MELODI：用于轻量级排序的相似性–身份解耦

Quoc Phong Nguyen
Amazon International Machine Learning 澳大利亚墨尔本
qphong@amazon.com

Vuong Le Amazon International Machine Learning 澳大利亚墨尔本
levuong@amazon.com

Volodymyr Vaskovych
Amazon International Machine Learning 澳大利亚墨尔本
vaskovyc@amazon.com

Julien Monteil
Amazon International Machine Learning 澳大利亚墨尔本
jul@amazon.com

arXiv:2611.31940v1 [cs.IR] 24 Nov 2026

摘要

针对延迟受限的候选排序，我们提出一种相似度–恒等映射解耦框架，将目标–历史序列兼容性与仅目标信息分离。我们研究了计算池化相似度的两种顺序——先相似度后池化和先池化后相似度——并指出，对于每个固定目标，只要相似度算子关于其历史参数具有线性性，两者即可等价，从而证明成本更低的先池化后相似度形式是合理的。我们将该框架实例化为用于解耦恒等映射与相似度的多头高效轻量算子（MELODI）。其相似度分支将目标条件多头历史池化与单个关于池化历史输入呈线性的无残差交叉层相结合，而其恒等分支则取决于具体问题。在公开基准数据集上，MELODI 的表现达到或超过了所评估的目标感知基线和目标无关基线。在两个大规模工业数据集上，其精确率–召回率曲线下面积（AUPRC）、归一化折损累计增益（NDCG）和召回率均等于或高于所评估的目标感知基线。处理器测量表明，在候选项数量较大时，其延迟低于所评估的候选项条件基线；通过请求内历史复用，延迟还可进一步降低。

CCS 概念

• **信息系统** → **推荐系统**；学习排序； • **计算方法** → 神经网络。

关键词

推荐系统、学习排序、目标感知排序、用户行为建模、交叉注意力、特征交互、高效推理

ACM 参考格式：

Quoc PhongNguyen、VuongLe、Volodymyr Vaskovych 和 Julien Monteil。2027。MELODI：面向轻量级排序的相似性–身份解耦。载于《第 *20*届*ACM*网络搜索与数据挖掘国际会议（*WSDM '27*）论文集》。 ACM，美国纽约州纽约市，17页。https: //doi.org/XXXXXXX.XXXXXXX

特此授权免费制作本作品的全部或部分数字版或纸质副本，用于个人或课堂使用，前提是不得出于营利或商业利益而制作或分发副本，且副本必须附有本声明以及首页上的完整引文。本作品中由作者以外的其他方拥有版权的组成部分，必须予以尊重。经注明出处后，允许编写摘要。以其他方式复制、再出版、发布到服务器或重新分发至列表，均须事先获得明确许可和/或支付费用。请通过 permissions@acm.org 申请许可。*WSDM '27*，香港 © 2027 版权所有归所有者/作者所有。出版权授权给 ACM。ACM ISBN 978-1-4503-XXXX-X/2027/02 https://doi.org/XXXXXXX.XXXXXXX

1 引言

大规模推荐系统通常将候选生成与排序分开：候选生成将数百万个物品缩减至数百或数千个，随后由更复杂的模型对候选列表进行排序 [6, 31]。候选生成可以分别编码用户或查询以及物品表示，并使用简单的相似度评分器 [12, 31]。这种分解方式使物品表示能够预先计算，并可在目录规模下高效检索 [31]。相比之下，排序针对较小的候选集进行，因此可以采用针对候选项的早期融合，将每个入围候选项与用户历史和请求特征联合处理 [4, 35], ，不过实时服务仍然限制了对每个候选项执行的计算。

目标感知编码器利用这一针对每个候选项的计算预算来区分用户同时存在的不同兴趣。假设某位购物者最近的活动在跑步装备和摄影器材之间交替进行。在对跑鞋进行评分时，模型应重点关注体育相关互动；而在对相机镜头进行评分时，则应重点关注摄影相关互动。候选无关向量会共同表示这两种兴趣，因此可能无法识别哪些事件支持每个候选项。不带目标感知的模型，例如自注意力序列推荐器（SASRec） [13] ，会为每个候选项复用同一个历史序列摘要。目标感知模型，例如深度兴趣网络（DIN） [35], 、深度兴趣演化网络（DIEN） [34],、深度会话兴趣网络（DSIN） [8], 和基于搜索的兴趣模型（SIM） [20] ，则会让部分计算以每个候选项为条件。目标感知 Transformer，例如

行为序列 *Transformer* （BST）将每个候选项与历史序列联合编码，因此对于每个候选项都必须分别评估其编码器 [4]。DIN 则会为每个（历史项，候选项）对评估一个多层感知机 （MLP）。随着交互历史和候选集规模的增长，目标感知算子必须能够高效扩展。

候选条件化决定了对于每个目标而言哪些历史事件具有重要性，但它并未消除另一项建模要求：目标感知排序必须将目标恒等映射与用户–目标兼容性保留为彼此独立的信号。目标的 恒等映射携带仅目标信息，例如基准吸引力和目标内部的特征交互，而其与相关历史的 相似度 则代表用户—目标兼容性。例如，在圣诞节期间，节日装饰可能会变得广受欢迎，因为评分器可以将其仅目标恒等映射与日历上下文结合起来，而不要求购物者的历史中存在匹配的季节性模式。兼容性则是另一种关系：如果某位购物者近期的活动包括油漆、装置和工具，那么电钻可能会获得较高评分，因为它与该购物者的装修历史相匹配。DIN

可以将目标单独提供给其评分器，但其加权池化会将目标条件化的历史内容与兼容性结合在同一个向量中。相反，我们将这些作用解耦：相似度分支计算显式的用户–目标交互特征，恒等分支单独处理目标，而评分器则将这两种信号与任何上下文特征结合起来。

我们提出了一种目标感知排序框架，将用户–目标相似度与目标恒等映射分离，并通过以下三项贡献保留可复用的历史序列路径：

• **相似度–身份框架**。我们形式化了先相似度后池化与先池化后相似度（SP/PS），并陈述了证明该设计合理性的基本条件：当相似度算子关于其历史参数为线性时，更廉价的 PS 顺序在目标感知池化后能够准确恢复 SP 输出（命题3.3）。

• **MELODI架构**。我们将该框架实例化为用于解耦恒等映射与相似度的多头高效轻量算子（MELODI）。其相似度分支将多头点积交叉注意力 [28] 与一个基于*Deep &Cross Network v2*（DCN-v2）的单个交叉层相结合 [29]。这两种基本组件本身都已存在；新增元素包括：一种关于池化历史为线性的双输入交叉且不含残差、作用于共享专家的逐头门控，以及一个独立的恒等分支。三者共同使历史序列路径与目标无关，并可在不同候选项之间复用（第 4 节）。

• **公开与工业场景下的评估**。我们在两个公开数据集和两个工业数据集上评估排序质量、组件变体、分析性浮点运算（FLOP）次数以及处理器延迟（第 5 节，补充材料第 B 和 D 节）。预先指定的公开配置在表 2 的所有报告指标上均高于每个基线。在这两种场景下，对于较大的候选集，MELODI 的处理器延迟低于所评估的候选条件基线；而在公开基准中，当候选集大小不超过 $C = 500$ 时，DIN 仍然更快。组件对比界定了测量结果支持哪些默认配置。

2 相关工作

用于排序的序列模型。在下游排序阶段，可复用的目标无关编码器（如 GRU4Rec [11]、SASRec [13], 和 BERT4Rec [27]）会生成一个历史序列表示，并将候选项交互限制在评分器中。多通道用户兴趣记忆网络同样会维护与候选项无关的长历史记忆，并避免在请求时对候选项与行为序列进行交互 [19]。

候选条件池化会针对每个目标改变历史序列表示。DIN为每个（历史项、候选项）对计算MLP激活，并使用标量权重池化出一个目标感知的历史序列 [35]DIEN通过目标条件的门控循环单元演化兴趣 [34], DSIN在自注意力和双向长短期记忆网络之后激活会话表示 [8],而SIM执行目标条件的粗略搜索和精确搜索 [20]。两阶段兴趣网络（TWIN）面向长达 10^4–10^5 个行为的终身序列，其长度远超我们的50个行为，并缓存固有于物品的特征的投影，使目标注意力的计算成本可控 [3]。AutoAttention以原始字段嵌入上的点积取代DIN的MLP激活，并学习全局共享的字段对权重，这些权重会在训练期间被剪枝 [33]。协同行为网络（CAN）将目标参数化的MLP应用于每个行为，并对输出进行求和池化 [2], 这是一种先计算相似度再池化的算子，其关于历史项是非线性的。

联合目标–历史编码器将候选条件化移入序列模型内部。BST 对组合后的目标–历史序列应用自注意力，因此每个候选项都需要重新运行完整编码器 [4]。在排序中，分层序列转导单元 (HSTU) 可在一次处理中对候选项微批次进行评分，同时屏蔽候选项之间的注意力，并缓存历史序列的键和值 [32]。我们提出的 MELODI 同样复用历史序列的键和值，但采用单个交叉注意力层，对不含自注意力的历史序列进行编码，并使用目标依赖的逐头门控；其交叉层的计算量与池化后的历史序列输入呈线性关系。

特征交互与相似性–身份解耦。 因子分解机 [22], Wide& Deep [5], DeepFM [10], 和 DCN-v2 [29] 均处理扁平特征。我们采用具有两个输入（$\mathbf{u}$，$\mathbf{t}$）的 DCN-v2 式交叉层、块对角专家和逐头门控。相关方法改进了不同阶段：卷积神经网络特征生成 (FGCNN) [16]通过在扁平特征嵌入上进行卷积来生成交互，而 Feng 等人提出的解耦嵌入方法 [7] 将注意力和表示嵌入表分离，但仍会将检索到的行为聚合为一个表示。在我们提出的 MELODI 中，评分器接收一个显式的用户–目标交互向量，以及一个独立的仅目标向量。

3 相似性–身份解耦：框架

3.1 问题设置

我们研究排序阶段，并将候选集视为已知，而不对候选生成进行建模。排序器在单请求延迟预算内，基于相同的近期交互历史对每个候选项进行评分。设历史序列为$\mathbf{X}$ $= [\ _1 \quad _S]^\top \in \mathbf{x}$，……，$\mathbb{R}^{S\times D_x}$ $= \ _1 \ldots \ _C \subset \mathbb{R}^{D_t} \quad D_x$ $\mathbf{x}$，候选集为 C $\{\mathbf{t}$，…，$\mathbf{t}\}$。将和 D_t 区分开来，使该框架能够涵盖池化历史输入与目标输入维度不同的相似度算子。排序器为每个候选项输出一个标量。

设计问题在于构造一种中间表示，使其信息丰富，同时能够以较低成本为每个候选项进行计算。我们将其分为相似度信号（取决于$\mathbf{X}$ 和$\mathbf{t}_c$）以及恒等映射信号（仅取决于$\mathbf{t}_c$）。相似度分支是第 3.2–3.3 节的重点。恒等分支可以采用原始目标嵌入、目标多层感知机、DCN-v2 目标交叉，或将它们组合起来，以捕捉目标自身的信息，例如基准吸引力和目标内部的特征交互。

该框架不依赖于嵌入源。物品表示可以来自内容编码器、协同过滤或自监督序列模型。在相似度算子之前，可以分别对每个原始历史序列或目标嵌入应用非线性适配器。定义 3.2 随后适用于经过适配的向量。用户、上下文和其他请求级特征可以输入最终评分器或单独的特征交叉模块，而不会改变相似度分支是否具有历史线性。

3.2 相似度分支：池化相似度的两种构造

对于给定的目标$\mathbf{t} \in \mathbb{R}^{D_t}$（以下省略候选索引 c），令 $\boldsymbol{\alpha} \in \Delta^{S-1}$ 表示一个非负注意力权重向量

表1：框架和架构中使用的数学符号。

符号	描述	符号	描述
框架			
$\mathcal{C} = \{\mathbf{t_c}\}_{c=1}^{C}$	大小为 C的候选集	$\mathbf{X} = [\mathbf{x}_i]_{i=1}^{S} \in \mathbb{R}^{S \times D_x}$	长度为 S的历史矩阵
$\mathbf{x}_i \in \mathbb{R}^{D_x}$，$\mathbf{t} \in \mathbb{R}^{D_t}$	历史项和通用目标；$\mathbf{t}$ 上的候选索引为已丢弃	D_x，D_t	框架历史和目标宽度
$\boldsymbol{\alpha} = [\alpha_i]_{i=1}^{S} \in \Delta^{S-1}$	在 S历史项目上的非负注意力分布 $\mathbf{u}(\mathbf{X}, \mathbf{t}) \in \mathbb{R}^{D_x}$		目标感知池化历史
$\phi : \mathbb{R}^{D_x} \times \mathbb{R}^{D_t} \to \mathbb{R}^{M}$	相似度算子；在调用 SP/PS 等价性时关于其历史参数是线性的（定义 3.2）	M	相似度算子输出宽度
$\boldsymbol{\psi}_{\mathrm{SP}}$ $\boldsymbol{\psi}_{\mathrm{PS}} \in \mathbb{R}^{M}$	先相似度后池化与先池化后相似度的顺序	$\mathbf{A}(\mathbf{t}) \in \mathbb{R}^{M \times D_x}$	$\phi_{\mathbf{A}(\mathbf{x}\ ,\mathbf{t})}$ 中的矩阵 $= \mathbf{A}(\mathbf{t})\mathbf{x}$
$\boldsymbol{\eta} : \mathbb{R}^{D_t} \to \mathbb{R}^{D_x}$	$\phi(\mathbf{x}, \mathbf{t})$ 中的目标映射 $= \mathbf{x} \odot \boldsymbol{\eta}(\mathbf{t})$	$\odot$	逐元素乘积
$\mathbf{M}$，$\mathbf{M}(\mathbf{t}) \in \mathbb{R}^{D_x \times D_t}$	框架示例中的线性且依赖目标的映射 $\mathbf{U}_k \in \mathbb{R}^{D_x \times r}$，$\mathbf{V}_k \in \mathbb{R}^{r \times D_t}$		专家的框架因子 k；$\mathbf{M}$（$\mathbf{t}$）将它们混合，使用 $g_k(\mathbf{t})$
MELODI 架构			
B	批量大小	D_{raw}	原始预训练嵌入宽度
D	阶段1后的物品嵌入宽度	H, d	注意力头数量和每头宽度
$D_u = Hd$	拼接后的注意力与相似度宽度；不必相等	K, r	低秩专家数量和专家秩
$\mathbf{X}_{\text{原始}}$，$\mathbf{T}_{\text{原始}}$	原始历史序列和候选项，位于 $\mathbb{R}^{S \times D_{\text{原始}}}$ 和 $\mathbb{R}^{C \times D_{\text{原始}}}$	$\tilde{\mathbf{X}}$，$\mathbf{x}_i$，$\tilde{\mathbf{t}}$	投影后的、归一化历史、历史项，以及给定宽度下的目标 D
$\mathbf{W}_q$，$\mathbf{W}_k$，$\mathbf{W}_v$，$\mathbf{W}_g$，h	查询、键、共享值，以及逐头门控投影矩阵	$\mathbf{Q}$，$\mathbf{K}$，$\mathbf{V}$	投影后的查询、键和值张量；下标 h 选择一个 head
$\boldsymbol{\alpha}_h \in \mathbb{R}^{S}$	头 h中一个候选项的注意力权重；批处理形状为 $C \times S$	$\mathbf{u}_h \in \mathbb{R}^{d}$	头 h中每个候选项的池化历史
$\mathbf{u} = [\mathbf{u}_1; \ldots; \mathbf{u}_H] \in \mathbb{R}^{D_u}$	拼接后的池化历史	$\mathbf{v}_t$，$\mathbf{v}_t^{(h)}$	$\mathbb{R}^{D_u}$ 中$\mathbf{W}_v$ 下的目标及其在 $\mathbb{R}^{d}$ 中的头-h 切片
$\mathbf{U}_k \in \mathbb{R}^{d \times r}$，$\mathbf{V}_k \in \mathbb{R}^{r \times d}$	MELODI 针对专家 k的因子；它们的乘积的秩至多为 r	$\mathbf{g}_h \in \mathbb{R}^{K}$，$g_{h,k}$	头 h 及其专家-k 分量的门控向量
$\boldsymbol{\eta}_h$，$\mathbf{b}_h \in \mathbb{R}^{d}$	每个头的目标映射输出和偏差；第3阶段使用 $\boldsymbol{\eta} : \mathbb{R}^{D} \to \mathbb{R}^{D_u}$	$\mathbf{c}_h \in \mathbb{R}^{d}$	每个头的相似度输出
$\mathbf{c} \in \mathbb{R}^{D_u}$	拼接相似度信号	$\tilde{\mathbf{t}}\mathrm{id} \in \mathbb{R}^{D}$	恒等分支目标表示
服务和基线			
$T_{\text{共享}}$ $T_{\text{每-候选}}$	仅历史和逐候选服务成本	$\mathbf{h}$	DIN 激活单元中的历史项表示

满足 $\sum_i \alpha_i = 1$。权重 $\boldsymbol{\alpha}$可以依赖于（$\mathbf{X}$，$\mathbf{t}$），且分析允许使用任意注意力函数。令 ϕ：$\mathbb{R}^{D_x} \times \mathbb{R}^{D_t} \to \mathbb{R}^{M}$ 表示一个相似度算子。我们将先相似度后池化 （SP）和先池化后相似度 （PS）定义为

$$\text{(SP):}\quad \boldsymbol{\psi}_{\mathrm{SP}}(\mathbf{X}, \mathbf{t}) \triangleq \sum_{i=1}^{S} \alpha_i\, \phi(\mathbf{x}_i, \mathbf{t}) \in \mathbb{R}^{M}, \tag{1}$$

$$\text{(PS):}\quad \boldsymbol{\psi}_{\mathrm{PS}}(\mathbf{X}, \mathbf{t}) \triangleq \phi\big(\mathbf{u}(\mathbf{X}, \mathbf{t}),\ \mathbf{t}\big) \in \mathbb{R}^{M}, \tag{2}$$

其中 $\mathbf{u}(\mathbf{X}, \mathbf{t}) \triangleq \sum_{i=1}^{S} \alpha_i \mathbf{x}_i \in \mathbb{R}^{D_x}$ 是目标感知池化历史向量。SP 在聚合前应用 ϕ，而 PS 则先构造 $\mathbf{u}$，再调用一次 ϕ。我们显式写出 $\mathbf{u}(\mathbf{X}, \mathbf{t})$ 的两个参数，因为目标依赖注意力会改变正在评分的候选项对应的池化历史。

框架中的 DIN。在其历史池化阶段，DIN 对应于 PS 形式，其中 $\phi(\mathbf{u}, \mathbf{t}) \triangleq \mathbf{u}$。根据定义 3.2，该恒等算子是历史线性的，但 DIN 从原始历史序列到最终输出并非线性。它的激活单元会为每个历史项计算非线性、目标依赖的注意力权重，其下游评分器也可能是非线性的。SP/PS 条件并未约束这两个组成部分，因此无法消除 DIN 逐项激活的成本。由于 $\mathbf{u}$ 结合了历史内容与目标条件化注意力权重，兼容性仍是隐式的，而非作为独立特征显现。DIN 还为每个历史项分配一个标量权重，使所有维度共享相同的目标条件化注意力分布。相比之下，MELODI 使用 H 注意力头（第 4 节）学习不同的池化分布，然后对每个目标感知池化历史应用显式的目标依赖历史线性相似度算子 ϕ（第 3.3 节）。

注释 *3.1*（两种构造之间的保真度—效率权衡）。这两种构造存在权衡。**SP 更适合保证建模保真度**：ϕ 会单独考察每个历史项 $\mathbf{x}_i$，并在聚合之前将其与目标进行比较，因此输出反映了逐项相似度。**PS 更适合提高计算效率**：对于每个（用户、目标）对，它只需评估一次算子，而非 S，从而将评估次数减少至原来的 S。这一权衡促使我们寻找一类算子，使两种顺序产生相同的输出，从而让成本更低的 PS 形式能够复现保真度更高的 SP 信号。

SP/PS 等价性的充分条件。所需条件仅涉及 ϕ 的第一个参数，即历史参数；它不限制 ϕ 对 $\mathbf{t}$ 的依赖方式。

定义 *3.2*（历史线性相似度）。对于每个固定的 $\mathbf{t}$，ϕ是历史线性的，当 $\phi(\cdot, \mathbf{t})$ 关于其历史参数是线性的。

此性质仅适用于相似度算子 ϕ。它不要求注意力权重 $\boldsymbol{\alpha}$、池化历史映射 $\mathbf{u}(\mathbf{X}, \mathbf{t})$、任何嵌入适配器或最终评分器关于整个原始交互历史$\mathbf{X}$ 都是线性的。

命题 3.3（历史线性条件下 SP/PS 的等价性）。对于任意历史序列 $\mathbf{X}$、目标 $\mathbf{t}$ 和注意力分布 $\boldsymbol{\alpha} \in \Delta^{S-1}$，历史线性相似度算子满足 $\boldsymbol{\psi}_{\mathrm{SP}}$（$\mathbf{X}$、$\mathbf{t}$）$= \boldsymbol{\psi}_{\mathrm{PS}}$（$\mathbf{X}$、$\mathbf{t}$）。

Proof。对于固定的$\mathbf{t}$，历史参数具有线性性 给出

$$\boldsymbol{\psi}_{\mathrm{SP}} = \sum_{i=1}^{S} \alpha_i \phi(\mathbf{x}_i, \mathbf{t}) = \phi\Big(\sum_{i=1}^{S} \alpha_i \mathbf{x}_i, \mathbf{t}\Big) = \boldsymbol{\psi}_{\mathrm{PS}}. \tag{3}$$

□

命题 3.3 允许采用依赖候选项的注意力和任意非线性的目标映射，因为只有历史槽位必须保持线性。因此，PS 顺序可以通过对每个 ϕ 候选项调用一次来恢复 SP 池化相似度。该命题直接源于线性性，其作用体现在架构层面：它确定了能够实现这种精确重排序的相似度算子。

3.3 历史线性相似度算子示例

最一般的历史线性算子形式为 $\phi_{\mathrm{A}}(\mathbf{x},\ \mathbf{t})\ =\mathbf{A(t)}\,\mathbf{x}$，其中矩阵值函数 $\mathbf{A}:\ \mathbb{R}^{D_t}\ \rightarrow\ \mathbb{R}^{M\times D_x}$ 任意。目标信息仅通过 $\mathbf{A(t)}$ 进入，因此无论 A 如何依赖于 $\mathbf{t}$，都有 $\sum_i\ \alpha_i\mathbf{A(t)x}_i\ =\mathbf{A(t)u}$。我们重点讨论逐元素特化情形 $M=D_x$，

$$\phi(\mathbf{x},\mathbf{t})\ =\ \mathbf{x}\odot\boldsymbol{\eta}(\mathbf{t})\ \in\ \mathbb{R}^{D_x},\tag{4}$$

其中 $\mathbf{A(t)}\ =\mathrm{diag}(\boldsymbol{\eta}(\mathbf{t}))$，对于目标映射 $\boldsymbol{\eta}$：$\mathbb{R}^{D_t}\ \rightarrow\mathbb{R}^{D_x}$。下面四种情况逐步增加该映射的复杂度。

(1) 直通式目标映射： $\boldsymbol{\eta}(\mathbf{t})\ =\mathbf{t}$。当 $D_x=D_t$ 时，$\phi(\mathbf{x},\ \mathbf{t})\ =\ \mathbf{x}\odot\mathbf{t}$，其逐元素求和可恢复点积 $\mathbf{u}^{\top}\mathbf{t}$。下游多层感知机也可以接收完整向量 $\mathbf{u}\odot\mathbf{t}$，并学习如何为其各个维度分配权重。

(2) 线性目标映射： $\boldsymbol{\eta}(\mathbf{t})\ =\mathbf{Mt}$。$\phi_{\mathrm{cross}}(\mathbf{x},\ \mathbf{t})\ =\mathbf{x}\ \odot(\mathbf{Mt})$，其中 $\mathbf{M}\ \in\mathbb{R}^{D_x\times D_t}$ 为可学习参数。将其各元素相加，得到双线性形式 $\mathbf{u}^{\top}\mathbf{Mt}$。

(3) 输入相关混合（MELODI 的选择）： $\boldsymbol{\eta}(\mathbf{t})\ =\mathbf{M(t)}\,\mathbf{t}$。其中 $\mathbf{M(t)}=\ \sum_k\ g_k(\mathbf{t})\mathbf{U}_k\mathbf{V}_k$，目标相关矩阵是由 K 低秩专家（$\mathbf{U}_k$，$\mathbf{V}_k$）构成的 softmax 门控混合，其门控权重为 $g_k(\mathbf{t})$。这一设计受到 DCN-v2 [29] 的低秩混合专家交叉层启发。尽管存在 softmax 非线性，$\boldsymbol{\eta}$ 仅作用于 $\mathbf{t}$，因此该算子仍具有历史线性，且 PS 等价性得以保留。我们将该算子独立应用于每个 H 注意力头，并使用逐头门控权重 $g_{h,\ k}(\mathbf{t})$，使每个注意力头都产生自身的目标相关混合和相似度向量。其架构实现详见第 4 节。

(4) 任意非线性目标映射，例如， $\boldsymbol{\eta}$ = 多层感知机。仅依赖 $\mathbf{t}$ 的任意非线性函数都能保持历史线性，因为 $\boldsymbol{\eta}(\mathbf{t})$ 在对历史项求和时是常数。

备注 *3.4*（表达能力—效率权衡）。允许 ϕ 以非线性方式依赖其历史参数会扩大算子类，但此时 SP 与 PS 通常不再等价，因为非线性运算与池化不可交换。因此，要保持二者等价，就必须限制 ϕ 具有历史线性，而非限制整个模型。要实现 SP 语义，非线性算子必须针对每个历史项分别调用一次。这种限制以牺牲 ϕ 历史槽位上的表达能力为代价，换取成本更低的 PS 形式，适用于延迟受限排序（第 5.2 节）。

4 MELODI架构

我们提出的 MELODI 接收原始交互历史 $\mathbf{X}^{\mathrm{raw}}\ \in\mathbb{R}^{S\times D_{\mathrm{raw}}}$ 和 C 原始候选嵌入 $\mathbf{T}^{\mathrm{raw}}\in\mathbb{R}^{C\times D_{\mathrm{raw}}}$，并为每个候选项生成一个分数。其相似度分支和恒等分支实现了第 3 节的框架。图 1 展示了整体设计。相似度分支由以下阶段 1–3 组成，数据流见图 2，完整公式见补充材料 C 节。恒等分支将每个目标映射为 $\tilde{\mathbf{t}}_{\mathrm{id}}\in\mathbb{R}^{D}$，而相似度分支输出 $\mathbf{c}\in\mathbb{R}^{D_u}$。该框架允许采用其他仅作用于目标的变换，但所有主要配置都使用单个 DCN-v2 目标自交叉。组件研究会明确改变这一选择（第 5.3 节和补充材料 D 节）。最后由一个多层感知机对 $[\mathbf{c};\tilde{\mathbf{t}}_{\mathrm{id}}]$ 进行评分。

阶段1：嵌入与归一化。将冻结的预训练项目嵌入投影到维度 D，然后对历史序列和目标应用共享的层归一化 LN，得到 $\tilde{\mathbf{x}}_i,\ \ \tilde{\mathbf{t}}_c\in\mathbb{R}^D$。只有历史序列会添加位置编码。共享归一化使查询和键在点积注意力中处于一致的尺度上，因为冻结嵌入的幅值各不相同。

阶段2：多头点积交叉注意力。每个候选项通过带有查询、键和值投影的多头交叉注意力关注用户历史。这些投影将阶段1的宽度 D 映射为 H 个宽度为 d 的注意力头，生成逐头池化历史 $\mathbf{u}_h\ \in\ \mathbb{R}^d$，以及宽度为 $D_u\ =\ Hd$ 的拼接结果。该公式支持 $D_u\neq D$。此阶段以批量矩阵乘法取代DIN的MLP激活单元，后者会显式构造一个 $[B,S,C,4D]$ 的成对张量。第5.1节和第5.2节测量由此产生的延迟，补充材料第B节报告了理论浮点运算次数。

阶段3：逐头门控交叉相似度。拼接后的池化历史 $\mathbf{u}\ \in\ \mathbb{R}^{D_u}$ 通过一种DCN-v2风格的算子与目标投影 $\mathbf{v}_t\ =\mathbf{W}_v\tilde{\mathbf{t}}\ \in\ \mathbb{R}^{D_u}$ 结合；对于每个固定目标，该算子关于 $\mathbf{u}$ 呈线性。与标准DCN-v2 [29]相比，该算子使用两个不同的输入，而非对同一个向量进行自交叉；在各注意力头之间共享 K 个低秩 d 维专家；为每个注意力头分配一个独立的 softmax门控，该门控根据完整的 $\tilde{\mathbf{t}}\ \in\mathbb{R}^D$ 计算；并省略残差。因此，专家矩阵包含 $2Kdr$ 个参数，而门控包含 HKD。补充材料第C节在一般设置和等宽设置下，将此参数量与DCN-v2进行了比较。

输出 $\mathbf{c}\ \in\mathbb{R}^{D_u}$ 实现了 $\phi(\mathbf{u}\ \tilde{\mathbf{t}})\ =\mathbf{u}\ \odot\boldsymbol{\eta}(\tilde{\mathbf{t}})$，并采用块对角目标映射。softmax 门控和专家混合使该映射关于目标项呈非线性，但关于 $\mathbf{u}$ 不呈非线性，因此保留了 PS 等价性。在评估的递归下，一层仍保留这一性质。省略其残差可使 $\mathbf{c}$ 保持为纯粹的交互信号，不过添加一个 $+\mathbf{u}$ 残差也仍将保持历史线性。更深的 $\tilde{}$ 特征交互模块可以在评分器中接在 $[\mathbf{c};\mathbf{t}_{\mathrm{id}}]$ 之后，而不会改变等价性。补充材料C节给出了相关公式，并与 DCN-v2 进行了并列比较。

备注 *4.1*（推理效率）。在历史线性条件下，PS 将相似度算子调用从 S 减少为每个候选项调用一次。MELODI 的交叉注意力还支持请求内历史复用。候选项展平评估会将共享历史复制到 C 候选行中，并为每一行重新编码。复用路径则只计算一次嵌入、投影、位置编码、层归一化、$\mathbf{K}$和$\mathbf{V}$，将所得张量扩展到各个候选项，并在请求结束后丢弃这些张量。公开基准测试和工业基准测试会在一次请求内比较这两条路径（第5.1节和第5.2节），补充材料A节解释了为什么复用发生在组装位置化历史之后，以及为什么追加目标项的Transformer无法复用该历史。

注释 *4.2*（跨候选项摊销）。在请求内局部复用下，每请求计算可分为仅依赖历史的计算（执行一次）和依赖目标项的计算（针对每个候选项执行）。它们的开销分别为 T_{shared} 和 $T_{\text{per-cand}}$。候选项展平评估执行 $C(T_{\mathrm{shared}}+T_{\text{per-cand}})$

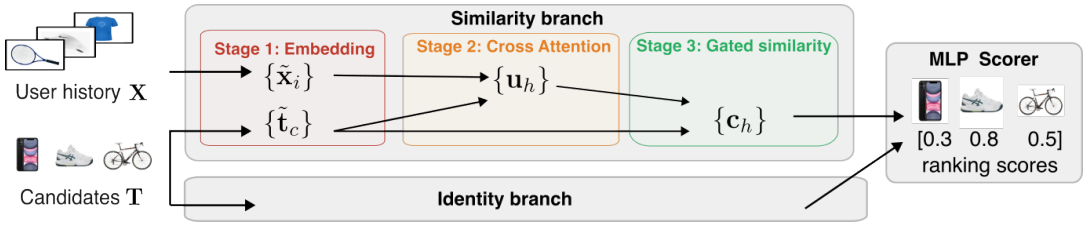

图1：MELODI 的双信号设计。上排：相似性信号分支应用共享嵌入投影和层归一化 (LN) [1]（阶段1），通过多头点积交叉注意力生成$\{\mathbf{u}_h\}$（阶段2），再通过不含残差连接的逐头门控交叉层生成 c（第3阶段）。下排：身份信号分支（虚线）应用问题相关的仅目标变换。

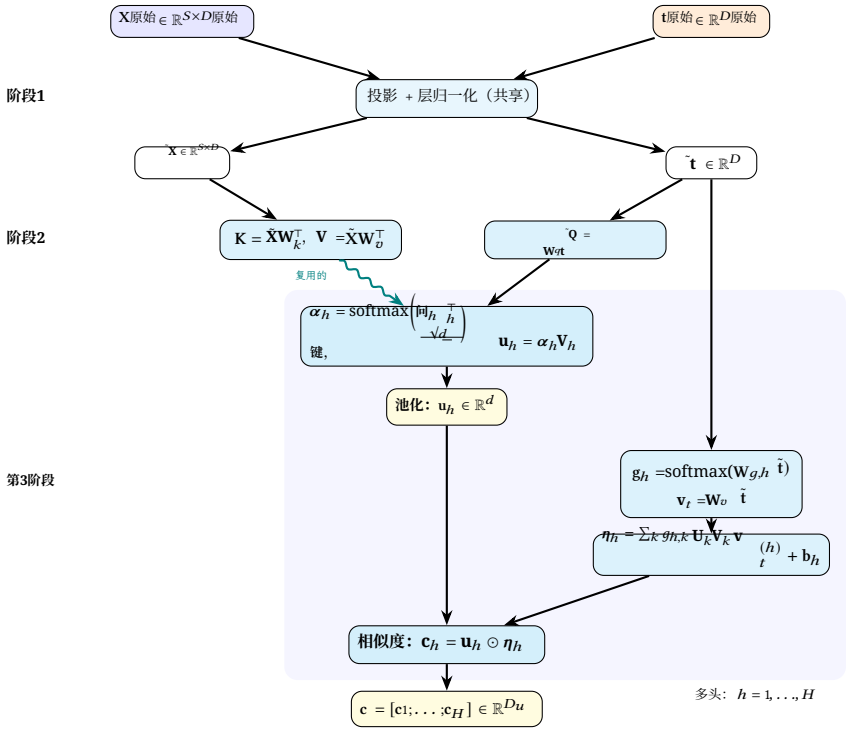

图 2：相似度分支的详细数据流。阶段1以宽度 D 生成 X 和 t。阶段2将它们投影到宽度为 d 的 H 个注意力头，并计算池化历史 $\mathbf{u}_h$。第3阶段将目标投影为 $\mathbf{v}_t \in \mathbb{R}^{D_u}$，其中 $D_u = Hd$，并应用逐头门控专家，生成 η_h 和 $\mathbf{c}_h$。青绿色波浪箭头标记可缓存的仅历史键和值（备注 4.1）。

计算，而复用执行 $T_{\text{shared}} + CT_{\text{per-cand}}$。其摊销后的单候选项开销为 $T_{\text{shared}}/C + T_{\text{per-cand}}$。随着 C 增大，该开销趋近于 $T_{\text{per-cand}}$。在 Industrial-B 上，当 $C = 1,000$ 时，共享历史计算约占复用路径延迟的 3%。

5 实验

我们围绕四个研究问题组织评估：

- **研究问题1：公开排名质量**。与目标感知和与目标无关的基线相比，固定的预先指定 MELODI 配置是否能提升排名指标?
- **RQ2：候选集扩展**。正如备注 4.1 和 4.2 所预期的那样，候选数量和请求内历史复用如何影响延迟?
- **RQ3：工业迁移**。在记录的工业请求上，公开数据中的质量—延迟趋势是否仍然存在?
- **研究问题4：组件证据**。哪些组件比较支持固定的架构默认设置?

正文规定了解释结果所需的比较设计，而补充材料 G 节和 F 节提供了复现所需的架构、优化和计时细节。

5.1 公开数据集评估

数据集与评测协议。我们使用 Steam [13]（281,354 名用户，11,961 件物品）和 2019 MovieLens-25M (ML-25M) 版本 [9]（162,541 名用户，32,720 件物品），并采用相同的逐用户留一法协议。我们保留所有 Steam 前缀，得到 2,706,396 个训练示例，

并保留每位用户最近的 20 个 ML-25M 前缀，得到 3,226,917 个训练示例。每个测试项均与 99 个唯一负样本进行排序比较，这些负样本与用户的完整序列不重合，且在各模型间保持一致。这种采样评测协议很常见，但采样指标可能会得出与全目录排序不同的模型排序 [15]。因此，我们还对每个模型的一次训练运行进行重新评分，不重新训练；评分时分别与通过 MiniLM 最近邻搜索检索出的 999 个负样本以及全目录进行比较，并将并列情况计为测试项落后（表4）。详情见补充材料第 G 节。

模型输入。 所有模型共用冻结的384维物品文本嵌入（补充材料第G节）、一个可训练的投影至 D=64，以及最多截取50个物品的交互历史。

基线嵌入和Transformer模型。 主要基线共用嵌入投影、可学习位置编码和层归一化。**SASRec** [13]与目标无关。一个双层因果自注意力Transformer将历史序列编码为固定向量，评分采用点积$\mathbf{u}^\top\mathbf{t}$。目标感知*Transformer*（TA-Transformer）基线将每个候选项附加到历史序列中，应用一层具有一个64维头的双向自注意力，并对候选位置输出进行评分。我们预先设定该深度和头数，以匹配工业对照模型。补充材料第G节记录了其从公开设置适配到工业设置的过程。

DIN基线和目标归一化。**DIN** [35]使用目标感知MLP激活单元计算每个物品的注意力权重，将历史序列池化为一个向量$\mathbf{u}$，并通过MLP($[\mathbf{u},\mathbf{t}]$)进行评分。**DIN + DCN-v2** 通过在（$\mathbf{u},\mathbf{t}$）上使用DCN-v2双线性交叉网络扩展DIN。将其输出记为$\mathbf{c}_{\mathrm{DCN}}$，模型通过MLP($[\mathbf{u},\mathbf{t},\mathbf{c}_{\mathrm{DCN}}]$)进行评分。作为主要结果报告的SASRec和两种DIN变体均对目标应用共享的层归一化。表3还报告了未对目标进行此归一化的SASRec。我们还评估了CAN [2], , 其目标参数化MLP作用于每个历史项；以及DIEN [34], , 其目标条件化门控循环单元用于建模兴趣演化（表4）。在Steam上，当 C = 1,000时，它们的请求级延迟约为2.0毫秒和16.4毫秒。

固定的 MELODI 配置。 我们预先指定的配置使用一个包含 4 个专家的无残差交叉层、一个 DCN-v2 目标自交叉，以及一个仅接收交叉输出的多层感知机。我们在两个数据集上均使用该配置，不针对每个数据集单独进行选择。

SASRec 目标归一化。 将共享的 LN 应用于目标后，SASRec 在 Steam (0.3736 → 0.4311) 上的平均倒数排名 (MRR) 提升了 15。4%，在 ML-25M (0.4658 → 0.6597) 上提升了 41。6%。如果不进行目标归一化，点积会出现与目标无关性无关的尺度不匹配。这一修正的影响大于大多数测得的架构效果，因此我们在主要比较中采用归一化变体，并在表 3 中保留两种变体。

训练、模型选择和指标。 所有模型均使用贝叶斯个性化排序 (BPR) [23], , 检查点选择使用验证集 MRR。我们报告命中率 (HIT)@{1，5、10}、NDCG@{5、10} 和 MRR。表 2 对三个训练随机种子取平均；正文中的其他主要排序结果均针对每个模型使用一次训练运行。补充材料 G 节给出了优化、批处理和模型规模详情（表17）。

研究问题1：公开排名质量。在三个训练随机种子下（表 2），预先指定的 MELODI 配置在两个数据集的每项指标上都取得了最高均值；在这五个模型中，它还在每个随机种子下的表3六项指标上均排名最高。其平均倒数排名相较于 DIN，在 Steam 上提升 +3.5%，在 ML-25M 上提升 +4.4%。在表 2 的三个指标中，其相较于最强基线的优势在 Steam 上为（0.0089–0.0111），在 ML-25M 上为（0.0067–0.0155）。对于每个模型的一次训练运行，基于测试用户的 MELODI 与各基线之差的配对自助法95%区间（10,000 次重采样）在两个数据集的 HIT@10、NDCG@10 和平均倒数排名上均不包含零。TA-Transformer 是一种具有竞争力的中游目标感知基线：其平均倒数排名在 Steam 上比归一化SASRec 高 0.0146，而在 ML-25M 上比其低 0.0083；但在每项指标上，它仍低于两个 DIN变体和 MELODI。相较于 TA-Transformer，MELODI 的平均倒数排名在 Steam 上提高 0.0295，在 ML-25M 上提高 0.0505。参数量匹配的TA-Transformer 在 Steam 的六项采样指标上数值均更高，但 16× 速度慢了 C = 1,000（补充材料E节）。

测试配置间的一致性。在{4、8}个专家和{1、2}层的四种组合中（表3），所有模型变体在Steam上的平均倒数排名均比最强基线高0.0104至0.0192，在ML-25M上高0.0183至0.0278。在两个数据集上，四种变体的HIT@10和NDCG@10也均超过该基线。根据验证集MRR进行选择时，两个数据集都会选出8专家、2层的变体。该变体在ML-25M上具有最高的测试集平均倒数排名。在Steam上，其0.4801的测试集平均倒数排名比8专家、1层的变体低0.0007。

更难的候选集。在表4所列的模型中，MELODI在两个数据集上使用困难负样本时的平均倒数排名最高，其相对于CAN的最小优势是在Steam上，为 +0.0019 。在全量排序下，MELODI在Steam上的平均倒数排名比DIN高0.0009，在ML-25M上比DIN + DCN-v2高0.0079，尽管DIN在Steam上的HIT@1更高（0.0186对0.0177）。在两个数据集的全部三种协议下，CAN和DIEN的表现仍低于MELODI，并且MELODI减去所列基线的每个配对自助法95%平均倒数排名区间均高于零。在两个数据集上，SASRec与TA-Transformer的平均倒数排名顺序在采样排序和全量排序之间发生反转，这与Krichene和Rendle [15]的结论一致。

研究问题2：候选集扩展与可缓存性。我们测量不同候选集大小下的评分延迟，采用候选项展平后的历史序列重新编码和请求内局部复用，以检验备注 4.1 和 4.2 中的服务机制。补充材料 B 节提供了补充性的浮点运算次数估算。

测量协议。我们在一台配备 32 个虚拟核心的专用处理器实例上，对一位用户、一段 S = 50 历史序列以及与其共享该历史序列的C 个候选项进行计时。对于每个模型，每次重复实验都在同一进程中测量候选项展平后的重新编码和请求内局部复用。调用之间不保留任何状态。我们报告五次进程隔离重复实验的最小值的中位数，以及配对加速比的中位数。补充材料 F 节介绍软件环境、计时流程、执行顺序、变异性和输出等价性检查。

研究问题1–研究问题2：质量—延迟比较。图 3 是补充图 4 的公开数据集对应版本，其中 TA-Transformer 是目标感知参考模型。预先指定的 MELODI 配置在 Steam 和 ML-25M 上的 NDCG@10 分别比 TA-Transformer 高 0.0332 和 0.0430。在 C = 1,000 个候选项且采用请求内局部复用时，其延迟分别为 1.97 和 2.00 毫秒；相比之下，TA-Transformer 为 15.09 和 13.74 毫秒，DIN 为 9.89 和 9.87 毫秒。MELODI 比 TA-Transformer 快7.7× 和 6.9×，比 DIN 快 5.0× 和 4.9×。此处的基线延迟在不同运行之间相差 9–25%，而重复运行的结果为 8.6–8.7× 和 3.8–4.0× （补充材料 F 节）。由于各数据集共享 S、 D和架构，且嵌入表大小不会影响每请求成本，因此其延迟几乎相同。

候选集边界与部署机制。 在请求内局部复用的情况下，在两个数据集上，候选项数为$C \in$ {1、10、100}时 DIN 的速度最快，

表2：公开数据集结果（HIT@10、NDCG@10、平均倒数排名）：三个训练随机种子上的均值 ± 样本标准差。MELODI 是第 4 节预先规定的单层、4 专家配置，在两个数据集上均相同，且不会针对每个数据集进行选择。表3报告了每种 {4、8} 专家和 {1、2} 层组合各进行一次训练运行的结果。SASRec 使用归一化目标。

	Steam			ML-25M		
模型	HIT@10	NDCG@10	MRR	HIT@10	NDCG@10	MRR
SASRec（与目标无关）	0.7677 ±0.0032	0.4997 ±0.0029	0.4279 ±0.0029	0.9327 ±0.0017	.7234 ±.0038	.6595 ±.0043
TA-Transformer（1 层，1 个头）	.7741 ±.0043	0.5128 ±0.0062	0.4425 ±0.0065	.9283 ±.0016	0.7158 ±0.0031	0.6512 ±0.0035
DIN	0.7817 ±0.0029	0.5253 ±0.0045	0.4561 ±0.0049	0.9399 ±0.0021	0.7347 ±0.0063	.6719 ±.0074
DIN + DCN-v2	0.7851 ±0.0029	0.5300 ±0.0021	0.4609 ±0.0017	.9436 ±.0003	0.7467 ±0.0022	0.6862 ±0.0029
MELODI（4 个专家，1 层）	**.7940** ±.0019	**.5409** ±.0016	**.4720** ±.0016	**.9503** ±.0006	**.7605** ±.0016	**.7017** ±.0019

表3：每个模型进行一次训练运行得到的完整公开数据集结果。粗体表示每个数据集和指标的最佳结果。“SASRec（仅历史层归一化）”省略了目标上的共享层归一化，而主 SASRec 行应用了该归一化。补充材料 G 节给出了批处理细节。

数据集	模型	HIT@1	HIT@5	HIT@10	NDCG@5	NDCG@10	平均倒数排名
Steam	SASRec（仅历史层归一化）	0.2168	0.5554	0.7198	0.3917	0.4451	0.3736
	SASRec	0.2738	0.6198	0.7704	0.4540	0.5029	0.4311
	TA-Transformer（1 层，1 个头）	0.2809	0.6235	0.7695	0.4599	0.5073	0.4370
	DIN	0.3082	0.6465	0.7848	0.4855	0.5304	0.4616
	DIN + DCN-v2	0.3060	0.6439	0.7823	0.4834	0.5284	0.4598
	MELODI（4 个专家，1 层）	0.3196	0.6563	0.7923	0.4963	0.5405	0.4720
	MELODI（4 个专家，2 层）	0.3238	0.6628	0.7960	0.5022	0.5455	0.4771
	MELODI（8 个专家，1 层）	**0.3272**	0.6678	0.8000	**0.5064**	**0.5494**	**0.4808**
	MELODI（8 个专家，2 层）	0.3257	**0.6680**	**0.8019**	0.5058	0.5493	0.4801
ML-25M	SASRec（仅历史层归一化）	0.3140	0.6472	0.7714	0.4901	0.5304	0.4658
	SASRec	0.5084	0.8587	0.9333	0.6992	0.7236	0.6597
	TA-Transformer（1 层，1 个头）	0.5027	0.8545	0.9302	0.6942	0.7189	0.6546
	DIN	0.5295	0.8700	0.9408	0.7151	0.7383	0.6762
	DIN + DCN-v2	0.5405	0.8753	0.9433	0.7234	0.7457	0.6850
	MELODI（4 个专家，1 层）	0.5634	0.8879	0.9510	0.7412	0.7619	0.7033
	MELODI（4 个专家，2 层）	0.5698	0.8913	0.9531	0.7461	0.7664	0.7086
	MELODI（8 个专家，1 层）	0.5686	0.8912	0.9525	0.7458	0.7660	0.7081
	MELODI（8 个专家，2 层）	**0.5745**	**0.8949**	**0.9548**	**0.7505**	**0.7701**	**0.7128**

表 4：在采样的（99 个均匀负样本）、困难的（来自固定 MiniLM 检索器的 999 个负样本）以及全目录排序设置下的单次运行平均倒数排名（测试项与用户序列之外的每个其他项进行比较）。

模型	采样	Hard	Full
Steam			
SASRec	.4311	.0636	.0351
TA-Transformer（1L，1H）	.4370	.0623	.0340
DIN	.4616	.0748	.0457
DIN + DCN-v2	.4598	.0731	.0434
CAN	.4611	.0785	.0388
DIEN	.4495	.0703	.0380
MELODI（4 个专家，1 层）	**.4720**	**.0804**	**.0466**
ML-25M			
SASRec	.6597	.0839	.0374
TA-Transformer（1L，1H）	.6546	.0942	.0383
DIN	.6762	.0947	.0471
DIN + DCN-v2	.6850	.1061	.0485
CAN	.6630	.0940	.0378
DIEN	.6769	.0975	.0415
MELODI（4 个专家，1 层）	**.7033**	**.1099**	**.0565**

其中 MELODI 的速度慢 2.2–2.3×, 1.7×，且比 1.9× 慢。MELODI 有一个候选项评分之前的历史序列开销固定，而 DIN 的激活单元能够高效地向量化，直到其 $[B, S, C, 4D]$ 张量变得很大。从 $C = 100$ 到 1，000，DIN 的延迟从 0.63 毫秒上升到约 9.9 毫秒，MELODI 的延迟则从 1.21 毫秒上升到约 2.0 毫秒，因此两者的排序会在其间某个依赖硬件的点发生反转；更细致的扫描将该点确定在 $C = 500$ 和 700 之间，而 DIN 与 DCN-v2 的交叉点则在 200 和 300 之间（补充材料 F 节）。因此，在这款处理器上，对于每个请求评分约 100 个候选项的排序器，MELODI 的 NDCG@10 更高（Steam 上为 +0.0101，ML-25M 上为 +0.0236），但 DIN 的延迟更低 1.9×；而候选项数约为 1，000 时，MELODI 的 NDCG@10 更高且延迟更低。$C = 1$ 部署机制也解释了为什么 MELODI 在训练循环中逐个评分目标时可能更慢。

请求内历史复用。 在 $C = 1$，000 条请求中，请求内局部复用将 Steam 上 MELODI 的延迟从 27.86 ms 降至 1.97 ms，将 ML-25M 上的延迟从 29.30 ms 降至 2.00 ms。配对收益中位数在 Steam 上为 14.21×，在 ML-25M 上为 14.68×（重复实验范围见补充材料 F 节）。该收益从 1.00×（$C = 1$）增长至 1.17×（$C = 10$），并在 $C = 100$ 时达到 1.43–1.45×。由于自注意力会将每个候选项与其历史序列混合，复用对 TA-Transformer 延迟的影响仅为 1.03–1.05×。共享目标无关输入路径可使 DIN 获得 1.67–1.89×

表5：公开的质量–延迟对比。上半部分：NDCG@10、候选项展平延迟（Flat.，毫秒）、请求内局部复用延迟（Reuse，毫秒）以及 $C = 1,000$ 条请求下的配对复用收益中位数。NDCG@10 来自一次训练运行，Δ 是其与 TA-Transformer 的差值。下半部分：较小 C下的请求内局部复用延迟。延迟是五次重复实验最小值的中位数。收益是单次重复实验内的 Flat./Reuse 比率的中位数，而不是所显示中位数的商。TA-Trans. 指 TA-Transformer，DIN+DCN 指 DIN + DCN-v2。

	Steam					ML-25M				
模型	NDCG@10	Δ	扁平。	复用	收益	NDCG@10	Δ	扁平。	复用	收益
TA-Transformer（1 层，1 个头）	0.5073	0.0000	14.76	15.09	1.03×	0.7189	0。0000	14。52	13。74	1.05×
DIN	0.5304	+0.0231	16.57	9.89	1.67×	0.7383	+0.0194	18.70	9.87	1.89×
DIN +DCN-v2	0.5284	+0.0211	17.74	10.58	1.80×	0.7457	+0.0268	16.64	10.19	1.66×
MELODI（4 个专家，1 层）	0.5405	+0.0332	27.86	1.97	14.21×	0.7619	+0.0430	29.30	2.00	14.68×

	Steam				ML-25M			
C	TA-Trans.	DIN	DIN+DCN	MELODI	TA-Trans.	DIN	DIN+DCN	MELODI
1	0.60	0.36	0.67	0.80	0.60	0.35	0.67	0.80
10	0.87	0.51	0.82	0.85	0.87	0.51	0.82	0.85
100	1.50	0.63	1.07	1.21	1.49	0.63	1.07	1.21

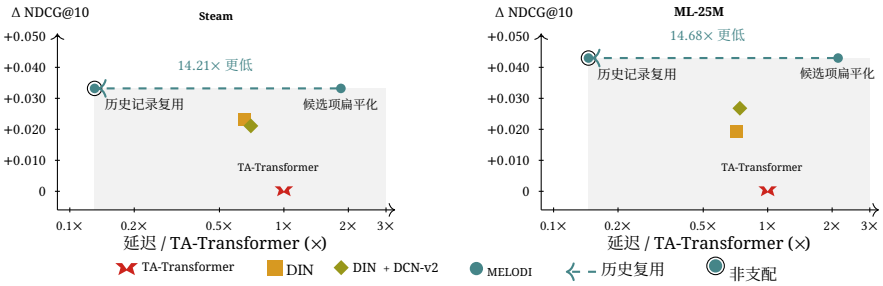

图3：相对于 TA-Transformer，在 $C = 1{,}000$处的公开质量增益与延迟。质量是相对于 TA-Transformer 的单次运行 NDCG@10 差值。基线点使用相对于 TA-Transformer 的请求内局部复用延迟。MELODI 的虚线箭头表示请求内局部复用降低了其延迟，左上方的点则表示在更低延迟下具有更高的 NDCG@10。灰色区域表示被复用后的 MELODI 支配的区域，黑色圆环标记非支配的已绘制运行点。与目标无关的 SASRec（Steam 上的 NDCG@100.5029**和 ML-25M 上的**0.7236**）被省略，以聚焦于候选项条件模型。**

benefit 和 DIN + DCN-v2 的收益为 1.66–1.80×，尽管它们的激活单元仍依赖候选项。

5.2 工业数据集评估

数据集与协议。 我们评估了两个大规模工业排序数据集，分别记为*Industrial-A*和*Industrial-B*，它们由日志请求组成。两者均为专有数据集，无法公开发布。Industrial-A 测试集包含来自 5,836 个请求的 591,286 个候选行。Industrial-B 测试集包含来自 5,625 个请求的 1,088,853 个候选行。每个参与评估的候选项都已通过上游候选生成，因此二元购买任务使用展示候选集中的困难负样本。补充材料第 G 节给出了时间划分和报告的细分组。

模型输入。 对于每个数据集，所有模型都接收相同的请求级特征和物品表示。序列模型还接收相同的两个事件历史分区。

基线与配置。 每个数据集都有各自的表格基线、TA-Transformer 和 DIN。两个序列基线都针对相同的两个事件历史分区分别使用一个分支。MELODI 在两个数据集中均使用一个相似度交叉层和一个 DCN-v2 目标自交叉，但两个数据集采用各自独立配置的宽度、专家数量和事件历史交叉。具体配置见补充材料 G。

训练、模型选择与指标。 所有模型均使用二元交叉熵，并依据验证集 NDCG@20 进行选择。我们报告四项指标：受试者工作特征曲线下面积（AUROC）、AUPRC、NDCG@20 和 Recall@20。每项指标均报告相对于表格基线的百分比变化，数值根据保留三位小数的结果计算。

RQ3：工业迁移。Industrial-A。 总体而言（表6），MELODI 在全部四项指标上均排名最高。相较于表格基线，其 AUPRC 提升了 25.0%，NDCG@20 提升了 11.3%。在两个细分组中，其 NDCG@20、Recall@20 和 AUPRC 均为最高；在报告精度下，它在 Segment-2 的 AUPRC 上与 TA-Transformer 并列，而 DIN 的 Segment-1 AUROC 更高（+2.6% 对 +2.5 %）。

Industrial-B。 总体而言（表 7），MELODI 的 AUPRC、NDCG@20 和 Recall@20 均为最高，而 DIN 的 AUROC 最高（+2.5 % 对比 +2.0%）。与表格基线相比，MELODI 的 AUPRC 提高了 7.8%，NDCG@20 提高了 3.2%，Recall@20 提高了 4.2%。无论是否提供用户标识符，其 Recall@20 均为最高，且在报告精度下，NDCG@20 并列最高。提供标识符时，其 AUPRC 也并列最高。

延迟和可缓存性。我们在用于公开基准测试的同一台商用处理器实例上，对包含 50 个历史项目和 1，000 个候选项的单个用户进行计时，该实例使用 32 个虚拟核心。补充材料 F 节介绍了计时流程，以及相对于每个表格

表6：Industrial-A：相对于表格基线的变化百分比（前20项N值：NDCG@20，前20项R值：Recall@20）。

	总体				Segment-1				Segment-2			
模型	AUROC	AUPRC	前20项N值	前20项R值	AUROC	AUPRC	前20项N值	前20项R值	AUROC	AUPRC	前20项N值	前20项R值
表格基线	–	–	–	–	–	–	–	–	–	–	–	–
TA-Transformer	+2.5	+22.2	+8.8	+5.8	+2.3	+28.6	+11.1	+3.8	+3.1	**+11.1**	+6.9	+7.1
DIN	+2.8	+16.7	+8.8	+6.0	**+2.6**	+22.4	+11.8	+4.6	+3.4	+7.4	+6.4	+6.7
MELODI	**+2.9**	**+25.0**	**+11.3**	**+7.1**	+2.5	**+32.7**	**+15.4**	**+5.5**	**+3.6**	**+11.1**	**+7.9**	**+8.1**

表7：工业数据集 **ustrial-B：相较于表格基线的变化百分比，整体以及按标识符可用性划分，如表所示 . 6.**

	整体				标识符可用				标识符不可用			
模型	AUROC	AUPRC	前20项N值	前20项R值	AUROC	AUPRC	前20项N值	前20项R值	AUROC	AUPRC	前20项N值	前20项R值
表格基线	–	–	–	–	–	–	–	–	–	–	–	–
TA-Transformer	+2.0	+6.5	+2.3	+2.3	+2.3	−3.3	+3.5	+4.9	+1.6	**+12.5**	**+2.3**	+0.9
DIN	**+2.5**	+6.5	+2.7	+3.1	**+3.1**	**+3.3**	**+5.6**	+7.4	**+1.9**	+10.2	+1.6	0.0
MELODI	+2.0	**+7.8**	**+3.2**	**+4.2**	+2.8	**+3.3**	**+5.6**	**+8.4**	+1.0	+10.2	**+2.3**	**+1.3**

表 8：组件消融实验（平均倒数排名，4 个专家）。第 3 行和第 4 行包含耦合干预和独立干预。补充材料 D 节给出了增量及详细信息。

# 配置	Steam	ML-25M
1 **锚点：1 层、无残差、仅交叉评分器**	**.4720**	**.7033**
2 – 目标自交叉	.4702	.6882
3a + 目标残差	.4722	.7023
3b +池化历史残差	.4716	.7023
4a + 第二层；恒等映射也加深	.4771	.7086
4b +第二层；恒等映射固定	.4794	.7019
5 + 未交叉池化历史和投影后的目标	.4758	.7038
6 +仅投影目标	.4643	.6934
7 – 相似度交叉；**u** 至评分器	.4649	.6924

基线的延迟：候选项展平路径会在候选项之间复制历史序列，而请求内局部复用路径则会为请求中的所有候选项编码一份共享历史。

质量—延迟结果。补充图 4 绘制了相对于每个表格基线的 NDCG@20 增益与延迟开销之间的关系。MELODI 在 Industrial-A 上将 NDCG@20 提高了 11.3%，在 Industrial-B 上提高了 3.2%。在两个数据集上，其报告的 NDCG@20 均高于 TA-Transformer 和 DIN，并且其候选项展平路径分别快 5.0–7.6× 和 2.0–2.9× 。

请求内历史复用。每个请 求仅对历史序列编码一次，可使 MELODI 的延迟进一步降低 2。6–3.8×，比 TA-Transformer 低 13。0– 28.7× ，比 DIN 低 5。2–11.1× 。两项基准测试均对相同的两条路径计时，因此收益差异反映了剩余计算中有多少是依赖候选项的。在公开 D=64 配置中，目标相关路径开销较低，重复的历史序列处理占候选项展平成本的主要部分，因此在 C=1,000 处，理论 FLOP比率接近实测的 14。21–14.68× 收益。工业配置的 D=256 、表格特征，以及更宽的相似度输出和评分器，在复用后仍保留了更多依赖候选项的计算，这解释了较小的实测收益。

5.3 组件消融证据

RQ4：组件证据。表 8 将组件变更与锚点配置进行了比较。补充材料 D 节详细说明了孤立和耦合干预。两个观察到的模式促成了固定默认值的设定。移除经过测试的目标自交叉恒等分支后，Steam 上的平均倒数排名下降 0。0018，ML-25M 上下降 0。0151。仅添加投影目标后，Steam 上的平均倒数排名下降 0。0077，ML-25M 上下降 0。0099。在提供投影目标时，添加相应的未交叉池化历史（第 5 行而非第 6 行）会在两个数据集上得到更高的数值。移除相似度交叉本身（第 7 行）会使 Steam 上的平均倒数排名下降 0。0071，ML-25M 上下降 0。0109，并在 $C = 1,000$ 处节省约 0。8ms；不过，在 ML-25M 上使用困难负样本时，平均倒数排名会提高 0。0011。

两种残差变体对平均倒数排名的影响最多为 0.0010。在恒等容量固定的情况下，增加一个相似度层会使 Steam 上的平均倒数排名变化 +0.0074 ，在 ML-25M 上变化 −0.0014 ，因此其影响方向取决于数据集。专家数量变化同样会因配置不同而有所差异（表 11）。我们保留一个层，以确保所述的 SP/PS保证适用；同时省略其残差，使该分支仅输出历史序列与目标之间的交互。每种配置仅对应一次训练运行，因此几千分之一的平均倒数排名差异尚未得到确证。

6 结论

我们提出了一个面向目标感知排序的相似性–身份解耦框架。命题 3.3 给出了在线性条件成立时，更廉价的 PS 顺序能够精确恢复 SP；MELODI 通过多头交叉注意力和无残差目标依赖交叉层实现了这一机制。以三个训练随机种子的结果取平均后，其固定公开配置在表 2 的每项报告指标上均超过所有基线。在 $C = 1,000$ 个候选项时，请求内局部复用使其延迟低于候选条件基线，而在 $C = 500$ 之前 DIN 仍然更快。在工业评估中，MELODI 以更低的延迟达到了或超过了目标感知基线所报告的 AUPRC、NDCG 和召回率。单次运行的组件比较为固定默认值提供了依据，主要包括目标自交叉，以及将投影目标与池化历史配对，但无法证明这些影响很小。仍存在一些局限：评估是离线进行的，未开展在线 A/B 测试；工业数据无法发布；延迟仅在一种处理器类型上测量。

7 伦理考量

工业评估引发了两个问题：日志曝光中的偏差，以及对行为数据的处理。

训练信号中的曝光偏差和位置偏差。模型基于展示候选项的日志购买标签进行训练，这些标签反映了日志策略的曝光偏差和位置偏差。MELODI 不会对其进行校正，因此可能继承并延续这些偏差。该框架与训练损失无关，因此采用逆倾向加权 [24] 或用于估计倾向得分的随机曝光留出集等校正方法，无需进行架构更改。这种偏差限制了对工业结果的解读。

隐私与数据处理。交互历史属于行为数据，即使没有明确的用户标识符，也可能识别出用户。因此，请求本地的历史张量需要与其来源历史采用相同的访问控制，并且在当前候选集完成评分后应立即丢弃。

参考文献

[1] Jimmy Lei Ba、Jamie Ryan Kiros 和 Geoffrey E. Hinton。2016。层归一化。*arXiv* 预印本 *arXiv:1607.06450* (2016)。[2] Weijie Bian、Kailun Wu、Lejian Ren、Qi Pi、Yujing Zhang、Can Xiao、Xiang-Rong Sheng、Yong-Nan Zhu、Zhangming Chan、Na Mou、Xinchen Luo、Shiming Xiang、Guorui Zhou、Xiaoqiang Zhu 和 Hongbo Deng。2022。CAN：用于点击率预测的特征协同行为网络。载于第*15*届*ACM*网页搜索与数据挖掘会议（*WSDM*）论文集。57–65。[3] Jianxin Chang、Chenbin Zhang、Zhiyi Fu、Xiaoxue Zang、Lin Guan、Jing Lu、Yiqun Hui、Dewei Leng、Yanan Niu、Yang Song 和 Kun Gai。2023。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。载于第*29*届*ACM SIGKDD*知识发现与数据挖掘会议论文集。3785–3794。[4] Qiwei Chen、Huan Zhao、Wei Li、Pipei Huang 和 Wenwu Ou。2019。阿里巴巴电子商务推荐的行为序列Transformer。载于第*1*届面向高维稀疏数据的深度学习实践国际研讨会论文集。1–4。[5] Heng-Tze Cheng、Levent Koc、Jeremiah Harmsen、Tal Shaked、Tushar Chandra、Hrishi Aradhye、Glen Anderson、Greg Corrado、Wei Chai、Mustafa Ispir 等。2016。推荐系统的Wide & Deep学习。载于第*1*届推荐系统深度学习研讨会论文集。7–10。[6] Paul Covington、Jay Adams 和 Emre Sargin。2016。用于YouTube推荐的深度神经网络。载于第*10*届*ACM*推荐系统会议论文集。191–198。[7] Ningya Feng、Junwei Pan、Jialong Wu、Baixu Chen、Ximei Wang、Qian Li、Xian Hu、Jie Jiang 和 Mingsheng Long。2025。长序列推荐模型需要解耦嵌入。载于国际学习表征会议（*ICLR*）。[8] Yufei Feng、Fuyu Lv、Weichen Shen、Menghan Wang、Fei Sun、Yu Zhu 和 Keping Yang。2019。用于点击率预测的深度会话兴趣网络。载于第*28*届国际人工智能联合会议（*IJCAI*）论文集。2301–2307。[9] GroupLens研究。2019。MovieLens 25M数据集。https://grouplens.org/datasets/movielens/25m/。于2019年12月发布。[10] Huifeng Guo、Ruiming Tang、Yunming Ye、Zhenguo Li 和 Xiuqiang He。2017。DeepFM：用于点击率预测的基于因子分解机的神经网络。载于第*26*届国际人工智能联合会议论文集。1725–1731。[11] Balázs Hidasi、Alexandros Karatzoglou、Linas Baltrunas 和 Domonkos Tikk。2016。基于循环神经网络的会话推荐。载于国际学习表征会议（*ICLR*）。[12] Po-Sen Huang、Xiaodong He、Jianfeng Gao、Li Deng、Alex Acero 和 Larry Heck。2013。利用点击数据学习用于网页搜索的深度结构化语义模型。载于第*22*届*ACM*信息与知识管理国际会议论文集。2333–2338。[13] Wang-Cheng Kang 和 Julian McAuley。2018。自注意力序列推荐。载于*IEEE*国际数据挖掘会议（*ICDM*）。197–206。[14] Diederik P. Kingma 和 Jimmy Ba。2015。Adam：一种随机优化方法。载于国际学习表征会议（*ICLR*）。[15] Walid Krichene 和 Steffen Rendle。2020。关于物品推荐的采样指标。载于第*26*届*ACM SIGKDD*知识发现与数据挖掘国际会议论文集。1748–1757。[16] Bin Liu、Ruiming Tang、Yingzhi Chen、Jinkai Yu、Huifeng Guo 和 Yuzhou Zhang。2019。用于点击率预测的卷积神经网络特征生成

点击率预测。在万维网大会。1119–1129。[17] Vinod Nair 和 Geoffrey E. Hinton。2010。修正线性单元改进受限玻尔兹曼机。在国际机器学习会议第*27*届会议论文集。807–814。[18] Adam Paszke、Sam Gross、Francisco Massa、Adam Lerer、James Bradbury、Gregory Chanan、Trevor Killeen、Zeming Lin、Natalia Gimelshein、Luca Antiga 等。2019。PyTorch：一种命令式风格的高性能深度学习库。在神经信息处理系统进展，第32卷。[19] Qi Pi、Weijie Bian、Guorui Zhou、Xiaoqiang Zhu 和 Kun Gai。2019。长序列用户行为建模在点击率预测中的实践。在第*25*届 *ACM SIGKDD*知识发现与数据挖掘国际会议论文集。2671–2679。[20] Qi Pi、Xiaoqiang Zhu、Guorui Zhou、Yujing Zhang、Zhe Wang、Lejian Ren、Ying Fan 和 Kun Gai。2020。利用终身序列行为数据进行基于搜索的用户兴趣建模，以实现点击率预测。在第*29*届 *ACM*信息与知识管理国际会议论文集。2685–2692。[21] Nils Reimers 和 Iryna Gurevych。2019。Sentence-BERT：使用孪生BERT网络的句子嵌入。在*2019*年自然语言处理中的经验方法会议与第*9*届自然语言处理国际联合会议论文集（*EMNLP-IJCNLP*）。3982–3992。[22] Steffen Rendle。2010。因子分解机。在*IEEE*国际数据挖掘会议（*ICDM*）。995–1000。[23] Steffen Rendle、Christoph Freudenthaler、Zeno Gantner 和 Lars Schmidt-Thieme。2009。BPR：基于隐式反馈的贝叶斯个性化排序。在第二十五届人工智能中的不确定性会议论文集（*UAI*）。452–461。[24] Tobias Schnabel、Adith Swaminathan、Ashudeep Singh、Navin Chandak 和 Thorsten Joachims。2016。推荐即干预：学习与评估去偏。在第*33*届国际机器学习会议论文集。1670–1679。[25] Sentence Transformers。2022。all-MiniLM-L6-v2。Hugging Face 模型卡片。https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2 修订版 1110a243fdf4706b3f48f1d95db1a4f5529b4d41；访问日期：2026年8月22日。[26] Nitish Srivastava、Geoffrey Hinton、Alex Krizhevsky、Ilya Sutskever 和 Ruslan Salakhutdinov。2014。丢弃：一种防止神经网络过拟合的简单方法。机器学习研究期刊15，56（2014），1929–1958。[27] Fei Sun、Jun Liu、Jian Wu、Changhua Pei、Xiao Lin、Wenwu Ou 和 Peng Jiang。2019。BERT4Rec：基于 Transformer 双向编码器表示的序列推荐。在第*28*届 *ACM*信息与知识管理国际会议论文集。1441–1450。[28] Ashish Vaswani、Noam Shazeer、Niki Parmar、Jakob Uszkoreit、Llion Jones、Aidan N Gomez、Łukasz Kaiser 和 Illia Polosukhin。2017。Attention is All You Need。在神经信息处理系统进展。[29] Ruoxi Wang、Rakesh Shivanna、Derek Cheng、Sagar Jain、Dong Lin、Lichan Hong 和 Ed Chi。2021。DCN V2：改进的深度交叉网络及网页规模学习排序系统的实践经验。在*2021*年万维网大会论文集。1785–1797。[30] Wenhui Wang、Furu Wei、Li Dong、Hangbo Bao、Nan Yang 和 Ming Zhou。2020。MiniLM：面向任务无关压缩的预训练 Transformer 深度自注意力蒸馏。在神经信息处理系统进展，第33卷。5776–5788。[31] Xinyang Yi、Ji Yang、Lichan Hong、Derek Zhiyuan Cheng、Lukasz Heldt、Aditee Kumthekar、Zhe Zhao、Li Wei 和 Ed Chi。2019。面向大语料库物品推荐的采样偏差校正神经建模。在第*13*届 *ACM*推荐系统会议论文集。269–277。[32] Jiaqi Zhai、Lucy Liao、Xing Liu、Yueming Wang、Rui Li、Xuan Cao、Leon Gao、Zhaojie Gong、Fangda Gu、Jiayuan He、Yinghai Lu 和 Yu Shi。2024。行动胜于言语：用于生成式推荐的万亿参数序列转换器。在第*41*届国际机器学习会议论文集（*ICML*）。58484–58509。[33] Zuowu Zheng、Xiaofeng Gao、Junwei Pan、Qi Luo、Guihai Chen、Dapeng Liu 和 Jie Jiang。2022。AutoAttention：用于用户行为建模的注意力自动字段对选择。在*IEEE*国际数据挖掘会议论文集（*ICDM*）。803–812。[34] 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。在*AAAI*人工智能会议论文集。5941–5948。[35] Guorui Zhou、Chengru Song、Xiaoqiang Zhu、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。在第*24*届 *ACM SIGKDD*知识发现与数据挖掘国际会议论文集。1059–1068。

补充材料

服务时推理效率

图2中的青绿色波浪箭头标示了可复用的历史序列路径：同一请求中的候选项仅在目标上有所不同。

候选项展平评估和请求内局部复用。 在 MELODI 的交叉注意力架构中，历史序列编码路径完全独立于目标项：嵌入查找、投影、位置编码、层归一化以及键/值投影（图 2 中的 $\mathbf{K}$，$\mathbf{V}$）仅取决于历史序列。候选项展平评估会为每个候选项行重复并编码该历史序列。请求内局部复用则只编码一次，并将所得张量扩展到 C 个候选项。两种路径都会在调用结束后丢弃这些张量。其余逐候选项计算包括注意力分数 $\boldsymbol{\alpha}_h$、池化 $\mathbf{u}_h$、逐元素交叉项 $\mathbf{c}_h = \mathbf{u}_h \odot \boldsymbol{\eta}_h$、恒等分支和评分器。追加目标项的 Transformer 不具备这样的完整目标无关路径。

预计算边界。 令 I 表示物品全集。冻结嵌入使目标侧的 Q 和目标映射输入独立于当前历史序列。历史序列的K 和 V 则不同：由于位置编码是在层归一化及键/值投影之前添加的，它们同时取决于物品和序列位置。因此，精确的请求内局部复用需要在组装好位置化历史后计算这些表示。存储物品-位置表示需要 $O(\mathbb{I}|SD_u)$ 而非 $O(\mathbb{I}|D_u)$ 的空间。补充材料 F 节中的激进工业基准排除了这项组装历史序列的计算，应将其视为在线延迟的下界，而非仅针对物品的查找。

B 分析性浮点运算次数核算

在历史线性性条件下，PS 将调用次数从 ϕ S 次降至每个候选项一次。交叉注意力则使键和值路径独立于目标，并可在候选项之间复用。我们计算针对包含 S 个物品的历史序列对 $C = 100$ 个候选项进行评分时的浮点运算次数，其中包括评分路径中的运算（投影、注意力、交叉网络、多层感知机），但不包括嵌入表查找。候选项展平路径会为每个候选项重复编码历史序列。请求内局部复用只评估一次，并将这一次评估计入总量。两种路径都不会跨请求保留状态。

对于 $C = 100$处的受控计算，我们单独考察可复用的历史路径。不同候选数量下的测量比较见表5。

采用请求内局部复用后，预先指定的 MELODI 配置使用 43.83M 次浮点运算，仅为 DIN 的 176.17M 次的四分之一。若为每个候选项重新编码历史序列，MELODI 的开销将升至 618.50M 次浮点运算，因此共享该路径在 $C = 100$时带来 14.11× 的分析性收益。DIN 的收益较小，为 2.39×，因为其独立于目标的嵌入、投影和层归一化可以共享，而其激活单元仍取决于目标与历史序列。目标增强型 Transformer 的分析性收益为 1.00×，因为其前向传播会在自注意力之前扩展候选项。

在 $C = 1,000$ 个候选项时，MELODI 的理论收益为 16.02×，而 Steam 和 ML-25M 上测得的中位数分别为 14.21× 和 14.68×。在候选数量较少时，固定调用开销在复用路径中占比更大，因此实测的耗时收益低于 FLOP比率。

计数约定。我们统计乘加 （MAC）运算和逐元素运算（softmax、层归一化）。随后，我们将 FLOPs = 2×MACs 约定应用于它们的总和，这会对非 MAC 项进行重复计数。这些项所占比例很小，并且在各模型中采用相同处理方式，因此比起绝对总量，比例受影响更小；绝对总量是估算值，而非指令计数。

C MELODI架构方程与跨层比较

本补充章节给出了 MELODI 前向传播五个阶段的完整方程（第 4 节中交叉引用），并将第3阶段中的交叉层与标准 DCN-v2 自交叉进行了比较。正文中的符号总结于表 1 中。以下将在首次引入时定义其他批量计算量。图 2 详细展示了相似度分支的数据流。

阶段1：嵌入与归一化

给定任意源维度的原始冻结嵌入 $\mathbf{x}_i^{\text{raw}}$，$\mathbf{t}_c^{\text{raw}}$，共享投影和层归一化会生成 $\tilde{\mathbf{X}} \in \mathbb{R}^{S\times D}$ 和$\tilde{\mathbf{T}} \in \mathbb{R}^{C\times D}$，其各行满足

$$\tilde{\mathbf{x}}_i = \text{Dropout}\big(\text{LN}(\text{proj}(\mathbf{x}_i^{\text{raw}}) + \text{pos}_i)\big) \in \mathbb{R}^D, \tag{5}$$

$$\tilde{\mathbf{t}}_c = \text{LN}(\text{proj}(\mathbf{t}_c^{\text{raw}})) \in \mathbb{R}^D. \tag{6}$$

阶段2：多头点积交叉注意力

投影将宽度 D映射到 $D_u = Hd$，它不一定等于 D:

$$\mathbf{Q} = \text{reshape}_H(\tilde{\mathbf{T}}\mathbf{W}_q^\top) \in \mathbb{R}^{H\times C\times d}, \tag{7}$$

$$\mathbf{K} = \text{reshape}_H(\tilde{\mathbf{X}}\mathbf{W}_k^\top) \in \mathbb{R}^{H\times S\times d}, \tag{8}$$

$$\mathbf{V} = \text{reshape}_H(\tilde{\mathbf{X}}\mathbf{W}_v^\top) \in \mathbb{R}^{H\times S\times d}, \tag{9}$$

$$\mathbf{V}_t = \text{reshape}_H(\tilde{\mathbf{T}}\mathbf{W}_v^\top) \in \mathbb{R}^{H\times C\times d}, \tag{10}$$

$$\boldsymbol{\alpha}_h = \text{softmax}\left(\frac{\mathbf{Q}_h\mathbf{K}_h^\top}{\sqrt{d}}\right), \tag{11}$$

$$\mathbf{u}_h = \boldsymbol{\alpha}_h\mathbf{V}_h. \tag{12}$$

此处 $\boldsymbol{\alpha}_h \in \mathbb{R}^{C\times S}$和 $\mathbf{u}_h \in \mathbb{R}^{C\times d}$。对于一个候选项，$\mathbf{V}_t$的一行记为 $\mathbf{v}_t = [\mathbf{v}_t^{(1)}; \ldots; \mathbf{v}_t^{(H)}] \in \mathbb{R}^{D_u}$。第3阶段独立地作用于这些头切片。

阶段3a：与 DCN-v2 的比较

第3阶段的相似度算子在结构上与 DCN-v2 相关，但在若干方面有所不同。表10总结了这些差异。

这两个算子都采用逐元素乘积和低秩专家混合。我们的双输入形式、残差移除、多头分块对角结构、逐头门控以及单层限制，共同定义了我们的历史线性相似度算子。

第3b阶段：门控交叉方程

现在，我们将 MELODI 相似度分支中使用的交叉层与单层 DCN-v2 自交叉并列写出。阶段1的目标是 $\tilde{\mathbf{t}} \in \mathbb{R}^D$。注意力使用 H 个宽度为 d 的注意力头，因此池化历史为 $\mathbf{u} = [\mathbf{u}_1; \ldots; \mathbf{u}_H] \in \mathbb{R}^{D_u}$，其中有 $D_u = Hd$。第3阶段使用目标值投影 $\mathbf{v}_t = \mathbf{W}_v\tilde{\mathbf{t}} \in \mathbb{R}^{D_u}$，并将其划分为 $\mathbf{v}_t^{(h)} \in \mathbb{R}^d$。每个注意力头的门控仍接收完整的 $\tilde{\mathbf{t}}$。

表 9：在共享公共设置（$S = 50$， $D = 64$）中对 $C = 100$ 候选项进行评分所需的浮点运算次数。“扁平”会为每个候选项重新编码共享历史序列，而“复用”则在请求中仅编码一次。收益为扁平/复用。

模型	扁平。	复用	收益
DIN	421.38M	176。17M	2.39×
DIN + DCN-v2	429.04M	183.84M	2.33×
MELODI（1 层，4 个专家）	618.50M	43。83M	14.11×
MELODI（2 层，4 个专家）	635.56M	60。88M	10.44×
MELODI（1 层，8 个专家）	635.42M	60.74M	10.46×

表 10：我们的用户–目标交叉层（位于 MELODI 的相似度分支中）与标准 DCN-v2 自交叉的结构比较。

方面	**DCN-v2 自交叉**	**我们的用户—目标交叉**
输入	一个自交叉后的特征向量$\mathbf{x}$。	池化历史$\mathbf{u} \in \mathbb{R}^{D_u}$和目标 $\tilde{\mathbf{t}} \in \mathbb{R}^{D}$，以及一个目标投影 $\mathbf{v}_t \in \mathbb{R}^{D_u}$。
核心操作	$\mathbf{x}_{\ell+1} = \mathbf{x}_0 \odot (\mathbf{W}_\ell \mathbf{x}_\ell + \mathbf{b}_\ell) + \mathbf{x}_\ell$	$\mathbf{c} = \mathbf{u} \odot \boldsymbol{\eta}(\tilde{\mathbf{t}})$（无残差），其中 $\boldsymbol{\eta}$ 由$\mathbf{v}_t$和门控作用于 $\tilde{\mathbf{t}}$。
残差连接	包含，将前一层的状态$\mathbf{x}_\ell$向前传递。	已移除。输出仅包含相似度，而恒等分支负责处理仅目标内容。
头部结构	单流，不含注意力头。	每个头都作用于 $\mathbf{u}$和$\mathbf{v}_t$ 的 d 维切片；拼接后得到 $D_u = Hd$。
专家结构	混合 of 专家 is 全局：$\mathbf{M}_\ell(\mathbf{x}_\ell) = \sum_k g_{\ell,k}(\mathbf{x}_\ell)\mathbf{U}_{\ell,k}\mathbf{V}_{\ell,k}$，跨所有维度稠密。	相同的 d维专家在各注意力头之间复用，生成 H 个独立块 $\mathbf{M}_h(\tilde{\mathbf{t}})$。相较于宽度为$D$的DCN-v2层，专家项比例为 d/D。
门控	单门控向量 $\mathbf{g}_\ell(\mathbf{x}_\ell) \in \mathbb{R}^K$ 在所有维度间共享——度。	逐头门控向量 $\mathbf{g}_h(\mathbf{t}) \in \mathbb{R}^K$，因此每个头会分别混合 K 专家，而不是在所有维度之间共享同一个混合结果跨越所有维度。
深度	交叉层可以堆叠，以建模高阶特征交互。	历史线性和 PS等价性分析适用于单层算子（第 3.3 节）。堆叠变体不属于这一保证的范围，并会改变模型类。
在流水线中的作用	作用于扁平特征的特征交互模块。	相似度算子的实例化 ϕ 相似度–恒等映射解耦框架（第 3 节）。

对目标进行 DCN-v2 自交叉（参考算子）。 仅应用于目标的单个 DCN-v2 交叉层计算结果为

$$\mathbf{g}(\tilde{\mathbf{t}}) = \mathrm{softmax}(\mathbf{W}_{\mathrm{gate}}\tilde{\mathbf{t}}) \in \mathbb{R}^K, \tag{13}$$

$$\mathbf{M}(\tilde{\mathbf{t}}) = \sum_{k=1}^{K} g_k(\tilde{\mathbf{t}})\, \mathbf{U}_k \mathbf{V}_k \in \mathbb{R}^{D\times D}, \tag{14}$$

$$\tilde{\mathbf{t}}_{\mathrm{crossed}} = \tilde{\mathbf{t}} \odot \big(\mathbf{M}(\tilde{\mathbf{t}})\,\tilde{\mathbf{t}} + \mathbf{b}\big) + \tilde{\mathbf{t}} \in \mathbb{R}^D, \tag{15}$$

其中，$\mathbf{U}_k \in \mathbb{R}^{D\times r}$ 和 $\mathbf{V}_k \in \mathbb{R}^{r\times D}$ 是全局（非注意力头结构化）低秩专家。门控 $\mathbf{g}$ 是一个共享向量，而残差连接 $+\tilde{\mathbf{t}}$ 会重新注入投影后的目标。

MELODI 的用户–目标交叉层。 我们将类似的运算符分别应用于两个不同的输入（$\mathbf{u}$ $\tilde{\mathbf{t}}$），对每个头执行，不使用残差，并通过逐头门控选择块对角专家：

$$\mathbf{g}_h(\tilde{\mathbf{t}}) = \mathrm{softmax}(\mathbf{W}_{g,h}\,\tilde{\mathbf{t}}) \in \mathbb{R}^K, \qquad h = 1,\dots,H, \tag{16}$$

$$\mathbf{f}_{h,k} = \mathbf{U}_k \mathbf{V}_k\, \mathbf{v}_t^{(h)} \in \mathbb{R}^d, \qquad k = 1,\dots,K, \tag{17}$$

$$\mathbf{c}_h = \mathbf{u}_h \odot \Big(\sum_{k=1}^{K} g_{h,k}(\tilde{\mathbf{t}})\, \mathbf{f}_{h,k} + \mathbf{b}_h\Big) \in \mathbb{R}^d, \tag{18}$$

$$\mathbf{c} = \mathrm{concat}(\mathbf{c}_1,\dots,\mathbf{c}_H) \in \mathbb{R}^{D_u}, \tag{19}$$

其中 $\mathbf{U}_k \in \mathbb{R}^{d\times r}$ 和 $\mathbf{V}_k \in \mathbb{R}^{r\times d}$ 是跨注意力头共享的矩形低秩扰动，并在每个 d 维头子空间内独立应用。

紧凑形式。公式（17）–（18）等价地可写为 $\mathbf{c} = \mathbf{u} \odot \boldsymbol{\eta}(\tilde{\mathbf{t}})$，其中块对角目标映射为

$$\boldsymbol{\eta}(\tilde{\mathbf{t}}) = \big[\mathbf{M}_1(\tilde{\mathbf{t}})\,\mathbf{v}_t^{(1)} + \mathbf{b}_1;\ \dots;\ \mathbf{M}_H(\tilde{\mathbf{t}})\,\mathbf{v}_t^{(H)} + \mathbf{b}_H\big] \in \mathbb{R}^{D_u}, \tag{20}$$

$$\mathbf{M}_h(\tilde{\mathbf{t}}) = \sum_{k=1}^{K} g_{h,k}(\tilde{\mathbf{t}})\, \mathbf{U}_k \mathbf{V}_k \in \mathbb{R}^{d\times d}. \tag{21}$$

向量化计算。 每个头的运算都可以作为批量张量缩并来计算。令 $N = BC$ 合并 B 示例

和 C候选项，并令 $\mathbf{V}_{t,\,H} \in \mathbb{R}^{N\times H\times d}$表示按头划分的投影目标。在本段中，$\mathbf{u}$、$\boldsymbol{\eta}$ 和 $\mathbf{c}$ 表示 $\mathbb{R}^{N\times D_u}$ 中堆叠的张量。对于专家 k、秩坐标 ℓ、头 h、批次–候选索引 n 以及头坐标 j，q，定义

$$\mathcal{Z}_{k\ell hn} = \sum_{q=1}^{d} (\mathbf{V}_k)_{\ell q} (\mathbf{V}_{t,H})_{nhq}, \qquad \mathcal{Z} \in \mathbb{R}^{K\times r\times H\times N}, \tag{22}$$

$$\mathcal{E}_{kjhn} = \sum_{\ell=1}^{r} (\mathbf{U}_k)_{j\ell} \mathcal{Z}_{k\ell hn}, \qquad \mathcal{E} \in \mathbb{R}^{K\times d\times H\times N}. \tag{23}$$

将 K专家矩阵堆叠为$\mathbf{V}_{1:K} \in \mathbb{R}^{K\times r\times d}$和$\mathbf{U}_{1:K} \in \mathbb{R}^{K\times d\times r} \in \mathbb{R}^{N\times K\times D_u}$。将 E 重新索引为$\mathbf{E}$后，每头门控$\mathbf{G} \in \mathbb{R}^{N\times K\times D_u}$会对专家输出进行加权并求和，其中每个$g_{h,k}$在其 d坐标上重复：

$$\boldsymbol{\eta} = \textstyle\sum_{k=1}^{K} \mathbf{E}_{:,k,:} \odot \mathbf{G}_{:,k,:} + \mathbf{b} \in \mathbb{R}^{N\times D_u}, \quad \mathbf{c} = \mathbf{u} \odot \boldsymbol{\eta} \in \mathbb{R}^{N\times D_u}. \tag{24}$$

每头目标矩阵$\mathbf{M}_h(\tilde{\mathbf{t}})$是由秩为$r$的专家构成的 softmax 门控混合。其分块对角组合的形状为 $\mathrm{blkdiag}(\mathbf{M}_1, \ldots, \mathbf{M}_H) \in \mathbb{R}^{D_u\times D_u}$，而 DCN-v2 使用具有全局门控的稠密宽度为$D$的矩阵。专家权重矩阵分别包含 $2Kdr$和 $2KDr$个权重，因此比率为 d/D。仅当 $D_u = D$时，该比率才等于 $1/H$。

权重数量。 计算权重矩阵数量（不包括偏差）时，宽度为 D 的 DCN-v2 自交叉层使用 $K(2Dr + D)$ 个权重用于专家和门控。我们的算子使用 $K(2dr + HD)$ 个权重，因为各个注意力头共享专家，且每个 H 个门控都能看到完整目标。仅当 $2r(D-d) > D(H-1)$ 时，它才更小。由于 d 可配置，因此这一条件不一定成立。如果 $D_u = D$，那么 $d = D/H$，该条件便简化为 $2r > H$。对于 H=4、d=64、r=32、K=4 和 D=256，权重数量分别为 20，480 和 66，560。相比之下，在公开实验中，$d = D$ 和 $D_u = 4D$，因此这种块结构并未减少专家项相对于宽度为 D 的 DCN-v2 层的权重数量，而每头门控还增加了权重。

阶段 4：恒等分支

恒等分支取决于具体问题（第 3.1 节）。在我们的实验中，我们对目标采用 DCN-v2 自交叉，其公式见上文“对目标进行 DCN-v2 自交叉”段落。因此 $\tilde{\mathbf{t}}_{\mathrm{id}} \triangleq \tilde{\mathbf{t}}_{\mathrm{crossed}} \in \mathbb{R}^D$。

阶段 5：MLP 评分

在实验中，我们采用仅交叉形式，其中多层感知机接收相似度分支输出 $\mathbf{c}$ 和恒等分支˜输出 $\mathbf{t}_{\mathrm{id}}$：

$$\hat{y} = \mathrm{MLP}\big([\mathbf{c}\,;\,\tilde{\mathbf{t}}_{\mathrm{id}}]\big) \in \mathbb{R}. \tag{25}$$

评分器使用宽度为 D的隐藏层，并采用修正线性单元 激活函数 [17],，后接一个标量输出层。补充材料 D 节中的消融实验 #5 将未交叉池化历史 $\mathbf{u}$和投影后的目- 标 ˜$\mathbf{t}$ 加入该输入中。相对于仅交叉评分器，Steam 上的平均倒数排名变化为 +0.0038，ML-25M 上的变化为 +0.0005。

D 公共数据集上的消融实验

表 8 报告了 4专家配置的消融实验。锚点是单层、无残差配置，其多层感知机仅接收交叉输出，并启用恒等分支。#3b 行和 #4b 行分别单独考察池化历史残差和相似度深度干预，#3a 行和 #4a 行则是二者的耦合变体。

• **观察到的范围。** 在测试的干预中，Steam 的变化范围为 −0.0077 至 +0.0074，而 ML-25M 的变化范围为 −0.0151 至 +0.0053。

• **#1 与 #2（恒等分支）。** 移除目标自交叉后，Steam 上的平均倒数排名变化为 −0.0018，ML-25M 上的变化为 −0.0151。ML-25M 上观察到的更大降幅促使我们保留经过测试的恒等分支（第 3.1 节）。

• **#1 与 #3（残差连接）。** 之所以报告两种形式，是因为所评估的递推将其 专家输入 作为残差。在第一层中，该输入是目标，而不是池化历史 $\mathbf{u}$；当专家输入的宽度与 D_u 不同时，则不定义残差。第 #3b 行改为加入一个显式的 +$\mathbf{u}$ 项。令 $\mathbf{1} \in \mathbb{R}^{D_u}$ 表示全 1 向量。由于 $\mathbf{u} \odot \boldsymbol{\eta}(\mathbf{t}) + \mathbf{u} = \mathbf{u} \odot (\boldsymbol{\eta}(\mathbf{t}) + \mathbf{1})$，该项仍然是历史线性的。目标残差和池化历史残差在 Steam 上使平均倒数排名变化 +0.0002 和 −0.0004。在 ML-25M 上，两种变化均为 −0.0010。

• **#1 对比 #4（交叉深度）。** 第 #4b 行在将恒等分支固定为一层并使用四个专家的情况下，单独考察相似度深度。增加第二个相似度层后，Steam 上的 MRR 变化为 +0.0074，ML-25M 上为 −0.0014，因此准确率变化方向取决于数据集。第 #4a 行还加深了恒等分支，MRR 分别变化了 +0.0051 和 +0.0053。第 3.3 节的 SP/PS保证适用于单层算子，而堆叠变体会改变模型类。因此，我们保留一层。

• **#1 对比 #5（多层感知机中的未交叉输入）。** 将未交叉池化历史 $\mathbf{u}$ 和投影后的目标 ˜$\mathbf{t}$ 加入交叉特征后，Steam 上的 MRR 变化为 +0.0038，ML-25M 上为 +0.0005。

• **#1 对比 #6（仅使用投影目标）。** 仅加入 ˜$\mathbf{t}$ 后，Steam 上的 MRR 变化为 −0.0077，ML-25M 上为 −0.0099。第 #5 行在两个数据集上都优于第 #6 行，因此观测值支持将投影目标与其预期进行比较的未交叉池化历史一同使用。

专家数量

表11：专家数量翻倍的影响。Δ 为使用 8 个专家时的平均倒数排名减去使用 4 个专家时的平均倒数排名。所有变体均不使用残差、采用仅交叉评分，并包含目标自交叉。“Pinned”将身份信号分支固定为一层和 4 个专家，而“Both”则同时更改两个信号分支。

	Steam		ML-25M	
配置	Both	Pinned	Both	已固定
1 个相似度层	+0.0088	+0.0035	+0.0048	−0.0032
2 个相似度层	+0.0030	−0.0088	+0.0042	+0.0031

在恒等容量固定的情况下，将专家数量加倍，在 Steam 上分别使一层和两层的平均倒数排名变化 +0.0035 和 −0.0088，在 ML-25M 上则分别变化 −0.0032 和 +0.0031。不同深度和数据集上的变化方向并不一致。耦合变体在四项比较中均提升了平均倒数排名，这表明恒等分支变化会改变专家数量所呈现的表面影响。我们保留四个专家作为固定默认值，因为这些单次运行的比较不足以据此作出选择。

固定默认值

表 12 列出了固定默认值、影响和建议。

E 容量匹配的 TA-Transformer

表 2 中的 TA-Transformer 使用 L 层和 1 头，以匹配工业对照模型。为区分架构与容量，我们训练了两层、两个头的变体，其前馈层宽度分别为$2D$（95,041 个可训练参数）和 $4D$（128,065 个），后者接近 MELODI 的 132,885。所有其他设置均与公开协议一致：学习率 10^{-3}、 $D = 64$，丢弃率 0.1，训练批次大小 1,024，评估批次大小 256，最多 200 个训练轮次，耐心值为 10，以及 $S = 50$。除参数匹配的 Steam 模型使用三个随机种子外，每个变体均只训练一次。延迟测试采用请求内局部复用，数量为 $C = 1{,}000$，并在同一台 32 核处理器上进行。我们禁用了 PyTorch 的嵌套张量路径；对于这种请求形状，该路径会使双头模型慢约 $4.5\times$。禁用该路径对分数的影响不超过 5×10^{-8}。

在 Steam 上，在三个随机种子下，参数匹配模型在六项采样指标上均超过 MELODI（平均倒数排名 0.4821 ± 0.0048 对比 0.4720 ± 0.0016）。在其中一次运行中，使用 999 个检索负样本时，它也处于领先地位（MELODI 减去该模型： -0.0040 $[-0.0046, -0.0035]$ 平均倒数排名）；但在全目录排序下，MELODI 以 $+0.0026$ $[+0.0022,$ 的 $+0.0031]$ 优势领先。在 Steam 上，较小的双头模型与 MELODI 的采样平均倒数排名持平，但在困难排序和全量排序下的平均倒数排名更低。在 ML-25M 上，参数匹配模型的采样平均倒数排名为 0.6921，高于 DIN + DCN-v2 的 0.6850；与此同时，MELODI 在所有报告的指标上仍然更高，并且在采样排序、困难排序和全量排序下分别以 +0.0113、 +0.0043 和 +0.0083 平均倒数排名领先该模型，所有配对自助法 95% 区间均高于零。在 $C = 1{,}000$ 个候选项下，参数匹配模型比 MELODI 慢约 $16\times$，因为它会针对每个候选项重新运行目标增强自注意力。

F 延迟基准测试

公共和工业延迟基准测试使用相同的商用处理器实例，该实例有 32 个虚拟核心，且不使用图形处理器加速：这是一个配备 64GiB 内存的 x86 平台，由第三代英特尔至强可扩展处理器（Ice Lake）提供支持。主要测量结果针对一位用户，使用 50 个历史项目和 1,000 个候选项进行评分。

公开数据集。 第5.1节所述的公开测试也会改变候选数量。每个模型都在全新的独立进程中使用 PyTorch 2.1 [18] 运行，且软件镜像、线程数和分配器设置相同。五种固定的随机置换模型顺序会改变每个模型在各次重复实验中的位置。每条路径都会记录 30 次预热调用后 300 次计时调用中的最小值。候选项展平路径和请求内局部复用路径在同一进程中运行，分别进行预热，且两者的运行顺序在各次重复实验中交替。两者的得分一致，最大绝对差异为 2.2×10^{-8}。表5列出了各次重复实验最小值的中位数。在 $C = 1{,}000$ 时，MELODI 的两条路径的最大值与最小值之差均不超过 $1.04\times$，而 DIN 的复用路径则达到 $2.56\times$。在各次重复实验中， $C = 1{,}000$ 时 MELODI 的配对性能优势在 Steam 上为 13.90–14.50×，在 ML-25M 上为 14.47–14.98×。使用 $C \in \{1, 10, 50, 100, 150, 200, 300, 500, 700, 1{,}000, 2{,}000\}$进行的第二次运行复现了所有单元格，误差在 $C \le 100$ 以内，且 MELODI 在 $C = 1{,}000$ 时的延迟为 1.96 和 1.95ms。DIN 的复用路径在最多 时比 MELODI 快（两个数据集上分别为 1.49 对 1.64 ms），从 $C = 700$ 开始则更慢（Steam 上为 5.08 对 1.71 ms，ML-25M 上为 2.16 对 1.72 ms）；DIN + DCN-v2 在 $C = 200$ 至 300 之间开始慢于 MELODI。 $C = 1{,}000$ 时的基线延迟与表5相差 9–25%，例如 Steam 上 DIN 的延迟为 7.38 ms，而表5中为 9.89 ms，因此相对于基线的比率取决于运行情况。

工业数据集。 工业基准测试采用 5 次预热迭代和 20 次计时迭代，并报告其平均值。其候选项展平路径将 1,000 个候选项与重复的历史记录一起进行批处理，因此会针对每个候选项重新编码该历史记录。请求内局部复用路径首先对一条历史记录执行嵌入查找和投影、位置编码、层归一化、$\mathbf{W}_k$ 和 $\mathbf{W}_v$，并处理填充、事件掩码和共享请求特征。随后，在对目标嵌入和查询路径进行批处理的同时扩展这些张量，并在候选项之间批处理注意力、时间衰减、事件历史池化、交叉模块、身份信号分支和评分器。因此，报告的 2.6–3.8× 倍缩减，是在同一请求中比较候选项展平后的重新编码与显式的单条历史记录复用。固定的历史长度使比较标准化，但并未涵盖所有观察到的工业历史记录。

Industrial-A。 复用将 MELODI 的延迟从 $40.0\times$ 降至 $15.6\times$，达到表格基线水平，降低了 $2.6\times$。复用后的模型比 TA-Transformer 快 $13.0\times$，比 DIN 快 $5.2\times$。

Industrial-B。 复用将 MELODI 的延迟从 $18.7\times$ 降至 $4.9\times$，达到表格基线水平，降低了 $3.8\times$。复用后的模型比 TA-Transformer 快 $28.7\times$，比 DIN 快 $11.1\times$。Industrial-B 使用 d=32，而 Industrial-A 使用 d=64。两者的配置在专家数量以及事件历史分支之间的交互方面也有所不同。在这两个工业数据集上，TA-Transformer 仍然最慢，因为它对每个候选项都需要在单独的目标–历史序列上执行自注意力。

激进预计算下界

Tab. 16 报告了一个激进下界，该下界不包括位置化历史 $\mathbf{K}$, $\mathbf{V}$、候选项 $\mathbf{Q}$ 以及归一化目标表示的构建。Industrial-A 的计时路径保留了掩码、时间衰减、共享特征投影、注意力、池化、交叉模块和评分器。Industrial-B 的计时协议也将掩码、时间衰减权重和共享特征投影移至计时区域之外。由于无法通过仅依据物品的查找恢复位置化的 $\mathbf{K}$，$\mathbf{V}$，两个基准测试都假设这些内容已针对组装好的历史序列完成计算。相较于请求内局部复用，它们的变化幅度低于 10%，且没有一致的变化方向。

G 实验详情

公开数据集协议

数据与划分。两个数据集均采用按时间顺序留一法：最后两个物品分别作为验证集和测试正样本，之前的物品构成训练序列。Steam 数据集根据 Kang 和 McAuley 的评论数据转储构建 [13]，包含 281,354 个用户和 11,961 个物品，共有 3,550,458 条经过过滤的交互记录。在迭代式用户–物品五核心过滤之前，我们会合并连续重复项。ML-25M 在预处理后包含 162,541 个用户和 32,720 个物品。历史序列会在右侧补齐，或截断为最近的 50 个物品。每个保留的训练前缀都会生成一个（历史序列、正向下一物品）样本。我们保留 Steam 的所有前缀（2,706,396 个样本），以及每位用户的

表 12：固定默认值及其观察到的平均倒数排名影响。每项影响均基于单次训练运行进行比较。

组件	影响（Steam / ML-25M）	建议
从恒等映射中移除目标自交叉分支	−0.0018/ −0.0151	保留经过测试的恒等分支，因为移除后报告的 MRR
池化历史残差	−0.0004/ −0.0010	省略以保留纯粹的交互信号
第二个相似度层	+0.0074/ −0.0014	使用一个层以保留既定的SP/PS保证
未交叉池化历史和投影后的多层感知机中的投影目标	+0.0038 / +0.0005	保留固定的仅交叉评分器
不带未交叉池化历史的投影目标历史序列	−0.0077/ −0.0099	如果评分器接收未交叉输入，则将目标与池化表示配对历史序列
专家数量	在集合中改变符号—置	使用 4 个专家作为固定默认值

表 13：更大型的 TA-Transformers：使用 99 个采样负例和请求局部延迟的单次运行指标，延迟为 $C = 1,000$（毫秒；Steam / ML-25M）。延迟来自一次基准测试运行。

		Steam			ML-25M			
模型	参数量	HIT@10	NDCG@10	MRR	HIT@10	NDCG@10	MRR	延迟
TA-Transformer（1L，1H）	61.6k	.7695	.5073	.4370	.9302	.7189	.6546	15.0 / 17.1
TA-Transformer（2L，2H）	95.0k	.7960	.5421	.4727	.9407	.7399	.6783	29.3 / 29.0
TA-Transformer（2L、2H、4D）	128.1k	**.8080**	**.5562**	**.4867**	.9461	.7519	.6921	32.4 / 31.3
MELODI	132.9k	.7923	.5405	.4720	**.9510**	**.7619**	**.7033**	**2.0 / 2.0**

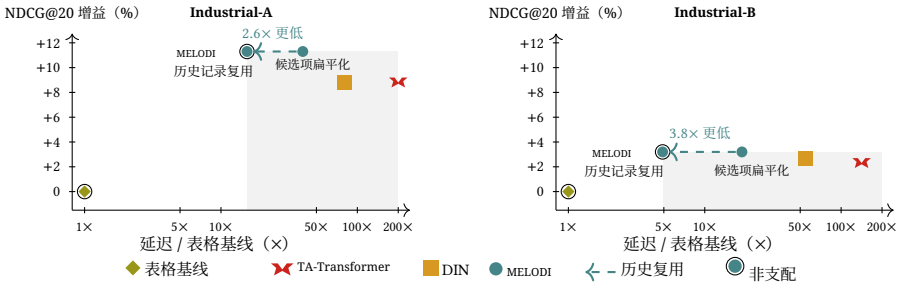

图4：在具有32个虚拟核心的商用处理器实例上，相对于表格基线的工业质量增益与延迟。质量指标是相对于表格基线的NDCG@20 增益（%）。对数横轴表示模型延迟与表格基线延迟之比，因此左上方更优。在两个数据集中，MELODI均比TA-Transformer和DIN更快，且报告的NDCG@20更高。蓝色虚线箭头表示实测的候选扁平化到请求本地复用的缩减，灰色区域表示被复用的MELODI所支配的区域，黑色圆环标记非支配的实测运行点。

表14：具有32个虚拟核心的商用处理器实例上的Industrial-A延迟。延迟除以表格基线延迟，NDCG@20 增益表示相对于表格基线的百分比变化。

模型	延迟（×）	NDCG@20 增益（%）
表格基线	**1.0**	–
TA-Transformer	201.8	+8.8
DIN	80.9	+8.8
MELODI（候选项展平）	40.0	**+11.3**
MELODI（历史记录复用）	15.6	**+11.3**

表 15：在配备 32 个虚拟核心的商用处理器实例上测得的 Industrial-B 延迟，报告方式同表 14。

模型	延迟（×）	NDCG@20 增益（%）
表格基线	**1.0**	–
TA-Transformer	142.1	+2.3
DIN	55.1	+2.7
MELODI（候选项展平）	18.7	**+3.2**
MELODI（历史记录复用）	4.9	**+3.2**

最新的 20 个 ML-25M 前缀（3,226,917）。

模型输入。两个数据集均使用来自 all-MiniLM-L6-v2 检查点 [25] 的冻结 384 维物品文本嵌入，并通过可训练的投影映射至 D=64。该检查点采用基于 MiniLM 编码器 [30]的 Sentence-BERT 框架 [21]。Steam 物品由于全零冻结嵌入无法区分或排序这些项目，因此在其元数据转储中缺失（<1%）的项目使用占位文本。

基线与配置。对于公开数据集，TA-Transformer 针对单个交互历史设有一个分支。该分支将候选项追加到序列中，应用单层单头双向Transformer，并对候选项位置的输出进行评分。除此之外，它与通用嵌入流程相同，并使用 BPR，而非

表 16：在具有 32 个虚拟核心的商用处理器实例上进行激进预计算时的在线延迟下界（1 位用户 $\times 50$ 历史序列 $\times 1$，000 个候选项）。延迟以各数据集的表格基线延迟为除数。

数据集 / 模型	扁平。	复用	下界
Industrial-A：MELODI	40.0×	15.6×	15.7×
Industrial-B：MELODI	18.7×	4.9×	4.6×

表 17：可训练参数（不含冻结的物品嵌入表）以及在每个公开数据集上训练的配置。在 Steam 和 ML-25M 上的数量相同。

模型	参数	配置
SASRec	127,936	2
TA-Transformer（1 层，1 个头）	61,569	1
TA-Transformer（2 层，2 个注意力头）	95,041	1
TA-Transformer（2 层，2 个注意力头，$4D$）	128,065	1
DIN	52,802	1
DIN + DCN-v2	90,314	1
CAN	109,437	1
DIEN	90,113	1
MELODI（4 个专家，1 层）	132,885	
MELODI（4 个专家，2 层）	167,273	
MELODI（8 个专家，1 层）	166,953	13
MELODI（8 个专家，2 层）	235,409	
无相似度交叉的 MELODI	115,205	

工业模型的逐点二元交叉熵。公开数据中不包含未知物品标识符，因此省略了可学习的未知标记参数。TA-Transformer 有 61,569 个可训练参数（不含冻结的物品表），其固定深度和注意力头数量与工业对照模型一致。公开版 MELODI 配置使用四个注意力头（H=4），每个头的宽度为 d=64 （D_u=256），并设置 r=32；评估的配置则改变 $K \in \{4, 8\}$以及交叉深度，取值为{1, 2}。

模型规模和配置。表17列出了可训练参数数量，以及在各数据集上训练的配置。预先指定的 MELODI 配置包含 132,885 个可训练参数， 2.2× TA-Transformer 包含 61,569 个， 2.5×DIN 包含 52,802 个。两个 SASRec 配置仅在目标归一化方面不同。三个 TA-Transformer 配置分别是基线，以及补充材料E节中的两个较大变体，其前馈层宽度为 $2D$ 和 $4D$。十三个 MELODI 配置包括表3中的四个网格单元和九个组件变体（表8和表11）。标题行采用第5.1节中的固定配置；若根据验证集MRR进行选择，则会选中 8 个专家、 2 层的变体。所有报告的运行均采用以下优化设置，我们的 DIEN 实现还加入了权重为 1 的辅助下一行为损失 [34]。

训练与模型选择。训练采用 BPR，每个样本使用一个负样本，Adam [14] 的学习率为 10^{-3}，在历史层归一化后使用 [26] 的丢弃率 0.1，批量大小为 1,024。早停以验证集MRR为监控指标，耐心值为 10，最多训练 200 个训练轮次。评估批次包含 1,024 个样本；DIN 和 TA-Transformer 则为 256 个。DIN 将 $[B, S, C, 4D]$ 激活张量具体化，而 TA-Transformer 为批次中的每个候选项构造一个目标增强序列。

评估。 每个留出测试物品都与 99 个负样本进行评分；这些负样本从该用户完整序列之外均匀抽取且互不重复，并由所有模型共享。最终测试集包含 Steam 的 281,354 个样例和 ML-25M 的 162,541 个样例。对于表 2 中每个模型的一次训练运行，以及表 3 中其余 MELODI 配置的一次训练运行，将所有与真实物品并列的情况都计为不利于真实物品时，每个秩均保持不变，因此这些检查点的所有报告值都不受平局处理方式影响。对于表 4，我们将并列情况计为不利于真实物品。Steam 测试物品中只有 0.5% 曾出现在可见历史中，因此其任务是下一物品预测，而非对已看过的物品重新排序。

工业数据集协议

数据与划分。 两个工业数据集都包含已记录的请求，并设有较晚的留出测试时段。Industrial-A 从其训练窗口中提取验证数据，而 Industrial-B 则使用较晚的窗口进行验证和测试。Industrial-A 包含来自 5,836 个请求的 591,286 条测试行，涵盖两个互不重叠的匿名化排序情境：Segment-1（232,908 条）和 Segment-2（358,378 条）。Industrial-B 包含来自 5,625 个请求的 1,088,853 条测试行，其中 562,927 条带有用户标识符，另有 525,926 条的用户标识符缺失或未知。这些细分组反映的是标识符可用性，而非历史长度。

模型输入。 对于每个数据集，所有模型都接收相同的请求级特征和物品表示。序列模型还会接收行为历史，这些历史按事件类型划分，并截取最近的事件。

基线与配置。 每个数据集使用各自的表格基线。TA-Transformer 为每种事件类型设置一个事件历史分支。每个分支都会追加目标，应用单层双向 Transformer 编码器，并使用目标位置输出。它不包含交叉网络或直接目标输入。DIN 同样为每种事件类型设置一个分支。其 MLP 激活函数（MLP($[\mathbf{h}、\mathbf{t}、\mathbf{h} \odot \mathbf{t}、\mathbf{h} - \mathbf{t}]$）计算逐物品权重。分别池化的事件历史分支进入 DCN-v2 目标交叉，投影目标进入评分器。MELODI 遵循第 4 节。两个数据集都使用事件历史池化、一个相似度交叉层、一个目标自交叉、 D=256、 H=8和 r=32。Industrial-A 使用 d=64 (D_u=512)、 K=8以及事件历史分支之间的逐头门控交叉。Industrial-B 使用 d=32 (D_u=256)、 K=4以及这些分支之间的 DCN-v2 交叉，但不包含直接目标输入。

训练与评估。 两个数据集均使用二元交叉熵和验证集 NDCG@20 进行检查点选择。固定比例训练负样本是来自同一请求的未购买候选项。评估会对留出的日志候选项进行排序，并将购买行为视为正向结果。

表18：数据集统计与评估方案。

	Steam	ML-25M
用户	281,354	162,541
物品	11,961	32,720
物品表大小	11,963	32,722
训练示例	2,706,396	3,226,917
每位用户的训练前缀	All	最近 20 个
嵌入源	all-MiniLM-L6-v2（384 维）	all-MiniLM-L6-v2（384 维）
隐藏维度 D	64	64
划分	留一法	留一法
测试样例	281,354	162,541
平均原始序列长度	12.6	153.5
模型历史长度限制	50	50
训练批次大小	1,024	1,024
评估批次大小	1,024（DIN 和 TA-Transformer：256）	1,024（DIN 和 TA-Transformer：256）