arXiv:2603.02653v1 [cs.IR] 3 Mar 2026

AlphaFree：免用户、ID和图神经网络的推荐

Minseo Jeon大韩民国首尔崇实大学
m
inseojeon@soongsil.ac.kr

Junwoo Jung大韩民国首尔崇实大学
jun
woojung@soongsil.ac.kr

Daewon Gwak大韩民国首尔崇实大学
dae
won1@soongsil.ac.kr

Jinhong Jung*大韩民国首尔崇实大学
jinhong@ssu.ac.kr

摘要

我们能否设计出不依赖用户、ID 和图神经网络的有效推荐系统？推荐系统是各个领域个性化内容分发的核心，其中，top-K 物品推荐是一项基础任务，旨在从历史交互中检索最相关的物品。现有方法依赖一些根深蒂固的设计惯例，且往往未经重新审视便直接采用，例如存储每个用户的嵌入（依赖用户）、根据原始 ID 初始化特征（依赖 ID），以及使用图神经网络（依赖图神经网络）。这些依赖带来了诸多局限，包括高昂的内存成本、冷启动和过平滑问题，以及对未见过的交互泛化能力较差。

在这项工作中，我们提出了 AlphaFree，这是一种不依赖用户、ID 和图神经网络的新型推荐方法。我们的核心思路是：无需用户嵌入即可即时推断偏好（无用户）；以预训练语言模型生成的语言表示（LR）取代原始 ID（无 ID）；并通过与相似物品进行增强及采用对比学习来捕捉协同信号，而不使用图神经网络（无 GNN）。在多个真实世界数据集上进行的大量实验表明，AlphaFree 始终优于竞争方法：相较于非基于 LR 的方法，性能最高提升约 40%；相较于基于 LR 的方法，最高提升 5.7%；同时，在使用高维 LR 时，GPU 内存使用量最高可减少 69%。

CCS概念

• **信息系统** →**推荐系统**。

关键词

无用户推荐、无ID表示、无GNN架构、协同增强、对比学习

ACM参考格式：

Minseo Jeon、Junwoo Jung、Daewon Gwak 和 Jinhong Jung。2026年。Al- phaFree：免用户、ID和GNN的推荐。载于*2026*年*ACM Web*会议论文集（*WWW '26*），*2026*年*4*月*13*日至*17*日，阿联酋迪拜。ACM，纽约，纽约州，美国，13页。https://doi.org/10.1145/ 3774904.3792355

1 引言

推荐系统在在线服务中发挥着核心作用，能够通过去除.....在电子商务等领域提供个性化内容，

*通讯作者。

本作品采用知识共享署名 4.0 国际许可协议进行许可。*WWW '26*，阿联酋迪拜© 2026年版权所有者/作者持有版权。ACM ISBN
979-8-4007-2307-0/2026/04
https://doi.org/10.1145/3774904.3792355

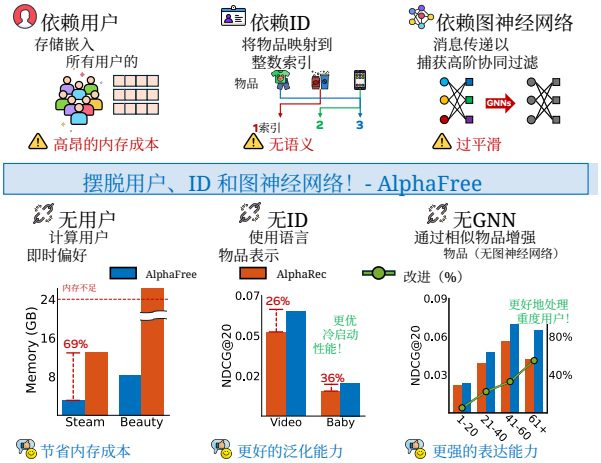


图 1：AlphaFree 的动机与有效性概览。（上）现有推荐系统依赖用户嵌入、物品ID 或图神经网络，从而导致较高的内存开销、冷启动局限性和过平滑。（下）AlphaFree 通过在不依赖用户、ID 和图神经网络的情况下执行推荐，克服了这些挑战：内存占用减少 69%，冷启动情况下准确率提高 26–36%，重度用户的改进最高约达 50%。

流媒体和社交媒体 [34, 44, 49]。该领域的一项基础任务是头部 K 物品推荐，旨在根据用户的历史交互 [19, 20,36]，检索 K最相关的物品。多年来，人们提出了许多方法来完成这一任务，从经典的协同过滤 [18, 31, 51, 52, 56]和基于神经网络的方法 [11, 13, 32, 62, 65]，到较新的基于语言模型的技术 [1, 9, 50, 55],，都致力于提升性能。在这一发展过程中，推荐系统建模主要受到若干惯例的影响，这些惯例已成为事实上的标准：存储用户嵌入、使用基于 ID 的特征进行初始化，以及采用图神经网络（GNN）捕捉协同信号。基于这些方面，我们将现有方法分为（1）用户依赖方法，（2）依赖 ID 的方法，以及（3）依赖图神经网络的方法。

用户依赖方法从历史交互中提取用户和物品嵌入，并要求在训练和推理期间持续维护用户嵌入。尽管用户嵌入编码了个体偏好，但当交互稀疏 [4]或存在噪声 [8, 15],时，它们会变得不可靠；而且其内存成本随用户数量线性增长，在电子商务等用户众多的领域，这一成本会变得相当高。*ID*依赖方法将原始离散标识符（ID）作为用户和物品的初始特征。虽然这种表示对于已见实体简单且有效，但缺乏语义含义，对未见实体的泛化能力也很差

，无法泛化到未见过的用户或物品。依赖图神经网络的方法使用图神经网络从显式图结构（例如用户–物品图）中捕获高阶协同信号。然而，这类方法也需要保留用户嵌入（因此本质上依赖用户）以及图结构，导致内存使用量较高；此外，由于过平滑，其表达能力也受到限制 [2, 37, 53]。

已有一些研究尝试部分解决这三种方法的局限性。例如，FISM [22] 和基于 SLIM 的方法 [43]既不存储用户嵌入，也不使用图神经网络；然而，它们依赖基于 ID 的特征，且难以捕捉高阶协同信号。LightGCN [11]和 XSimGCL [64] 利用图神经网络捕捉高阶结构，但同样依赖基于 ID 的特征、存储用户嵌入，并受到过平滑问题的影响。RLMRec [50] 和 AlphaRec [55] 等近期方法利用语言模型来挖掘物品的语义表示，但仍然依赖用户或依赖图神经网络。图 1 总结了这些依赖关系和局限性，启发了我们的研究；而表 1 表明，现有方法都无法同时解决这些问题，因此引出了一个问题：我们能否设计出不依赖用户、*ID* 和图神经网络的有效推荐系统？

本文提出 AlphaFree，一种面向 top-K 物品推荐的新颖方法，它摆脱了对用户、ID 和图神经网络的依赖，同时在各类基准测试上取得了强劲性能。AlphaFree 建立在三项核心设计原则之上：

- **无用户**：它不存储用户嵌入，而是根据交互历史即时计算潜在偏好，从而降低内存使用并改善冷启动处理。
- **无ID**：它用源自语言模型的物品语义表示替代原始 ID，从而能够泛化到未见物品并捕捉语义相似性。
- **无GNN**：它通过用行为相似且语义相似的物品增强表示并应用对比学习，在不使用图神经网络的情况下捕捉协同信号。

通过去除全部三项依赖，AlphaFree 相比现有方法实现了更高的内存效率、更好的泛化能力以及更高的准确率。我们的贡献如下：

- **方法**。 我们提出 AlphaFree，一种无需用户、ID 和图神经网络即可产生推荐结果的新颖方法。
- **技术**。 我们通过以下方式有效实现了设计原则：(1) 使用行为和语义上相似的物品进行增强，(2) 即时偏好建模，以及 (3) 在原始视图和增强视图之间进行对比学习。
- **实验**。 通过大量实验，我们证明，与最先进的方法相比，Al- phaFree 在推荐准确率和内存效率方面表现更优。

2 相关工作

依赖用户、ID 或图神经网络的传统方法。 大多数经典方法都依赖用户，因为它们需要用户嵌入。基于矩阵分解的方法，如 MF-BPR [31, 52] 和 NCF [13], , 从用户 - 物品矩阵中提取用户嵌入和物品嵌入。基于图的方法，如 Light-GCN [11] 和 XSimGCL [64], , 在用户-物品图上执行消息传递，以学习用户和物品的嵌入。尽管这些方法直接在嵌入空间中编码用户偏好，但其内存成本会随着用户数量的增加而上升；在电子商务等用户数量庞大的领域，这一成本相当可观（见表2）。

此外，包括上述模型在内的传统方法都依赖*ID*，由于缺少用户或物品内容且依赖用户-物品交互，它们会使用原始离散 ID（例如用户或物品各自唯一的整数索引）初始化嵌入。这些 ID 通常会被映射为独热编码，或被映射为随机初始化的特征，再经过训练。虽然这种方法简单直接，且无需领域知识便能有效学习交互，但在交互稀疏时难以捕捉有意义的模式，也难以泛化到新实体。

随着图神经网络的兴起，许多依赖图神经网络的方法应运而生，它们依靠在用户-物品图上进行消息传递来捕获高阶协同信号。例如，LightGCN 使用简化的图神经网络层，在图中传播用户和物品嵌入，通过多层传递捕获高阶信号。XSimGCL [64] 进一步引入图对比学习 [59, 63, 69] ，以提升表达能力。这些方法利用高阶协同过滤信号来提升性能，但本质上仍依赖用户，需要为所有用户生成嵌入。此外，它们还存在过平滑问题：经过反复消息传递后，嵌入会变得过于相似。由于对大量邻居进行过度平均，这一问题在高阶节点中尤为严重，从而限制了性能。

部分摆脱用户、ID 或图神经网络依赖的方法。已有一些研究尝试部分减轻这些依赖。无用户方法会舍弃用户嵌入，仅保留物品嵌入。沿着这一思路，SLIM [43] 直接学习物品间的相似度，而 FISM [22] 则通过物品的因子化矩阵对其进行建模。然而，这两种方法都仍然依赖ID。MF-BPR 和 NCF 等早期方法在图神经网络出现之前便已提出，因此自然不依赖图神经网络，但仍依赖用户和ID。随着语言模型的发展，近期提出了 RLMRec [50]和 AlphaRec [55] ，它们利用从物品内容中提取的语言表示（LR），在训练过程中取代ID，从而实现无ID。不过，这两种方法都存储用户嵌入，因此仍然依赖用户；此外，AlphaRec 还依赖图神经网络。具体而言，基于LR的方法在用户依赖性方面面临内存消耗上的挑战：大多数语言模型会生成高维嵌入，随后用这些嵌入为每位用户计算嵌入[1]。据我们所知，现有工作尚未能同时消除所有这些依赖，而我们的 AlphaFree 成功实现了这一点。

3 预备知识

符号说明。 我们将所有物品的集合记为 $\mathcal{I}$， t_i是物品 i 的相关文本内容（例如标题）。令 $I \subseteq \mathcal{I}$表示任意用户的交互集，即 $I = \{i_1, \cdots \ i_{|I|}\}$。令 $\mathcal{H}$表示数据集中所有交互集的集合，即 $\mathcal{H} = \{I \mid I \subseteq \mathcal{I}\}$。**问题定义**。 该问题定义如下：

问题1（前K 物品推荐）。给定一个由交互集组成的训练集合$\mathcal{H}_{\text{train}}$，目标是学习推荐模型的参数$\theta$，以便为查询交互集 I_q推荐排名前K 的物品列表$\mathcal{R}^{I_q}$：

$$\mathcal{R}_{I_q} := \texttt{argtop}_K\Big(\{y_\theta(i \mid I_q) \mid i \in \mathcal{I}\}\Big)$$

[1]例如，在初始阶段，LightGCN 使用 64 维嵌入 [11], , 而 AlphaRec 采用来自 text-embedding-3-large 的 3,072 维嵌入 [41, 55]。

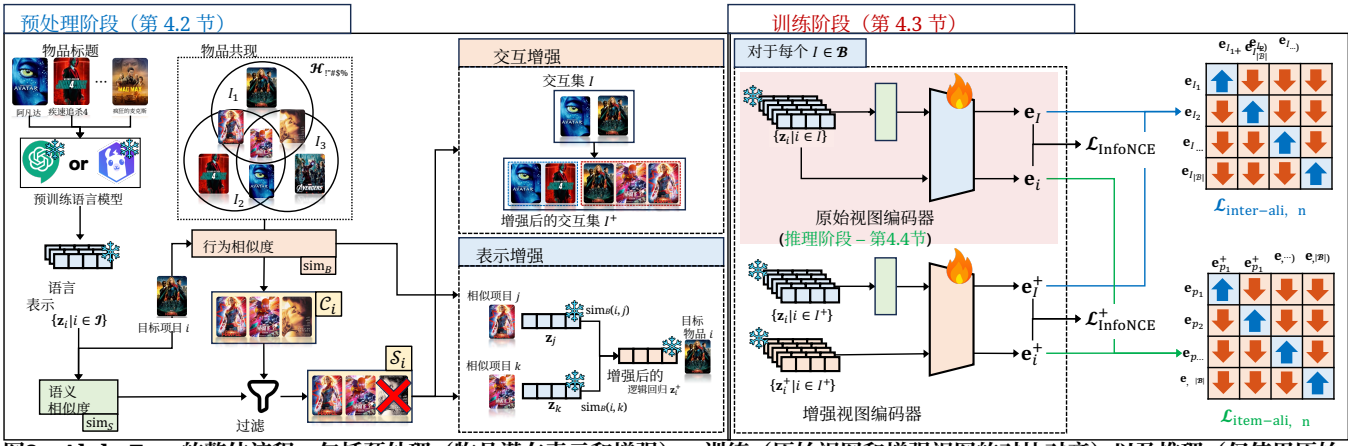


图2：AlphaFree的整体流程，包括预处理（物品潜在表示和增强）、训练（原始视图和增强视图的对比对齐）以及推理（仅使用原始视图编码器）。符号说明见表4。

其中，$y_\theta(i \mid I_q)$ 表示模型 θ 为物品 i 针对 I_q 生成的推荐分数，而 $\texttt{argtop}_K(\cdot)$ 返回分数最高的前 K 个物品组成的列表。

4 所提方法

我们针对问题1提出了AlphaFree，这是一种新颖的推荐方法，不依赖用户、ID和图神经网络。

4.1 概述

我们描述 AlphaFree 的概述，其中图 2 展示了其整体流程。如第 1 节所述，我们基于以下原则设计 AlphaFree：

- **无用户：**我们采用无用户机制，从交互集即时推断潜在偏好，从而消除用户嵌入并提高内存效率[2]。
- **无ID：**我们使用预训练语言模型从物品内容得到的语言表示（LRs）替换原始物品ID，从而增强泛化。
- **无GNN：**我们避免使用图神经网络，转而用相似物品增强嵌入，通过避免消息传递和过平滑来提高效率与表达能力。

通过去除这些依赖，我们表明一个简单轻量的模型能够实现高效的内存使用（在使用高维LR时尤为重要），同时在多种基准测试上取得更高的推荐准确率。

AlphaFree 分为三个阶段：预处理、训练和推理（详情请参阅附录 C）。预处理阶段（第 4.2 节）计算所有物品的语言表示，并检索行为和语义上相似的物品，以增强交互和表示。训练阶段（第 4.3 节）以批次为单位学习物品和交互集的嵌入，并通过与增强后的交互进行对比学习来注入高阶协同过滤信号。与训练不同，推理阶段（第4.4节）仅基于查询交互集 I_q 进行推荐，不使用增强。由于训练后的参数嵌入了增强后交互中的信息，因此能够有效捕捉 I_q 中的潜在偏好。

[2]这里，“无用户”并不意味着去除个性化，而是旨在通过即时计算来消除存储用户嵌入的需要。

4.2 预处理阶段（算法 1）

此阶段旨在对 $\mathcal{I}$ 中的所有物品的语言表示进行预处理，并利用语言表示和 $\mathcal{H}_{\text{train}}$ 为目标物品识别相似物品，再将其用作高阶协同过滤信号，以增强交互和表示。请注意，此阶段仅在训练前执行一次。

4.2.1 物品的语言表示。与依赖 ID 的方法不同，基于语言表示的方法（如 AlphaRec [55]）提供了经验证据，表明物品潜在表示本身就能捕捉协同过滤信号，这意味着无需依赖原始 ID 也能获得更好的性能。受此启发，我们采用预训练语言模型 语言模型(·)，实现**无 ID** 设计，如下所示：

$$\mathbf{z}_i \leftarrow \texttt{LM}(t_i), \tag{1}$$

其中，$\mathbf{z}_i \in \mathbb{R}^{d_{\text{LR}}}$ 表示物品 i 的语言表示，输出维度为 d_{LR}；t_i 是物品 i 的文本描述（即其标题）。任何预训练语言模型都可用于此目的（例如 BERT、LLaMA 或 GPT），语言模型的详细配置见附录 B。

4.2.2 行为和语义相似的物品。对于每个物品 i，我们使用 $\mathcal{H}_{\text{train}}$ 和 $\{\mathbf{z}_j\}$ 来识别相似物品集合 S_i。每个 $I \in \mathcal{H}_{\text{train}}$ 都提供行为信号，表明用户消费了哪些物品；而 $\mathbf{z}_j$ 则捕捉物品 j 的语义。我们介绍一种简单而有效的方法，利用这些信息获取 S_i。

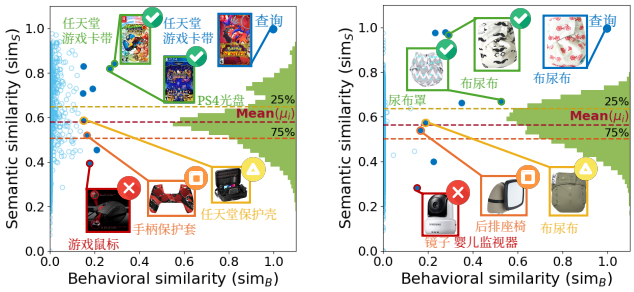
行为相似度。对于 $\mathcal{H}_{\text{train}}$，我们采用共现 [25] 作为行为相似度，因为它能够以简单高效的方式捕捉协同过滤信号，并已在以往工作中得到广泛应用 [39, 58, 68]。其直觉是，经常一起被消费的物品很可能具有相同的协同模式，而共现则利用交互作为隐式桥梁来捕捉高阶信号。形式上，该度量定义如下：

$$\texttt{sim}_B(i,j) = \sum_{I \in \mathcal{H}_{\text{train}}} \mathbb{I}(i \in I) \cdot \mathbb{I}(j \in I), \tag{2}$$

其中，$\mathbb{I}(\cdot)$ 在给定谓词成立时返回 1，否则返回 0。**语义相似性。**我们还考虑语义相似性，它能够捕捉物品的内在含义。由于语言表示通常通过点积进行比较 [24, 61]，我们采用这一度量，定义如下：

$$\texttt{sim}_S(i,j) = \mathbf{z}_i^\top \mathbf{z}_j, \tag{3}$$

其中，$\mathbf{z}_i$ 和 $\mathbf{z}_j$ 分别是物品 i 和 j 的语言表示。



(a) 视频。 **(b) 婴儿。**

图3：查询物品 i的行为相似性与语义相似性示例（最小-最大归一化），其中 $K_c = 10$，蓝色圆圈表示基于$\mathtt{sim}_B$的C_i，绿色直方图显示$\mathtt{sim}_S$的分布。基于$\mathtt{sim}_S$ 的25%阈值会产生过于语义相似的物品，75%阈值则会纳入关联较弱的物品，而 μ_i 提供了平衡的权衡。

行为与语义过滤。 虽然$\mathtt{sim}_B$ 提供了有用的共现信号，但可能包含噪声候选项。为优化选择，我们提出了基于两种相似度的过滤策略。具体而言，我们首先选取与物品 i的$\mathtt{sim}_B$ 最高的前 K_c 个物品，构建候选集 C_i：

$$C_i = \mathtt{argtop}_{K_c}\Big(\{\mathtt{sim}_B(i,j) \mid j \in \mathcal{I}\}\Big). \tag{4}$$

随后，我们通过仅保留语义相似性超过均值 μ_i 的项来过滤C_i，如下所示：

$$\mathcal{S}_i = \{j \in C_i \mid \mathtt{sim}_S(i,j) \geq \mu_i\}, \quad \mu_i = \frac{1}{|\mathcal{I}|}\sum_{j\in\mathcal{I}} \mathtt{sim}_S(i,j). \tag{5}$$

图3展示了我们策略背后的直觉以及对 μ_i的选择。给定查询物品 i（蓝色），候选首先通过$\mathtt{sim}_B$ 检索，再使用 μ_i依据$\mathtt{sim}_S$ 进行细化。语义相似度分布显示了25%、 μ_i和75%的取值；25%截断过严，75%截断过松，而 μ_i 提供了一个平衡的阈值，在保留语义相关物品的同时过滤掉不相关物品。这些截断值下的性能见附录D，其中 μ_i 提供了更好的性能。

4.2.3 交互与表示增强。对于每个物品 i，我们获取其相似物品集合 S_i，将其作为后续增强所需的高阶协同过滤信号，使我们的模型在训练和推理阶段均保持**无GNN**。

交互增强。为丰富协同过滤信号，我们将相似物品加入每个 $I \in \mathcal{H}_{\text{train}}$ 中。对于每个物品 $i \in I$，我们使用 S_i并与 I取并集，得到增强后的集合：

$$I^+ = I \cup \mathtt{union}\Big(\{\mathcal{S}_i \mid i \in I\}\Big). \tag{6}$$

这在交互层面注入了高阶协同过滤信号，添加了未被明确观测到的新交互。

表示增强。我们还利用 S_i增强每个物品 i 的表示。在传统的物品-物品协同过滤中，频繁在交互中共现的物品被视为彼此相关，并能提供有用的协同过滤信号。受此启发，我们根据行为相似度对相似物品S_i的语言表示进行加权组合：

$$\mathbf{z}_i^+ = \frac{\sum_{j\in\mathcal{S}_i} \mathtt{sim}_B(i,j)\cdot \mathbf{z}_j}{\sum_{j\in\mathcal{S}_i} \mathtt{sim}_B(i,j)}. \tag{7}$$

这在表示层面注入了高阶协同过滤信号，通过基于共现的行为信号增强$\mathbf{z}_i$。

4.3 训练阶段（算法 2）

此阶段使用原始视图和增强视图（或数据），并在每个训练轮次中通过小批次 $\mathcal{B} \subseteq \mathcal{H}_{\text{train}}$ 学习模型参数 Θ。对于每个$\mathcal{B}$，我们对两个视图进行编码并将其对比对齐，从而向 Θ中注入高阶协同过滤信号。

4.3.1 原始视图和增强视图的编码器。我们将说明两个编码器如何为每个 $I \in \mathcal{B}$生成嵌入。

原始视图。 我们使用多层感知机层计算 I及其组成物品的嵌入，从而对原始视图进行编码：

$$\mathbf{e}_I = \mathtt{MLP}\Big(\mathtt{mean}\big(\{\mathbf{z}_i \mid i \in I\}\big)\Big), \text{ and } \mathbf{e}_i = \mathtt{MLP}(\mathbf{z}_i),\ \forall i \in I,$$

其中$\mathbf{e}_I$ 根据交互 I捕获任意用户的潜在偏好，并且是即时计算的，从而使我们的模型**无用户**，无需存储用户嵌入。

该结构可视为不使用图神经网络的AlphaRec [55] 的对应方案，但仅靠它无法捕获高阶协同过滤信号。因此，AlphaRec依赖图神经网络来学习此类信号，从而既依赖用户又依赖图神经网络。然而，AlphaFree通过对以下增强视图进行编码来避免使用图神经网络，从而消除了这些依赖。

增强视图。 我们使用另一个多层感知机层，根据预处理阶段获得的增强后的语言表示$\{\mathbf{z}_i^+\}$计算 I^+ 的嵌入，从而对增强视图进行编码：

$$\mathbf{e}_I^+ = \mathtt{MLP}^+\Big(\mathtt{mean}\big(\{\mathbf{z}_i \mid i \in I^+\}\big)\Big), \text{ and } \mathbf{e}_i^+ = \mathtt{MLP}^+(\mathbf{z}_i^+),\ \forall i \in I^+,$$

其中$\mathbf{e}_I^+$表示针对 I的交互级增强，因此使用原始语言表示；而$\mathbf{e}_i^+$则对应于基于增强后的语言表示、针对 i 的表示级增强。

用于推荐的 InfoNCE 损失。接下来，我们利用每个视图中的嵌入来学习推荐结果。一种常见方法是 BPR 损失 [52]，，它将一个正样本与一个负样本进行对比。然而，近期研究 [3, 55], 已通过实验证明，在推荐任务中，将一个正样本与多个负样本进行对比的 InfoNCE，其效果优于 BPR。因此，我们也采用该方法。为简洁起见，我们首先给出对数 *softmax*（或将其负值视为局部损失）的一般定义：

$$\ell^{(\tau)}(\mathbf{a},\mathbf{p};C) = \log\frac{\exp\big(\sigma(\mathbf{a},\mathbf{p})/\tau\big)}{\sum_{\mathbf{c}\in C}\exp\big(\sigma(\mathbf{a},\mathbf{c})/\tau\big)},$$

其中，$\sigma(\cdot,\cdot)$ 表示余弦相似度，τ表示温度参数，$\mathbf{a}$和$\mathbf{p}$ 分别表示锚点 样本和正样本 的嵌入，C 表示 softmax 中纳入考虑的候选嵌入集合。该式量化了锚点–正样本对相对于 C中其他候选项的相对接近程度。

对于每个 $I \in \mathcal{B}$，我们随机采样一个正样本物品 $p \sim U(I)$，并将 N_I 定义为从 $\mathcal{I} \setminus I$ 中抽取的负样本集合。于是，原始视图的 InfoNCE 损失如下：

$$\mathcal{L}_{\text{InfoNCE}}(\mathcal{B}) = -\frac{1}{|\mathcal{B}|}\sum_{I\in\mathcal{B}} \ell^{(\tau_r)}(\mathbf{e}_I,\mathbf{e}_p;\mathcal{E}_I), \tag{8}$$

其中 $\mathrm{E}_I = \{\mathbf{e}_i \mid i \in N_I \cup \{p\}\}$，而对于每个负样本项 $i \in N_I$，$\mathbf{e}_i$ = 多层感知机($\mathbf{z}_i$)。这会促使$\mathbf{e}_I$更接近$\mathbf{e}^p$，同时将其与负样本区分开来。类似地，增强视图的损失

表示为：

$$\mathcal{L}^{+}_{\text{InfoNCE}}(\mathcal{B}) = -\frac{1}{|\mathcal{B}|}\sum_{I\in\mathcal{B}} \ell^{(\tau_r)}(\mathbf{e}^{+}_I, \mathbf{e}^{+}_p; \mathcal{E}^{+}_I), \tag{9}$$

其中 $\mathrm{E}^{+}_I = \{\mathbf{e}^{+}_i \mid i \in N_I \cup \{p\}\}$，而 $\mathbf{e}^{+}_i$ = 多层感知机$^{+}(\mathbf{z}^{+}_i)$ 对应每个负样本物品 $i \in N_I$.[3] 这与针对负样本的 $\mathbf{e}^{+}_I$和$\mathbf{e}^{+}_p$形成对比。为简便起见，两种损失使用相同的 τ_r。为提高效率，我们在每个批次中计算所有物品的 $\mathbf{e}_i$ 和 $\mathbf{e}^{+}_i$（见附录 C）。

4.3.2 跨视图对齐。原始视图捕捉物品消费情境，但缺少高阶协同过滤信号；增强视图则提供此类信号，但可能过度强调这些信号。因此，我们对齐这两个视图，使模型能够有效整合它们互补的信息。为此，受 CLIP 风格对比损失 [48],的启发，我们对其进行调整，使两个视图在交互层面和物品层面上都能对齐。

交互级对齐。 我们首先对齐交互集的原始与增强嵌入。遵循 CLIP 框架，每对 $(\mathbf{e}_I, \mathbf{e}^{+}_I)$ 都被视为 $I \in \mathcal{B}$ 中的正样本，而 B 中其他交互嵌入则作为负样本，从而提升训练过程的效率和稳定性。为了确保 $\mathbf{e}_I$ 和$\mathbf{e}^{+}_I$之间的双向一致性，我们将交互级对齐损失设计为对称形式，如下所示：

$$\mathcal{L}_{\text{inter-align}}(\mathcal{B}) = -\frac{1}{2|\mathcal{B}|}\sum_{I\in\mathcal{B}}\Big(\underbrace{\ell^{(\tau_a)}(\mathbf{e}_I, \mathbf{e}^{+}_I; \mathcal{E}^{+}_{\mathcal{B}})}_{\text{original}\to\text{augmented}} + \underbrace{\ell^{(\tau_a)}(\mathbf{e}^{+}_I, \mathbf{e}_I; \mathcal{E}_{\mathcal{B}})}_{\text{augmented}\to\text{original}}\Big), \tag{10}$$

其中，$\mathcal{E}_{\mathcal{B}} = \{\mathbf{e}_I \mid I \in \mathcal{B}\}$ 和 $\mathcal{E}^{+}_{\mathcal{B}} = \{\mathbf{e}^{+}_I \mid I \in \mathcal{B}\}$分别表示。对于每个 I，前者促使原始 $\mathbf{e}_I$ 比其他增强嵌入更接近其对应的增强 $\mathbf{e}^{+}_I$，而后者则针对其他原始嵌入，施加相反方向的约束。

物品级对齐。 接下来，我们对齐物品的原始与增强嵌入。在这一步中，我们只考虑正样本物品，因为目标是对齐两个视图中的相同物品对，而利用 B 中的正样本物品便足以实现这一目标；纳入所有其他物品则会增加计算成本。假设 $P_{\mathcal{B}}$ 是$\mathcal{B}$的正样本物品集合，即 $P_{\mathcal{B}} = \{p \sim U(I) \mid I \in \mathcal{B}\}$。那么，物品级对齐损失定义如下：

$$\mathcal{L}_{\text{item-align}}(\mathcal{B}) = -\frac{1}{2|P_{\mathcal{B}}|}\sum_{p\in P_{\mathcal{B}}}\Big(\underbrace{\ell^{(\tau_a)}(\mathbf{e}_p, \mathbf{e}^{+}_p; \mathcal{E}^{+}_{P_{\mathcal{B}}})}_{\text{original}\to\text{augmented}} + \underbrace{\ell^{(\tau_a)}(\mathbf{e}^{+}_p, \mathbf{e}_p; \mathcal{E}_{P_{\mathcal{B}}})}_{\text{augmented}\to\text{original}}\Big), \tag{11}$$

其中，$\mathcal{E}^{P_{\mathcal{B}}} = \{\mathbf{e}_p \mid p \in P_{\mathcal{B}}\}$和$\mathcal{E}^{+}_{P_{\mathcal{B}}} = \{\mathbf{e}^{+}_p \mid p \in P_{\mathcal{B}}\}$分别表示前者将$\mathbf{e}^{p}$ 与其他增强后的嵌入进行对比并对齐到$\mathbf{e}^{+}_p$，后者则将$\mathbf{e}^{+}_p$ 与其他原始嵌入进行对比并对齐回$\mathbf{e}^{p}$。

4.3.3 最终损失。 给定$\mathcal{B} \subseteq \mathcal{H}_{\text{train}}$，我们描述用于训练模型的最终损失 $\mathrm{L}_{\text{final}}(\cdot)$，其定义如下：

$$\mathcal{L}_{\text{final}}(\mathcal{B}) = \mathcal{L}_{\text{rec}}(\mathcal{B}) + \lambda_{\text{align}} \cdot \mathcal{L}_{\text{align}}(\mathcal{B}), \tag{12}$$

其中，$\mathrm{L}^{\text{rec}}(\cdot)$ 是推荐损失，$\mathrm{L}^{\text{align}}(\cdot)$ 是对齐损失，λ_{align} 是用于平衡这两部分的超参数。损失函数 $\mathrm{L}^{\text{rec}}(\cdot)$ 和 $\mathrm{L}^{\text{align}}(\cdot)$ 定义如下：

$$\mathcal{L}_{\text{rec}}(\mathcal{B}) = \mathcal{L}_{\text{InfoNCE}}(\mathcal{B}) + \mathcal{L}^{+}_{\text{InfoNCE}}(\mathcal{B}),$$
$$\mathcal{L}_{\text{align}}(\mathcal{B}) = \mathcal{L}_{\text{inter-align}}(\mathcal{B}) + \mathcal{L}_{\text{item-align}}(\mathcal{B}).$$

[3]我们测试了来自 $\boldsymbol{I} \setminus I^{+}$的负样本，但在初步实验中发现其性能与来自 $\boldsymbol{I} \setminus I$的负样本没有差异；因此，为了提高效率，我们在式 (8) 中复用 N_I。

请注意，AlphaFree 的参数仅包含多层感知机层的参数，即 $\Theta = \{\theta_{\text{MLP}}, \theta_{\text{MLP}^{+}}\}$，其中每个多层感知机层都是一个两层网络，将 d_{LR}映射到 d，并且每层都有一个权重矩阵和一个偏置。在训练过程中，对于每个训练轮次中的每个 B，我们计算 $\mathrm{L}_{\text{final}}(\mathrm{B})$，反向传播梯度，并通过梯度下降更新 Θ。

4.4 推理阶段（算法3）

在推理阶段，我们基于训练后的参数，针对查询交互集 I_q计算所有物品的推荐分数。此阶段的思路是使用原始视图中的参数 θ_{MLP}。主要原因有：（1）训练期间，增强视图中的高阶协同过滤信号已对齐到原始视图编码器多层感知机$(\cdot)$中；（2）因此，推理时无需对 I_q 进行增强，从而提高效率。对于每个物品 i，问题1中相对于 I_q 的推荐分数定义为：

$$y_{\theta_{\text{MLP}}}(i \mid I_q) = \sigma\big(\mathbf{e}_{I_q}, \mathbf{e}_i\big), \tag{13}$$

其中，$\sigma(\cdot,\ \cdot)$ 表示余弦相似度，$\mathbf{e}^{I_q}$ = 多层感知机(均值($\{\mathbf{z}_i \mid i \in I_q\}$))，以及$\mathbf{e}_i$ = 多层感知机$(\mathbf{z}_i)$。请注意，**我们的推理阶段，以及预处理和训练，均可在无需用户、ID 和图神经网络的情况下进行推荐**，因为它能够为任意用户即时高效地计算其 I_q 嵌入（无用户），利用物品潜在表示（无***ID***），并将高阶协同过滤信号对齐到原始视图编码器中，而无需依赖任何图神经网络（无***GNN***）。

4.5 复杂度分析

假设 $n = |\mathcal{I}|$、 $m = \sum_{I\in\mathcal{H}_{\text{训练}}}|I|$ 和 $h = |\mathcal{H}_{\text{train}}|$分别表示物品数、交互数和交互集数。令 d_{LR}表示语言表示的维度。其他项均为超参数。详细分析和证明见附录G。

时间复杂度。 AlphaFree 的时间复杂度如下：

定理1（AlphaFree 的时间复杂度）。 在预处理阶段，*AlphaFree* 耗时 $O(T_{\text{LM}}n + mn + (d_{\text{LR}} + \log K_c)n^2)$，其中 T_{LM}是语言模型所用的时间。其训练阶段每个训练轮次需要 $O((T_i dn + K_c m + dh)d_{\text{逻辑回归}} + (n_s + |\mathcal{B}|)hd)$时间，其中 $T_i = h/|\mathcal{B}|$是遍历所有批次所需的迭代次数。其推理阶段耗时 $O(d_{\text{LR}}dn + n\log K)$。 ■

假设包括 d_{LR}在内的超参数均为固定常数，则预处理阶段耗时 $O(mn + n^2)$，而训练阶段耗时 $O(m + h + n)$。尽管预处理阶段不具备线性可扩展性，但它只执行一次，并且可以使用近似 k-NN 进一步优化（见附录 A）。相比之下，训练阶段相对于 m、 h和n呈线性扩展，推理阶段也随 n线性扩展。

空间复杂度。 AlphaFree 的空间复杂度如下：

定理 2（AlphaFree 的空间复杂度）。 预处理阶段使用 $O(d_{\text{LR}}n + K_c(m + n))$ 空间。训练阶段需要 $O(m + d(n + d_{\text{逻辑回归}} + n_s + |\mathcal{B}|))$ 空间。推理阶段需要 $O((d_{\text{LR}} + d)n + K)$ 空间。 ■

当包括 d_{LR} 在内的超参数为固定常数时，预处理和训练阶段使用 $O(m+n)$ 空间，而推理阶段使用 $O(n)$ 空间。请注意，LightGCN 和 AlphaRec 在训练和推理时使用 $O(n + h + m)$ 空间，其中 h 和 m 分别表示用户数和边数；而 AlphaFree 在训练时将空间需求降至 $O(h)$，在推理时降至 $O(h + m)$。

表 1：按用户划分的留出集设置下，以 Recall@K 和 NDCG@K ($K = 20$) 衡量的推荐性能。我们标明各方法是否满足三项设计原则：UF（无用户信息）、IF（无ID信息）和 GF（无GNN）。最佳的非LR结果以红色标示，最佳的LR结果（不包括 AlphaFree）以蓝色标示。请注意，只有 AlphaFree 遵循所有设计原则、取得最高准确率，并成功处理了部分基于LR的方法无法处理的所有数据集。

模型	原则 UF	IF	GF	Recall@20 电影	Book	视频	Baby	Steam	Beauty	健康	NDCG@20 电影	Book	视频	Baby	Steam	Beauty	健康
MF-BPR	✗	✗	✓	0.0580	0.0436	0.0177	0.0150	0.1610	0.0063	0.0091	0.0533	0.0387	0.0095	0.0074	0.1537	0.0032	0.0048
FISM-BPR	✓	✗	✓	0.0861	0.0623	0.0392	0.0150	0.1801	0.0079	0.0104	0.0801	0.0540	0.0220	0.0076	0.1431	0.0044	0.0060
LightGCN	✗	✗	✗	0.0860	0.0712	0.0732	0.0359	0.2013	0.0201	0.0250	0.0754	0.0597	0.0409	0.0185	0.1518	0.0110	0.0142
XSimGCL	✗	✗	✗	0.0967	0.0818	0.0897	0.0390	0.2245	0.0253	0.0299	0.0866	0.0690	0.0500	0.0203	0.1664	0.0140	0.0170
RLMRec	✗	✓	✗	0.1046	0.0905	超时	超时	超时	超时	超时	0.0942	0.0741	超时	超时	超时	超时	超时
AlphaRec	✗	✓	✗	0.1219	0.0991	0.1088	0.0391	0.2360	内存不足	内存不足	0.1141	0.0829	0.0605	0.0210	0.1884	内存不足	内存不足
AlphaFree	✓	✓	✓	**0.1267**	**0.1014**	**0.1111**	**0.0412**	**0.2402**	**0.0361**	**0.0325**	**0.1194**	**0.0861**	**0.0615**	**0.0219**	**0.1938**	**0.0200**	**0.0184**
相较最佳逻辑回归的提升百分比				3.94%*	2.32%*	2.11%*	5.37%*	1.78%*	-	-	4.65%*	3.86%*	1.65%*	5.71%*	2.89%*	-	-
较最佳非逻辑回归模型的提升百分比				31.00%*	23.96%*	23.81%*	5.58%*	6.99%*	42.69%*	8.70%*	37.90%*	24.78%*	23.07%*	9.26%*	16.49%*	42.86%*	8.24%*

- 超时（o.o.t.）表示包括预处理在内的总时间超过了50小时，而内存不足（o.o.m.）表示GPU内存溢出。
- 标有 *的百分比增长具有统计显著性（p值 ≤ 0.05）。

表2：真实世界推荐数据集的统计信息。

数据集	电影	图书	视频	Baby	Steam	Beauty	健康
用户数	26,073	71,306	94,762	150,777	334,730	729,576	796,054
物品数	12,464	40,523	25,612	36,013	15,068	207,649	184,346
#Inter.	875,906	2,206,865	814,586	1,241,083	4,216,781	6,624,441	7,176,552

5 实验

我们展示了实验结果，以评估 AlphaFree 相较于解决问题1的最先进基线方法的有效性。

5.1 实验设置

数据集与竞争方法。 我们在表2中的七个真实世界数据集上进行了实验，每个数据集的详细信息见附录 B。我们将 AlphaFree 的基线方法分为以下几类：

- **非基于LR的方法。** 我们考虑了 MF-BPR [31]，，这是一种广泛使用的基线方法，会存储用户嵌入；还考虑了 FISM-BPR [22]，，这是一种知名的无用户模型。我们还纳入了 LightGCN [11] 和 XSimGCL [64]，，它们是流行的基于GNN的方法。这些方法都依赖ID，不使用语言表示。

- **基于LR的方法。** 我们纳入了 RLMRec [50] 和 AlphaRec [55]，，它们是当前最先进的无ID方法。本节使用的语言模型详情见附录 B，附录E还提供了关于不同语言模型影响的补充分析。

训练与评估流程。 每个数据集都被划分为训练集、验证集和测试集，具体比例和划分策略将在以下小节中介绍。我们使用 Recall@20 在验证集上调整每个模型（包括 AlphaFree）的超参数。确定超参数后，我们根据 Recall@20 和 NDCG@20 在测试集上评估各模型。每项实验均使用不同的随机种子运行五次，并报告平均性能。设置详情请参阅附录 B。

可复现性。 论文中使用的代码和数据集可在 https://github.com/minseojeonn/AlphaFree获取。

5.2 推荐性能

我们按照近期研究的方法 [55, 66]，，采用按用户划分的留出集评估了 top-K 推荐的性能。具体而言，

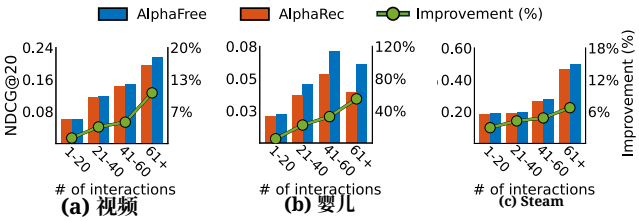


图4：AlphaFree 和 AlphaRec 在不同交互数量的分组中的 NDCG@20 性能。绿色线条表示 AlphaFree 相较于 AlphaRec 的相对改进。AlphaFree 始终表现出更优性能，且交互越多，改进幅度越大。

我们将每位用户的交互历史随机划分为训练集、验证集和测试集，比例为 4:3:3。结果如表 1 所示，我们观察到以下发现：

- AlphaFree 优于基于LR的方法，Recall@20 和 NDCG@20 的提升最高分别达到 5.37% 和 5.71%；与非基于LR的方法相比，AlphaFree 在这两项指标上的提升均高达 42%，表现显著更优，同时也是唯一满足所有设计原则的方法。

- 在基于LR的方法中，只有 AlphaFree 能够成功处理所有数据集，而其他方法在大型数据集上会失败。RLMRec 即使在较小的数据集上也会超时，AlphaRec 则在 Beauty 和健康等较大的数据集上因内存不足而无法运行，这凸显了无用户设计的优势。

- AlphaFree 得益于其无ID特性，因为 MF-BPR 和 FISM-BPR 等依赖ID的方法准确率较低，这表明基于逻辑回归的无ID方法（包括 AlphaFree）能够提供更丰富的语义和更好的性能。

- 凭借无GNN特性，AlphaFree 在准确率和效率方面始终优于 LightGCN 和 AlphaRec 等依赖GNN的方法。

5.3 不同交互分组下的性能

我们分析了不同交互分组的性能，以探究采用无GNN设计的影响。为此，我们将 AlphaFree 与第二佳模型 AlphaRec（表 1）进行了比较。AlphaRec 同样使用物品潜在表示，但仍依赖图神经网络。我们根据每个 $I \in \mathcal{H}_{\text{train}}$ 的 61+，将其分为四组（即 1–20 、 21–40 、 41–60 和 61+），

表 3：训练期间未见过的新用户交互集上的性能（最佳结果以粗体标出；次佳结果加下划线）。AlphaFree 能够有效地为新用户提供推荐，性能优于所测试的基线方法。

模型	Recall@20 电影	图书	视频	婴儿	NDCG@20 电影	图书	视频	Baby
FISM-BPR	0.0570	0.0494	0.0601	0.0135	0.0853	0.0601	0.0397	0.0090
LightGCN	0.0401	0.0302	0.0438	0.0125	0.0614	0.0423	0.0274	0.0076
AlphaRec	0.0681	0.0621	0.0732	0.0216	0.0980	0.0837	0.0474	0.0138
AlphaFree	**0.0706**	**0.0646**	**0.0922**	**0.0300**	**0.1027**	**0.0850**	**0.0596**	**0.0188**
% 增幅	3.63%	3.94%	25.90%	38.65%	4.89%	1.51%	25.71%	35.95%

$|I|$，并测量了每组的 NDCG@20。图4展示了结果，右侧 y 轴显示了我们的方法相较于 AlphaRec 的提升百分比。AlphaFree 在所有分组中均持续优于 AlphaRec，且随着 $|I|$ 增加，相对改进幅度也随之增大。尤其是，61+ 组（即重度用户）的改进比其他组更为显著，这表明，随着这些用户的表示被过度平均（即过度平滑），采用图神经网络的 AlphaRec 在他们身上的性能有限，而我们的无图神经网络设计则避免了这一问题。

5.4 对未见交互集的泛化

我们评估了 AlphaFree 在训练期间未见过的交互集上的泛化性能，即用户冷启动设置[4]。对于每个数据集，我们首先按照 8:1:1 的比例将用户划分为训练集、验证集和测试集。在评估时，我们从验证和测试数据中每个用户的交互集 I 中采样 10% 的交互（至少一个）以形成 I_q，并将剩余交互用作真实值。根据表 3，我们观察到以下结果：

- 与所测试的基线方法相比，AlphaFree 始终展现出更优的泛化性能，在 Recall@20 和 NDCG@20 上的提升幅度分别最高达到 38.65% 和 35.95%。
- 作为一种依赖用户的方法，AlphaRec 的泛化能力有限，相较于 AlphaFree 表现较弱，这表明它难以为新用户学习有效的嵌入。
- 对于依赖ID的方法，无用户的 FISM-BPR 的表现优于 LightGCN，而 AlphaFree 进一步利用语言表示提升了性能。

这些结果验证了无用户架构具有更强的泛化能力面向交互较少的新用户，同时利用语言表示还能带来额外收益，超越基于 ID 和依赖用户的模型。

5.5 空间效率

我们分析了 AlphaFree 通过无用户设计实现的空间效率。运行时间分析见附录 I。**基于LR的方法的内存使用**。对于基于LR的模型，语言模型（LM）会生成固定大小的语言表示 d_{LR}，且难以灵活调整，因而成为空间效率的一大瓶颈。因此，我们分析了 AlphaFree 和 AlphaRec 在训练和推理阶段每个训练轮次的平均 GPU 内存使用量，两者使用相同的语言表示。如图5所示，AlphaFree 的内存使用量低于 AlphaRec；后者无法在更大的数据集上运行

[4]在这一实验设置中，每位新用户在测试时平均提供 1.09–2.95 次交互作为查询（详情见附录 B）。

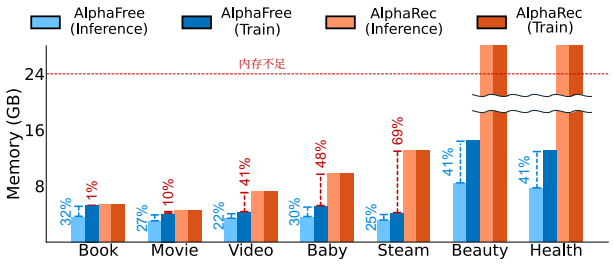
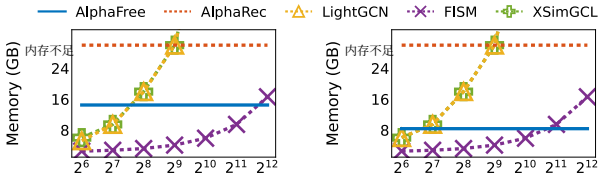


图5：AlphaFree 和 AlphaRec 在训练和推理期间的 GPU内存使用量，表明 AlphaFree 始终需要较少内存，而 AlphaRec 在 Beauty 和 Health 等大型数据集上会内存不足（内存不足）。



(a) 训练 **(b) 推理**

图6：在不同嵌入维度下，Beauty数据集上的GPU内存使用量。对于AlphaFree和AlphaRec，维度由语言模型固定（$d_{\text{LR}} = 2^{12}$），其中AlphaRec会出现内存不足。其他基线方法在维度较小时所需内存较少，但内存使用量会迅速增长，最终在 d_{LR}之后超过AlphaFree。

例如 Beauty 和健康。具体而言，在训练期间，AlphaFree 相比 AlphaRec 最多可减少 69% 的内存使用量。这得益于 AlphaFree 的无用户设计（即不存储用户嵌入），而 AlphaRec 则不同。我们还发现，AlphaFree 在推理时所需的内存最多比训练时少 41%，因为推理阶段仅使用原始视图编码器。

嵌入维度对内存使用量的影响。与上述基于LR的模型不同，传统方法的嵌入维度 d 可以轻松调整。对于AlphaFree和AlphaRec，$d_{\text{LR}} = 2^{12}$由语言模型固定；而对于其他方法，我们将 d 从2^6 调整到 2^{12}，以考察它们在Beauty数据集上的内存使用情况。如图6所示，这些方法在 d较小时使用的GPU内存较少，但随着 d 增大，内存使用量也会增加。特别是，LightGCN和XSimGCL（即依赖用户的方法）的增长速度比 FISM-BPR（即无用户方法）更快。由于采用了高维LR，AlphaFree在 d较小时比这些方法使用更多内存，但其准确率高得多（第5.2节）；而这些方法在维度 d较大时会过拟合。

5.6 消融实验

我们通过消融实验进一步分析了各组件的贡献：**不使用语义过滤**会禁用用于选择相似物品的语义过滤模块；**不使用交互对齐**会移除交互级对齐损失 $\text{L}_{\text{inter-align}}$；**不使用物品对齐**会移除物品级对齐损失 $\text{L}_{\text{item-align}}$；而**不使用全部对齐**会通过设置 $\lambda_{\text{align}} = 0$移除所有对齐损失，并且仅使用 $\text{L}_{\text{InfoNCE}}$。图 7 展示了消融实验的结果，移除任何组件都会导致性能下降，

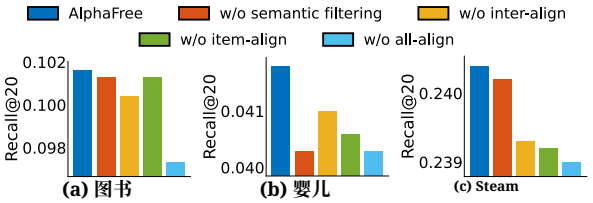


图 7：消融实验结果表明，各模块均带来正向贡献，但在不同数据集上的效果有所不同；完整模型（AlphaFree）的表现最佳。

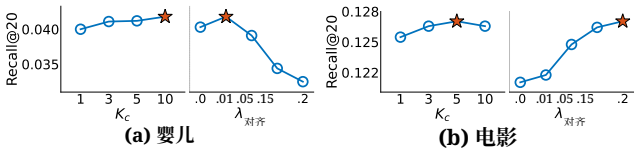


图 8：超参数 K_c 和 λ_{align}的影响，其中最佳点以星形符号标记。不同数据集呈现出不同趋势，这表明需要对这些超参数进行适当调优，以适应各种数据特征。

这验证了每个模块都能为 AlphaFree 带来正向贡献，但在不同数据集上的影响各异。值得注意的是，完整版本的 AlphaFree 始终表现最佳。

5.7 超参数的影响

我们分析了 K_c 和 λ_{align}的影响，它们分别控制相似物品的数量和跨视图对齐的强度。由于数据密度会影响增强，我们选取了Movie和Baby，它们的数据密度分别为0.27%和0.02%，以考察其在不同密度数据集上的影响。

影响 K_c。 如图8所示，在更稀疏的Baby数据集中，较大的 K_c 有利于AlphaFree；而在Movie数据集中，$K_c = 5$左右时性能最佳，这表明较大的 K_c 不适合数据密度更高的数据集。

影响 λ_{align}。 在数据密度更高的Movie数据集中，较大的 λ_{align} 通过增强两个视图之间的一致性来提升性能。在更稀疏的Baby数据集中，过强的对齐会降低性能，因为稀疏数据中的增强可能包含噪声；在这种情况下，较弱的对齐反而可能更有帮助。

5.8 定性案例研究

我们开展了一项案例研究，从定性角度验证AlphaFree的无用户和无ID设计的有效性。为此，我们在Steam数据集的$\boldsymbol{H}_{\text{train}}$上训练AlphaFree，并考察其如何适应新的交互。具体而言，我们设置了一个未见过的I_q = {EuroTruck、Banished}，其上下文由#模拟和#沙盒等多个标签构成。图9中间展示了AlphaFree针对 I_q 生成的推荐结果，其中这些标签占主导地位，表明AlphaFree能够有效捕捉 I_q的上下文。接下来，将Witcher2（#动作和#冒险）添加到 I_q 中，并将其输入AlphaFree而无需重新训练，得到左侧的推荐列表；其中动作游戏数量增加或首次出现，并且这些标签成为主导。相比之下，将使命召唤添加到 I_q时，AlphaFree得到右侧结果，

第一人称射击游戏兴起或新近出现的地方，以及 #第一人称

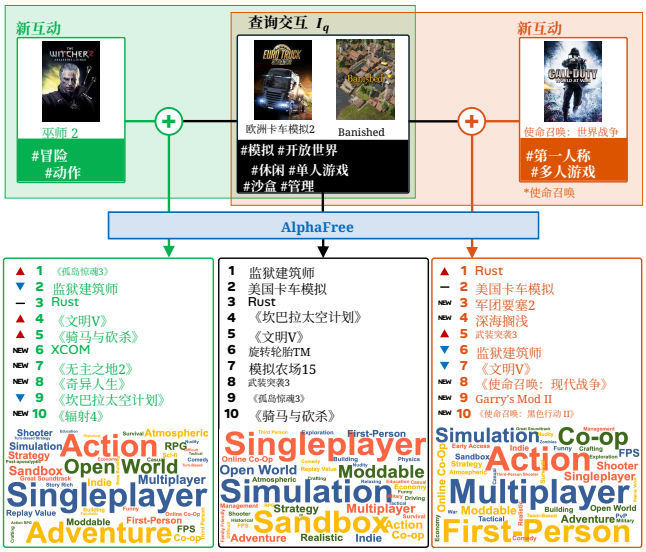


图 9：Steam数据集上的案例研究。中间展示了未见过的 I_q 的结果，左右两侧则展示了在 I_q 中添加新物品后的更新，以及推荐游戏的用户标签词云。

和 #多人游戏 标签占据主导地位。这些结果表明，AlphaFree 能有效捕捉查询上下文（无ID），并适应新的交互（无用户）。

6 结论

本文提出了 AlphaFree，这是一种新颖的 Top-K 推荐框架，不依赖用户、ID 和图神经网络。本文展示了，即使是多层感知机这类简单轻量的模型，也能通过消除对用户、ID 和图神经网络的依赖，实现出色的推荐性能。具体而言，AlphaFree 为每个物品和交互集扩充行为上及语义上相似的物品，并在训练期间对比对齐原始视图和增强视图，使模型无需依赖图神经网络即可利用高阶协同过滤信号。在推理阶段，仅使用由轻量多层感知机组成的原始视图编码器，针对给定查询即时生成推荐结果，同时不依赖用户、ID 和图神经网络。在各种基准测试和设置上进行的大量实验表明，AlphaFree 优于最先进的方法：与非基于LR的方法相比，准确率最高提升约 40%；与基于LR的方法相比，准确率提升 5.7%；同时 GPU 内存使用量最多减少 69%。我们在附录 A 中讨论了 AlphaFree 的局限性和未来方向。

致谢

我们谨向所有匿名审稿人致以谢意，感谢他们认真审阅并提出意见。本研究得到韩国政府（科学技术信息通信部，MSIT）资助的韩国信息通信技术规划与评估院（IITP）“地方智能化创新人力资源开发”项目（IITP-2026-RS-2022-00156360，占50%）以及“融合安全核心人才培养事业支持项目”（IITP-2026-RS-2024-00426853，占50%）的支持。

参考文献

[1] Keqin Bao、Jizhi Zhang、Yang Zhang、Wenjie Wang、Fuli Feng 和 Xiangnan He。2023年。TALLRec：一种将大语言模型与推荐对齐的有效且高效的微调框架。载于 RecSys。[2] Deli Chen、Yankai Lin、Wei Li、Peng Li、Jie Zhou 和 Xu Sun。2020年。从拓扑视角衡量并缓解图神经网络的过平滑问题。载于 AAAI。[3] Yuxin Chen 、 Junfei Tan 、 An Zhang 、 Zhengyi Yang 、 Leheng Sheng、Enzhi Zhang、Xiang Wang 和 Tat-Seng Chua。2024年。关于用于推荐的 Softmax 直接偏好优化。载于 NeurIPS。[4] Zefeng Chen、Wensheng Gan、Jiayang Wu、Kaixia Hu 和 Hong Lin。2023年。推荐系统中的数据稀缺：综述。CoRR abs/2312.10073（2023）。[5] Minjin Choi、Sunkyung Lee、Seongmin Park 和 Jongwuk Lee。2025年。用于基于会话的推荐的神经知识线性物品间模型。载于 SIGIR。[6] Jaewan Chun、Geon Lee、Kijung Shin 和 Jinhong Jung。2024年。超图上的带重启的随机游走：快速计算及其在异常检测中的应用。《数据挖掘与知识发现》38卷3期（2024），1222–1257页。[7] Abhinav Dubey 等。2024年。Llama 3模型家族。CoRR abs/2407.21783（2024）。[8] Yunjun Gao、Yuntao Du、Yujia Hu、Lu Chen、Xinjun Zhu、Ziquan Fang 和 Baihua Zheng。2022年。通过自引导去噪学习实现鲁棒推荐。载于 SIGIR。[9] Shijie Geng、Shuchang Liu、Zuohui Fu、Yingqiang Ge 和 Yongfeng Zhang。2022年。推荐即语言处理（RLP）：统一的预训练、个性化提示与预测范式（P5）。载于 RecSys。[10] Gyeongmin Gu、Minseo Jeon、Hyun-Je Song 和 Jinhong Jung。2025年。用于有符号二部图的高效轻量表示学习。《神经网络》192卷（2025），107708。[11] Xiangnan He、Kuan Deng、Xiang Wang、Yan Li、Yong-Dong Zhang 和 Meng Wang。2020年。LightGCN：简化并强化用于推荐的图卷积网络。载于 SIGIR。[12] Xiangnan He、Ming Gao、Min-Yen Kan 和 Dingxian Wang。2017年。BiRank：面向二部图排序的方法。《IEEE知识与数据工程汇刊》29卷1期（2017），57–71页。[13] Xiangnan He、Lizi Liao、Hanwang Zhang、Liqiang Nie、Xia Hu 和 Tat-Seng Chua。2017年。神经协同过滤。载于 WWW。[14] Yupeng Hou、Jiacheng Li、Zhankui He、An Yan、Xiusi Chen 和 Julian J. McAuley。2024年。连接语言与物品以实现检索和推荐。CoRR abs/2403.03952（2024）。[15] Tinglin Huang、Yuxiao Dong、Ming Ding、Zhen Yang、Wenzheng Feng、Xinyu Wang 和 Jie Tang。2021年。MixGCF：一种改进的基于图神经网络的推荐系统训练方法。载于 KDD。[16] Albert Q. Jiang、Alexandre Sablayrolles、Arthur Mensch、Chris Bamford、Devendra Singh Chaplot、Diego de Las Casas、Florian Bressand、Gianna Lengyel、Guillaume Lample、Lucile Saulnier、Lélio Renard Lavaud、Marie-Anne Lachaux、Pierre Stock、Teven Le Scao、Thibaut Lavril、Thomas Wang、Timothée Lacroix 和 William El Sayed。2023年。Mistral 7 B。CoRR abs/2310.06825（2023）。[17] Jeff Johnson、Matthijs Douze 和 Hervé Jégou。2021年。利用 GPU 进行十亿规模相似性搜索。《IEEE大数据汇刊》7卷3期（2021），535–547页。[18] Yu-Chin Juan、Yong Zhuang、Wei-Sheng Chin 和 Chih-Jen Lin。2016年。用于点击率预测的场感知因子分解机。载于 RecSys。[19] Jinhong Jung、Woojeong Jin、Lee Sael 和 U Kang。2016年。使用带重启的有符号随机游走在有符号网络中进行个性化排序。载于 ICDM。[20] Jinhong Jung、Namyong Park、Lee Sael 和 U Kang。2017年。BePI：一种用于十亿规模带重启的随机游走的快速、节省内存的方法。载于 SIGMOD。[21] Jinhong Jung、Jaemin Yoo 和 U Kang。2020年。有符号图扩散网络。CoRR abs/2012.14191（2020）。[22] Santosh Kabbur、Xia Ning 和 George Karypis。2013年。FISM：用于 Top-N 推荐系统的因子分解物品相似性模型。载于 SIGKDD。[23] Wang-Cheng Kang 和 Julian J. McAuley。2018年。自注意力序列推荐。载于 ICDM。[24] Vladimir Karpukhin、Barlas Oguz、Sewon Min、Patrick Lewis、Ledell Wu、Sergey Edunov、Danqi Chen 和 Wen-tau Yih。2020年。用于开放域问答的稠密段落检索。载于 EMNLP。[25] George Karypis。2001年。基于物品的 Top-N 推荐算法评估。载于 CIKM。[26] Kyungho Kim、Sunwoo Kim、Geon Lee、Jinhong Jung 和 Kijung Shin。2025年。多行为推荐系统：综述。载于 PAKDD。[27] Kyungho Kim、Sunwoo Kim、Geon Lee 和 Kijung Shin。2025年。用于多行为推荐的自监督混合专家框架。载于 CIKM。[28] Sang-Hong Kim 和 Ha-Myung Park。2023年。利用多路随机划分森林高效分布式构建近似k近邻图。载于 SIGKDD。[29] Geonwoo Ko、Minseo Jeon 和 Jinhong Jung。2025年。级联行为图上的个性化排序，实现精准多行为推荐。arXiv 预印本 arXiv:2502.11335 abs/2502.11335（2025）。[30] Geonwoo Ko 和 Jinhong Jung。2024。有符号有向图中不依赖社会假设的解耦表示学习。《信息科学》665（2024），120373。[31] Yehuda Koren、Robert M. Bell 和 Chris Volinsky。2009。推荐系统的矩阵分解技术。Computer 42，8（2009），30–37。[32] Geon Lee、Kyungho Kim 和 Kijung Shin。2024。重新审视 LightGCN：意料之外的不灵活性、不一致性，以及改进推荐的方法。载于 RecSys。[33] Jong-whi Lee 和 Jinhong Jung。2023。通过时间感知随机游走扩散改进动态图学习。载于 AAAI。[34] Jaeri Lee 和 U Kang。2025。通过用户特定表示进行上下文感知序列捆绑推荐。载于 CIKM。[35] Seunghan Lee、Geonwoo Ko、Hyun-Je Song 和 Jinhong Jung。2024。MuLe：用于多行为推荐的多粒度图学习。载于 CIKM。[36] Sangkeun Lee、Sang-il Song、Minsuk Kahng、Dongjoo Lee 和 Sang-goo Lee。2011。基于图上随机游走的多维推荐实体排序。载于 RecSys。[37] Qimai Li、Zhichao Han 和 Xiao-Ming Wu。2018。深入理解用于半监督学习的图卷积网络。载于 AAAI。[38] Greg Linden、Brent Smith 和 Jeremy York。2003。Amazon.com 推荐：物品到物品协同过滤。IEEE Internet Comput. 7，1（2003），76–80。[39] Yaokun Liu、Xiaowang Zhang、Minghui Zou 和 Zhiyong Feng。2023。通过共现嵌入增强解决多兴趣推荐中的长尾问题。载于 RecSys。[40] Yury A. Malkov 和 Dmitry A. Yashunin。2020。使用分层可导航小世界图进行高效且稳健的近似最近邻搜索。IEEE Trans. Pattern Anal. Mach. Intell. 42，4（2020），824–836。[41] Arvind Neelakantan 等。2022。通过对比预训练实现文本与代码嵌入。CoRR abs/2201.10005（2022）。[42] Jianmo Ni、Jiacheng Li 和 Julian J. McAuley。2019。使用远程标注评论和细粒度方面为推荐结果提供依据。载于 EMNLP。[43] Xia Ning 和 George Karypis。2011。SLIM：用于 Top-N 推荐系统的稀疏线性方法。载于 ICDM。[44] Haekyu Park、Jinhong Jung 和 U Kang。2017。推荐系统中矩阵分解与带重启的随机游走的比较研究。载于 BigData。[45] Ka Hyun Park、Junghun Kim、Jinhong Jung 和 U Kang。2025。PiGLeT：用于半监督链路符号预测的概率消息传递。载于 ICDM。[46] Seungcheol Park、Jeongin Bae、Beomseok Kwon、Minjun Kim、Byeongwook Kim、Se Jung Kwon、U Kang 和 Dongsoo Lee。2025。统一均匀量化与二值编码量化，实现大语言模型的精准压缩。载于 ACL。[47] Apurva Pathak、Kshitiz Gupta 和 Julian J. McAuley。2017。SIGIR。[48] Alec Radford、Jong Wook Kim、Chris Hallacy、Aditya Ramesh、Gabriel Goh、Sandhini Agarwal、Girish Sastry、Amanda Askell、Pamela Mishkin、Jack Clark、Gretchen Krueger 和 Ilya Sutskever。2021。通过自然语言监督学习可迁移视觉模型。CoRR abs/2103.00020（2021）。[49] Shaina Raza、Mizanur Rahman、Safiullah Kamawal、Armin Toroghi、Ananya Raval、Farshad Navah 和 Amirmohammad Kazemeini。2024。推荐系统综合综述：从理论走向实践。CoRR abs/2407.13699（2024）。[50] Xubin Ren、Wei Wei、Lianghao Xia、Lixin Su、Suqi Cheng、Junfeng Wang、Dawei Yin 和 Chao Huang。2024。基于大语言模型的推荐表示学习。载于 WWW。[51] Steffen Rendle。2010。因子分解机。载于 ICDM。[52] 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。载于 UAI。[53] T. Konstantin Rusch、Michael M. Bronstein 和 Siddhartha Mishra。2023。图神经网络过平滑问题综述。CoRR abs/2303.10993（2023）。[54] Badrul Munir Sarwar、George Karypis、Joseph A. Konstan 和 John Riedl。2001。基于物品的协同过滤推荐算法。载于 WWW。[55] Leheng Sheng、An Zhang、Yi Zhang、Yuxin Chen、Xiang Wang 和 Tat-Seng Chua。2025。语言表示可能正是推荐系统所需：发现与潜力。载于 ICLR。[56] Harald Steck。2019。稀疏数据的浅层自编码器：简单有效。载于 WWW。[57] Gemini 团队。2023。Gemini：一系列能力强大的多模态模型。CoRR abs/2312.11805（2023）。[58] Andreas Veit、Balazs Kovacs、Sean Bell、Julian J. McAuley、Kavita Bala 和 Serge J. Belongie。2015。利用异质二元共现学习视觉服装风格。载于 ICCV。[59] Petar Velickovic、William Fedus、William L. Hamilton、Pietro Liò、Yoshua Bengio 和 R. Devon Hjelm。2018。深度图互信息最大化。CoRR abs/1809.10341（2018）。[60] Fali Wang、Zhiwei Zhang、Xianren Zhang、Zongyu Wu、Tzuhao Mo、Qiuhao Lu、Wanjing Wang、Rui Li、Junjie Xu、Xianfeng Tang、Qi He、Yao Ma、Ming Huang 和 Suhang Wang。2025。大语言模型时代的小语言模型综合综述：技术、增强方法、应用、与大语言模型协作及可信性。ACM Trans. Intell. Syst. Technol.

16，6（2025），145:1–145:87。[61] Liang Wang、Wei Zhao 和 Jing ming Liu。2021。通过双动量对比对齐跨语言句子表示。载于 *EMNLP*。[62] Xiang Wang、Xiangnan He、Meng Wang、Fuli Feng 和 Tat-Seng Chua。2019。神经图协同过滤。载于 *SIGIR*。[63] Yuning You、Tianlong Chen、Yongduo Sui、Ting Chen、Zhangyang Wang 和 Yang Shen。2020。带增强的图对比学习。载于*NeurIPS*。[64] Junliang Yu、Xin Xia、Tong Chen、Lizhen Cui、Nguyen Quoc Viet Hung 和 Hongzhi Yin。2024。XSimGCL：迈向极简的用于推荐的图对比学习。*IEEE*知识与数据工程汇刊 36，2（2024），913–926。[65] Junliang Yu、Hongzhi Yin、Xin Xia、Tong Chen、Lizhen Cui 和 Quoc Viet Hung Nguyen。2022。图增强是否必要？用于推荐的简易图对比学习。载于 *SIGIR*。[66] An Zhang、Yuxin Chen、Leheng Sheng、Xiang Wang 和 Tat-Seng Chua。2024。推荐中的生成式智能体。载于 *SIGIR*。[67] Yanzhao Zhang、Mingxin Li、Dingkun Long、Xin Zhang、Huan Lin、Baosong Yang、Pengjun Xie、An Yang、Dayiheng Liu、Junyang Lin、Fei Huang 和 Jingren Zhou。2025。Qwen3嵌入：借助基础模型提升文本嵌入与重排序。*CoRR* abs/2506.05176（2025）。[68] Zhihui Zhou、Lilin Zhang 和 Ning Yang。2023。面向冷启动物品推荐的对比协同过滤。载于 *WWW*。[69] Yanqiao Zhu、Yichen Xu、Feng Yu、Qiang Liu、Shu Wu 和 Liang Wang。2020。深度图对比表示学习。*CoRR* abs/2006.04131（2020）。

A 局限性与未来方向

对文本元数据与语言模型的依赖。 AlphaFree对物品的文本元数据以及预训练语言模型存在固有依赖。这种依赖源于其无ID设计所不可避免的权衡：该设计以语言表示替代原始物品ID，从而应对基于ID的建模所带来的局限性。因此，在没有文本元数据的场景中，AlphaFree以及其他基于语言表示的方法都无法直接应用。此外，所得表示的质量本质上受所选语言模型表达能力的限制（见附录E）。因此，AlphaFree的性能可能因文本元数据的质量以及所采用的具体语言模型而在不同领域间有所差异。不过，本研究（包括 [55], ）仅使用物品标题，其语言表示为推荐提供了有益信号。由于此类标题在大多数领域中广泛可得，AlphaFree有望具备实际适用性。然而，如何有效利用物品标题之外的其他文本元数据（例如类别、价格、描述等），以及不同语言模型选择所带来的权衡，尚未得到系统研究，这使其成为未来研究中的一个重要开放问题。**预处理开销与可扩展性。** 我们的方法还涉及额外的预处理开销与可扩展性考量。在数据增强过程（第4.2节）中，AlphaFree计算物品之间的成对相似度，目前对 n 个物品需要$O(n^2)$的时间。此外，如表9所示，使用预训练语言模型生成物品的语言表示约占预处理时间的98–99%，因而预处理成本高于ID依赖方法。这些开销主要影响初始构建阶段，并可能成为超大规模数据集的瓶颈。不过需要注意，该成本仅产生一次，且不会影响推理效率，因为推理时仅使用原始视图编码器。为提升可扩展性，相似度计算的二次复杂度可通过采用近似 k-NN搜索方法（例如HNSW [40], 、IVF [17], 和MRDF [28]）来缓解。此外，采用轻量级语言模型 [46, 60] 可进一步缩短预处理时间，而系统分析其在表示质量方面的相关权衡仍是未来工作的重要方向。

未来方向。 本研究聚焦于基于静态用户–物品交互历史的top-K 物品推荐。一个自然的扩展是将所提出的设计原则应用于更具表达能力的设定，包括序列推荐 [23], 、基于会话的推荐 [5], 、多行为建模 [26, 27, 35], 、捆绑推荐 [34], 、时间动态 [33], 以及符号或显式反馈 [10, 21, 30, 45]。此外，尽管本研究中的行为相似度基于共现计算，未来工作仍可探索其他基于图的相似性，例如基于随机游走的方法 [6, 20] 或双向排序方法 [12, 29], ，以在所提出的框架内进一步考察其有效性或权衡。

B 详细实验设置

机器。 我们使用了一台配备 AMD 5955WX、RTX 4090（24GB 显存）和 128GB 主存的工作站。

实现。 我们使用 Python 3.9、PyTorch 1.13 和 CUDA 11.7 实现了 AlphaFree。对于基线方法，我们使用了以下来源提供的官方开源实现：

- **MF-BPR**：https://github.com/RUCAIBox/RecBole/blob/master/recbole/model/general_recommender/bpr.py
- **FISM**[5]：https://github.com/RUCAIBox/RecBole/blob/master/recbole/model/general_recommender/fism.py
- **LightGCN**：https://github.com/gusye1234/LightGCN-PyTorch
- **XSimGCL**：https://github.com/Coder-Yu/SELFRec
- **RLMRec**：https://github.com/HKUDS/RLMRec
- **AlphaRec**：https://github.com/LehengTHU/AlphaRec

数据集详情。 遵循以往研究 [14, 42, 55], ，我们使用了来自亚马逊评论 [14][6] 的多个类别的数据集，包括电影、图书、视频、婴儿、健康和 Beauty。为与 AlphaRec [55], 进行公平比较，我们使用了该研究[7]提供的预处理电影和图书版本。对于其他数据集，我们使用了 [14]中的 5-core 版本。我们还使用了 Steam 评论 [47][8]的 5-core 版本。我们将物品标题用作文本。

语言表示。 对于电影和图书，我们采用了 [55], 提供的预处理语言表示，这些表示基于 text-embedding-3-large，维度为 $d_{\text{LR}} = 3,072$，AlphaRec 和 AlphaFree 均使用了这些表示。对于其他未预处理语言表示的数据集，我们使用 LLaMA-3.1-8B [7]（$d_{\text{LR}} = 4{,}096$）获取语言表示。RLMRec 按照原始研究的建议，使用了 GPT 3.5 Turbo 和 ada-002（$d_{\text{LR}} = 1{,}536$）。

超参数调优。 我们使用Adam优化器进行训练。对于基线方法，我们采用其各自论文中报告的超参数。对于我们的方法，我们将嵌入维度 d 固定为 64，学习率 η 设为 0.0005，批量大小 $|\mathcal{B}|$ 设为 4096，权重衰减设为 10^{-6}，负样本数量 n_s 设为 256。我们最多训练 500 个训练轮次（$T_{\text{epoch}} = 500$），并依据验证指标进行早停（耐心值 = 20）。我们通过网格搜索为关键超参数选择最优取值：相似物品数量 $K_c \in \{1, 3, 5, 10\}$、对齐温度 $\tau_a \in \{0.01, 0.05, 0.15, 0.2\}$、推荐温度 $\tau_r \in \{0.15, 0.2\}$以及对齐损失权重 $\lambda_{\text{align}} \in \{0.01, 0.05, 0.15, 0.2\}$。

[5]我们基于相应代码仓库中提供的原始代码，对 FISM 进行了调整，使其使用 BPR 损失而非 BCE损失。[6]https://amazon-reviews-2023.github.io/data_processing/5core.html [7]https://github.com/LehengTHU/AlphaRec [8] https://cseweb.ucsd.edu/~jmcauley/datasets.html#steam_data

表4：常用符号

符号	描述
$\mathcal{I}$	物品集合，其中 $n = \lvert\mathcal{I}\rvert$
I & I^+	原始交互集和增强后的交互集，分别。
$\mathbf{H}_{\text{train}}$	所有交互集的集合，其中 $I \in \mathcal{H}_{\text{train}}$
$\mathcal{B}$	小批次，即 $\mathcal{B} \subseteq \mathcal{H}_{\text{train}}$
K_c	相似物品的数量
p	从 I中随机采样的正样本物品
N_I	从$\mathcal{I} \setminus I$中抽取的负样本集合
$\mathcal{S}_i$	与物品 i相似的物品集合
$\mathbf{z}_i$ & $\mathbf{z}_i^+ \in \mathbb{R}^{d_{\text{LR}}}$	物品 i的原始与增强潜在表示
$\mathbf{e}_*$ & $\mathbf{e}_*^+ \in \mathbb{R}^{\times d}$	$* \in \{i,\ I\}$的原始与增强嵌入
$\text{sim}_B(\cdot,\ \cdot)$	两个物品之间的行为相似度
$\text{sim}_S(\cdot,\ \cdot)$	两个物品之间的语义相似性
d_{LR} & d	LR 和嵌入的维度
λ_{align}	对齐损失的比例

用户冷启动设置的统计数据。如第5.4节所述，此划分会产生用户冷启动设置，其中验证数据和测试数据中每个交互集的物品平均数量介于 1.03 至 2.95 之间，如下表所示。

表5：用户冷启动设置下每个交互集物品数量的统计数据。

数据集	平均值	最大值	最小值
电影	2.95	142	1
Book	2.70	74	1
视频	1.02	18	1
Baby	1.03	19	1

C 算法详解

我们在算法 1–3 中给出了 AlphaFree 各阶段的详细算法。预处理阶段（算法 1）采用逐物品的方式描述，而非直接遵循公式。相似物品搜索可以通过对物品执行并行循环（第 4 行）来并行化，前提是将所得物品级输出通过交互级并集操作进行汇总（第 9–10 行）。请注意，计算 $\texttt{sim}_B(i,\ j)$ 等价于统计同时消费过两个物品 i 和 j的用户数量。因此，可以很容易地以向量化方式实现为$\texttt{sim}_B(i,\ j) = \mathbf{B}ij$，其中 $\mathbf{B} = \mathbf{A}^\top\mathbf{A}$，而 $\mathbf{A}$ 表示由 $\mathcal{H}_{\text{train}}$构建的用户–物品交互矩阵。在我们的实现中，我们直接以这种矩阵形式计算 $\texttt{sim}_B$，并将 $\mathbf{A}$ 表示为稀疏矩阵（例如压缩稀疏行）。在训练阶段，尽管我们在第 4.3.1 节中仅描述了对 I、 I^+和 N_I 中的物品计算嵌入，但为了利用并行计算效率，我们的实现会在每个批次开始前预先计算所有物品的嵌入（算法 2 第 4 行）。

D 语义过滤详解

我们在第 4.2.2 节分析了语义阈值 κ 的影响，其中公式 (5) 可重写为 $\mathsf{S}_i = \{j \in \mathsf{C}_i \mid \text{相似度}_S(i,\ j) \geq \kappa\}$。我们将阈值 κ 设为语义相似度分布的第 25、50 和 75 百分位数，并测量 Al- phaFree 的保留率（即 $(\Sigma_{i\in\mathcal{I}} \lvert\mathcal{S}_i\rvert)/(\Sigma_{i\in\mathcal{I}} \lvert C_i\rvert)$）和 Recall@20。如表 6 所示，当 κ 对应于 μ_i时，性能最佳，并根据行为相似度实现了均衡的物品选择。请注意，在两个数据集中，我们都使用了不同的语言模型（即，电影领域中的 Large 3 和婴儿领域中的 LLAMA-3.1-8B），这表明我们的过滤策略可能适用于不同的语言模型。

算法 1 AlphaFree 的预处理阶段

输入： -

$\mathcal{H}_{\text{train}}$：交互集的训练集合 - $\{t_i \mid i \in \mathcal{I}\}$：物品内容集合 - K_c：相似物品数量

输出： -

$\{(I \mapsto I^+) \mid I \in \mathcal{H}_{\text{train}}\}$：增强后的交互集集合 - $\{(\mathbf{z}_i \mapsto \mathbf{z}_i^+) \mid i \in \mathcal{I}\}$：增强后的物品潜在表示集合

1：为每个 $i \in \mathcal{I}$ 计算 $\mathbf{z}_i \leftarrow$ 语言模型(t_i)
2：为每个 $i \in \mathcal{I}$ 计算 $H(i) \leftarrow \{I \mid i \in I, I \in \mathcal{H}_{\text{train}}\}$
3：为每个 $I \in \mathcal{H}$ 初始化 $I^+ \leftarrow I$
4: **对每个** $i \in$，**我执行** 训练
5: **对每个** $j \in$，**我执行**
6: 计算 相似度$_B(i,\ j) \leftarrow \lvert H(i) \cap H(j)\rvert$ ▷ 若满足则跳过 $\lvert H(i)\rvert \cdot \lvert H(j)\rvert = 0$
7: 计算相似度$_S(i,\ j) \leftarrow \mathbf{z}_i^\top \mathbf{z}j$
8: 根据式 (4) 和式 (5)，用 K_c 计算 μ_i 和 S_i
9: **对每个** $I \in H$（i）**执行**
10: $I^+ \leftarrow I^+ \cup \mathsf{S}_i$
11: 根据公式（7）计算$\mathbf{z}_i^+$
12: 为 $\mathbf{z}_i \mapsto \mathbf{z}_i^+$**对于每个** $i \in \mathbf{I}$，以及 $I \mapsto I^+$ **对于每个** $I \in \mathbf{H}_{\text{train}}$
13: **返回** $\{(I \mapsto I^+) \mid I \in \mathcal{H}_{\text{train}}\}$ 和 $\{(\mathbf{z}_i \mapsto \mathbf{z}_i^+) \mid i \in \mathcal{I}\}$

算法 2 AlphaFree 的训练阶段

输入： -

$\{(I \mapsto I^+) \mid I \in \mathcal{H}_{\text{train}}\}$：交互集及其增强的集合 - $\{(\mathbf{z}_i \mapsto \mathbf{z}_i^+) \mid i \in \mathcal{I}\}$：物品潜在表示及其增强的集合 - τ_r和 τ_a：温度超参数 - λ_{align}：对齐损失的比率 - T_{epoch}和 η：训练轮次和学习率的数量，分别为。

输出：

- θ_{MLP} **：原始视图编码器的参数集合 1：在** $\Theta = \{\theta_{\text{MLP}}, \theta_{\text{MLP}^+}\}$ **中初始化模型参数 2：对于** T_{epoch} **中的每个训练轮次，执行** ▷ $I^+ \leftarrow I$ **和** $\mathbf{z}_i^+ \leftarrow \mathbf{z}$ 3 $\subseteq$ **：对于每个小批次 B** $\mathbf{H}_{\text{train}}$**，执行 4：对每个** $i \in \mathbf{I}$**，计算** $\mathbf{e}_i \leftarrow$ **MLP(**$\mathbf{z}_i$**) 和** $\mathbf{e}_i^+ \leftarrow$ **MLP⁺(**$\mathbf{z}_i^+$**) 5：对于每个** $I \in \mathbf{B}$**，执行 6：计算** $\mathbf{e}_I \leftarrow$ **MLP(mean({**$\mathbf{z}_i \mid i \in I$**})) 7：计算** $\mathbf{e}_I^+ \leftarrow$ **MLP⁺ (mean({**$\mathbf{z}_i^+ \mid i \in I^+$**})) 8：从** I **中随机采样** p**，并从** $\mathbf{I} \setminus I$ **中采样** N_I **9：计算用于** $\mathbf{L}_{\text{rec}}$ ▷ **（公式 (8)-(9)）** 10 align ▷ 10) (11) **的局部** 11 **损失：计算用于 L（公式 (- ）的局部损失：计算** $\mathbf{L}_{\text{final}}$**(B)** $\mathbf{L}_{\text{rec}}$**(B)** $^{12} + \lambda_{\text{align}} \cdot$ $\mathbf{L}_{\text{align}}$**(B)（公式 ( )）12：更新** $\theta \leftarrow \theta - \eta\nabla_\theta \mathbf{L}_{\text{final}}$**(B)（针对每个** $\theta \in \Theta$**）13：返回** θ

MLP

算法3 AlphaFree 的推理阶段

输入： -

I_q：查询交互集合 - $\{\mathbf{z}_i \mid i \in \mathcal{I}\}$：物品潜在表示集合 - θ_{MLP}：原始视图编码器的参数集合 - K：用于 top-K 推荐的参数

输出： -

R^{I_q}：针对 I_q 推荐的前 K 名列表：1：计算 $\mathbf{e}^{I_q}$ ← 多层感知机 (mean({$\mathbf{z}_i \mid i \in I_q$})) 2：计算 $\mathbf{e}_i \leftarrow$ 多层感知机($\mathbf{z}_i$) **对每个** $i \in \mathcal{I}$3：$\mathrm{R}^{I_q} \leftarrow \text{argtop}_K(\{\sigma(\mathbf{e}^{I_q},\ \mathbf{e}_i) \mid i \in \mathcal{I}\})$ ▷ y_θ多层感知机 $(i \mid I_q) := \sigma(\mathbf{e}^{I_q},\ \mathbf{e}_i)$ 4：**返回**R^{I_q}

表 6：不同语义阈值的影响 κ。

数据集	阈值 κ	保留率	Recall@20
电影	25%	67.1%	0.1265
	平均值（μ_i）	**80.2%**	**0.1267**
	75%	91.6%	0.1264
Baby	25%	43.9%	0.0414
	均值（μ_i）	**63.9%**	**0.0418**
	75%	84.5%	0.0413

E 语言模型的影响

我们考察了 AlphaFree 在不同语言模型上的推荐性能。根据语言模型参数的数量，轻量级语言模型（例如 Qwen-0.6B 和 Gemma-1B）生成的嵌入较小 d_{LR}，而中等规模的语言模型（例如 LLaMA-8B 和 Mist-7B）使用较大的 d_{LR}，如表 8 所示。我们比较了 AlphaFree 和 AlphaRec 在不同语言模型上的表现，并进一步将它们与基于图神经网络的方法进行比较，以考察显式结构建模的作用。从表 8 中，我们观察到以下几点：

- 对于 LLaMA-8B 和 Mist-7B 这类表达能力足够强的语言模型，丰富的语言表示使 AlphaFree 能够在不显式使用图神经网络的情况下有效捕捉高阶协同过滤信号，从而取得出色的性能。
- 对于 Qwen-0.6B 和 Gemma-1B 这类表达能力相当有限的小型语言模型，通过图神经网络（如 AlphaRec 或 XSimGCL）引入显式结构偏置仍然有益。
- 然而，这类基于图神经网络的设计会带来显著的内存开销；如图5所示，随着 d语言表示规模的增长，这一问题会愈发突出。
- 通过去除图神经网络，AlphaFree 在充分发挥高表达能力语言表示优势的同时，实现了更高的内存效率，并取得了最先进的推荐性能。

F 已验证超参数

我们在表7中报告了 AlphaFree 在每个数据集上已验证的超参数。

表7：AlphaFree 的已验证超参数。

超参数。电影	图书	视频	婴儿	Steam	Beauty	健康
K_c	5	5	10	10	3	10	5
λ_{align}	0.2	0.2	0.05	0.01	0.01	0.01	0.01
τ_a	0.2	0.1	0.01	0.2	0.2	0.1	0.2
τ_r	0.15	0.15	0.2	0.2	0.2	0.2	0.15
d_{LR}	3,072	3,072	4,096	4,096	4,096	4,096	4,096

G 计算复杂度分析

时间复杂度分析。我们从理论上分析 AlphaFree 各阶段的时间复杂度，其中算法 1、2 和 3 分别展示了预处理、训练和推理阶段。假设 $n = |\mathcal{I}|$表示物品数量，$m = \sum_{I \in \mathcal{H}_{\text{train}}} |I|$表示交互总数，$h = |\mathcal{H}_{\text{train}}|$表示交互集数量（$h \le m$），$K_c$ 表示相似物品数量（$K_c \le n$），$|\mathcal{B}|$ 表示小批次大小，d_{LR} 表示语言表示的维度，d表示 MLP 层的嵌入维度。

引理 3（预处理时间复杂度）。算法 *1* 的时间复杂度为 $O(T_{\text{LM}}n + mn + (d_{\text{LR}} + \log K_c)n^2)$，其中 T_{LM} 表示语言模型生成嵌入的时间。

表 8：AlphaFree 在不同语言模型上的推荐性能，分别以 R@20（Recall@20）和 N@20（NDCG@20）衡量。对于每种语言模型，加粗值和下划线值分别表示最佳分数和次佳分数。在所有测试的语言模型中，最佳 R@20 和 N@20 分别以红色和蓝色标示。请注意，LightGCN 和 XSimGCL 不依赖语言模型；因此，它们在不同语言模型下的性能相同，并在此报告以供比较。

	方法	Qwen-0.6B [67] (d_{LR}=1,024) R@20		Gemma-1B [57] (d_{LR}=1,152) @20	N@20	LLaMA-8B [7] (d_{LR}=4,096) R@2		Mist-7B [16] (d_{LR}= 4,096)	N@20
Video	**AlphaFree**	____	0.057	0.094	0.052	**0.112**	0.062	**0.118**	**0.065**
	AlphaRec	**0.059**	**0.099**	**0.056**	0.109	____	.115	0.064	
	LightGCN	0.073	0.041	0.073	0.041	0.073	0.041	0.073	0.041
	XSimGCL	0.090	0.050	0.090	0.050	0.090	0.050	0.090	0.050
Baby	**AlphaFree**		0.018	0.034	0.018	**0.041**	**0.022**	**0.045**	**0.024**
	AlphaRec	**0.040**	**0.021**	0.034	0.018	0.039	0.021	0.043	0.023
	LightGCN	0.036	0.019	0.036	0.019	0.036	0.019	0.036	0.019
	XSimGCL	0.039	0.020	**0.039**	**0.020**	0.039	0.020	0.039	0.020
Steam	**AlphaFree**		**0.187**	**0.234**	**0.190**	**0.240**	**0.194**	**0.241**	**0.193**
	AlphaRec	0.226	0.182	0.230	0.186	0.236	0.188	0.238	0.189
	LightGCN	0.201	0.152	0.201	0.152	0.201	0.152	0.201	0.152
	XSimGCL	0.225	0.166	0.225	0.166	0.225	0.166	0.225	0.166

证明。该算法需要 $O(T_{\text{LM}}n)$ 时间，因为它会计算所有物品的语言表示（第 1 行）。构建 $H(i)$ 的倒排表需要 $O(m)$ 时间，因为通过遍历 H_{train} 中的所有交互来构建每个 i 的 $H(i)$（第 2 行）。假设每个 $H(i)$ 都已排序，则 $\text{sim}_B(\cdot)$ 的每次交集运算需要 $O(|H(i)| + |H(j)|)$ 时间；随后还需要 $\sum_{i \le j}(|H(i)| + |H(j)|) = (n+1)\sum_i |H(i)| = O(nm)$ 时间（第 6 行）。计算所有 $\text{sim}_S(\cdot)$ 需要 $O(d_{\text{LR}}n^2)$ 时间。然后，若使用最小堆排序，则需要 $O(n^2 \log K_c + nK_c)$ 时间，为所有物品找到相似物品 S_i（第 8 行）。获取 I^+ 和 z_i^+ 分别需要 $O(mK_c)$ 和 $O(nK_c d_{\text{LR}})$（第 9 至 11 行）。将以上各项相加即可证明结论。■

引理4（训练时间复杂度）。在算法 *2* 中，每个训练轮次耗时 $O((T_i dn + K_c m + dh)d_{\text{逻辑回归}} + (n_s + |\mathcal{B}|)hd)$，其中$T_i = h/|\mathcal{B}|$表示遍历所有批次的迭代次数，$n_s$表示负样本的数量。

证明。我们首先分析处理每个小批次 $\mathcal{B}$ 所需的时间。对所有物品执行多层感知机操作（第 4 行）需要 $O(d_{\text{LR}}dn)$ 时间。对于每个 I，需要 $O(d_{\text{LR}}(|I|+d))$ 时间；对于每个 I^+（第 6 行和第 7 行），需要 $O(d_{\text{LR}}(|I^+|+d))$ 时间，其中 $|I^+| = O(K_c|I|)$，因此总开销为 $O(d_{\text{LR}}(K_c|I| + d))$。使用拒绝采样时，采样 p 和 N_I 的期望时间为 $O(n_s)$（第 8 行）。接着，L_{rec} 需要 $O(|\mathcal{B}|n_s d)$ 时间，而 L_{align} 需要 $O(|\mathcal{B}|^2 d)$ 时间（第 9 行和第 10 行）。考虑所有批次后，可表示如下：

$$\begin{aligned}&\sum_{\mathcal{B} \subseteq \mathcal{H}_{\text{train}}} \left(d_{\text{LR}}dn + \sum_{I \in \mathcal{B}} d_{\text{LR}}(K_c|I| + d) + n_s + |\mathcal{B}|n_s d + |\mathcal{B}|^2 d \right) \\ &\quad = T_i d_{\text{LR}} dn + d_{\text{LR}} K_c m + d_{\text{LR}} dh + T_i n_s + hn_s d + |\mathcal{B}|hd \\ &\quad = O((T_i dn + K_c m + dh)d_{\text{LR}} + (n_s + |\mathcal{B}|)hd,\end{aligned}$$

其中 $n_s \le n$。■

引理5（推理的时间复杂度）。算法*3*的时间复杂度为 $O(d_{\text{LR}}dn + n \log K)$，其中 K 是用于获取前 K 个物品的推荐的参数。

证明。该算法需要 $O(d_{\text{LR}}(|I_q|+d))$ 的时间来执行平均和多层感知机操作（第1行）。对于所有物品，计算其嵌入需要 $O(d_{\text{LR}}dn)$ 的时间（第2行）。接着，计算所有物品的分数并使用最小堆选出前 K 个物品，需要 $O(dn+n\log K)$ 的时间（第3行）。综合以上各项即可证明结论。■

空间复杂度分析。我们分析各阶段的空间复杂度。

引理6（预处理空间复杂度）。*AlphaFree* 的预处理阶段消耗 $O(d_{\text{LR}}n+K_c(m+n))$ 空间。

证明。所有物品的语言表示 $\{\mathbf{z}_i\}$ 和 $\{\mathbf{z}_i^+\}$ 需要 $O(d_{\text{LR}}n)$ 空间（第1行和第11行）。所有物品的 $\{H(i)\}$ 使用 $O(m)$ 空间（第2行）。集合 I^+ 使用 $O(K_c|I|)$ 空间，因此对于所有 $I\in\mathcal{H}_{\text{train}}$，需要 $O(K_cm)$ 空间（第10行）。查找并存储 S_i 时，使用堆需要 $O(K_c)$ 空间；所有物品总共需要 $O(K_cn)$ 空间（第8行）。注意，对于每个 i，算法会以原地方式计算每个 j 的 i 相似度分数，只需 $O(n)$ 空间，无需存储先前 i 的结果（第5至7行）。综合以上各项即可证明该结论。■

引理7（训练空间复杂度）。*AlphaFree* 的训练阶段使用 $O(m+d(n+d_{\text{逻辑回归}}+n_s+|\mathcal{B}|))$ 空间。

证明。输入 $\mathcal{H}_{\text{train}}$ 使用 $O(m)$ 空间。每个多层感知机层占用 $O(d_{\text{LR}}d)$ 空间（第 1 行）。隐层嵌入 $\{\mathbf{e}_i\}$ 和 $\{\mathbf{e}_i^+\}$ 使用 $O(dn)$ 空间（第 4 行）。对于第 9 行，计算 L_{rec} 的局部损失时，E_I 和 E_I^+ 采用原地方式，因此占用 $O(dn_s)$ 空间。对于第 10 行，计算 $\mathrm{L}_{\text{align}}$ 的局部损失时，$\mathcal{E}_{\mathcal{B}}$、$\mathcal{E}_{\mathcal{B}}^+$、$\mathcal{E}^{P_{\mathcal{B}}}$ 和 $\mathcal{E}_{P_{\mathcal{B}}}^+$ 采用原地方式，因此占用 $O(|\mathcal{B}|d)$ 空间。将上述所有项合并即可证明该结论。■

引理8（推理空间复杂度）。*AlphaFree* 的推理阶段 *AlphaFree* 使用 $O((d_{\text{LR}}+d)n+K)$ 空间。

证明。计算所有 $\{\mathbf{e}_i\}$ 以及 $\mathbf{e}^{I_q}$需要 $O((d_{\text{LR}}+d)n)$ 空间（第 2 行），而在使用堆的情况下，选出排名前 K 个物品需要 $O(K)$ 空间（第 3 行）。■

H 过平滑案例研究

图10展示了一个案例研究，用于比较 AlphaRec 和 AlphaFree，以考察婴儿数据集上的过平滑问题。对于交互数量众多的重度用户（用户A）（$|I_A|=191$），AlphaRec 赋予的分数普遍较高且方差极低，这表明其物品嵌入因图神经网络而变得过度相似（即发生了过平滑）。然而，AlphaFree 无需图神经网络即可构建用户表示，从而保留嵌入
为用户A提升多样性并避免过平滑。对于交互较少的轻度用户（用户B）（$|I_B|=17$），AlphaRec 得分更高且分布更广，表明即使在稀疏交互的情况下，它生成的嵌入也更相似，但不够稳定；而 AlphaFree 的相似度较低，分布更紧凑、稳定，这表明它能够抵御此类影响。

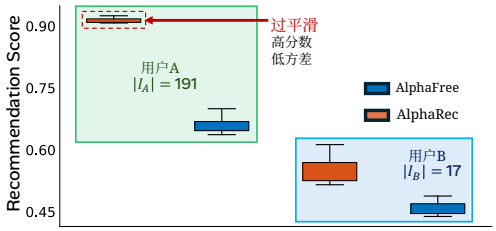


图10：AlphaRec与AlphaFree在Baby数据集上针对重度用户和轻度用户的过平滑案例研究。

I 运行时间实验

本节评估了 AlphaFree 与基于LR的方法（即 RLMRec 和 AlphaRec）以及 LightGCN 在预处理、训练和推理各阶段的运行时间。在预处理阶段，基于LR的方法主要使用语言模型计算物品潜在表示，其他辅助操作则因方法而异；具体而言，AlphaFree 会计算相似物品。LightGCN 在预处理期间初始化嵌入，并对用户–物品交互矩阵进行归一化。预处理阶段只执行一次，我们测量了其实际耗时 T_p，单位为秒。对于训练阶段，我们测量每个训练轮次的平均运行时间 T_t，单位为秒。在推理阶段，我们测量为单个用户生成推荐结果所需的平均时间 T_i，单位为微秒。

表9展示了不同数据集上的结果。AlphaRec与AlphaFree的预处理时间相近，因为LR生成部分主导了这一阶段。尽管二者在训练时间上的性能也相当，但AlphaRec在Beauty和健康等大型数据集上会出现显存不足错误（需要超过24 GB显存）。RLMRec在所有评估数据集的预处理过程中都会超时（超过50小时）。虽然LightGCN在预处理和训练中最快，但由于GNN传播，其推理延迟更高，并且如第5.2节所述，其推荐性能低于基于LR的方法。从推理时间的角度来看，Al- phaFree实现了与LightGCN相当的延迟。相比之下，AlphaRec的推理延迟相对更高。值得注意的是，AlphaRec和AlphaFree都在语言表示的维度 d_{LR} 超过3,000的条件下运行，而LightGCN则使用64维嵌入进行计算。

表9：AlphaFree与基于LR的方法及LightGCN的运行时间比较。T_p（秒）、T_t（秒）和 T_i（μ秒）分别表示预处理、训练（每个训练轮次）和推理（每个查询用户）的时间。基于LR的方法在 T_p 下方括号中的数值表示预处理期间生成语言表示的比例。

方法 RLM			AlphaRec			AlphaFree			LightGCN		
数据集	T_t	T_i	T_p	T_t	T_i	T_p	T_t	T_i	T_p	T_t	T_i
视频 超	n/a		11(98%)	5	19	563(98%)	9	14	10	1	13
Baby 超时	n/a		797(98%)	12	23	802(98%)	13	18	39	1	18
Steam 超时	n/a		342(98%)	91	28	315(98%)	45	3	84	8	9
Beauty	n/a		4614(99%)	不适用 (内存不足)					92	36	97
健康 超	n/a		4211(99%)	不适用 (内存不足)					563	39	90