arXiv:2302.03487v1 [cs.IR] 6 Feb 2023

PIER：电子商务中的基于排列级兴趣的端到端重排序框架

Xiaowen Shi*
美团
中国北京
shixiaowen03@美团.com

Fan Yang*
美团
中国北京
yangfan129@meituan.com

Ze Wang†
美团
中国北京
wangze18@meituan.com

Xiaoxu Wu
美团
中国北京
wuxiaoxu04@美团.com

Muzhi Guan
美团
中国北京
guanmuzhi@meituan.com

Guogang Liao
美团
中国北京
liaoguogang@meituan.com

Yongkang Wang
美团
中国北京
wangyongkang03@meituan.com

Xingxing Wang
美团
中国北京
wangxingxing04@meituan.com

Dong Wang
美团
中国北京
wangdong07@meituan.com

摘要

重排序在学术界和工业界都日益受到关注，它通过建模商品之间的相互影响来重新排列排序列表，从而更好地满足用户需求。许多现有的重排序方法直接将初始排序列表作为输入，并通过精心设计的上下文级模型生成最优排列，这带来了重排序前评估问题。同时，在实践中评估所有候选排列会产生无法接受的计算成本。因此，为了更好地平衡效率与效果，在线系统通常采用两阶段架构：首先使用束搜索等启发式方法生成适量的候选排列，然后将其输入评估模型，以获得最优排列。然而，两个阶段中的现有方法都可以从以下方面加以改进。在生成阶段，启发式方法仅使用点式预测分数，缺乏有效的判断机制。在评估阶段，大多数现有的上下文评估模型仅考虑项目上下文，缺乏更细粒度的特征上下文建模。

本文提出了一种名为 *PIER* 的新型端到端重排序框架，以应对上述挑战。该框架仍采用两阶段架构，主要包含两个模块：FPSM 和 OCPM。受长期用户行为建模方法的启发，我们在 FPSM 中应用 SimHash，根据用户的排列级兴趣，从全排列中高效选出前 K 个候选项。随后，我们在 OCPM 中设计了一种新颖的全向注意力机制，以更好地捕捉排列中的上下文信息。最后，我们引入对比学习损失，以端到端的方式联合训练这两个模块，并利用 OCPM 的预测值指导 FPSM 生成更优的排列。离线实验结果表明，PIER 在公开数据集和工业数据集上的表现均优于基线模型，并且我们已成功将 PIER 部署于美团外卖平台。

关键词

重排序、端到端学习、推荐系统

ACM参考格式：

Xiaowen Shi、Fan Yang、Ze Wang、Xiaoxu Wu、Muzhi Guan、Guogang Liao、Yongkang Wang、Xingxing Wang和Dong Wang。2023年。PIER：电子商务中基于排列级兴趣的端到端重排序框架。载于第*29*届*ACM SIGKDD*知识发现与数据挖掘会议（*KDD '23*），*2023*年*8*月*6*日至*10*日，美国加利福尼亚州长滩。ACM，纽约州纽约，9 页。https://doi.org/10.1145/xxxxxx.xxxxxx

1 引言

京东和美团等电子商务应用拥有大量商品。为了提高用户的决策效率，通常会根据用户的兴趣提供一个包含有限商品的物品列表。如图1所示，美团外卖平台向用户展示了一个名为“猜你喜欢”的物品列表，其中包含三个商品。

随着深度学习技术的快速发展，许多精心设计的排序模型被提出，以提升推荐性能，主要聚焦于特征交互（例如 Wide&Deep [9], DeepFM [14], xDeepFM [16]）、用户兴趣建模（例如 DIN [27], DIEN [26], ETA [8]）等。然而，现有的大多数排序方法只对当前商品的点击率进行建模，却忽略了上下文物品之间至关重要的相互影响。为建模展示商品的排列对用户行为的影响，我们引入了重排序阶段，对排序阶段生成的初始列表重新排序。

*贡献相同。列出顺序为随机。
†通讯作者。

允许免费为个人或课堂使用而制作本作品全部或部分内容的数字版或纸质副本，前提是这些副本不得用于盈利或商业利益，也不得为此目的进行分发，并且副本须包含本声明及首页的完整引文。对于本作品中归他人所有而非 ACM 所有的组件，必须尊重其版权。注明出处后可以摘录。若要以其他方式复制、重新出版、发布至服务器或分发至列表，则须事先获得特定许可和/或支付费用。请向 permissions@acm.org 申请许可。*KDD '23*，*2023*年*8*月*6*日至*10*日，美国加利福尼亚州长滩©2023年计算机协会。ACM ISBN 978-1-4503-xxxx-x/xx/xx...$15.00https://doi.org/10.1145/xxxxxx.xxxxxx

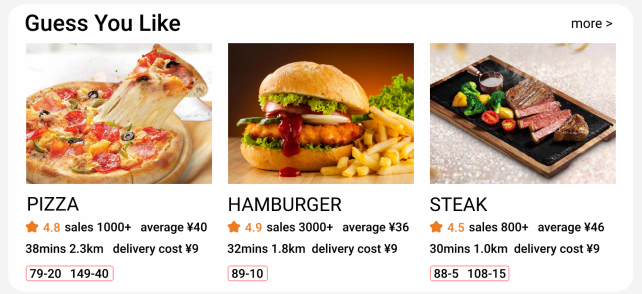

图 1：美团的猜你喜欢。

现有的重排序方法可分为两类[12]。第一类是逐步贪心重排序策略[3, 12, 13, 29], , 它依次决定每个位置的展示结果。这类方法只考虑前序信息，却忽略后续信息，因此不足以获得最优结果。与贪心策略不同，另一种方案是上下文级重排序策略 [1, 6, 11, 20, 25],, 它使用上下文级评估模型来捕捉商品之间的相互影响，并重新预测每个商品的点击率。PRM 等方法 [20], 直接将初始排序列表作为输入，并根据上下文模型给出的预测值生成最优排列。这些单阶段方法存在重排序前评估问题 [25],: 也就是说，由于重排序后的顺序与初始顺序不同，输入重排序后的列表会得到不同的预测结果。为解决重排序前评估问题，一个直接的方案是评估所有可能的排列；这种方案虽能达到全局最优，但复杂度过高，无法满足工业系统严格的推理时间限制。因此，现有的大多数重排序框架采用两阶段架构 [11, 25] , 由排列生成和排列评估组成。具体而言，它们先使用束搜索等 [21]启发式方法生成适量的候选排列，然后将其输入评估模型，以获得最优排列。

为了提升两阶段架构下重排序阶段的性能，一方面，生成的候选排列应尽可能包含最优排列；另一方面，上下文级评估模型应尽可能准确地进行预测。然而，现有的两个阶段中的方法仍可从以下方面进行改进：

• 生成阶段。一些启发式方法，例如束搜索算法 [11],仅使用逐点预测分数（即项目点击率）来生成候选排列，却忽略了一个排列中各商品与其上下文之间的相互影响。此外，由于生成阶段独立于评估阶段，评估结果无法指导生成过程。因此，生成的候选排列缺乏有效的质量评判。• 评估阶段。与逐点商品预测不同，排列的评估需要利用多种类型的上下文信息，才能充分建模商品之间的相互影响

除了商品上下文的影响外，特征上下文也会产生细粒度的影响。这些特征构成了多种频道，用户可能会对某个频道中的特征感兴趣。例如，价格敏感型用户会更加关注上下文中的价格信息对比，我们将其称为多特征频道竞争问题。

为了解决上述问题，我们提出了一种新颖的端到端重排序框架，称为基于排列级兴趣的端到端重排序（PIER）。我们的框架仍遵循两阶段范式，主要包含两个模块：细粒度排列选择模块（FPSM）和全向上下文感知预测模块（OCPM）。受长期用户行为建模方法的启发 [4, 8],，我们在 FPSM 中应用 SimHash，根据用户的排列级兴趣，以高效的方式从全排列中选出前 K 个候选项。随后，在 OCPM 中，我们设计了一种新颖的全向注意力和上下文感知预测机制，以更好地捕获上下文信息，并预测排列中每个商品的列表级点击率。最后，我们将这两个模块整合到一个框架中，并以端到端的方式进行训练。我们引入了一种比较学习损失，利用 OCPM 的预测值来指导 FPSM 生成更优的排列。

The 我们工作的主要贡献总结如下 :

• 我们提出了一种名为 PIER 的新型重排序框架，将生成模块和评估模块整合到一个模型中，并可通过端到端的方式进行训练。

• 我们在公开数据集和来自美团的真实工业数据集上进行了大量离线实验。实验结果表明，PIER 行之有效。值得注意的是，PIER 已部署于美团外卖平台，并在多项指标上取得了显著提升。

2 相关工作

工业推荐系统通常包含三个阶段[1] :
召回、 [28], 排序 [15, 24] 和重排序 [20]。召回阶段旨在从整个商品集合中召回数千个相关商品。排序阶段 [9, 14, 27]逐点预测召回商品的点击率（或转化率等）。重排序阶段旨在从排序模型给出的初始列表中找出最佳排列（例如，使总点击量最大化）。本文主要关注重排序阶段。

典型的重排序方法可分为两类。第一类是逐步贪心重排序策略 [3, 12, 13,29], , 它通过循环神经网络或近似解，依次确定每个位置的展示结果。Seq2slate [3]使用指针网络，MIRNN 则使用 [29]门控循环单元逐一确定商品顺序。同样，快手的客户端短视频重排序框架 [13]将以有序候选项列表为输入的逐点预测模型与束搜索相结合，依次生成最终列表。这些方法忽略了后续信息，因此无法获得最优结果。

[1]本文不讨论广告相关阶段（例如混合排序 [7, 18], 和拍卖 [17]）。

另一类是上下文级重排序策略 [1, 6, 11,20, 25], ，它使用上下文级评估模型来捕捉商品之间的相互影响，并重新预测每个商品的点击率/转化率。PRM [20] 和 DLCM [1] 等方法以初始排序列表为输入，使用循环神经网络或自注意力对上下文级信号进行建模，并输出每个商品的预测值。最优排列根据预测值排序得到。这类方法带来了重排序前评估问题， [25] 并导致结果次优。同样，EXTR [6]等方法会估计每个候选商品在每个候选位置上的预估点击率；这类方法本质上是逐点模型，因此难以提取准确的上下文。为了对排列的准确上下文进行建模，一个直接的解决方案是通过精心设计的上下文级模型评估所有可能的排列。这是一种全局最优方法，但复杂度过高，无法满足工业系统严格的推理时间约束。为了降低复杂度，PRS [11] 采用束搜索首先生成少量候选排列，再通过排列级排序模型为每个排列打分。尽管启发式方法有效，但它们只使用点式预测分数，缺乏有效的判断机制，仍有进一步改进的空间。

在我们的场景中，我们仍然采用两阶段架构，并专注于在在线耗时约束下，通过优化两个阶段来提升整体性能。

3 预备知识

通常，一个工业推荐系统由三个连续阶段组成：召回、排序和重排序 [11, 15, 20, 24, 25, 28]。给定一个用户及其输入排序列表 $C = \{a_i\}_{i=1}^{N_o}$，向用户最终展示的 N_d 商品被组织为展示列表 $P = \{a_i\}_{i=1}^{N_d}$，其中 $N_d \leq N_o$。

如果不存在重排序阶段，则从 C 中选取排名前N_d 的商品作为 G 进行展示。因此，重排序阶段的任务是学习一种重排序策略 $\pi: C \rightarrow P^*$，旨在从 C 中选择并重新排列商品，随后推荐一个更优的展示列表 P^*，以提升点击率和商品交易总额等指标。

4 方法

我们展示了 PIER 的整体结构（**P**排列级**I**基于兴趣的 **E**端到端**R**重排序）如图所示2。具体而言，我们将排序列表 $C = \{a_i\}_{i=1}^{N_o}$ 和用户的排列级点击行为序列 $\mathcal{B} = \{b_i\}_{i=1}^{M}$ 作为 PIER 的输入。然后，我们通过全排列算法在 $C = \{p_i\}_{i=1}^{T}$ 上生成候选排列 C。接下来，我们使用细粒度排列选择模块（FPSM），从大量候选排列中选出前K 个排列。最后，我们使用全方位上下文感知预测模块（OCPM）计算每个排列的分数，并选择最佳排列 p^* 作为要展示的重排序列表。

在 FPSM 中，我们提出了一种基于 SimHash [5, 8, 19] 计算的时间感知汉明距离。通过对用户排列级点击行为与候选排列之间的距离进行排序，选出前K个排列。在 OCPM 中，我们设计了一种新颖的全向注意力单元，用于对每个排列中的上下文信息建模，并输出该排列中每个商品的列表级 pCTR。根据输出分数（即列表级 pCTR 之和）选出最佳排列。FPSM 与 OCPM 的关系类似于推荐系统中的匹配和排序阶段。我们将两者整合到一个框架中，以生成最优重排序列表。接下来，我们将分别详细介绍 FPSM 和 OCPM。

4.1 细粒度排列选择模块

考虑到时间开销，一些重排序方法会采用束搜索等启发式方法 [11] 生成候选排列，并使用精心设计的预测模型来选择最优排列，但这些启发式方法与建模目标并不一致，导致性能欠佳。受 ETA 和 SDIM 等长期用户行为建模方法 [4], 的启发，我们提出了 FPSM，通过 SimHash 选择候选项。这里，我们将用户的历史点击行为作为目标，然后计算它与所有候选排列之间的距离。如果距离更近，我们便认为该排列能更好地匹配用户兴趣，从而带来更高收益。通过这种方式，我们不仅可以降低时间复杂度，还能通过训练以端到端的方式一致地进行选择，

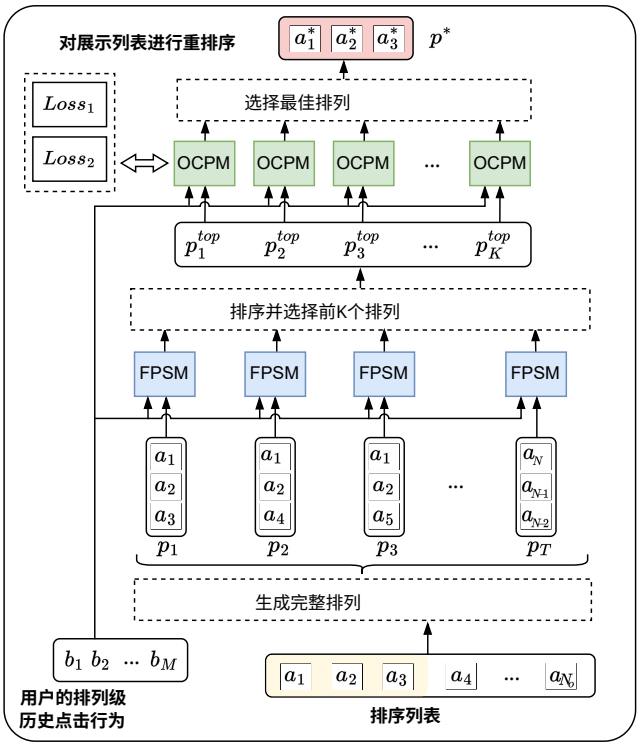

图2：我们的框架 PIER 概览。PIER 以排序列表和用户的排列级历史点击行为作为输入，并借助 FPSM 和 OCPM 输出用于展示的重排序列表。

[2]为便于在图中说明，我们将每个排列中的商品数量 N_d 设为 3。

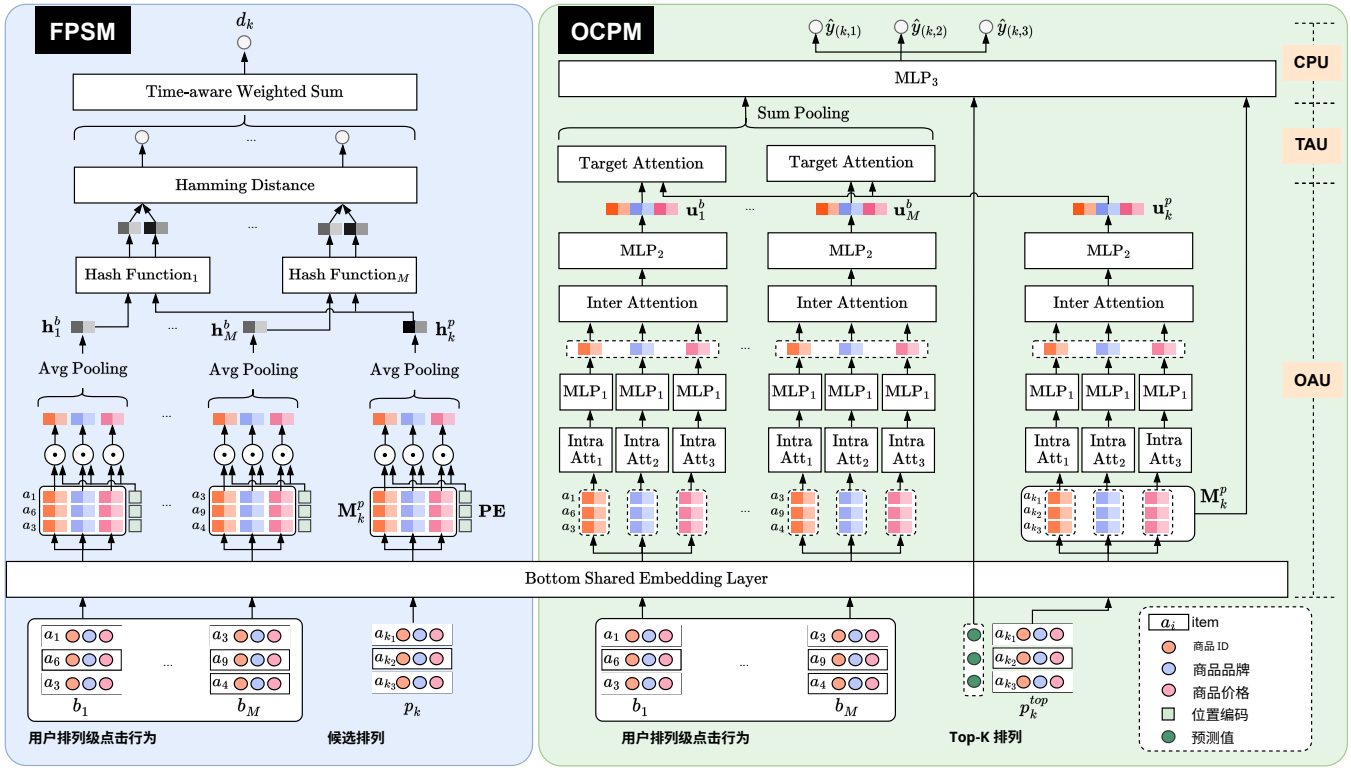

图 3：细粒度排列选择模块（FPSM）和全向上下文感知预测模块（OCPM）的结构。FPSM 以用户排列级点击行为和一个候选排列为输入，并输出该排列的距离分数。OCPM 以用户排列级点击行为和一个由 FPSM 选出的 Top-K 排列为输入，并输出该排列中每个商品的列表级 pCTR 预测值。彩色版本效果最佳。

FPSM 和预测模型。接下来，我们将介绍如何通过 FPSM 选择前 K 个排列。

如图3左侧所示，我们首先使用底部的共享嵌入层从原始输入中提取嵌入。对于排列 p_k，我们将嵌入矩阵$\mathbf{M}_k^p$表示如下：

$$\begin{aligned}\mathbf{M}_k^p &= \begin{bmatrix} \mathbf{E}_{0;}^{p_k} \\ \mathbf{E}_{1;}^{p_k} \\ \dots \\ \mathbf{E}_{N_d;}^{p_k} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{E}_{;0}^{p_k} & \mathbf{E}_{;1}^{p_k} & \dots & \mathbf{E}_{;N_f}^{p_k} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{e}_{0;0}^{p_k} & \mathbf{e}_{0;1}^{p_k} & \dots & \mathbf{e}_{0;N_f}^{p_k} \\ \mathbf{e}_{1;0}^{p_k} & \mathbf{e}_{1;1}^{p_k} & \dots & \mathbf{e}_{1;N_f}^{p_k} \\ \dots & \dots & \dots & \dots \\ \mathbf{e}_{N_d;0}^{p_k} & \mathbf{e}_{N_d;1}^{p_k} & \dots & \mathbf{e}_{N_d;N_f}^{p_k} \end{bmatrix} \in \mathbb{R}^{N_d \times N_f \times D}, \end{aligned} \tag{1}$$

其中，N_d 表示每个排列中的商品数量，N_f表示每个商品中的特征字段数量（如 ID、类别、品牌等），D 表示嵌入转换后的特征字段的维度，$\mathbf{e}_{i;j}^{p_k} \in \mathbb{R}^D$ 表示排列 p_k 中第 i 个商品的第 j 个特征字段的嵌入，$\mathbf{E}_{i;}^{p_k} \in \mathbb{R}^{N_f \times D}$ 表示排列 p_k 中第 i 个商品的嵌入矩阵，而$\mathbf{E}^{p_k}{}_{;j} \in \mathbb{R}^{N_d \times D}$表示排列 p_k 中第 j 个特征字段的嵌入矩阵。类似地，用户排列级历史点击行为中第 m 个排列的嵌入矩阵可表示为$\mathbf{M}_m^b \in \mathbb{R}^{N_d \times N_f \times D}$。

接下来，我们为每个排列生成位置编码矩阵$\mathbf{PE} \in \mathbb{R}^{N_d \times D}$，如下所示：

$$\begin{aligned} \mathrm{PE}_{(i,2d)} &= \sin(i/10000^{2d/D}), \\ \mathrm{PE}_{(i,2d+1)} &= \cos(i/10000^{2d/D}), \\ \mathbf{PE} &= \begin{bmatrix} \mathrm{PE}_{(0,0)} & \mathrm{PE}_{(0,1)} & \dots & \mathrm{PE}_{(0,D)} \\ \mathrm{PE}_{(1,0)} & \mathrm{PE}_{(1,1)} & \dots & \mathrm{PE}_{(1,D)} \\ \dots & \dots & \dots & \dots \\ \mathrm{PE}_{(N_d,0)} & \mathrm{PE}_{(N_d,1)} & \dots & \mathrm{PE}_{(N_d,D)} \end{bmatrix} \in \mathbb{R}^{N_d \times D}. \end{aligned} \tag{2}$$

然后，将每个特征字段的嵌入矩阵分别与位置编码矩阵 **PE** 相乘，再通过平均池化将其合并为相应的排列表示$\mathbf{h}_k^p$，如下所示：

$$\mathbf{h}_k^p = \frac{1}{N_f} \sum_{i=1}^{N_f} \text{Avg-Pool}\left(\mathbf{E}_{;i}^{p_k} \odot \mathbf{PE}\right), \quad \forall k \in [N_o]. \tag{3}$$

类似地，用户的排列级历史点击行为中第 m个排列的表示形式为$\mathbf{h}_m^b$。

在我们的场景中，用户更有可能点击与自身兴趣更接近的排列。我们使用用户的排列级历史点击行为来表示用户的兴趣，并计算用户兴趣与每个候选排列之间的距离。具体而言，我们采用随机投影方案（SimHash）[5, 8, 19] 来计算用户点击过的排列的表示形式与候选排列的表示形式之间的相似度。首先，我们生成 M 个与 M 用户排列级行为相对应的哈希函数。对于每个候选排列 p_k，我们使用$\mathbf{h}_k^p$ 个 M 不同的哈希函数对其表示形式进行哈希，并计算它与每种用户排列级行为之间的汉明距离，如下所示：

$$\mathrm{Sim}(p_k, b_m) = \mathrm{Hash}_m(\mathbf{h}_k^p, \mathbf{h}_m^b), \ \ \forall m \in [M], \forall k \in [N_o]. \quad (4)$$

与此同时，行为发生得越近，就越能反映用户当前的兴趣，因此在相似度计算中会获得更高的权重。因此，我们根据每种行为的发生时间为这些距离加权，以获得时间感知的汉明距离，如下所示：

$$d_k = \sum_{m=1}^{M} w_m \cdot \mathrm{Sim}(p_k, b_m), \ \ \forall k \in [N_o]. \quad (5)$$

其中，w_m 是第 m种行为的时间感知权重。

最后，我们根据距离分数对候选排列进行排序，并选择距离分数最小的前-K 个排列 $\mathrm{P}^{top-K} = \{p_i^{top}\}_{i=1}^{K}$，作为 FPSM 的输出。

由于 FPSM 与 OCPM 共享底层嵌入层，并且哈希函数和位置编码使用固定的随机向量，因此它没有需要训练的独立参数。为了确保训练期间所选前 K 个排列的质量，我们提出了一种对比损失，以提升 FPSM 的性能。对比损失的详细信息见第 4.3.2 节。

4.2 全向上下文感知预测模块

对于 FPSM 选出的每一种候选排列，我们都会使用 OCPM，通过三个连续的单元来预测每件商品的预估点击率：全向注意力单元（OAU）、目标注意力单元（TAU）和上下文感知预测单元（CPU）。OCPM 的架构如图 3 右侧所示，接下来我们将分别介绍 OCPM 的各个单元。

4.2.1 全向注意力单元（*OAU*）。向用户展示一种排列时，一方面，用户可能会关注不同的展示信息，例如价格、配送费、评分等；另一方面，不同商品同一特征之间的竞争关系也会影响用户的行为。例如，将价格昂贵的商品排在价格便宜的商品前面，可能会激发用户点击便宜商品的意愿。从这个角度来看，我们在 OCPM 中设计了一个全向注意力单元，以有效地对每种排列中的信息进行建模。

具体而言，OCPM 首先使用与 FPSM 相同的底部共享嵌入层，从原始输入中提取嵌入。然后，我们使用N_f参数独立的自注意力层 [22]分别计算各字段中不同商品之间的相互影响，并输出相应的矩阵$\mathbf{H}_j^{p_k}$，如下所示：

$$\mathbf{H}_j^{p_k} = \mathrm{soft\,max}\left(\frac{\mathbf{Q}_j^{p_k}\mathbf{K}_j^{p_k\top}}{\sqrt{D}}\right)\mathbf{V}_j^{p_k}, \ \ \forall j \in [N_f], \forall k \in [K], \quad (6)$$

其中，$\mathbf{Q}_j^{p_k}$、$\mathbf{K}_j^{p_k}$、$\mathbf{V}_j^{p_k}$分别表示目标排列中第 k 个排列的第 j 个字段对应的查询、键和值。 D表示每个特征的维度。这里，查询、键和值由$\mathbf{E}^{p_k;j}$经过线性变换得到，如下所示：

$$\mathbf{Q}_j^{p_k} = \mathbf{E}_{:j}^{p_k}\mathbf{W}_j^Q, \mathbf{K} = \mathbf{E}_{:j}^{p_k}\mathbf{W}_j^K, \mathbf{V} = \mathbf{E}_{:j}^{p_k}\mathbf{W}_j^V, \forall j \in [N_f], \forall k \in [K], \quad (7)$$

其中，$\mathbf{W}_j^Q$、$\mathbf{W}_j^K$、$\mathbf{W}_j^V \in \mathbb{R}^{D\times D}$。然后将$\mathbf{H}_j^{p_k}$输入多层感知机（MLP）层，以生成表示：

$$\begin{aligned} \mathbf{h}_j^{p_k} &= \mathrm{MLP}_1\left(\mathbf{H}_j^{p_k}\right), \ \ \forall j \in [N_f], \forall k \in [K], \\ \mathbf{Z}^{p_k} &= \left[\mathbf{h}_1^{p_k} \ \ \mathbf{h}_2^{p_k} \ \ \ldots \ \ \mathbf{h}_{N_f}^{p_k}\right] \in \mathbb{R}^{N_f\times D}, \ \ \forall k \in [K], \end{aligned} \quad (8)$$

之后，引入一个跨字段自注意力层，计算每个排列中不同字段之间的相互影响，并通过多层感知机层输出该排列的最终表示$\mathbf{u}_k^p$，如下所示：

$$\mathbf{u}_k^p = \mathrm{MLP}_2\left(\mathbf{H}_j^{p_k}\right) = \mathrm{MLP}_2\left(\mathrm{soft\,max}\left(\frac{\mathbf{Q}'_{p_k}\mathbf{K}'^{\top}_{p_k}}{\sqrt{D}}\right)\mathbf{V}'_{p_k}\right), \forall k \in [K], \quad (9)$$

其中，$\mathbf{Q}'_{p_k}$、$\mathbf{K}'_{p_k}$、$\mathbf{V}'_{p_k}$分别表示第 k 个排列对应的查询、键和值，由$\mathbf{Z}^{p_k}$经过线性变换得到，如下所示：

$$\mathbf{Q}'_{p_k} = \mathbf{Z}^{p_k}\mathbf{W}^{Q'}, \mathbf{K}'_{p_k} = \mathbf{Z}^{p_k}\mathbf{W}^{K'}, \mathbf{V}'_{p_k} = \mathbf{Z}^{p_k}\mathbf{W}^{V'}, \forall k \in [K]. \quad (10)$$

类似地，用户排列级历史点击序列中第 m 个排列的最终表示可表示为$\mathbf{u}_m^b$。通过这两个自注意力层，我们可以有效地建模不同商品和不同特征字段之间的关系。

4.2.2 目标注意力单元（*TAU*）。Zhou 等人 [27] 已证明，与目标更相关的历史行为能够为模型的预测提供更多信息。我们使用排列级目标注意力单元，对目标排列与历史行为中的每个排列之间的交互进行建模，如下所示：

$$\begin{aligned} \mathbf{w}_{m;k} &= \mathbf{u}_m^b \cdot \mathrm{MLP}_{\mathrm{Att}}\left(\mathbf{u}_k^p || \mathbf{u}_m^b || (\mathbf{u}_k^p \odot \mathbf{u}_m^b) || (\mathbf{u}_k^p - \mathbf{u}_m^b)\right), \\ \mathbf{w}_k &= \text{Sum-Pool}\left(\left[\mathbf{w}_{1;k}, \ldots, \mathbf{w}_{M;k}\right]\right), \forall m \in [M], \forall k \in [K]. \end{aligned} \quad (11)$$

其中，|| 表示拼接。TAU 的输出表示$\mathbf{w}_k$ 被视为用户针对目标排列的排列级兴趣表示，并作为下一个单元的输入。

4.2.3 上下文感知预测单元（*CPU*）。在上下文感知预测单元中，我们使用参数共享多层感知机层来预测每个排列中每个商品的列表级 pCTR。以第 k 个目标排列为例，CPU 的输入由四部分组成：第 k 个目标排列的表示 $\mathbf{u}_k^p$、用户对第 k 个目标排列的排列级兴趣 $\mathbf{w}_k$、原始预测值（即每个商品的点级预估点击率 курул</final>

商品) $\{v_i^{p_k}\}_{i=1}^{N_d}$ 和 k个目标排列的嵌入矩阵 $\mathbf{M}_k^p$。然后，预测第 k 个目标排列中第 t 个商品的列表级 pCTR 如下：

$$\hat{y}_{(k,t)} = \sigma\Big(\text{MLP}_3(\mathbf{u}_k^p||\mathbf{w}_k||\{v_i^{p_k}\}_{i=1}^{N_d}||\mathbf{M}_k^p)\Big), \tag{12}$$

其中 σ 是 Sigmoid 函数。在 PIER 中，每个排列的得分可通过对输出的列表级预估点击率求和轻松获得。不过需要注意的是，使用 PIER 框架时，可根据业务需求方便地调整每个排列的得分，例如使用商品交易总额而非点击率。

4.3 模型训练

在训练阶段，我们首先使用真实曝光的排列来训练 OCPM，然后通过添加对比学习损失联合训练这两个模块。

4.3.1 OCPM 预训练。为了准确评估选出的前K个排列，我们首先使用从在线日志中收集的样本对 OCPM 进行预训练。输入是实际展示的排列，标签则表示每个展示排列中的商品是否被点击。然后，OCPM 的损失计算如下：

$$Loss_1 = \sum_{t=1}^{N_d}\Big(-y_{(i,t)}\log(\hat{y}_{(k,t)}) - (1-y_{(i,t)})\log(1-\hat{y}_{(i,t)})\Big), \tag{13}$$

其中，下标 i 表示样本的索引，下标 t 表示展示商品的索引。

4.3.2 PIER 联合训练。 在联合训练阶段，由于我们将固定的随机向量用作“哈希函数”和“位置编码”，且 FPSM 与 OCPM 共享嵌入，因此 FPSM 无需更新梯度。嵌入更新时，哈希签名也会相应更新。那么，如何在训练过程中提升所选前K个排列的质量？为此，我们提出一种对比损失来实现这一目标。其核心思想是，所选前K个排列的平均预估点击率与未选排列中任意K个排列的平均预估点击率之间的差异应尽可能大。因此，联合训练阶段的最终损失为：

$$Loss_2 = -\sum_{k=1,k'\notin[K]}^{K}\Big(\frac{1}{N_d}\sum_{t=1}^{N_d}\hat{y}_{(k,t)} - \frac{1}{N_d}\sum_{t=1}^{N_d}\hat{y}_{(k',t)}\Big)^2, \tag{14}$$

其中，下标 i 表示样本的索引，下标 t 表示展示商品的索引。

最后，我们从数据集中抽取一批样本 B，并根据损失通过梯度反向传播更新 PIER：

$$L(B) = \frac{1}{|\mathcal{B}|}\sum_{\mathcal{B}}\Big(Loss_1 + \alpha\cdot Loss_2\Big), \tag{15}$$

其中 α 是用于平衡两种损失的系数。

5 实验

在本节中，我们进行了大量离线实验和在线 A/B 测试，以评估我们框架的有效性[3]。对于离线实验，我们将首先把 OCPM 与现有的基线方法进行比较，并分析其中不同设计的作用。在使用相同预测模型的前提下，接下来我们将比较 FPSM 与其他生成方法的性能。最后，我们将验证不同的超参数设置（例如 K、α）如何影响我们框架的性能。对于在线实验，我们将通过在线 A/B 测试，将我们的框架与美团平台上部署的现有策略进行比较。

5.1 实验设置

5.1.1 数据集。为验证我们框架的有效性，我们在公开数据集和工业数据集上进行了充分实验。对于公开数据集，我们选用 Avito 数据集；对于工业数据集，我们使用从美团外卖平台收集的真实世界数据。表 1 简要介绍了这些数据集。

• **Avito**。公开的 Avito 数据集包含来自 avito.ru 的用户搜索日志和元数据，其中包含超过 3600 万条广告、130 万名用户和 5300 万个搜索请求。完整特征包括用户搜索信息（例如，用户 ID、搜索 ID 和搜索日期）以及广告信息（例如，广告 ID、类别 ID、标题等）。每个搜索 ID 对应一个包含多个广告的搜索页面。对于每位用户，我们根据搜索日期按时间递增顺序排列其搜索页面，并将前 T-1 个搜索页面作为行为页面，将第 T 个搜索页面中的广告作为待预测的目标广告。为避免数据泄漏，我们使用 20150428 至 20150514 的数据作为训练集，并使用 20150515 至 20150520 的数据作为测试集。• **美团**。美团工业数据集采集自 2022 年 4 月的美团外卖平台，包含用户信息（例如，用户 ID、性别、年龄）和广告信息（例如，广告 ID、类别 ID、品牌 ID 等）。根据数据采集日期，我们按 8:2 的比例将数据集划分为训练集和测试集。

5.1.2 评估指标。对于离线实验，我们针对不同模块采用不同的评估指标。具体而言，我们使用广泛采用的曲线下面积（AUC）指标来评估预测模块的效果，并使用以下指标评估整个框架：

• **命中率（HR）@1** [2]。对于每条数据，仅当生成方法选出的前K个排列中包含最佳排列时，HR 才为 1。• **成本**。成本指不同重排序框架所需的总时间。对于在线实验，我们通过点击率（CTR）、商品交易总额（GMV）和推理时间，将我们提出的框架与现有方法进行比较。

5.1.3 超参数。MLP1、MLP2、MLP3 的隐藏层大小分别为 (128, 64, 32)、(60, 32, 20) 和 (50, 20)。学习率为 10^{-3}，优化器为 Adam，批量大小为 1,024。α 为 0.1，嵌入维度为 8，用户

表 1：数据集的统计数据。

数据集	#请求	#用户	#Ads
Avito	53,562,269	1,324,103	23,562,269
美团	230,525,531	3,201,922	98,525,531

[3]代码可在以下地址公开获取https://github.com/Lemonace/PIER_代码。

表2：两个数据集上的曲线下面积和对数损失实验结果。

模型	Avito		美团	
	AUC	对数损失	AUC	对数损失
DNN	0.6876	0.0483	0.6538	0.1917
DCN	0.6896	0.0483	0.6550	0.1916
PRM	0.7131	0.0481	0.6718	0.1899
EXTR	0.7114	0.0481	0.6704	0.1901
Edge-Rerank	0.7163	0.0479	0.6694	0.1909
OCPM	**0.7320**	**0.0471**	**0.6822**	**0.1891**

行为序列的长度为 5。对于 Avito 数据集，排序列表和重排序列表的长度均为 5，因此全排列的长度为 120，**K** 设为 10。对于 Metuan 数据集页面，我们从包含 10 个商品的初始排序列表中选取 3 个商品，因此全排列的长度为 720，**K** 设为 100。

5.2 OCPM 离线实验

5.2.1 基线方法。 我们将 OCPM 与点式和列表式代表方法作为基线进行比较。我们选择 DNN、DCN 作为点式基线，并选择快手的 PRM、EXTR 和 Edge-Rerank 作为列表式基线。以下简要介绍这些方法：

• **DNN** [10]。DNN 是一种用于点击率预测的基础深度学习方法，它采用多层感知机进行高阶特征交互。 • **DCN** [23]。DCN 在每一层显式进行特征交叉，无需手动进行特征工程，且只给 DNN 模型带来可忽略不计的额外复杂度。 • **PRM** [20]。PRM 通过应用自注意力机制来捕捉商品之间的相互影响，从而调整初始列表。 • **EXTR** [6]。外部性 Transformer 将包含所有广告位的目标广告视为查询，并将外部商品视为键和值，以对所有曝光场景中的外部性进行建模。 • **Edge-Rerank** [13]。Edge-Rerank 将端侧排序模型与自适应束搜索方法相结合，生成具有上下文感知能力的重排序结果。

5.2.2 性能比较。表2 总结了离线实验的结果。所有实验均重复进行了 5 次，并报告平均结果。根据实验结果，我们得到以下观察：i）所有重排序列表模型（例如 PRM、EXTR）通过对上下文商品之间的相互影响进行建模，相较于逐点模型（例如 DNN、DCN）均取得了显著提升，这验证了上下文对用户点击行为的影响。ii）与基于Transformer的模型（例如 PRM、ETXR）相比，Edge-Rerank 也提升了点击率预测效果，这是因为它使用了历史序列和先前商品信息。iii）与最先进的独立基线相比，我们提出的 OCPM 在 Avito/Metuan 数据集上分别带来了 0.0189/0.0104 的绝对曲线下面积提升，这对于工业推荐系统而言是显著的改进。

5.2.3 消融研究。 为了探究 OCPM 中不同模块的有效性，我们在 Avito 和美团数据集上进行了消融研究。所有实验重复 5 次，并报告平均曲线下面积：

表3：重排序模型不同部分的消融实验结果。

模型	Avito		美团	
	AUC	对数损失	AUC	对数损失
OCPM	**0.7320**	0.0483	**0.6822**	0.1917
- OAU	0.7198	0.0483	0.6774	0.1916
- TAU	0.7263	0.0481	0.6786	0.1899

• OCPM(-OAU) 屏蔽了全向注意力模块。OAU 旨在捕获项目上下文和特征上下文信息，从而对不同商品的相同属性之间的竞争关系进行建模。如表 3所示，曲线下面积下降了 0.0112/0.0048，这表明提取上下文信息至关重要，且所提出的 OAU 满足了这一要求。 • OCPM (-TAU) 不使用目标注意力单元。TAU 旨在计算目标排列与历史行为中的各个排列之间的交互。如表 3所示，在 Avito/Metuan 数据集上，曲线下面积分别下降了 0.0057/0.0036，这表明我们的 TAU 能够捕获用户历史页面级兴趣。

5.3 PIER 的离线实验

5.3.1 基线方法。 我们将结合 FPSM 和 OCPM 的完整框架与一些启发式生成方法进行比较，并将这些方法作为基线方法。所有这些方法都使用固定的 OCPM，但候选排列的生成方式各不相同。以下简要介绍这些方法：

• **随机 & OCPM**。我们随机选取 K 个排列作为候选项。

• **PRS（束搜索 & OCPM）** [11]。我们使用束搜索方法，根据累积点击率生成 K 个候选排列。

• **全排列 & OCPM**。我们直接将所有候选排列输入预测模型，并根据平均点击率选出 K 个排列。这可以视为我们框架的上界。

5.3.2 性能比较。 表 4 总结了离线实验的结果，我们有以下观察：i）直观来看，在公开数据集和工业数据集上寻找最佳排列时，PIER 相较于随机方法和束搜索方法取得了显著提升，而时间复杂度仅略有增加。一种合理的解释是，FPSM 能够在对比损失的引导下选出更优的排列。ii）在寻找最佳排列方面，与全排列方法相比，我们仍有很大的改进空间，但我们大幅降低了整个框架的时间成本。

5.3.3 消融研究。 为验证不同单元（即时间感知加权和对比损失）的影响，我们研究了 PIER 框架的两个消融变体：

• PIER（-时间感知加权）不对每种行为使用时间感知权重，而是将它们视为同等重要。
• PIER（-对比损失）移除了对比损失，以提升所选前K个排列的质量，即 $\alpha = 0$。

表4：不同方法的命中率和成本。

模型	Avito		美团	
	HR	成本（毫秒）	HR	成本（毫秒）
随机 & OCPM	0.09	**10.8**	0.13	**70.5**
PRS（束搜索 & OCPM）0.62		16.8	0.72	78.7
全排列 & OCPM	1.00	100.5	1.00	430.5
PIER（FPSM & OCPM）	**0.78**	17.5	**0.84**	85.3
- 时间感知加权	0.74	17.3	0.70	84.8
- 对比损失	0.52	17.5	0.61	85.1

由于移除某个单元需要对整个框架重新训练，因此我们尽量确保 OCPM 的曲线下面积尽可能接近，以保证消融实验的可比性。根据实验结果，我们得出以下发现：i）在距离计算中使用与不使用时间感知加权，对公开数据集的性能差异影响很小，但会在一定程度上影响工业数据集上的性能。这表明，在美团场景中，用户的兴趣在短时间内保持一致。ii）在两个数据集上，不使用对比损失都会导致性能明显下降。这表明，辅助损失有助于 FPSM 选择更好的排列。

表5：PIER各部分消融实验的结果

设置	Avito			美团		
	AUC	HR	成本（毫秒）	AUC	HR	成本（毫秒）
$\alpha = 0$	0.7337	0.52	17.5	0.6833	0.57	85.1
$\alpha = 0.01$	0.7336	0.63	17.4	0.6831	0.62	85.3
$\alpha = 0.05$	0.7329	0.71	17.6	0.6828	0.71	85.3
$\alpha = 0.1$	0.7320	0.78	17.5	0.6822	0.84	85.2
$\alpha = 0.3$	0.7108	0.81	17.4	0.6674	0.88	85.4
$\alpha = 0.5$	0.6927	0.88	17.6	0.6425	0.91	85.3
$K = 5$	0.7325	0.63	13.8	-	-	-
$K = 10$	0.7320	0.71	17.5	-	-	-
$K = 20$	0.7318	0.82	25.2	-	-	-
$K = 50$	0.7299	0.93	46.9	0.6827	0.78	49.2
$K = 100$	-	-	-	0.6822	0.84	85.3
$K = 200$	-	-	-	0.6813	0.92	157.3
$K = 300$	-	-	-	0.6799	0.95	223.3

5.4 超参数分析

我们分析了两个超参数的敏感性： α， K。具体来说， α 是对比损失的权重，而 K 是 FPSM 选择的排列数量。结果如表5所示，公开数据集和工业数据集呈现相同趋势，我们得出以下结论：

i) 当 α 在一定范围内增加时，OCPM的曲线下面积保持在相对较好的水平，同时HR有所提升。当

表6：在线A/B测试的实验结果。

模型	CTR	商品交易总额	耗时（毫秒）	超时
基础排序模型	0%	0%	**0**	0.0%
PRM	1.21%	0.92%	6.1	0.0%
束搜索与评估器	3.17%	2.54%	10.5	0.052%
全排列与评估器	-	-	52.4	64.39%
PIER	**5.46%**	**5.83%**	11.1	**0.061%**

当α 超过一定水平时，对比损失对OCPM的影响增大，曲线下面积开始下降，导致HR指标的置信度降低。

ii) 更改 K 主要会影响该框架的命中率和平均成本。在一定范围内增加 K 可以快速提升命中率表现。超过一定范围后，命中率的提升会变缓。与此同时，平均成本会随着K的增加而上升。因此，在实际应用中，应合理设置 K 的值，以平衡效果和效率。

5.5 线上结果

我们通过在线 A/B 测试，将 PIER 与其他模型（例如基础排序模型、PRM 等）进行比较，并将所有模型部署在美团外卖平台上。具体而言，我们在 2022年4月9日 至 2022年4月25日（一周）期间，使用全量生产流量的 1% 进行在线 A/B 测试。结果发现，PIER 的点击率和商品交易总额分别提升了 5.46% 和 5.83%。此外，我们重点关注耗时这一重要指标，因为它决定了模型能否应用于大规模工业场景。如表 6 所示，全排列无法部署，因为其超时率增加了 64.39%。与束搜索相比，PIER 的点击率提升了 2.29%，商品交易总额提升了 2.31%，而超时率仅略有上升，系统可以接受这一影响。目前，PIER 已部署上线并服务于主要流量，为业务带来了显著增长。

6 结论

本文提出了一种名为 PIER 的新型端到端重排序框架，该框架包含两个主要模块：FPSM 和 OCPM。受长期用户行为建模方法的启发，我们在 FPSM 中应用 SimHash，从全排列中选择前 K 个候选项。为了更好地捕获排列中的上下文信息，我们在 OCPM 中设计了一种新型全向注意力机制。最后，我们引入对比学习损失，以引导 FPSM 生成更优的排列，并以端到端方式联合训练这两个模块。离线实验和在线 A/B 测试均表明，PIER 的性能显著优于其他现有的重排序基线方法；目前，我们已将 PIER 部署在美团外卖平台上。

参考文献

[1] Qingyao Ai、Keping Bi、Jiafeng Guo 和 W Bruce Croft。2018。学习用于排序优化的深度列表式上下文模型。发表于 第 *41* 届国际 *ACM SIGIR* 信息检索研究与开发会议。135–144。[2] Areej Alsini、Du Q Huynh 和 Amitava Datta。2020。命中率：标签推荐的评估指标。*arXiv* 预印本 *arXiv:2010.01258*（2020）。[3] Irwan Bello、Sayali Kulkarni、Sagar Jain、Craig Boutilier、Ed Chi、Elad Eban、Xiyang Luo、Alan Mackey 和 Ofer Meshi。2018。Seq2slate：使用循环神经网络进行重排序和列表优化。*arXiv* 预印本 *arXiv:1810.02019*（2018）。[4] Yue Cao、Xiaojiang Zhou、Jiaqi Feng、Peihao Huang、Yao Xiao、Dayao Chen 和 Sheng Chen。2022。在点击率预测中对长期用户行为建模：采样就是你所需要的一切。发表于 第 *31* 届 *ACM*国际信息与知识管理会议。2974–2983。[5] Moses S Charikar。2002。基于舍入算法的相似度估计技术。发表于 第 *34* 届 *ACM*计算理论研讨会。380–388。[6] Chi Chen、Hui Chen、Kangzhi Zhao、Junsheng Zhou、Li He、Hongbo Deng、Jian Xu、Bo Zheng、Yong Zhang 和 Chunxiao Xing。2022。EXTR：电子商务赞助搜索中考虑外部性的点击率预测。发表于 *ACM SIGKDD*知识发现与数据挖掘会议论文集。2732–2740。[7] Dagui Chen、Qi Yan、Chunjie Chen、Zhenzhe Zheng、Yangsu Liu、Zhenjia Ma、Chuan Yu、Jian Xu 和 Bo Zheng。2022。信息流中的分层约束自适应广告曝光。发表于 第 *31* 届 *ACM*国际信息与知识管理会议。3003–3012。[8] Qiwei Chen、Changhua Pei、Shanshan Lv、Chao Li、Junfeng Ge 和 Wenwu Ou。2021。点击率预测模型中的端到端用户行为检索。*arXiv* 预印本 *arXiv:2108.04468*（2021）。[9] Heng-Tze Cheng、Levent Koc、Jeremiah Harmsen、Tal Shaked、Tushar Chandra、Hrishi Aradhye、Glen Anderson、Greg Corrado、Wei Chai、Mustafa Ispir 等。2016。用于推荐系统的宽深度学习。发表于 首届推荐系统深度学习研讨会论文集。7–10。[10] Paul Covington、Jay Adams 和 Emre Sargin。2016。用于 YouTube推荐的深度神经网络。发表于 第 *10* 届 *ACM* 推荐系统会议论文集。191–198。[11] Yufei Feng、Yu Gong、Fei Sun、Junfeng Ge 和 Wenwu Ou。2021。从排列视角重新审视推荐系统。*arXiv* 预印本 *arXiv:2102.12057*（2021）。[12] Yufei Feng、Binbin Hu、Yu Gong、Fei Sun、Qingwen Liu 和 Wenwu Ou。2021。GRN：用于上下文级推荐的生成式重排序网络。*arXiv* 预印本 *arXiv:2104.00860*（2021）。[13] Xudong Gong、Qinlin Feng、Yuan Zhang、Jiangling Qin、Weijie Ding、Biao Li、Peng Jiang 和 Kun Gai。2022。移动设备上的实时短视频推荐。发表于 第 *31* 届 *ACM*国际信息与知识管理会议。3103–3112。[14] Huifeng Guo、Ruiming Tang、Yunming Ye、Zhenguo Li 和 Xiuqiang He。2017。DeepFM：一种基于因子分解机的点击率预测神经网络。发表于 第 *26* 届国际人工智能联合会议。1725–1731。[15] Xiang Li、Shuwei Chen、Jian Dong、Jin Zhang、Yongkang Wang、Xingxing Wang 和 Dong Wang。2023。用于点击率预测的决策上下文交互网络。arXiv 预印本 arXiv:2301.12402（2023）。[16] Jianxun Lian、Xiaohuan Zhou、Fuzheng Zhang、Zhongxia Chen、Xing Xie 和 Guangzhong Sun。2018。xDeepFM：结合显式和隐式特征交互的推荐系统。载于第 24 届 ACM SIGKDD 国际知识发现与数据挖掘会议论文集。1754–1763。[17] Guogang Liao、Xuejian Li、Ze Wang、Fan Yang、Muzhi Guan、Bingqi Zhu、Yongkang Wang、Xingxing Wang 和 Dong Wang。2022。NMA：具有外部性的神经多槽拍卖，用于在线广告。arXiv 预印本 arXiv:2205.10018（2022）。[18] Guogang Liao、Ze Wang、Xiaoxu Wu、Xiaowen Shi、Chuheng Zhang、Yongkang Wang、Xingxing Wang 和 Dong Wang。2022。Cross dqn：用于信息流广告分配的交叉深度 Q 网络。载于 ACM Web Conference 2022 论文集。401–409。[19] Gurmeet Singh Manku、Arvind Jain 和 Anish Das Sarma。2007。用于网络爬取的近重复内容检测。载于第 16 届万维网国际会议论文集。https://doi.org/10.1145/1242572.1242592[20] Changhua Pei、Yi Zhang、Yongfeng Zhang、Fei Sun、Xiao Lin、Hanxiao Sun、Jian Wu、Peng Jiang、Junfeng Ge、Wenwu Ou 等。2019。用于推荐的个性化重排序。载于第 13 届 ACM 推荐系统会议论文集。3–11。[21] D Raj Reddy 等。1977。语音理解系统：五年研究工作的成果总结。计算机科学系。卡内基梅隆大学，匹兹堡，宾夕法尼亚州 17（1977），138。[22] Ashish Vaswani、Noam Shazeer、Niki Parmar、Jakob Uszkoreit、Llion Jones、Aidan N Gomez、Łukasz Kaiser 和 Illia Polosukhin。2017。Attention is all you need。神经信息处理系统进展 30（2017）。[23] Ruoxi Wang、Bin Fu、Gang Fu 和 Mingliang Wang。2017。用于广告点击预测的深度交叉网络。载于 ADKDD'17 论文集。1–7。[24] Zhe Wang、Liqin Zhao、Biye Jiang、Guorui Zhou、Xiaoqiang Zhu 和 Kun Gai。2020。Cold：迈向下一代预排序系统。arXiv 预印本 arXiv:2007.16122（2020）。[25] Yunjia Xi、Weiwen Liu、Xinyi Dai、Ruiming Tang、Weinan Zhang、Qing Liu、Xiuqiang He 和 Yong Yu。2021。通过效用最大化实现用于推荐的上下文感知重排序。arXiv 预印本 arXiv:2110.09059（2021）。[26] Guorui Zhou、Na Mou、Ying Fan、Qi Pi、Weijie Bian、Chang Zhou、Xiaoqiang Zhu 和 Kun Gai。2019。用于点击率预测的深度兴趣演化网络。载于 AAAI 人工智能大会论文集，第 33 卷。5941–5948。[27] Guorui Zhou、Xiaoqiang Zhu、Chenru Song、Ying Fan、Han Zhu、Xiao Ma、Yanghui Yan、Junqi Jin、Han Li 和 Kun Gai。2018。用于点击率预测的深度兴趣网络。载于第 24 届 ACM SIGKDD 国际知识发现与数据挖掘会议论文集。1059–1068。[28] Han Zhu、Xiang Li、Pengye Zhang、Guozheng Li、Jie He、Han Li 和 Kun Gai。2018。学习用于推荐系统的基于树的深度模型。载于第 24 届 ACM SIGKDD 国际知识发现与数据挖掘会议论文集。1079–1088。[29] Tao Zhuang、Wenwu Ou 和 Zhirong Wang。2018。电子商务搜索中的全局优化互影响感知排序。载于第 27 届国际人工智能联合会议论文集。3725–3731。