRec [37]表明，可以通过基于 verbalizer 的方法对大语言模型进行微调，并将输出 logits 转换为候选项目上的概率分布。在 [11],中，研究者构建了一个全连接图，使大语言模型能够考虑准确性、多样性和公平性等不同方面。

C. 高效序列建模

随着输入长度不断增加，基于 Transformer 架构的大语言模型（LLM）的计算复杂度已成为一个重要瓶颈。这一点在检索增强生成（RAG）、思维链（CoT）和系统提示等应用中尤为明显，因为这些应用需要更长的输入才能达到理想性能。为应对这一挑战，研究人员探索了两种主要方法：（1）提高 Transformer 架构本身的效率；（2）将输入或输入的一部分压缩为更少的词元。高效 Transformer 架构的典型例子包括 Linformer、 [31],环形注意力[22],和注意力汇 [34]。

另外，研究人员也探索了将输入或输入的一部分压缩为更少词元的方法。一些值得关注的方法包括 Gist [25], , 它通过添加要旨令牌并微调大语言模型，将提示压缩为更短的要旨令牌（例如，4 个词元）。ICAE [12] 通过微调编码器，将文本压缩为少量可学习令牌，并使用冻结解码器（预训练大语言模型）还原原始文本。SepLLM [5] 和 AutoCompressor [8] 进一步拓展了这一方法：将输入划分为更短的片段，并依次压缩每个片段。近期一项研究 [9] 评估了这些压缩方法在 MTEB基准多种任务中的效果，发现它们在检索增强生成和长文档问答等任务中表现良好，但在重排序和合成召回任务中的可靠性有限。

III. 所提方法

本文将探讨以下研究问题：

- **RQ1**：我们能否利用个性化专家压缩较长的用户交互历史?
- **RQ2**：随着新事件的出现，压缩后的个性化专家会如何衰减?
- **RQ3**：个性化专家的不同位置如何影响推荐结果?
- **RQ4**：个性化专家捕获了哪些信息?

A. 增加序列长度可提升推荐性能

序列推荐模型旨在根据用户的交互历史（UIH）预测用户接下来会与哪个项目交互。给定 UIH 为 x_0，x_i，$\cdots$，x_n，序列推荐模型 θ 会根据以下内容推荐下一个项目 x^*：

$$x^* \propto p_\theta(x|x_0, x_i, \cdots, x_n) \tag{1}$$

其中，θ 表示模型，x 表示项目的呈现形式，可以是物品ID（HSTU [38]）或物品嵌入（HLLM [6]）。这一过程可以自回归地运行，以推荐更多项目。因此，序列推荐模型可以使用下一物品预测损失进行训练，类似于大语言模型中的下一词元预测。当使用物品嵌入作为项目的表示形式时，我们采用对比损失，将其下一位置的输出嵌入和输入嵌入作为正样本对，并将其他用户/序列的输入嵌入作为负样本对（更多细节见 [6]）。

我们发现，HLLM [6]等序列推荐模型的性能会随着序列长度的增加而提升。

为验证这一观察结果，我们在 MerRec 数据集 [19] 上使用不同的序列长度，对两种最先进的序列推荐模型（HSTU [38] 和 HLLM [6]）进行了训练和性能评估。数据集的详情及处理方式见第 IV-A 节。

接下来 [6], ，我们截取每位用户交互序列中最近的交互，以模拟不同长度的用户交互历史，并将最后一个项目作为检索目标。实验使用了 [6]发布的代码和超参数：HSTU 训练至多 200 轮，HLLM 训练至多 5 轮。

图 1 展示了 HLLM 和 HSTU 在不同序列长度下的结果，结果表明，随着用户交互序列变长，它们的性能（例如 Recall@5）稳步提升。这种扩展性使序列推荐模型在建模用户长期交互方面颇具吸引力。然而，由于这些模型采用了 Transformer 架构，其计算成本会随序列长度呈二次增长（详见第 III-C 节），这给产品中扩展序列长度带来了实际阻碍。

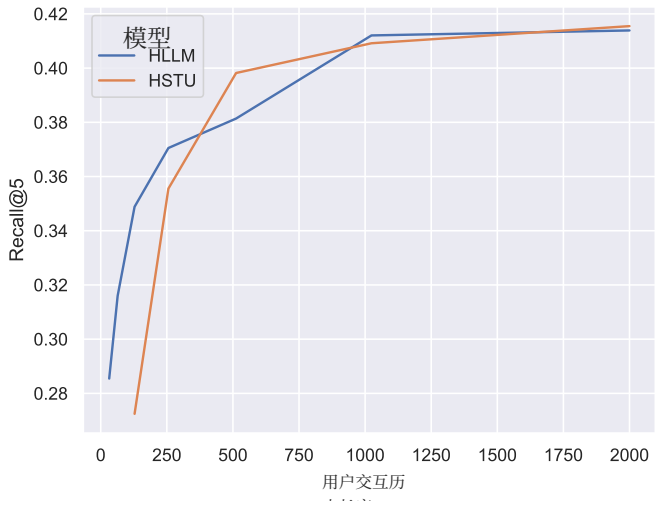

图 1. 在 MerRec 数据集上，随着序列长度从 128 增至 2000，HSTU 和 HLLM 模型的性能（Recall@5）稳步提升。即使序列较短，HLLM 也能取得良好性能；随着序列变长，HSTU 则缩小了差距。

B. 借助个性化专家实现高效扩展

在上一节中，我们展示了 HSTU 和 HLLM 等序列推荐模型能够随着用户交互历史变长而稳步提升推荐性能，但代价是计算成本呈二次增长。给定一段交互 $s_j$as $[x_0^j, x_1^j, \cdots, x_{n_j}^j]$，且 UIH 由多个片段 $[s_0, s_1, \cdots, s_m] = [x_0^0, x_1^0, \cdots, x_{n_0}^0, x_0^1, x_1^1, \cdots, x_{n_m}^m]$组成。片段可以定义为一段会话数据、一天的数据或固定数量的事件，具体取决于应用场景。朴素的序列推荐模型基于完整的 UIH 预测下一个项目，形式如下：

$$x^* \propto p_\theta(x|x_0^0, x_1^0, \cdots, x_{n_0}^0, x_0^1, x_1^1, \cdots, x_{n_m}^m) \quad (2)$$

本文（图2）提出一种方法，使用可学习令牌压缩每个片段，再利用这些可学习令牌预测下一个项目。该方法可表示为：

$$z_{s_j} \propto p_\phi(z|z_{s_0}, z_{s_1}, \cdots, x_{s_{j-1}}; x_0^j, x_1^j, \cdots, x_{n_j}^j; y) \ \forall j \quad (3)$$

$$x^* \propto p_\theta(x|z_{s_0}, z_{s_1}, \cdots, z_{s_{m-1}}, x_0^m, x_1^m, \cdots, x_{n_m}^m) \quad (4)$$

这里 y 是可学习令牌（也可以使用多个可学习令牌），z_{s_j} 是片段 s_j的压缩信息。 ϕ 是压缩模型， θ 是推荐模型，它们可以共享同一组参数，即 $\theta = \phi$。z_{s_j} 仅取决于片段 s_j 中的物品以及先前片段的压缩信息；对新项目 x^* 的预测仅取决于最近片段中的物品以及先前片段的压缩信息。图2也对此进行了说明。鉴于这些可学习令牌能够相应地捕获每个片段的必要信息，我们将这些可学习令牌称为个性化专家。

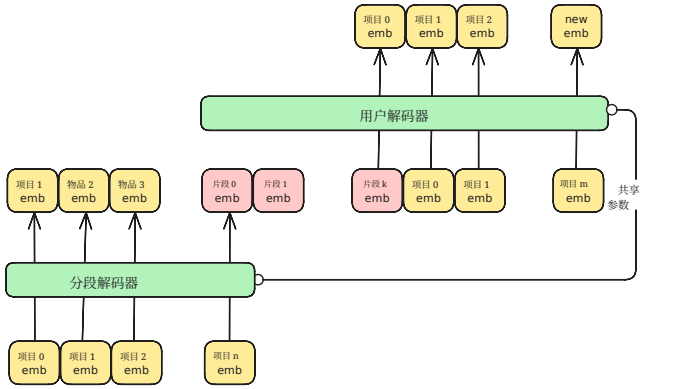

图2. 所提方法架构概览。该模型将长序列划分为多个片段，并利用分段解码器将每个片段“压缩”为分段嵌入。随后，将这些分段嵌入与近期片段中的物品嵌入相结合，以进行序列推荐。这种设计能够显著提高效率。解码器与分段解码器可以共享参数。

1) 训练：所提方法也可以采用下一物品预测作为损失函数进行训练，与原始序列推荐模型类似。通过仔细组织 UIH 并控制注意力掩码，公式 3 中的两个步骤可以合并为一个步骤。为此，我们在每个片段的末尾插入可学习令牌 y ，然后将一个 UIH 中的所有片段展平为一个长序列，得到$[x_0^0, x_1^0, \cdots, x_{n_0}^0, y_0, x_0^1, x_1^1, \cdots, y_1, \cdots, x_{n_m}^m]$。根据公式3，每个项目 x 和可学习令牌 y 都可以关注自身、同一片段中位于其前面的项目，以及前面片段中的所有可学习令牌。如图3所示，这可以通过调整注意力掩码来实现。算法 1 提供了用于生成注意力掩码的 Python 代码，图4展示了一个示例。

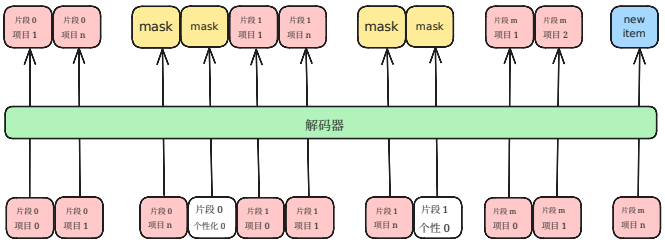

图3。每个项目 x 和可学习令牌 y 都可以关注自身、同一片段中位于其之前的项目，以及前面片段中的所有可学习令牌。黄色表示在训练期间计算损失时被屏蔽的位置（可学习令牌）。请注意，为避免图示过于拥挤，此处未显示表示词元关注其他位置的箭头。

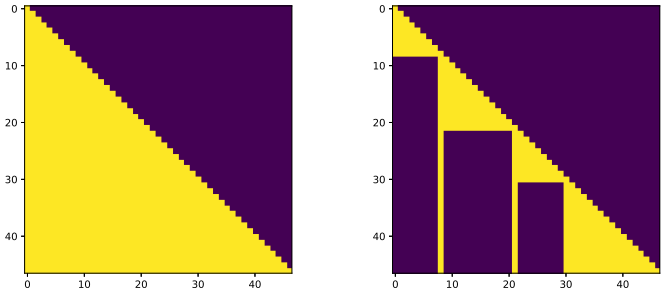

图4。注意力掩码示意图。第 i 行、第 j 列为黄色表示位置 i 可以关注位置 j。左：HSTU 和 HLLM 用于下一物品预测的常规因果掩码；右：经过修改的注意力掩码，用于阻止一个片段中的项目关注其他片段。此处以包含四个片段的 UIH 为例，每个片段的长度相应为$[8, 12, 8, 16]$，并且在每个片段之后（最后一个片段除外）追加一个可学习令牌。

清单 1. 用于创建注意力掩码的 Python 代码 def c r e a t e a t t e n t i o n m a s k (片段长度 : L i s t [i n t] , 可学习数量 : L i s t [i n t]) −> t o r c h . Tensor : # 首先创建因果掩码 u i h 长度 =sum(片段长度) +sum(个人数量) 掩码 =t o r c h . ones ((ui h 长度 , u i h 长度)) 掩码 =t o r c h . t r i l (掩码) # 将因果掩码改为片段偏移量 = 0for i in range (len (片段长度)) : x 偏移量 = 0for j in range (i) : 掩码 [y 偏移量 : 片段长度 [i] + 个人数量 [i] +y 偏移量 , x 偏移量 : x 偏移量 +片段长度 [j]] = 0x 偏移量 += 个人数量 [j] +片段长度 [j]y 偏移量 += 个人数量 [i] +片段长度 [i] return 掩码

训练时采用下一物品预测，而与可学习令牌对应的位置不参与该损失的计算（图3中的黄色方框）。

2) 推理： 在推理过程中，片段会逐个处理，而不是将所有片段展平为完整的 UIH，从而降低推理成本（将在第III-C节中进行分析）。预测新项目（公式3）只需使用这些可学习令牌的激活值，这不仅降低了计算成本，也减少了内存使用量。这可以通过缓存所有先前片段中可学习令牌的键值缓存来实现，这种方法通常用于加速大语言模型的推理 [18], [26]。

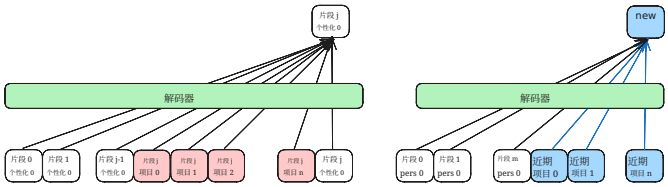

图5. 在推理时，我们首先为每个片段生成并保存可学习令牌的激活值；随后，这些激活值将作为键值缓存，用于新项目预测。

3) 更短的历史记录和冷启动：所提方法 用可学习令牌压缩长历史中的每个片段。对于历史较短或处于冷启动状态的用户，其历史可能只包含一个片段（最后一个片段），因此不会使用可学习令牌。上述训练和推理方法仍可应用于此处。

C. 复杂度分析

图 6 展示了 Transformer 解码器层的示意图。对于具有 L 个注意力层、d 个内部维度且输入序列长度为 n 的 Transformer，其训练和推理的计算复杂度均可粗略表示为 $C_b = \mathcal{O}(L(n^2d + nd^2))$。

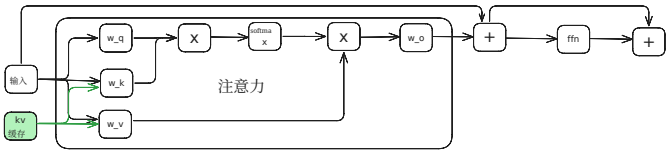

图 6. Transformer 解码器层示意图。它包含注意力层和前馈神经网络（FFN）层。此处忽略归一化层。如果存在键值缓存，则会将其拼接到键和值输入中。

对于所提方法，不失一般性，假设输入序列被均匀划分为 m 个片段，因此每个片段包含 $\frac{n}{m}$ 个项目，并且每个片段都附加 $\frac{k}{m}$ 个可学习令牌（因此总计有 k 个可学习令牌）。总序列长度为 $n + k$，因此所提方法的训练复杂度为 $C_t = \mathcal{O}(L((n+k)^2d + (n+k)d^2))$。训练的计算成本将增加至：

$$\mathbf{S} = \frac{\mathcal{O}(L((n+k)^2d + (n+k)d^2))}{\mathcal{O}(L(n^2d + nd^2))} \tag{5}$$

$$= \frac{\mathcal{O}((1+\alpha)^2 n + (1+\alpha)d)}{\mathcal{O}(n+d)} \tag{6}$$

$$\approx 1 + \alpha \tag{7}$$

其中 $\alpha = \frac{k}{n} \ll 1$ 是可学习令牌与原始长序列的比率。给定 $k \ll n$，所提方法只会给训练带来可忽略的额外开销。

在推理期间，计算片段 j 时只需要该片段中的项目以及其所有前序片段的可学习令牌，因此该片段的序列长度为 $\frac{n}{m}+\frac{k}{m}j$，复杂度为：

$$C_i=\sum_{j=0}^{m}\mathcal{O}(L((\frac{n}{m}+\frac{k}{m}j)^2d+(\frac{n}{m}+\frac{k}{m}j)d^2)) \quad (8)$$

$$=\mathcal{O}(L(\frac{(n+k)^2}{m}d+(n+k)d^2)) \quad (9)$$

与将所有片段展平为一个长序列的基线相比，所提方法在推理时可节省的计算成本为：

$$\mathbf{S}=\frac{\mathcal{O}(L(\frac{(n+k)^2}{m}d+(n+k)d^2))}{\mathcal{O}(L(n^2d+nd^2))} \quad (10)$$

$$=\frac{\mathcal{O}(\frac{(1+\frac{k}{n})^2}{m}n^2d+(1+\frac{k}{n})nd^2}{\mathcal{O}(L(n^2d+nd^2))} \quad (11)$$

$$=\frac{\mathcal{O}(\frac{(1+\alpha)^2}{m}n+(1+\alpha)d)}{\mathcal{O}(n+d)} \quad (12)$$

其中，$\alpha=\frac{k}{n}$ 和 m 是 UIH 中的片段数。已知所有片段中的可学习令牌总数为 k，因此 $m\leq k-1$。在我们使用 HSTU 的实验中：$n=1280$、$m=5$、$k=4$ 和 $d=64$，我们得到 $\mathbf{S}\approx 0.238$，也就是约四分之一的计算成本。第 IV-B1 节比较了实验期间的计算成本。此外，由于片段压缩带来的开销可以摊销，自回归生成更多项目时还能获得更高的计算比率。

为实现这一计算节省，我们只需缓存可学习令牌的键和值张量的激活值；这会为每个 UIH 额外占用 $\mathcal{O}(2Lkd)$ 的空间。

IV. 实验

本节报告了回答上述四个研究问题的结果。所有实验均使用配备 8 个 H100 GPU 的单节点机器。我们使用 HLLM 中提供的相同超参数。

A. 数据集

为评估所提方法的效果和效率，我们使用 MerRec 数据集 [19] 和 Ekstra Bladet 新闻推荐数据集（EB-NeRD） [17]。

1）MerRec 数据集：MerRec 数据集是来自 Mercari（这一 C2C 电商平台电子商务平台）的大规模、高度多样化的物品交互事件序列数据子集。该数据集的一大优势是规模庞大，且包含较长的交互序列。相比之下，Amazon Books 和 Pixel8M 数据集中的用户平均分别只有 2.8 次和 17.8 次交互；而在 MerRec 数据集中，用户平均有 288.9 次交互，并且有 119756 名用户至少有 2000 次交互。这使 MerRec 数据集非常适合用于衡量序列推荐模型在更长的

表 I MerRec 数据集统计信息

特征	不同值数量	描述
用户 ID	5,569,367	全局唯一用户账户 ID。
序列 ID	69,144,727	用户级唯一序列 ID。
会话 ID	227,167,616	用户级唯一会话 ID。
事件 ID	1,274,814,848	动作事件 ID。
产品 ID	1,403,098	品牌 ID 与 c2 ID 的拼接。
品牌名称	1,554,523,806 至-kens	商品品牌的文本标签。
品牌 ID	20,001	商品品牌的 ID。
c2_名称	1,989,892,371 到-kens	项目的 c2级类别文本标签类别。
c2 ID	3073	项目的 c2级类别 ID。
序列 2000	119,756	包含 ≥的用户级唯一序列 2000 个事件。

表 II MERREC 数据集中的样本数据。某些字段可能为空。更多信息请参阅 HuggingFace。

c2_名称	品牌名称	事件类型
Indie	Face	商品浏览
耳环	Chanel	商品点赞
黑胶唱片	好莱坞唱片	购买_开始

用户交互序列下的扩展能力。该数据集的一些关键统计数据如表 I 所示：

用户可以通过六种事件类型与产品进行交互：点击、点赞、加入购物车、出价、发起交易和完成交易。在实验中，我们只考虑至少有 2000 次交互的用户，并移除这些用户从未交互过的产品。最终得到 119756 个用户/用户交互序列和 1255665 种独特产品。我们为选定的用户和产品重新生成用户 ID 和产品 ID。实验中只使用每位用户最近的 2000 次交互。对于 HLLM 使用的物品描述，我们采用 c2 名称和品牌名称，与产品 ID 的定义保持一致。表 II 展示了一些示例。

2) EB-NeRD：Ekstra Bladet 新闻推荐数据集（EB-NeRD）是一个用于新闻推荐系统的综合数据集。该数据集收集自知名丹麦报纸 Ekstra Bladet 的用户行为日志，为分析用户与新闻文章的交互提供了丰富的数据来源。具体而言，EB-NeRD 包含超过 100 万名独特用户，产生了逾 3700 万条曝光日志和 2.51 亿次交互。此外，该数据集还包含超过 125000 篇新闻文章，每篇文章都附有标题、摘要和正文等文本内容特征。这一宝贵资源支持对基于文本的特征进行探索，为推荐系统研究带来了新的机遇。

约有 44968 位用户至少有 512 次交互（点击）。在我们的实验中，只使用这些用户。本实验仅使用这些用户最近的 512 次交互。对于 HLLM 使用的物品描述，我们采用

文章的标题、副标题和类别字符串。

B. 通过个性化专家压缩 *UIH*（*RQ1*）

在本节中，我们将所提方法与 SASrec、HSTU 和 HLLM 进行了比较，以评估其性能。我们在 HSTU 和 HLLM 上实现了所提方法，以展示其对不同基于 Transformer 的序列推荐模型的适用性。对于 HLLM，我们使用 Llama 3.2 1B[10]（16 层，嵌入维度为 2048）作为项目大语言模型和用户大语言模型。对于 HSTU，我们使用 16 层和 64 的嵌入维度。超参数方面，我们遵循 HLLM [6] 中的实验设置：HLLM 的学习率为 $=1e-4$，HSTU/SASrec 的学习率为 $1e-3$；HLLM 的权重衰减为 $=0.01$，HSTU/SASrec 的权重衰减为 0.1；HLLM 的批量大小为 $=3$，HSTU/SASrec 的批量大小为 8。

对于，用户交互序列被划分为训练集（前1280个事件）和测试集（最后720个事件）。对于所提方法，训练序列进一步划分为两个片段：预训练片段（前1024个事件）和近期片段（剩余的256个事件）。对于基线方法，我们要么使用完整的训练序列（1280个事件），要么仅使用训练序列的近期片段（256个事件）。其他划分方法将在第IV-D节中研究和比较。按照 [6], ，我们使用召回率@k和归一化折损累计增益@k（归一化折损累计增益）衡量检索准确性，并将用户交互序列中的最后一个项目作为目标。

对于比较而言，与我们最相关的工作是Kuaiformer[21],，它将输入序列划分为早期、中期和近期片段，然后分别压缩各个片段。遗憾的是，其源代码无法获取，也没有在公开数据集上报告任何结果。其他相关工作，如SIM[7], 、TWIN [4] 和TWIN V2 [27], ，将完整长度的序列作为输入，然后进行采样或聚类，因此成本会远高于我们所提的方法。

MerRec 数据集和 EB-NERD 的结果分别列于表 III 和表 IV。表中显示，通过将预训练片段压缩为可学习令牌，所提方法在 HSTU 和 HLLM 上几乎能够保留基线模型使用完整序列（预训练 + 近期）时的性能。所提方法显著优于仅使用近期片段的基线方法。这证明了所提方法在压缩预训练片段并将其用于较短序列的序列推荐方面的有效性。这将大幅降低推理计算成本。

1）训练和推理的计算成本：表 V 展示了所提方法在 MerRec 数据集上的训练和推理时间，并与基线进行了比较。结果表明，与使用完整序列（1280）的 HLLM 相比，所提方法仅增加了微不足道的训练成本（$<$ 5%），却显著降低了推理成本（$>$ 11%）；与此同时，检索指标（召回率或归一化折损累计增益）非常相近。这些实验表明了所提方法的有效性。通过将片段压缩的开销分摊到多次推理中，还可以进一步降低推理成本。

2）专家规模的影响：为研究可学习令牌的数量（k）对检索性能的影响，我们在 HSTU 上评估了所提方法，并采用了不同数量的可学习令牌（从 $k=1$ 到 256）。我们采用与第 IV-B 节相同的训练和评估协议。表 VI 中的结果表明，可学习令牌的数量对检索性能没有显著影响：所有方法的表现都显著优于序列长度为 256 的基线，甚至略优于序列长度为 1280 的基线。然而，$k=16$ 的表现最差，这一点仍需进一步研究。

C. 个性化专家的衰减（*RQ2*）

根据第III-C节，为了摊销模型训练的成本，并将预训练片段压缩为可学习令牌，我们希望尽可能多次复用这些可学习令牌进行推理。本节评估这种复用的影响。我们采用与第 IV-B 节相同的序列分块方法。模型在训练序列上训练完成后，我们生成并保存预训练片段中可学习令牌的激活值。测试时，我们以 256 个事件为窗口，在测试序列上滑动近期片段。对于基线，HSTU 使用训练序列或训练序列中的近期片段（最新的 256 个事件）进行训练；随后在测试时，HSTU 接收一个输入序列，该序列的最后一个元素（最近一次交互）与所提方法的近期片段的最后一个元素对齐。如图7所示。

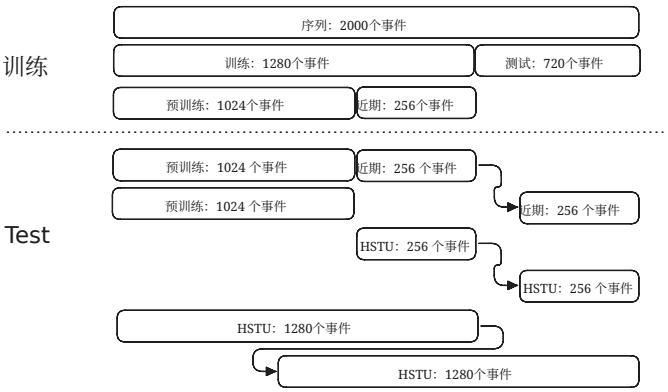

图 7. 用于测量个性化专家性能如何随滑动近期片段与固定预训练片段之间的时间距离变化而变化的序列设置示意图；该设置用于生成所提方法中可学习令牌的激活值。

结果如图 8 所示。该图表明，所提方法始终优于使用 256 个事件的 HSTU；随着滑动近期片段与固定预训练片段之间的时间距离增大（固定预训练片段会生成可学习令牌的激活值），性能差距并未缩小。所提方法甚至达到了使用 1280 个事件的 HSTU 的性能。将我们的方法应用于 HLLM 时，我们也观察到了类似的现象。

表 III 所提方法应用于 HSTU 和 HLLM 的性能，与最先进的序列推荐方法 SASRec、HSTU 和 HLLM 进行比较。对于所提方法，我们使用 $k = 4$ 个可学习令牌。k 的影响在第 IV-B2 节中进行研究。此处 R@K 表示召回率@K，N@K 表示归一化折损累计增益@K。

方法	预训练长度	近期长度	R@10	R@50	R@200	N@10	N@50	N@200
基线 SASRec	N.A.	256	47.40%	64.06%	75.50%	31.00%	34.73%	36.47%
基线 SASRec	N.A.	1280	49.77%	66.05%	77.18%	32.86%	36.52%	38.21%
基线 HSTU	N.A.	256	45.63%	61.79%	73.03%	29.54%	33.16%	34.87%
基线 HSTU	N.A.	1280	51.39%	67.05%	77.79%	34.02%	37.55%	39.18%
个性化 HSTU	1024	256	51.63%	67.31%	77.87%	34.10%	37.63%	39.24%
HLLM 基线	N.A.	256	47.48%	64.33%	75.30%	31.05%	34.83%	36.50%
HLLM 基线	N.A.	1280	49.87%	66.74%	77.51%	33.15%	36.93%	38.57%
个性化 HLLM	1024	256	49.80%	66.64%	77.47%	33.14%	36.92%	38.56%

表 IV 在 EB-NERD 数据集上，将所提方法应用于 HSTU 和 HLLM，并与最先进的序列推荐方法 SASRec、HSTU 和 HLLM 进行比较的性能。对于所提方法，我们使用 $k = 2$ 可学习令牌。对于 EB-NERD，训练序列为前500 个事件，测试集为最后12 个事件；所提方法的训练序列进一步划分为两个片段：预训练（前400 个事件）和近期（剩余100 个事件）。

方法	预训练长度	近期长度	R@10	R@50	R@200	N@10	N@50	N@200
基线 SASRec	N.A.	100	33.38%	41.07%	56.84%	24.51%	27.00%	30.53%
基线 SASRec	N.A.	500	38.79%	46.69%	62.59%	28.93%	31.49%	35.05%
基线 HSTU	N.A.	100	43.55%	61.38%	93.33%	29.04%	34.81%	42.16%
基线 HSTU	N.A.	500	44.79%	61.80%	93.20%	30.58%	36.08%	43.28%
个性化 HSTU	400	100	45.00%	62.56%	93.30%	30.89%	36.57%	43.62%
HLLM 基线	N.A.	100	38.09%	57.11%	92.25%	23.58%	29.69%	37.77%
HLLM 基线	N.A.	500	50.06%	66.18%	93.87%	36.27%	41.47%	47.83%
个性化 HLLM	400	100	48.04%	66.02%	93.69%	33.06%	38.90%	45.28%

表 V 所提方法与基线在 MerRec 数据集上的训练和推理时间比较。这里报告的是每轮训练或评估数据集所需的秒数。令人惊讶的是，HSTU 在序列长度为 256 和 1280 时的推理时间相近。

方法\时间（秒/轮次）	训练	推理
HSTU 256	116	8.6
HSTU 1280	360	8.78
个性化 HSTU	376（+4.4%）	8.7
HLLM 256	6260	123.85
HLLM 1280	27420	163.64
个性化 HLLM	27480 (+0.22%)	144.8 (-11.5%)

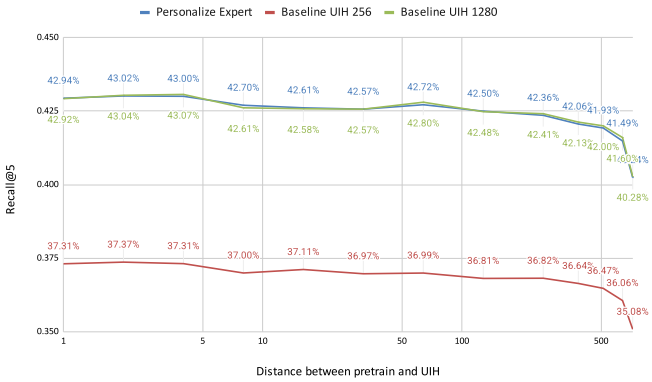

图 8. 在不同位置选取包含 256 个事件的近期片段时，采用所提方法的 HSTU 与采用固定预训练片段的原始 HSTU 的检索性能（Recall@5）。在此实验中，预训练片段始终为用户交互序列的前 1024 个事件。

图 8 表明，随着近期片段沿横轴滑动、离预训练片段越来越远，这些模型的性能（纵轴）持续下降。然而，所提方法始终取得了与使用 1280 个事件的基线 HSTU 相近的检索性能；也就是说，随着近期片段离预训练片段越来越远，这两种模型之间的性能差距并未扩大。这表明，在当前数据集和配置下，当近期片段与固定预训练片段相距 $720 - 256 = 464$ 个事件时，固定预训练片段生成的可学习令牌的性能不会衰减。这证实了先压缩预训练片段，再将其用于多次推理的有效性。

D. 如何放置可学习令牌（*RQ3*）

在以往的实验中，我们将训练序列划分为两个片段：预训练片段和近期片段，然后总是将可学习令牌插入预训练片段之后。然而，这可能并非最佳选择，因为近期事件可能包含更多与推荐相关的信息，因此应分配更多可学习令牌，例如 Kuaiformer。

表 VI 可学习令牌数量对检索性能的影响。结果表明，在当前实验设置下，可学习令牌的数量（从 1 到 256）对检索性能没有显著影响。

方法	预训练长度	# 个人令牌	近期长度	R@10	R@50	R@200	N@10	N@50	N@200
个性化 HSTU	1024	1	256	52.01%	67.66%	78.15%	34.22%	37.75%	39.34%
个性化 HSTU	1024	2	256	51.76%	67.50%	78.01%	33.96%	37.51%	39.10%
个性化 HSTU	1024	4	256	51.63%	67.31%	77.87%	34.10%	37.63%	39.24%
个性化 HSTU	1024	16	256	50.65%	66.96%	77.60%	33.08%	36.76%	38.38%
个性化 HSTU	1024	64	256	51.85%	67.61%	78.02%	34.16%	37.71%	39.29%
个性化 HSTU	1024	256	256	51.92%	67.64%	78.06%	34.11%	37.65%	39.24%

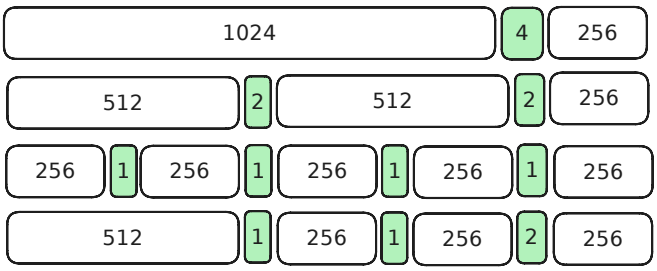

图 9. 将可学习令牌插入训练序列的四种设置示意图。绿色方框表示可学习令牌。方框中的数字表示片段的长度。

[21] 将由 256 个事件组成的序列划分为三个片段：早期（128 个事件）、中期（80 个事件）和近期（47 个事件）；早期和中期片段分别被“压缩”为 2 个和 5 个可学习令牌，而近期片段不进行压缩。本节中，我们研究插入可学习令牌的不同方法。我们考虑以下四种设置（亦见图 9）（T 包含最新事件的片段不进行压缩）：

1)（基线）将序列划分为两个片段 A（1024 个事件）和 T（256 个事件）。A 被压缩为 4 个词元；2) 将序列划分为三个片段 A（512 个事件）、B（512 个事件）和 T（256 个事件）。A 和 B 各被压缩为 2 个词元；3) 将序列划分为五个片段 A、B、C、D 和 T，每个片段包含 256 个事件。A、B、C 和 D 各被压缩为 1 个词元；4) 将序列划分为四个片段 A（512 个事件）、B（256 个事件）、C（256 个事件）和 T（256 个事件）。A、B 和 C 分别被压缩为 1、1 和 2 个词元。这与 Kuaiformer 的设置类似；

注意力运算所使用的注意力掩码根据算法 1 定义，因此每个片段的词元只能关注自身、**同一片段**中的前置词元，以及其前置的所有可学习令牌。这种机制可以促使模型将片段的信息压缩到可学习令牌中，因此在推理时，我们只需生成这些可学习令牌的激活值作为键值缓存，然后将其应用于新序列。

结果见表七。该表表明在预训练片段之后直接插入所有可学习令牌，效果最佳，优于将预训练片段进一步拆分为多个更小的片段，并在每个片段之后放置可学习令牌的做法。事实上，拆分得越细，检索性能就略微越差。这或许是因为设置 1 为模型提供了最大的灵活性，使其能够确定如何利用可学习令牌以及如何压缩信息。不过，设置 3 确实提高了效率，只是检索性能略有损失：我们可以逐步处理每个较小的片段，并聚合相应可学习令牌的激活值；由于 Transformer 的注意力运算具有二次复杂度，这种做法比压缩单个较长片段更高效。

E. 个性化专家的工作原理（*RQ4*）

了解可学习令牌中捕获了哪些信息，以及这些信息如何用于推荐，同样很有意义。为此，我们取可学习令牌的最后一层输出 x，并对该输出相应的预训练片段 P 进行非负矩阵分解（NMF） w：$\text{argmin}_w \|x - Pw\|$ s.t. $w \geq 0$。我们特别考察了来自权重最大的预训练片段中的项目。实验表明，可学习令牌的输出可以由预训练片段中的一组非常少量的项目表示，并且所选项目中的大多数通常都与目标相关。图 10 展示了一个示例。

在表 VIII 中，我们根据非负矩阵分解的权重选出的前 5 个项目，进一步列举了几个目标示例。表 VIII 表明，可学习令牌的输出确实捕捉到了与目标相关的信息。例如，对于第 3 行的目标“LEGO玩具/LEGO”，排名前五的项目都与“LEGO”相关。

V. 结论

在本文中，我们展示了 HSTU 和 HLLM 等序列推荐模型在处理更长用户交互序列方面的扩展潜力。为了解决长序列导致的计算成本呈二次增长的问题，我们提出了一种利用个性化专家的新方法。该方法将序列的一部分压缩为可学习令牌，然后将其与剩余序列结合进行推理。我们在

表七 不同设置下将可学习令牌置于训练序列中的检索性能。训练序列包含1280 个事件。

设置	R@5	R@10	R@50	R@200	N@5	N@10	N@50	N@200
1: [4]	43.15%	51.98%	67.58%	78.19%	31.41%	34.28%	37.80%	39.41%
2: [2, 2]	43.00%	51.78%	67.46%	77.93%	31.30%	34.15%	37.68%	39.27%
3: [1, 1, 1, 1]	42.78%	51.64%	67.53%	77.98%	31.13%	34.01%	37.59%	39.18%
4: [1, 1, 2]	42.88%	51.76%	67.46%	78.03%	31.17%	34.05%	37.59%	39.19%

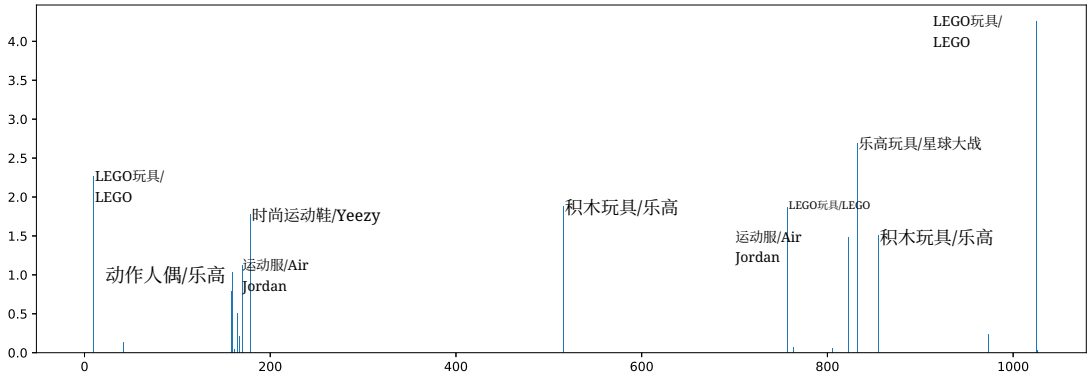

图10. 一个示例上的非负矩阵分解可视化（表 VIII 中的第3行）。提供了排名前10的项目的描述。
检索目标为“LEGO玩具/LEGO”。

表 VIII 预训练片段中由排名5 的项目表示的目标项目和可学习令牌。排名5 的项目根据可学习令牌输出相对于预训练片段的非负矩阵分解权重确定。目标项目表示为 C2名称/品牌名称，预训练片段中的项目表示为 C2名称/品牌名称=权重。

– – – –

目标	项目 1	项目 2	项目 3	项目 4	项目 5
靴子 / 无	项链 / 无=3.083	项链 / 无 = 2.764	耳环 / 克里斯汀·迪奥 = 1.380	项链 / Trifari = 1.128	项链 / 雅芳 = 1.109
连帽 / Torrid	休闲裤 / Torrid = 2.861	被套组 / 无 = 1.563	休闲裤 裤子 / LuLaRoe = 1.459	被套组 / 无 = 1.308	睡裤短裤 / Torrid = 1.270
LEGO 玩具 / LEGO	乐高玩具 / 乐高 = 4.254	乐高玩具 / 星球大战 = 2.688	乐高玩具 / 乐高 = 2.262	积木玩具 / 乐高 = 1.887	乐高玩具 / 乐高 = 1.866
内裤 / 维多-利亚的秘密	胸罩 / 为爱 & 柠檬 = 4.053	内裤 / Victoria’s Secret = 3.474	丁字裤和丁裤 / For Love & Lemons = 2.371	内裤 / Natori = 1.163	丁字裤和丁裤 / 精品店 = 1.030
Boys 2T-5T / 古驰	男童2T-5T码 / Air Jordan = 2.675	男童2T-5T码 / 乔丹 = 2.096	男童2T-5T码 / 乔丹 = 1.677	男童2T-5T码 / 马球衫 Ralph Lauren = 1.557	男童2T-5T码 / Air Jordan = 1.356
运动服 / Nike	球衣 / Wish = 3.124	短裤 / 无 = 2.523	泳裤 / American Eagle = 1.318	男童2T-5T码 / Air Jordan = 1.206	T恤 / 时尚 Nova = 1.071
Boys (4+) / 博柏利	男童2T-5T码 / Air Jordan = 3.752	男童（4+）/ 博柏利 = 3.466	男童2T-5T码 / ZARA = 3.044	男童2T-5T码 / 博柏利 = 2.207	靴子 / 马丁博士 = 1.760

HSTU 和 HLLM 这两种最先进的序列推荐模型上实现了所提方法，并在 MerRec 数据集上评估其性能。实验结果表明，所提方法有效且高效，能够在保持较高推荐准确率的同时降低计算成本。此外，我们还深入分析了可学习令牌所捕获的信息，并探究这些令牌的位置如何影响模型性能。研究结果表明，我们的方法可作为提升可扩展性的有效工具以及序列推荐模型的效率。总体而言，我们的工作有助于开发更高效、更有效的序列推荐模型，使其能够更好地处理较长的用户交互序列，并提升整体用户体验。

参考文献

[1] “Claude 3模型系列：Opus、Sonnet、Haiku。”[在线]。网址：https://api.semanticscholar.org/CorpusID:268232499

[2] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat 等，“GPT-4技术报告”，*arXiv*预印本 *arXiv:2303.08774*，2023年。

[3] Z. Chai, Q. Ren, X. Xiao, H. Yang, B. Han, S. Zhang, D. Chen, H. Lu, W. Zhao, L. Yu, X. Xie, S. Ren, X. Sun, Y. Tan, P. Xu, Y. Zheng, and D. Wu, “Longer：扩展工业推荐系统中的长序列建模”，2025年。 [在线]。网址：https://arxiv.org/abs/ 2505 04421 。J. Chang, C. Zhang, Z. Fu, X. Zang, L. Guan, J. Lu, Y. Hui, D. Leng, Y. Niu, Y. Song, and K. Gai, “TWIN：用于快手点击率预测中终身用户行为建模的两阶段兴趣网络”，2023年6月，arXiv:2302.02352 [cs]。 [在线]。网址：http://arxiv.org/abs/ 2302 02352 。G. Chen, H. Shi, J. Li, Y. Gao, X. Ren, Y. Chen, X. Jiang, Z. Li, W. Liu, and C. Huang, “SepLLM：通过将一个片段压缩为一个分隔符来加速大语言模型”，2024年12月，arXiv:2412.12094 [cs]。 [在线]。网址：http://arxiv.org/abs/2412. 12094[6] J. Chen, L. Chi, B. Peng, and Z. Yuan, “HLLM：通过用于物品与用户建模的层次化大语言模型增强序列推荐”，2024年9月，arXiv:2409.12740 [cs]。 [在线]。网址：https://arxiv.org/abs/ 2409 12740 。X.-H. Chen, B. He, Y. Yu, Q. Li, Z. Qin, W. Shang, J. Ye, and C. Ma, “Sim2Rec：一种用于优化现实世界长期用户参与度的序列推荐系统模拟器决策方法”，2023年5月，arXiv:2305.04832 [cs]。 [在线]。网址：http://arxiv.org/abs/ 2305 04832 。A. Chevalier, A. Wettig, A. Ajith, and D. Chen, “让语言模型适应上下文压缩”，2023年11月，arXiv:2305.14788 [cs]。 [ ] 2305 14788 在线。网址：http://arxiv.org/abs/。C. Deng, Z. Zhang, K. Mao, S. Li, X. Huang, D. Yu, and Z. Dou, “全注意力的灵丹妙药还是折中方案？基于要旨令牌的上下文压缩综合研究”，2024年12月，arXiv:2412.17483 [cs]。 [在线]。网址：http://arxiv.org/abs/ 2412.17483[10] A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Yang, A. Fan 等，“Llama 3模型群”，arXiv预印本 arXiv:2407.21783，2024年。[11] J. Gao, B. Chen, X. Zhao, W. Liu, X. Li, Y. Wang, W. Wang, H. Guo, and R. Tang, “Llm4rerank：用于推荐的基于大语言模型的自动重排序框架”，载于 THE WEB CONFERENCE 2025。[12] T. Ge, J. Hu, L. Wang, X. Wang, S.-Q. Chen, and F. Wei, “大语言模型中用于上下文压缩的上下文内自编码器”，2024年5月，arXiv:2307.06945 [cs]。 [在线]。网址：http://arxiv.org/abs/ 2307 06945 。B. Hidasi, “基于循环神经网络的基于会话的推荐”，arXiv预印本 arXiv:1511.06939，2015年。[14] Y. Hou, J. Zhang, Z. Lin, H. Lu, R. Xie, J. McAuley, and W. X. Zhao, “大语言模型是推荐系统的零样本排序器”，载于欧洲信息检索会议。Springer，2024年，第364–381页。[15] A. Hurst, A. Lerer, A. P. Goucher, A. Perelman, A. Ramesh, A. Clark, A. Ostrow, A. Welihinda, A. Hayes, A. Radford 等，“GPT-4o系统卡片”，arXiv 预印本 arXiv:2410.21276，2024年。[16] W.-C. Kang and J. McAuley，“自注意力序列推荐”，载于2018年IEEE国际数据挖掘会议（ICDM）。IEEE，2018年，第197–206页。[17] J. Kruse, K. Lindskow, S. Kalloori, M. Polignano, C. Pomo, A. Srivastava, A. Uppal, M. R. Andersen, and J. Frellsen, “EB-NeRD：用于新闻推荐的大规模数据集”，载于《2024年推荐系统挑战赛论文集》，系列：RecSysChallenge ’24。美国纽约：美国计算机协会，2024年，第 1 11 [ ] 10 1145 3687151 3687152 –。在线。网址：https://doi.org/。/。W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. E. Gonzalez, H. Zhang, and I. Stoica, “利用分页注意力高效管理大语言模型服务的内存”，载于《第29届ACM SIGOPS操作系统原理研讨会论文集》，2023年。[19] L. Li, Z. A. Din, Z. Tan, S. London, T. Chen, and A. Daptardar, “MerRec：用于消费者对消费者推荐系统的大规模多用途Mercari数据集”，2024年7月。 2024， 2402 14230 [ ] [ ] 2402 arXiv:。cs。在线。网址：http://arxiv.org/。 14230 [20] A. Liu, B. Feng, B. Xue, B. Wang, B. Wu, C. Lu, C. Zhao, C. Deng, C. Zhang, C. Ruan 等，“DeepSeek-V3技术报告”，arXiv预印本 arXiv:2412.19437，2024年。[21] C. Liu, J. Cao, R. Huang, K. Zheng, Q. Luo, K. Gai, and G. Zhou, “KuaiFormer：快手的基于Transformer的检索”，2024年11月，arXiv:2411.10057 [cs]。 [在线]。可访问：http://arxiv.org/abs/2411。10057 [22] H. Liu、M. Zaharia 和 P. Abbeel，“利用分块 Transformer 的环形注意力实现近乎无限的上下文”，2023 年。 [在线]。可访问：https://arxiv.org/abs/ 2310 01889 。Z. Liu、Y. Hou 和 J. McAuley，“多行为生成式推荐”，载于《第 33 届 ACM 信息与知识管理国际会议论文集》，2024 年，第 1575–1585 页。[24] Y. Luo、Q. Qin、H. Zhang、M. Cheng、R. Yan、K. Wang 和 J. Ouyang，“Molar：通过协同过滤对齐增强序列推荐的多模态大语言模型”，2024 年。 [在线]。可访问：https://arxiv.org/abs/ 2412 18176 。J. Mu、X. L. Li 和 N. Goodman，“学习使用要旨令牌压缩提示”，2024 年 2 月，arXiv:2304.08467 [cs]。 [在线]。可访问：http://arxiv.org/abs/ 2304 08467 。N. Shazeer，“快速 Transformer 解码：只需一个写入头即可”，2019 [ ] 1911 02150 。在线。可访问：https://arxiv.org/abs/。Z. Si、L. Guan、Z. Sun、X. Zang、J. Lu、Y. Hui、X. Cao、Z. Yang、Y. Zheng、D. Leng、K. Zheng、C. Zhang、Y. Niu、Y. Song 和 K. Gai，“TWIN V2：在快手扩展超长用户行为序列建模以增强点击率预测”，载于《第 33 届 ACM 信息与知识管理国际会议论文集》，10 月。 2024， 4890 4897， 2407 16357 第 – 页。arXiv：。[ ] [ ] cs。在线。可访问：http://arxiv.org/abs/。 2407 16357 [28] F. Sun、J. Liu、J. Wu、C. Pei、X. Lin、W. Ou 和 P. Jiang，“Bert4rec：利用来自 Transformer 的双向编码器表示进行序列推荐”，载于《第 28 届 ACM 信息与知识管理国际会议论文集》，2019 年，第 1441–1450 页。[29] G. Team、R. Anil、S. Borgeaud、J.-B. Alayrac、J. Yu、R. Soricut、J. Schalkwyk、A. M. Dai、A. Hauth、K. Millican 等，“Gemini：高能力多模态模型家族”，arXiv 预印本 arXiv:2312 11805，2023 年。。[30] A. Vaswani，“注意力即你所需”，《神经信息处理系统进展》，2017 年。[31] S. Wang、B. Z. Li、M. Khabsa、H. Fang 和 H. Ma，“Linformer：具有线性复杂度的自注意力”，2020 年。 [在线]。可访问：https://arxiv.org/abs/ 2006 04768 。J. Wei、Y. Tay、R. Bommasani、C. Raffel、B. Zoph、S. Borgeaud、D. Yogatama、M. Bosma、D. Zhou、D. Metzler 等，“大语言模型的涌现能力”，arXiv 预印本 arXiv:2206.07682，2022 年。[33] L. Wu、Z. Zheng、Z. Qiu、H. Wang、H. Gu、T. Shen、C. Qin、C. Zhu、H. Zhu、Q. Liu 等，“面向推荐的大语言模型综述”，《万维网》，第 27 卷，第 5 期，第 60 页，2024 年。[34] G. Xiao、Y. Tian、B. Chen、S. Han 和 M. Lewis，“具有注意力汇的高效流式语言模型”，2024 年。 [在线]。可访问：https://arxiv.org/abs/ 2309 17453 。W. Xu、Q. Wu、Z. Liang、J. Han、X. Ning、Y. Shi、W. Lin 和 Y. Zhang，“Slmrec：赋能小语言模型进行序列推荐”，arXiv 预印本 arXiv:2405.17890，2024 年。[36] A. Yang、B. Yang、B. Zhang、B. Hui、B. Zheng、B. Yu、C. Li、D. Liu、F. Huang、H. Wei 等，“Qwen2. 5 技术报告”，arXiv 预印本 arXiv:2412.15115，2024 年。[37] Z. Yue、S. Rabhi、G. d. S. P. Moreira、D. Wang 和 E. Oldridge，“LlamaRec：使用大语言模型进行排序的两阶段推荐”，arXiv 预印本 arXiv:2311.02089，2023 年。[38] J. Zhai、L. Liao、X. Liu、Y. Wang、R. Li、X. Cao、L. Gao、Z. Gong、F. Gu、M. He、Y. Lu 和 Y. Shi，“行动胜于言辞：用于生成式推荐的万亿参数序列转换器”，2024 年 5 月，arXiv:2402.17152 [cs]。 [ ] 2402 17152 在线。可访问：http://arxiv.org/abs/。C. Zhang、S. Wu、H. Zhang、T. Xu、Y. Gao、Y. Hu、D. Wu 和 E. Chen，“NoteLLM：用于笔记推荐的可检索大语言模型”，2024 年 3 月，arXiv:2403.01744 [cs]。 [在线]。可访问：http://arxiv.org/abs/ 2403 01744 。C. Zhang、H. Zhang、S. Wu、D. Wu、T. Xu、Y. Gao、Y. Hu 和 E. Chen，“Notellm-2：用于推荐的多模态大型表征模型”，arXiv 预印本 arXiv:2405.16789，2024 年。[41] K. Zhou、H. Wang、W. X. Zhao、Y. Zhu、S. Wang、F. Zhang、Z. Wang 和 J.-R. Wen，“S3-rec：通过互信息最大化进行序列推荐的自监督学习”，载于《第 29 届 ACM 信息与知识管理国际会议论文集》，2020 年，第 1893–1902 页。