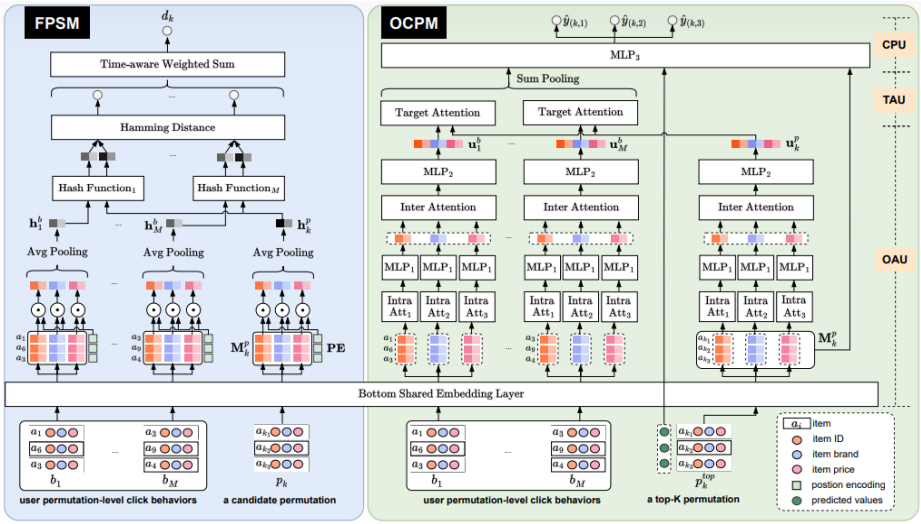
知乎




# 美团2023-基于兴趣的端到端重排框架，让推荐系统更懂你！-论文深度解析

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

21 人赞同了该文章

## Introduction

电子商务应用，如京东和美团，有大量的商品。为了 提高用户的决策效率，通常会根据用户的兴趣提供一 个包含有限商品的列表。随着深度学习技术的迅速发展，许多精心设计的 排序模型被提出以提高推荐性能，主要集中在特征交 互（如 Wide&Deep，DeepFM，xDeepFM ）， 用户兴趣建模（如 DIN，DIEN ，ETA）等 方面。然而，大多数现有的排序方法只建模当前商品 的点击率，而忽略了上下文商品之间的相互影响。为 了建模展示商品的排列对用户行为的影响，引入了重 排序阶段来重新排列排序阶段的初始列表。

为了提升两阶段架构下重排序阶段的性能，方法包括：

1. 生成候选排列时，应尽可能包含最优排列；

2. 上下文相关评估模型应准确预测；

现有方法可以从以下方面改进： 1. 优化候选排列生成方法； 2. 提高模型预测精度。

- 生成阶段存在一些问题。一些启发式方法仅使用逐点预测分数来生成候选排列，忽略了排列中每个物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，生成的候选排列的质量缺乏有效的判断。建议使用更有效的方法指导生成过程。
- 评估阶段使用各种上下文信息充分建模项目之间的相互影响，不仅考虑项目上下文还考虑特征上下文的细粒度影响，针对价格敏感用户更关注价格信息比较的特性提出多特征渠道竞争问题。
- 本文提出了一种名为PIER的新型重排序框架，将生成模块和评估模块集成到一起，通过端到端的方式进行训练。该框架有助于提升模型的性能。

## related work

一个工业级的推荐系统通常包含三个阶段，本文主要关注推荐系统中的重排阶段，介绍了两种典型的重排方法。

第一种是贪婪重排策略，逐个确定每个位置的显示结果。

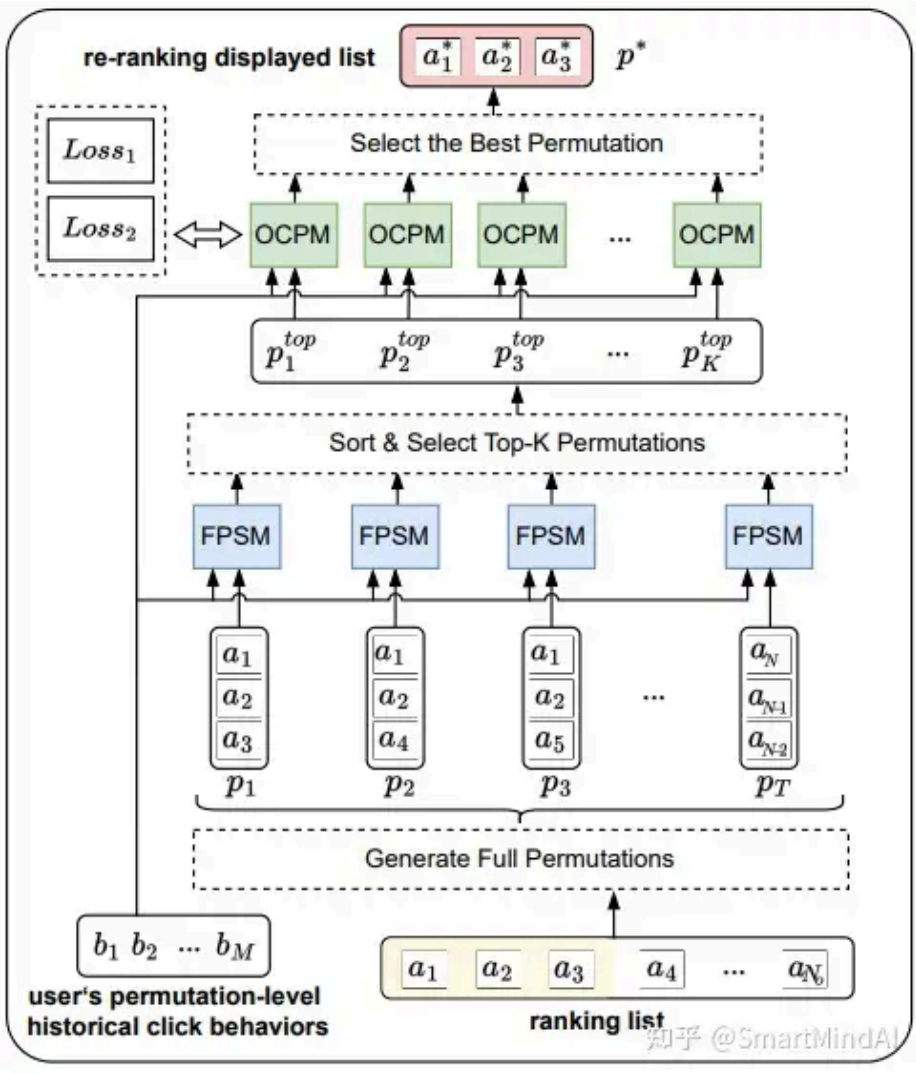
第二种是将有序候选列表作为输入的逐点预测模型与集束搜索相结合，顺序生成最终列表。这两种方法都忽略了后续信息，不足以获得最优结果。

知乎

工业推荐系统由三个阶段组成：匹配、排序和重排序。给定用户及其输入列表，最终显示的$N_d$个项被表示为显示列表。

## Methodology

我们展示了PIER的概述结构，通过全排列算法生成候选排列，并使用细粒度排列选择模块和全向上下文感知预测模块进行重新排序。在FPSM中，基于SimHash的时间感知汉明距离⁺被提出，而在OCPM中，一种新颖的全向注意力单元被设计以建模上下文信息。PIER将两者组合到框架中以生成最佳的重新排序列表。




### Fine-grained Permutation Selection Module

提出了一种名为FPSM的方法来选择前K个候选排列，方法借鉴了长期用户行为建模⁺方法。

首先使用共享嵌入层提取输入的嵌入，再根据嵌入矩阵计算⁺与所有候选排列之间的距离。通过这种方式，我们可以在保持一致的同时降低时间复杂度⁺，并通过端到端的方式训练FPSM和预测模型。 符号说明： $p_k$：排列 $\mathbf{M}_k^p$：嵌入矩阵 公式示例： 公式（1）描述了计算距离的过程：

$$dist(u, p_k) = \frac{u \cdot p_k}{\|u\| \cdot \|p_k\|}$$

知乎

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

这段内容主要介绍了排列中的物品数量、特征字段数量、嵌入转换后的特征字段维度、物品嵌入矩阵、特征字段嵌入矩阵以及位置编码✦矩阵等概念和生成方式。

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

这段内容主要介绍了如何将特征场的嵌入矩阵乘以位置编码矩阵，并通过平均池化合并成排列表示。具体步骤如下：将每个特征场的嵌入矩阵乘以位置编码矩阵，再通过平均池化合并成相应的排列表示。

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

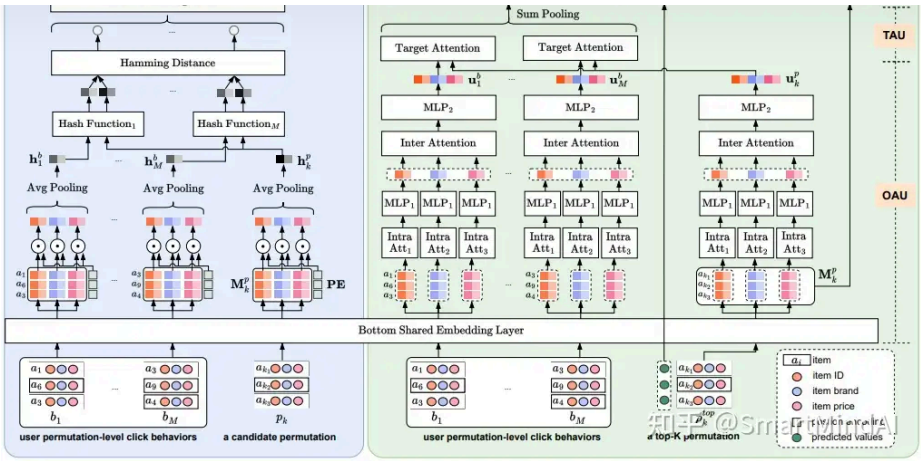
用户排列层次的历史点击行为被用来代表用户兴趣，并计算用户兴趣和每个候选排列之间的距离。通过SimHash计算用户点击过的排列的表示和候选排列的表示之间的相似度。利用$M$个不同的哈希函数✦，对每个候选排列进行表示并计算汉明距离。

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

本文提出了一种基于时间感知的海明距离的计算方法，该方法根据每个行为的发生时间对距离进行加权，从而更好地反映用户当前的兴趣。

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

这段内容主要介绍了全方位✦上下文感知预测模块（FPSM）和对比损失的细节，该模块用于预测项目的pCTR，并使用全方位注意力单元、目标注意力单元和上下文感知预测单元进行预测。该模块与OCPM共享底部嵌入层，并固定随机向量✦，无需训练独立参数。为了提高性能，提出了一种对比损失✦。




### Omnidirectional Attention Unit (OAU)

在向用户展示排列时，不同项目相同特征之间的竞争关系会影响用户行为，设计全方位的注意力单元来建模每个排列的信息。提取嵌入后，使用$N_f$个参数独立的自注意力层计算相互影响，并输出相应的矩阵。

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

该文研究了第$k$个目标排列中第$j$个字段的查询、键和值在特征维度$D$下的线性变换[+]问题，具体表达式为

$$\mathbf{Q}_j^{p_k} = \mathbf{E}_{:j}^{p_k} \cdot \mathbf{V}_j^{p_k} + \mathbf{K}_j^{p_k}。$$

文中进一步对算法和数据集进行阐述和分析。

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

将矩阵$\mathbf{W}_j^Q$、$\mathbf{W}_j^K$、$\mathbf{W}_j^V$应用于多层感知机（MLP）。这三个矩阵均为实数矩阵，其大小为$D \times D$。然后我们将$\mathbf{H}_j^{p_k}$输入到这些矩阵中进行多层感知机操作，从而生成新的表示。具体实现可参考文献详细研究内容。

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

其次采用了跨字段自注意力层和MLP层来计算每个排列中不同字段之间的相互影响，并输出排列的最终表示。

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

具体使用方程式$\mathbf{Q}'_{p_k}, \mathbf{K}'_{p_k}, \mathbf{V}'_{p_k}$表示第$k$个排列的查询、键和值，通过从$\mathbf{Z}^{p_k}$线性转换得到。

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

用户在排列层次历史点击序列中第m个排列的最终表示为$\mathbf{u}_m^b$，并通过两个自注意力层[+]有效建模不同物品和不同特征领域之间的关系。

### Target Attention Unit (TAU)

基于历史行为与目标排列的交互关系，指出更相关的历史行为能够提供更多信息帮助模型预测。提出使用目标注意力单元来建模交互，为排列级别的目标关注提供新思路。具体实现过程如上述公式

知乎

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

TAU输出$\mathbf{w}_k$表示用户对目标排列的排列级兴趣的表示，用作下一个单位的输入。 公式简述：TAU输出||用户兴趣表示||=输入。 （TAU: 目标排列用户兴趣算法） TAU: 是对目标排列的用户兴趣算法，用于表示用户对目标排列的排列级兴趣。该算法将用户对目标排列的偏好作为输入，并输出一个向量，该向量被视为用户对目标排列的排列级兴趣的表示，并用作下一个单位的输入。这有助于实现更有效的用户推荐系统。）

### Context-aware Prediction Unit (CPU)

在上下文感 知预测单元中，我们使用参数共享的 MLP 层来预测每 个排列中每个项目的列表式 pCTR。

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

通过使用PIER框架，可以根据业务需求调整排列得分。每个排列的得分可以通过对输出列表的pCTR进行求和获得，其中$\sigma$为Sigmoid函数。。

### Model Training

训练阶段，首先使用真实曝光的排列训练OCPM，再通过添加对比学习损失联合训练这两个模块。

### Pre-training of OCPM

为了评估前K个排列的准确性。为此，首先对OCPM模型进行预训练，使用在线日志样本。输入为实际排列，标签为点击与否。OCPM损失计算公式如下：

$L_{OCPM} = L_{non-click} + L_{click}$。

其中，$L_{non-click}$表示非点击损失，$L_{click}$表示点击损失。通过优化模型参数，可提高排列评估的准确性。

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

对一个多物品分类模型进行了详细的分析，该模型利用了深度神经网络✦，并在实际应用中取得了显著的效果。在模型中，$i$表示样本的索引，$t$表示展示物品的索引。通过调整神经网络的权重和偏差，该模型能够有效地识别出不同物品。此外，该模型还考虑了物品之间的相关性，从而提高了分类的准确性。

### Joint training of PIER

在联合训练阶段，固定随机向量作为哈希函数和位置编码，FPSM和OCPM共享嵌入，无需更新梯度。嵌入更新时，哈希签名相应更新。提出对比损失确保前K个排列质量提高，主要思想是平均预测点击率差异最大化。联合训练阶段最终损失为结合上述因素。

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

通过样本的抽取方法和梯度反向传播算法✦更新PIER以最小化损失的过程。最终得到的结果可用于评估和优化PIER模型的效果。

## Experiments

知乎

### Dataset

本文在公开数据集和工业数据集上进行了实验，验证了框架的有效性。选择了Avito数据集，包含超过3600万个广告、130万个用户和5300万个搜索请求。使用前T-1个搜索页面作为行为页面，将T-th搜索页面中的广告作为要预测的目标广告。工业数据集来自美团外卖平台，以8:2的比例划分为训练集和测试集。

表 1: Statistics of the datasets.

| Dataset | #Requests | #Users | #Ads |
|---|---|---|---|
| Avito | 53,562,269 | 1,324,103 | 23,562,269 |
| Meituan | 230,525,531 | 3,261,922 | 98,525,537 |

### Evaluation Metrics

在离线实验中，我们针对不同模块使用了不同指标。具体来说，使用AUC评估预测模块的有效性，并采用以下指标评估整个框架。

- 命中率\@1，提出只有在生成方法选择的前K个排列中包含最佳排列时，HR才为1。该结论适用于每个数据。
- 成本问题，具体指不同重排序框架的整体耗时对成本的影响。

通过在线实验，将提出的框架与现有方法进行比较，主要关注CTR、GMV和推理时间等指标。

表 2: The experimental results about AUC and Logloss on two datasets.

| Model | Avito | | Meituan | |
|---|---|---|---|---|
| | AUC | LogLoss | AUC | LogLoss |
| 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** |

### Hyperparameters

论文介绍了三个不同的多层感知器（MLP）模型，它们的隐藏层大小分别为$(128, 64, 32)$、$(60, 32, 20)$和$(50, 20)$。使用的参数包括学习率、优化器、批次大小、嵌入大小、用户行为序列长度等。在Avito数据集和Metuan数据集页面上，模型的全排列长度和K值也进行了设置。

### Offline Experiments For OCPM

### Baselines

将OCPM与逐点和列表式代表性方法进行比较，选用DNN和DCN作为逐点基准，同时选用Kuaishou的PRM、EXTR和Edge-Rerank作为列表式基准。通过这些方法简要介绍可见。

- DNN，这是一种基本深度学习方法，用于CTR预测。DNN通过MLP进行高阶特征交互。

知乎

- PRM算法通过应用自注意力机制[+]调整初始列表以捕获项目之间的相互影响。该方法可以有效地捕捉语义相关性并提高模型性能。
- Edge-Rerank，该方法结合了设备端排名模型和自适应集束搜索，生成上下文感知的重新排名结果。这种方法被用于处理复杂的用户查询和检索场景。具体地，该方法采用了一系列方法和技术，如建立和更新数据结构，求解系统控制理论问题等。因此，对于目标用户的查询，我们可以生成最相关且最具吸引力的结果。总之，Edge-Rerank是一个极具前景的方法，值得进一步研究。

### Performance Comparison

离线实验结果显示，重排序列表模型和基于transformer的模型在CTR预测上表现优异，OCPM在Avito / Metuan数据集上带来了显著的提升。

### Ablation Study

本文在Avito和美团数据集上进行了消融实验以探究OCPM中不同模块的有效性，实验重复5次并报告平均AUC。

- OCPM阻断了全向注意模块，OAU提取物品上下文和特征上下文信息。实验表明，AUC下降了0.0112和0.0048，这表明提取上下文信息对OCPM的建模至关重要，而OAU满足了这一要求。
- OCPM(-TAU)不使用目标注意力单元，而TAU用于计算目标排列与历史行为中每个排列的交互。在Avito/Metuan数据集上，AUC分别下降了0.0057/0.0036，这表明TAU能够捕捉用户历史页面级别的兴趣。

**表 3: Result of ablation experiment on different parts in re-ranking model.**

| Model | Avito | | Meituan | |
|---|---|---|---|---|
| | AUC | LogLoss | AUC | LogLoss |
| OCPM | **0.7320** | 0.0483 | **0.6822** | 0.1917 |
| - OAU | 0.7198 | 0.0483 | 0.6774 | 0.1916 |
| - TAU | 0.7263 | 0.0481 | 0.6795 | 0.1899 |



### Offline Experiments For PIER

### Baselines

本文将FPSM和OCPM结合框架与启发式生成方法进行比较，所有方法使用固定OCPM，但候选排列生成方式不同，简要介绍如下。

- 随机OCPM算法，通过随机选择K个排列作为候选，提高算法的效率和准确性。
- PRS方法基于点击率生成K个候选排列，使用集束搜索方法。
- 全排列与OCPM。我们通过将所有候选排列输入预测模型并基于平均点击率选择K个排列，达到上限。

### Performance Comparison

离线实验结果显示，PIER在寻找最佳排列上取得了很大改进，相较于随机方法和束搜索方法，其在公共数据集和工业数据集上的时间复杂度增加很小。一个合理的解释是FPSM可通过对比损失的指导来选择更好的排列。此外，与全排列方法相比，我们的框架大大降低了时间成本。

知乎

本文研究了PIER框架的两个消融变体，以验证不同单元（时间感知加权、对比损失）的影响。通过实验，我们发现时间感知加权对模型性能有显著影响，而对比损失对模型性能的影响相对较小。该研究为深度学习模型优化提供了有益的见解。

- 提出了一种名为PIER（-时间感知加权）的方法，该方法不将每个行为上的时间感知权重视为同等重要，而是将它们视为同等权重进行处理。这种方法对研究中的数据处理进行了改进，可有效地分析和预测任务性能，同时无需额外的建模步骤。该方法在实际应用中已被证实，是一种可靠的方法。
- PIER（对比损失去除）通过删除对比损失，提高了前K个排列的质量，具体来说，当α=0时效果更佳。

**表 5: Result of ablation experiment on different parts in PIER**

| Settings | Avito | | | Meituan | | |
|---|---|---|---|---|---|---|
| | AUC | HR | Cost (ms) | AUC | HR | Cost (ms) |
| $\alpha = 0$ | 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 | 225.3 |

### Hyperparameter Analysis

i) 当α在一定范围内增加时，OCPM的AUC保持相对较好水平，HR提高；超过一定水平，对比损失对OCPM影响增大，AUC下降，HR指标置信度下降。

ii) 改变K主要影响框架HR和平均成本。增加K可以迅速提高HR性能，但超过一定范围后HR增长缓慢；同时，平均成本随K增加而增加。在实践中应合理设置K值以平衡效果和效率。

### Online Results

比较了PIER与其他模型，并在美团外卖平台上进行了在线A/B测试。结果显示，PIER的点击率和总销售额分别提高了5.46%和5.83%。同时，还关注了时间成本指标，结果表明可以应用于大规模工业场景。全排列超时率增加64.39%，而PIER与束搜索相比，点击率和总销售额均有所提高，超时率影响增加甚少。目前，PIER已在线部署并为主流流量提供服务，为业务增长做出显著贡献。

知乎

提出了一种新型端到端重排序框架PIER，包含两个主要模块FPSM和OCPM。受长期用户行为建模方法启发，应用SimHash从全排列中选择前K个候选项。在OCPM中设计了一种新型全向注意力机制以捕捉排列上下文信息。通过引入对比学习损失联合训练两个模块，引导FPSM生成更好排列。实验和测试表明，PIER显著优于其他重排序基线，已在美团外卖平台上部署。

**原文《PIER: Permutation-Level Interest-Based End-to-End Re-ranking Framework in E-commerce》**

编辑于 2024-02-18 21:53 · IP 属地北京

推荐系统 美团 排序



**推荐阅读**

**多业务建模在美团搜索排序中的实践**

美团技术团... 发表于美团技术博...


**美团搜索粗排优化的探索与实践**

美团技术团... 发表于美团技术博...


**1500亿美团杀入社区团购，兴盛或遇最强对手！**

生鲜榜 发表于生鲜榜


**怎样做美团运营?**

简单