arXiv:2303.01130v1 [cs.IR] 2 Mar 2023

异构模型的蒸馏用于 Top-K 推荐

SeongKu Kang[1‡]、Wonbin Kweon[1], Dongha Lee[2], Jianxun Lian[3], Xing Xie[3], Hwanjo Yu[1*]浦项科技大学（浦项科技大学），韩国[2]延世大学，韩国 [3]微软亚洲研究院，北京，中国 {seongku, kwb4453, hwanjoyu}@postech.ac.kr,donalee@yonsei.ac.kr, jianxun.lian@outlook.com, xingx@microsoft.com

摘要

近年来，推荐系统通过采用异构模型集成取得了出色的性能。然而，这种方法成本极高，因为其所需资源和推理延迟与模型数量成正比，这仍是其在生产环境中应用的瓶颈。我们的工作旨在利用知识蒸馏（KD），将异构教师模型的集成知识迁移到轻量级学生模型中，在保持高准确率的同时降低高昂的推理成本。通过实证研究，我们发现，从异构教师模型迁移知识时，蒸馏效果会大幅下降。尽管如此，我们发现，教师的训练轨迹中蕴含着能够缓解这一难题的重要信号。本文提出了一种新的知识蒸馏框架 HetComp，通过迁移教师轨迹生成的由易到难的知识序列来指导学生模型。为了根据学生模型的学习状态提供指导，HetComp 使用动态知识构建来提供难度逐步提升的排序知识，并采用自适应知识迁移来逐步迁移更细粒度的排序信息。全面的实验表明，HetComp 显著提升了蒸馏质量以及学生模型的泛化能力。

美国计算机协会分类体系概念

• **信息系统 → 检索模型与排序；推荐系统；检索效率。**

关键词

知识蒸馏，模型压缩，由易到难学习

美国计算机协会参考文献格式：

SeongKuKang、Wonbin Kweon、DonghaLee、JianxunLian、XingXie、Hwanjo Yu。2023。面向 Top-K 推荐的异构模型蒸馏。载于*2023*年 *ACM Web* 会议论文集（*WWW '23*），*2023*年*5*月*1*日至*5*日，美国德克萨斯州奥斯汀。美国计算机协会，纽约州纽约市，美国，11页。https: //doi.org/10.1145/3543507.3583209

*通讯作者‡工作在微软实习期间完成。

允许为个人或课堂使用而免费制作本作品的全部或部分数字版或纸质版副本，前提是不得为盈利或商业利益而制作或分发副本，且副本首页须载有本声明及完整引文。必须尊重本作品中由作者以外人士拥有版权的组成部分。允许在注明出处的情况下进行摘编。其他复制、再版、发布到服务器或重新分发至邮件列表的行为，须事先获得明确许可和/或支付费用。请向 permissions@acm.org 申请许可。*WWW '23，2023*年*5*月*1*日至*5*日，美国德克萨斯州奥斯汀 © 2023 版权由所有者/作者持有。出版权已授权给美国计算机协会。美国计算机协会 ISBN 978-1-4503-9416-1/23/04...$15.00 https://doi.org/10.1145/3543507.3583209

1 引言

推荐系统（RS）已应用于各种场景，以辅助决策 [9]。推荐系统的核心是为每位用户提供个性化的物品排序列表。过去几十年来，研究者已探索了多种具有不同架构和损失函数的模型，从矩阵分解 [36] 到图神经网络 [9], , 以生成高质量的排序列表。众所周知，这些异构模型具有不同的归纳偏置，使模型偏好某些假设而非其他假设 [1, 51], , 因此，它们能更好地捕捉某些符合各自模型偏好的用户/物品偏好 [19, 54, 56]。因此，通过模型集成利用这些多方面的知识，通常能比单一模型显著提高准确率 [19, 32, 45, 51, 54]。然而，其根本局限在于推理所需的计算成本可能是单一模型的数倍，因此不适用于实时服务。

一种日益常见的降低推理延迟的方法，是通过知识蒸馏（KD） [11]将大型模型压缩为较小的模型。知识蒸馏通过从训练良好的庞大模型（教师）向紧凑模型（学生模型）迁移知识进行训练，从而有效缩小二者之间的性能差距。受其在计算机视觉领域巨大成功的启发，知识蒸馏已被积极研究用于压缩排序模型。近期的排序匹配方法 [5, 15, 25, 34, 41] 采用列表式学习，训练学生模型模拟教师模型给出的项目排列（即排序列表）。该方法在诸如 Top-K 推荐 [15, 25, 41], 文档检索 [34, 52], 和人员识别 [5]等众多面向排序的应用中表现出显著性能。然而，这些研究大多侧重于从与学生模型具有相同模型类型的同构教师模型进行蒸馏。对于具有不同架构和损失函数的异构教师模型开展的跨模型蒸馏，尚未得到充分研究。

根据我们的分析，我们发现，将知识从异构教师模型迁移时，先前知识蒸馏方法的效果会显著下降，最终导致学生模型无法保留集成准确率。具体而言，我们考察了学生模型与教师模型的排序预测之间的差异，发现与从同构教师模型学习的学生模型相比，从异构教师模型学习的学生模型存在明显更大的差异。幸运的是，我们发现教师模型的中间训练状态中蕴含着有助于缩小差异的重要线索；无论教师模型的类型如何，从教师模型较早训练状态学习的学生模型，其差异都明显小于从最终收敛的教师模型学习的学生模型。这表明，在教师模型的训练过程中，教师模型的知识对学生模型而言会逐渐变得更难以

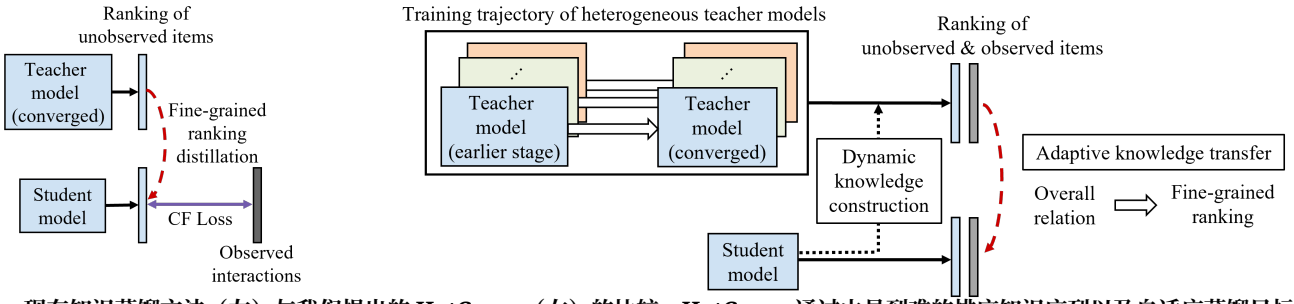

图 1：现有知识蒸馏方法（左）与我们提出的 HetComp（右）的比较。HetComp 通过由易到难的排序知识序列以及自适应蒸馏目标来监督学生模型。

学习。事实上，我们通过实验证明，教师模型的预测会变得越来越复杂，因为后期的预测包含更多样化且个性化的物品排名。

受这一观察的启发，我们提出的提升蒸馏效果的核心思路是，利用教师模型的训练轨迹，按由易到难的顺序动态传递知识（图 1）。随着教师模型在训练过程中逐渐掌握更难的知识，我们旨在监督学生模型逐步跟随这一序列，从而降低向已收敛教师模型学习的难度。我们的方法基于由易到难的学习范式 [2, 13, 43]，这一范式已在机器学习的多个领域得到广泛研究。它先使用简单样本训练模型，再逐步引入更困难的样本，使模型能够利用先前学到的概念，更容易地掌握难度更高的内容 [2, 13]。遵循这一思路，我们从教师模型训练早期阶段相对简单的知识开始蒸馏，然后沿着教师轨迹逐步增加难度，从而减少向异构教师模型学习时产生的巨大差异。

在这项工作中，我们提出了一种新的知识蒸馏框架 Het-Comp，它能有效压缩异构模型中有价值但难以处理的集成知识，从而生成一个轻量级模型，同时保持较高的推荐性能。HetComp 从以下几个方面具体实现了由易到难蒸馏：

- **迁移什么**：HetComp 通过动态知识构建 来监督学生模型，根据学生的学习状态提供由易到难的排列序列。它首先从每位教师的学习轨迹中识别合适的知识，然后动态构建目标知识，以指导学生模型。

- **如何迁移**：HetComp 采用自适应知识迁移，根据学生的学习状态调整蒸馏目标。它训练学生模型先关注目标排列中的整体关系，再逐步学习偏好物品的细粒度顺序。此外，我们还提出了一种新的迁移策略，以利用已观测和未观测的用户-物品交互知识，而这一点在以往研究中尚未得到考虑。我们的贡献总结如下：

- 我们揭示了异构模型之间排序知识迁移的难点，并从由易到难蒸馏的角度解决这一问题，这在推荐领域尚属首次。

- 我们提出了 HetComp，这是一种全新的知识蒸馏框架，能够有效地将异构模型集成压缩为一个紧凑的模型。HetComp 能显著减轻模型集成带来的巨大计算负担，同时保持其较高的准确率。

- 我们通过在真实世界数据集上进行大量实验，验证了 HetComp 的优越性。此外，我们还对所提出的方法进行了全面分析。

2 相关工作

知识蒸馏。 知识蒸馏（KD）已在多个领域被广泛用于模型压缩 [5, 11, 17, 37, 48, 55]。

知识蒸馏将大容量教师模型所捕获的知识迁移至轻量级学生模型，在保持性能相当的同时显著降低推理成本。鉴于单个教师模型提供的监督信号不足以实现准确监督，近期许多研究 [8, 28, 42, 45, 50, 51, 54] 采用多个教师模型，并证明这对于进一步提升学生模型的效果十分显著。值得注意的是，最先进的方法 [8, 45, 51] 利用在配置、架构、损失函数及诸多其他因素上各不相同的异构教师模型，以融合它们互补的知识；与单个教师模型或同构教师模型提供的单一视角相比，这能提供更全面的指导。**面向排序的知识蒸馏**。 知识蒸馏也已应用于排序问题。许多研究 [4, 20, 22, 40, 54] 迁移每个用户-物品对（或查询-文档对）的逐点重要性。然而，逐点方法无法同时考虑多个物品之间的关系，因此排序性能有限 [5, 15, 41]。近期方法 [5, 15, 25, 34, 41, 52] 将蒸馏过程建模为一项排序匹配任务。它们利用教师模型给出的排序顺序作为监督信号，训练学生模型保留教师模型的排列。通过直接迁移排序知识，这种方法在多种以排序为核心的应用中取得了最先进的性能，例如 Top-K 推荐 [15, 16, 25, 41]，文档检索 [34, 52]，和人员识别 [5]。此外，排序匹配方法可灵活应用于输出分数分布各异的异构模型之间的知识迁移，而逐点方法无法直接应用于这种情形 [19]。

由易到难学习。 受人类学习过程的启发，由易到难学习已在机器学习的各个领域得到广泛研究 [2, 6, 13, 21, 27, 29, 43, 46]。当直接优化非凸目标函数可能收敛到较差的局部极小值时，这种方法已得到广泛应用，并已证明在实现更好的泛化方面发挥着重要作用 [2]。课程学习 [2, 43] 通过逐步纳入

按先验知识界定的难度从低到高排列的数据样本来训练模型。另一方面，自步学习 [21] 会在训练过程中动态调整课程，通常依据训练损失 [21] 或验证集 [6, 49]上的表现。由易到难学习已应用于知识蒸馏，以提升计算机视觉 [14, 39] 和自然语言处理 [52, 55]中的蒸馏效果。 [3, 14, 39] 利用教师的优化路径为学生模型制定课程， [52]则在训练过程中逐步纳入越来越多的细粒度文档对。

备注。 现有面向推荐系统的知识蒸馏方法，侧重于从与学生模型具有相同模型类型的同构教师模型中进行蒸馏。对于架构和学习目标都与学生模型不同的异构教师模型，其蒸馏尚未得到充分研究。在这项工作中，我们展示了蒸馏异构教师模型集成的必要性与难度，并采用由易到难学习来应对这一问题。此外，先前采用由易到难学习的知识蒸馏研究侧重于分类 [14, 31, 39]，或依赖于领域特定特征 [55], ，因此难以将其应用于排序问题和推荐系统。我们的工作提供了一种专门用于压缩排序模型的解决方案：根据学生模型的学习状态，蒸馏由易到难的排序知识序列。

3 预备知识

3.1 问题定义

令 U 和 I 分别表示用户集合和物品集合。给定隐式用户-物品交互（例如点击）历史，推荐模型 $f: \mathrm{U} \times \mathrm{I} \rightarrow \mathbb{R}$ 学习每个用户-物品对的排序分数。基于预测分数，推荐系统为每位用户提供未观测物品中排名前 K 的排序列表，称为 Top-K 推荐。给定一组复杂的教师模型 $\mathrm{F} = \{f^1, f^2, ..., f^M\}$，我们的目标是将这些教师模型集成有效地压缩为一个轻量级学生模型 f。该学生模型的推理计算成本显著降低，因此更适用于实时服务和资源受限环境。我们采用模型无关的解决方案，使任何类型的推荐模型都能灵活地用作教师模型或学生模型，从而允许服务提供商根据自身环境使用任何首选模型。

我们采用具有不同架构和损失函数的异构教师模型。在本研究中，我们选择了六种在推荐系统研究中得到广泛研究的代表性模型类型：矩阵分解（MF） [36], 度量学习（ML） [12], 深度神经网络（DNN）[10], GNN（图神经网络） [9], AE（自编码器） [26], 基于物品的自编码器（I-AE） [38]。附录A.2 对教师模型及其集成进行了详细分析。

符号说明。 给定一个排序列表（即物品的排列） π， π_k表示 π中的第 k个物品，而 $r(\pi, i)$表示物品 i在 π中的排名，其中数值越小表示位置越靠前，即 $r(\pi, i) = 0$ 表示排名最高。请注意， π是针对每个用户 u定义的。为简化符号，本文中省略 u在 π中的表示。

3.2 排序匹配蒸馏

排序匹配蒸馏 [5, 15, 25, 34, 41] 训练学生模型模仿教师的排列。一种主流策略是根据 Plackett-Luce模型 [30], ，然后训练学生模型，使其最大化教师排列的似然 [47]。给定教师针对某个用户 u的排列 π^t，近期研究 [15, 25, 34] 会匹配排名靠前的项目（P）的排序，同时忽略 其余项目（N）的排序。列表式知识蒸馏损失定义为 [P 的排列概率的负对数似然； N] 如下：

$$\mathcal{L}_F(P, N) = -\log \prod_{k=1}^{|P|} \frac{\exp f(u, P_k)}{\sum_{j=k}^{|P|} \exp f(u, P_j) + \sum_{l=1}^{|N|} \exp f(u, N_l)} \quad (1)$$

P和 N通常分别选取少量排名靠前的项目，以及从大量剩余项目中随机抽取的一个项目子集 [15, 34]。通过最小化损失，学生模型学习保留 P中的细粒度顺序，同时惩罚 N中排名低于 P中排名最低项目的项目。值得注意的是， N内部项目的顺序不一定会被保留。

3.3 排序知识蒸馏研究

我们进行了分析，展示了压缩异构教师模型的集成知识这一任务的难度。此外，我们表明，有助于降低这一难度的线索可以从教师模型的中间训练状态中获得。在此，我们使用嵌入维度为6的矩阵分解作为学生模型，而所有教师模型的嵌入维度均为64。使用其他类型的学生模型也观察到了类似结果。我们仅通过蒸馏（公式1）训练学生模型。详细设置请参见第5节。

3.3.1 差异。 由于推荐准确率只能间接反映知识蒸馏的效果，我们引入一种新指标，直接评估学生模型对教师排列 π^t的学习程度。令 π表示学生模型预测的排列。我们将 π与 π^t之间的差异定义为

$$D@K(\pi, \pi^t) = 1 - NDCG@K(\pi, \pi^t), \quad (2)$$

其中， $D@K(\pi, \pi^t) = 0$ 表示学生模型完美保留了 π^t的前 K 项排序。 $NDCG$是一种广泛使用的列表式排序评估指标 [44]。在此，我们将 π^t视为最优排序。

$$NDCG@K(\pi, \pi^t) = \frac{DCG@K(\pi)}{DCG@K(\pi^t)}, \; DCG@K(\pi) = \sum_{k=1}^{K} \frac{2^{y_{\pi_k}} - 1}{\log(k+1)} \quad (3)$$

每个物品 i的相关性（即 y_i）通常定义为评分（针对显式反馈）或二元值（针对隐式反馈）。为提高 π^t中排名靠前物品的权重，我们采用参数化几何分布 [35], ，即 $y_i = \exp(-r(\pi^t, i)/\lambda)$，当 i位于 π^t的前 K 项内时取此值，否则为 0。 $\lambda \in \mathbb{R}^+$是控制分布尖锐程度的超参数。

3.3.2 观察与分析。 我们通过来自不同排序监督信号的蒸馏来训练学生模型（矩阵分解），并分析学生模型与监督信号之间的差异。在表1中，（a）表示同构教师模型，其模型类型（即矩阵分解）与学生模型相同，这也是以往大多数工作所采用的设置。（b）和（c）分别表示由六个具有不同初始化的同构教师模型组成的集成，以及由六个异构教师模型组成的集成。“NLL”表示学生模型在给定监督信号下的负对数似然（公式1），其值越低，说明学生模型对给定监督信号的模拟效果越好。我们针对每个用户计算指标，并报告平均值。在表1中，

表1：知识蒸馏后与给定监督信号的差异。

数据集	监督信号（教师）差异		
	类型	Recall@50	D@10 D@50 负对数似然
Amusic	（a）单教师（矩阵分解）	0.2202	0.6640 0.5167 0.5805
	（b）集成（矩阵分解）	0.2396	0.6699 0.5162 0.6048
	（c）集成（异构）	0.2719	0.7417 0.5958 0.7206
CiteULike	(a) 单教师（矩阵分解）	0.2604	0.5101 0.3716 0.5962
	(b) 集成（矩阵分解）	0.2763	0.5373 0.3910 0.5977
	(c) 集成（异构）	0.3144	0.6983 0.5269 0.6906

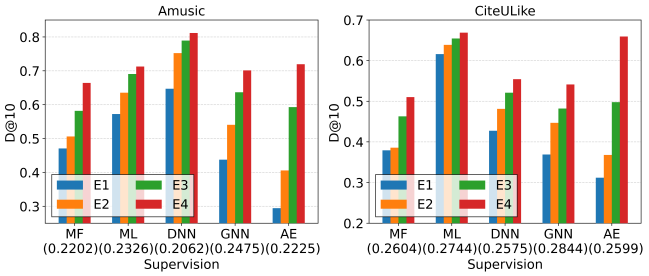

图2：各教师模型的中间训练状态（E1/E2/E3）和最终收敛状态（E4）下，与不同监督信号之间的差异。我们还在 x 轴下方标注了各收敛教师模型的 Recall@50。

我们观察到与压缩（a）和（b）相比，压缩（c）产生了明显更大的差异。换言之，当我们迁移异构教师模型的集成知识时，蒸馏的有效性会严重下降。这一有趣的观察表明，排列中的物品排序会对学习难度造成巨大差异。

为探究如此大的差异源自何处，我们分析了 (c) 中分别向各教师模型学习的情况。我们分别使用各个已收敛教师模型及其中间训练状态[1]进行蒸馏，独立训练学生模型（矩阵分解）。随后，我们在图2中展示了与各监督信号之间的差异。我们发现，与同构模型（即使用 *E4* 的矩阵分解）相比，从异构模型（即使用 *E4* 的其他模型）进行蒸馏始终会导致更大的差异。众所周知，异构模型具有不同的归纳偏置，使模型倾向于某些假设而非其他假设 [1, 19, 51]。从这个角度来看，向异构教师模型学习尤其具有挑战性，因为学生模型需要学习与其不太匹配的关系。有趣的是，教师模型准确率更高并不一定会导致更大的差异。例如，在 CiteULike 数据集上，GNN 的准确率高于矩阵分解，而自编码器的准确率与矩阵分解相当。然而，从自编码器学习时，差异要大得多。

一方面，无论模型类型如何，差异都会随着教师模型的训练（从 *E1* 到 *E4*）逐渐增大。也就是说，在训练过程中，教师模型的知识越来越难以模仿。如第5.2节所示，模型往往会先学习整体模式，再逐渐学习个性化偏好。我们还表明，教师模型的知识在训练过程中日益复杂，因为后续的预测会为每位用户提供更多样化、个性化的物品排名。这些观察促使我们将教师模型的训练轨迹用作学生模型的天然课程学习。

[1]对于每个教师，我们使用 4 个训练状态（或检查点），从 E1 到 E4，分别对应收敛训练轮次的 25%、50%、75% 和 100% 处的检查点。I-AE 的结果与 AE 相似，由于篇幅有限，故省略其结果。

4 方法

HetComp（面向推荐系统的**异构**模型**压缩**）框架基于由易到难学习的理念，利用教师模型的训练轨迹监督学生模型。HetComp 由两个旨在实现由易到难蒸馏的主要组件构成：

- **（第 4.1 节）动态知识构建**会根据学生的学习状态，动态生成难度适当的知识。该方法首先从每个教师的轨迹中识别出合适的知识，并构建目标知识以指导学生模型。该过程以个性化方式进行，在考虑每个用户不同学习难度的情况下，迁移其推荐结果。

- **（第 4.2 节）自适应知识迁移**会根据学生的学习状态调整蒸馏目标。该方法训练学生首先关注目标排列中的整体关系，然后逐步转向学习对偏好物品的细粒度顺序进行学习排序。此外，我们提出了一种新的策略，用于迁移已观测交互和未观测交互的知识，而此前的工作忽略了这一点。HetComp 的整体训练过程将在第 4.3 节中介绍。

4.1 动态知识构建

我们的知识构建旨在根据学生模型的状态，从教师模型的训练轨迹中生成要迁移的目标知识。这个过程需要满足以下要求：（1）它应体现来自每个教师模型、针对每位用户的知识具有不同的学习难度；我们的分析表明，学习难度因教师模型而异，而且针对每位用户的排序也自然具有不同的难度。（2）它应以模型无关方式运作，从而适用于任何类型的教师模型或学生模型。因此，我们避免使用因模型类型而异的统计量（例如训练损失）来决定要迁移的知识。由于数据稀疏性很高，难以获取每位用户足够的交互，我们也避免使用额外留出数据上的性能指标。

令 $\mathbf{T} = \{T^x\}_{x\in\{1,\dots,M\}}$ 表示教师模型的训练轨迹。对于每个教师模型 f^x，我们使用其在 E 个不同训练状态下的预测（即未观测物品的排序），即 $[\pi^{x,1}, \dots, \pi^{x,E}]$[2]。最后一个排列 $\pi^{x,E}$，对应于收敛后的最终预测。我们的分析表明，教师的知识在训练过程中会变得越来越难，因此 $\pi^{x,e}$ 比 $\pi^{x,e+1}$ 更容易模仿。因此，我们从 $\pi^{x,1}$ 开始，然后逐渐过渡到 $\pi^{x,E}$。我们用 v 表示 M 维选择向量，其中每个元素 $v[x] \in \{1, \dots, E\}$ 指示当前选取教师 f^x 的哪个训练状态来指导学生模型。在训练过程中，我们从每个教师的训练轨迹 T^x 中识别合适的知识，然后根据 v 整合这些知识，构建动态监督信号 πd。知识构建的概述见算法1。

为了控制蒸馏的难度，我们使用学生模型与教师模型的排序预测之间的差异。我们的核心思路是利用较早的预测来降低学习较晚预测的难度。对于每条教师轨迹 T^x，我们

[2]在本研究中，我们将 E 设为 4，并将其均匀分布在每位教师的轨迹中。我们通过实验证实，只要在教师训练过程中进行合理分布，E 几乎不会影响最终性能（附录 A.3）。

算法1： 动态知识构建

输入： 教师轨迹 T，学生模型 f，当前具有差异 d的选择变量 v
输出： 新的动态目标排列 π^{d}

由学生模型预测 $\pi\ f$
对每个教师 $x \in \{1, ..., M\}$ **执行**
  如果 $(v[x] < E)$ **且** $(\gamma^x > \alpha)$ **则**
    $v[x] = v[x] + 1$
    $d[x] = D@K(\pi, \pi^x, \min(v[x]+1, E))$
生成 $\pi^{\mathrm{d}} = g(\{\pi^{x,v[x]}\}_{x\in\{1,...,M\}})$

持续跟踪与下一训练状态 $D@K(\pi, \pi^{x,(v[x]+1)})$ 对应排列的差异；当差异减小到一定程度时，就转到下一个状态。为此，我们将相对差异比定义如下：

$$\gamma^x = \frac{d[x]}{D@K(\pi, \pi^{x,(v[x]+1)})}, \tag{4}$$

其中，$d[x]$ 表示学生模型开始学习 π^x时计算出的、相对于 π^x $_{v[x]+1}$的初始差异，$_{(v[x]+1)}$。请注意，$d[x]$ 被视为常量，而 D $@K(\pi,\ \pi^x,\ _{(v[x]+1)})$会随着学生模型在训练过程中的演进而减小。$\gamma^x$揭示了 $\pi^x,\ _{(v[x]+1)}$ 的学习难度降低的程度。随后，我们采用一种贪心策略：如果 γ^x 大于阈值 α，就转到下一个教师状态。$\alpha \geq 1$ 是一个控制转换速度的超参数，取值越低，转换越快。我们基于差异设计的贪心策略能够兼顾学生模型的学习状态和教师模型难度的变化，从而高效地进行课程学习。此外，由于差异可以针对任何模型进行度量，因此该策略以模型无关方式运作。

更新选择变量 v后，我们通过以下方式生成目标排列：

$$\pi^{\mathrm{d}} = g(\{\pi^{x,v[x]}\}_{x\in\{1,...,M\}}) \tag{5}$$

其中，g 是用于整合各个排列的排序集成函数。在HetCo mp中，可以灵活使用各种集成技术 [32]，，从简单的平均到带有可学习重要性权重的复杂方法均可。本工作采用一种简单的方法，以预测的一致性作为重要性；这一做法在近期研究中被广泛使用 [7, 19]（附录A.2.3）。

一旦对于所有 x 都有 $v[x]$ 等于 E，该知识就等价于传统知识蒸馏中使用的最终集成知识。然而，关键区别在于，学生模型现在更有能力从难度更高的知识中学习。

4.2 自适应知识迁移

我们介绍如何将动态构建的排列 π^{d}迁移给学生模型。首先，我们在第 4.2.1 节介绍根据学生模型学习状态进行自适应调整的蒸馏目标。随后，我们在第 4.2.2 节解释迁移已观测项目和未观测物品排序知识的策略。

4.2.1 自适应蒸馏目标. 考虑到目标排列会在学生模型训练过程中不断变化，在训练初期学习详细排序不仅没有必要，而且难度很大。我们首先训练学生模型，使其通过 L 学习目标排列中的整体关系。随后，一旦根据最终收敛的预测构建出目标排列（即 $v[x]$ 等于 E，$\forall x$），我们便通过 L_F 继续学习细粒度的排序顺序。通过修改公式1，我们将用于传递整体关系的蒸馏目标定义如下：

$$\mathcal{L}_O(P, N) = -\log \prod_{k=1}^{|P|} \frac{\exp\ f(u, P_k)}{\exp f(u, P_k) + \sum_{l=1}^{|N|} \exp f(u, N_l)}, \tag{6}$$

该方法仅推动 P中的项目排名高于 N中的项目，而不对 P中的项目之间施加任何约束。

4.2.2 已观测项目/未观测物品的知识迁移 .
以往的知识蒸馏方法 [15, 25, 40] 大多只迁移未观测物品的排序，而不考虑已观测项目的排序。我们认为，已观测项目之间的相对优先级同样是用户偏好的宝贵知识。利用已观测项目知识的一种直接方法，是迁移由教师模型构建的全体项目集合排列。然而，由于推荐系统模型大多通过协同过滤（CF）损失进行训练（例如，二元交叉熵 [10], BPR [36] 损失），该损失会惩罚未观测物品的得分低于已观测项目，因此“相关但尚未观测项目”在全体项目集合的排列中的排名，很可能低于其真实相关性所对应的排名。我们观察到，如果直接蒸馏整个项目排列，会导致学生模型预测中过度惩罚某些尚未观测项目，从而阻碍偏好的学习。

我们提出了一种新策略，以有效利用已观测和未观测的排序知识。我们分别迁移已观测物品和排名靠前的未观测物品的排序，以避免过度惩罚。令 P^- 表示排名靠前的未观测物品的排序，N表示从 π^{d}中得到的其余未观测物品。我们另外定义 P^+，即已观测物品的排序[3]。蒸馏损失定义如下：

$$\mathcal{L} = \mathcal{L}_{KD}(P^+, N) + \mathcal{L}_{KD}(P^-, N), \tag{7}$$

其中，L_{KD}根据学生模型的学习状态，可以是L_O（公式6）或 L_F（公式1）。请注意，我们的策略不会强制排名靠前的未观测物品（P^-）排在已观测物品（P^+）之后，从而避免将尚未观测项目推离排序列表的顶部。相反，该策略允许一些相关性较高的未观测物品自然地与已观测物品混排在排序列表顶部附近。L_{KD}（$P^+,\ N$）与协同过滤损失的区别在于：（1）它迁移已观测物品的排序；（2）它不会惩罚排名靠前的未观测物品。

4.3 整体训练过程

算法2总结了整个训练过程。知识构建以个性化方式针对每个用户 u 进行。此外，我们会持续进行知识构建和 L_O，直至学生模型从最终收敛的教师模型中学到知识（即 $v[x]$ 等于 E, $\forall x$ ）。我们每隔 p 个训练轮次进行一次知识构建，因为每个训练轮次都更改目标排列既耗时又没有必要。在本文中，我们将 p 设为 10。附录 A.3 对 HetComp 的离线训练成本进行了详细分析。

[3] P^+ 由已观测项目上的已收敛教师模型集成得到。值得注意的是，由于已观测项目集合是固定的，因此在学生模型训练开始前只需生成一次 P^+，而$\mathcal{L}_O$ 不会传递其详细排序。

算法2： HetComp 训练流程

输入： 教师轨迹 T、学生模型 f、更新周期 p
输出： 训练完成的学生模型 f

随机初始化学生模型 f
初始化选择变量 $v_u[x] = 1$ 和 $d_u[x] \quad \forall x, \quad \forall u$
通过已收敛教师的集成获得 $P_u^+ \qquad \forall u$
对于 $i = 1, \ldots, epoch_{max}$，**执行**
  对每个用户 $u \in \mathcal{U}$ **执行**
    如果 $(i \% p == 0)$ **且不满足** $(v_u[x] == E, \forall x)$，**则**
      通过动态知识更新 v_u，d_u 和 π_u^{d} 构建（算法1）
    如果不满足$(v_u[x] == E, \forall x)$ **则**
      $\mathrm{L} = \mathrm{L}_O(P_u^+, N_u) + \mathrm{L}_O(P_u^-, N_u)$
    else
      $\mathrm{L} = \mathrm{L}_F(P_u^+, N_u) + \mathrm{L}_F(P_u^-, N_u)$
    更新学生模型 f

5 实验

实验设置。我们使用三个真实世界数据集：Amazon Music（Amusic）、CiteULike 和 Foursquare。我们按照 60%/20%/20% 的比例，将每位用户的交互随机划分为训练集/验证集/测试集 [12]。我们采用两个 Top-K 排序指标：Recall@K（R@K）和归一化折损累积增益@K（N@K）[4]。对于学生模型，我们使用矩阵分解、机器学习和深度神经网络 [15, 22, 25]。所有教师模型的用户/物品嵌入维度（或自编码器的瓶颈层大小）设为 64，所有学生模型则设为 6，使学生模型使用的学习参数大约为教师模型的十分之一，与 [15, 22, 25]中的做法一致。

基线方法。我们将 HetComp 与多种蒸馏方法进行比较。除 MTD 外，所有比较方法都迁移收敛后的异构教师模型的集成排序（记为 **集成**），其生成方式与 HetComp 相同。第一组基线方法包括一种逐点知识蒸馏方法。

- **关系蒸馏 [40]** 迁移排名靠前项目的重要性。重要性由每个项目的排名位置定义。

第二组包括排序匹配知识蒸馏方法（第 3.2 节）。

- **RRD [15]** 是一种为推荐系统提出的排序匹配方法。它采用侧重于排名靠前项目的列表式损失。
- **MTD（多教师蒸馏）**：我们注意到，其他领域的多教师知识蒸馏方法 [8, 23, 28, 54] 通常会使用每个教师模型在每个数据实例上的可训练重要性。借鉴这一思路，我们提出 MTD，在每个用户的排序中利用每个教师模型的可训练重要性来迁移知识。

第三组包括用于排序的最先进的知识蒸馏方法，这些方法采用先进的方案来提高蒸馏质量。

- **CL-DRD [52]** 是文档
  检索领域最先进的知识蒸馏方法。它采用课程学习，其中学习难度由绝对排名位置预先定义。
  度由绝对排名位置预先定义。
- **DCD [25]** 是推荐系统领域最先进的知识蒸馏方法。它采用双重校正损失，与 RRD 结合使用，用于校正学生模型未能准确预测的内容。

请注意，直接使用预测分数的知识蒸馏方法 [11, 22, 48, 54] 并不适用，因为在我们的任务中，教师模型与学生模型的输出分数分布差异很大。

[4]这里，归一化折损累积增益是针对隐式反馈计算的。为避免混淆，在实验部分我们使用归一化折损累积增益来指代推荐准确率。

5.1 蒸馏效果比较

表 2 展示了采用不同知识蒸馏方法训练的学生模型的推荐性能。此外，表 4 总结了
最佳基线方法（即 DCD）和 HetComp 的差异值。“最佳教师”表示在每个数据集上，所有异构教师模型中性能最佳的教师模型。

- 对于所有数据集和学生模型，HetComp 的表现都显著优于所有基线方法，有效提升了学生模型的性能[5]。此外，从推荐物品数量（K）来看，与 R@50/N@50 相比，HetComp 在 R@10/N@10 上取得了更大的提升，这有利于那些旨在向用户提供最受青睐物品的现实服务。此外，表 4 显示，与最佳基线方法相比，HetComp 确实显著降低了差异。

- 与逐点知识蒸馏方法（即关系蒸馏）相比，其他直接迁移排序顺序的知识蒸馏方法始终表现更好，这再次说明了排序知识对于 Top-K 推荐的重要性。一方面，与 RRD 相比，MTD 的性能提升有限。由于交互数据的稀疏性较高，用户级可学习重要性很容易过拟合。

- 采用先进方案的排序匹配（即 CL-DRD、DCD）在一定程度上提升了蒸馏效果。然而，对于 CL-DRD，我们观察到其在不同数据集和学生模型上的表现存在显著差异。一个可能的原因是，它使用预定义规则来定义难度并控制难度级别，却未考虑学生模型的状态。另一方面，DCD 直接迁移学生模型未能预测的内容，表现始终优于 RRD。

- 表 5 展示了集成模型和 HetComp 的参数数量及推理延迟。我们逐步增大学生模型（矩阵分解）的规模，直至其达到与集成模型相当的性能。与因多次模型前向传播而产生较高推理成本的集成模型相比，HetComp 通过将知识蒸馏到紧凑的学生模型中，能够显著降低成本。

关于模型泛化能力的进一步比较。由于已知由易到难的学习范式能够产生泛化能力更强的模型 [13], ，我们进一步评估在教师模型提供的知识极其有限时，学生模型捕捉用户偏好的准确程度。我们将用户随机划分为两组，比例为 80%/20%（即 U_{g_1}，U_{g_2}）。随后，我们仅使用 U_{g_1} 中 80% 的训练交互来训练所有教师模型；也就是说，教师模型掌握的关于 U_{g_1} 的知识有限，且不了解 U_{g_2}。最后，我们使用 U_{g_1}[6] 上的全部训练数据并结合蒸馏来训练学生模型。也就是说，学生模型需要在教师模型提供不完整指导的情况下，独立从数据中学习。在表 3 中，我们观察到，所有情况下 HetComp 的表现都显著优于 DCD。这些结果表明，HetComp 训练出的学生模型具有更强的泛化能力，并且能够独立发现更准确的隐含偏好。我们认为，这一特性在持续添加新交互的场景中具有优势；将 HetComp 应用于持续学习的问题留待未来研究。 [33]

[5]我们还尝试了从各种教师模型（例如，最佳教师/同质教师集成）进行知识蒸馏。在所有情况下，HetComp 都能提升蒸馏效果（附录 A.4）。[6]对于未用于训练教师模型的训练交互，我们使用原始的协同过滤损失。

表 2：推荐性能对比。 *Imp* **表示 HetComp 相较于最佳基线的提升。**

	方法	Amusic				CiteULike				Foursquare			
		R@10	N@10	R@50	N@50	R@10	N@10	R@50	N@50	R@10	N@10	R@50	N@50
	最佳教师集成	0.0972	0.0706	0.2475	0.1139	0.1337	0.0994	0.2844	0.1392	0.1147	0.1085	0.2723	0.1635
		0.1096	0.0820	0.2719	0.1273	0.1550	0.1156	0.3144	0.1571	0.1265	0.1213	0.2910	0.1786
学生模型 模型：MF	无知识蒸馏	0.0449	0.0303	0.1451	0.0594	0.0568	0.0422	0.1372	0.0634	0.0726	0.0666	0.1806	0.1047
	RD	0.0522	0.0387	0.1602	0.0693	0.0610	0.0472	0.1514	0.0725	0.0778	0.0703	0.1921	0.1153
	RRD	0.0890	0.0659	0.2353	0.1077	0.0973	0.0740	0.2422	0.1113	0.0982	0.0905	0.2539	0.1446
	MTD	0.0901	0.0649	0.2279	0.1043	0.0993	0.0749	0.2425	0.1118	0.0955	0.0890	0.2402	0.1394
	CL-DRD	0.0883	0.0648	0.2375	0.1071	0.1033	0.0794	0.2512	0.1175	0.1001	0.0933	0.2528	0.1464
	DCD	0.0956	0.0675	0.2380	0.1079	0.1106	0.0851	0.2640	0.1246	0.1034	0.0965	0.2547	0.1491
	HetComp	**0.1036***	**0.0747***	**0.2469***	**0.1157***	**0.1379***	**0.1031***	**0.2814***	**0.1396***	**0.1118***	**0.1036***	**0.2722***	**0.1594***
	Imp	8.37%	10.67%	3.74%	7.23%	24.68%	21.15%	6.59%	12.04%	8.12%	7.36%	6.87%	6.91%
学生模型 模型：ML	无知识蒸馏	0.0447	0.0310	0.1522	0.0623	0.0210	0.0148	0.0859	0.0323	0.0184	0.0139	0.0804	0.0356
	RD	0.0706	0.0507	0.1874	0.0840	0.0835	0.0615	0.1914	0.0890	0.0729	0.0677	0.1811	0.1059
	RRD	0.0903	0.0643	0.2422	0.1074	0.0981	0.0701	0.2529	0.1116	0.0925	0.0813	0.2505	0.1366
	MTD	0.0843	0.0590	0.2293	0.1003	0.0944	0.0690	0.2519	0.1098	0.0909	0.0811	0.2440	0.1347
	CL-DRD	0.0866	0.0621	0.2409	0.1061	0.0989	0.0718	0.2583	0.1131	0.0931	0.0825	0.2541	0.1387
	DCD	0.0928	0.0653	0.2466	0.1086	0.1003	0.0724	0.2592	0.1144	0.0943	0.0845	0.2530	0.1399
	HetComp	**0.1020***	**0.0751***	**0.2470**	**0.1156***	**0.1251***	**0.0916***	**0.2686***	**0.1287***	**0.1039***	**0.0962***	**0.2645***	**0.1521***
	Imp	9.91%	15.01%	0.16%	6.45%	24.73%	26.52%	3.63%	12.50%	10.18%	13.85%	4.09%	8.72%
学生模型 模型：DNN	无知识蒸馏	0.0460	0.0324	0.1396	0.0597	0.0414	0.0339	0.1095	0.0518	0.0693	0.0665	0.1608	0.0987
	RD	0.0531	0.0378	0.1545	0.0670	0.0584	0.0445	0.1440	0.0671	0.0746	0.0683	0.1820	0.1060
	RRD	0.0851	0.0613	0.2255	0.1016	0.1034	0.0792	0.2552	0.1186	0.1016	0.0939	0.2584	0.1484
	MTD	0.0802	0.0563	0.2210	0.0958	0.0982	0.0710	0.2322	0.1058	0.0888	0.0797	0.2321	0.1305
	CL-DRD	0.0889	0.0623	0.2365	0.1047	0.1083	0.0816	0.2575	0.1183	0.1039	0.0983	0.2635	0.1536
	DCD	0.0919	0.0646	0.2404	0.1071	0.1114	0.0838	0.2668	0.1240	0.1060	0.1017	0.2671	0.1576
	HetComp	**0.1045***	**0.0768***	**0.2534***	**0.1190***	**0.1381***	**0.1050** *	**0.2864***	**0.1413***	**0.1136***	**0.1079** *	**0.2759***	**0.1642***
	Imp	13.71%	18.89%	5.41%	11.11%	23.97%	25.30%	7.35%	13.95%	7.17%	6.10%	3.29%	4.19%

∗ 表示与最佳基线相比，配对 t 检验结果具有显著性（0.05 水平）。

表 3：DCD 和 HetComp 在泛化设置下的推荐性能对比。

	User 组	方法	Amusic				CiteULike				Foursquare			
			R@10	N@10	R@50	N@50	R@10	N@10	R@50	N@50	R@10	N@10	R@50	N@50
	$\mathcal{U}_{g_1}$	最佳教师集成	0.0741	0.0562	0.2075	0.0948	0.1065	0.0789	0.2276	0.1111	0.1024	0.0977	0.2401	0.1457
			0.0972	0.0691	0.2344	0.1089	0.1304	0.0955	0.2805	0.1346	0.1141	0.1079	0.2655	0.1601
学生模型 模型：MF	$\mathcal{U}_{g_1}$	无知识蒸馏	0.0452	0.0307	0.1460	0.0600	0.0551	0.0430	0.1351	0.0642	0.0736	0.0683	0.1811	0.1063
		DCD	0.0784	0.0548	0.2126	0.0933	0.0955	0.0724	0.2256	0.1056	0.0984	0.0937	0.2317	0.1401
		HetComp	**0.0903***	**0.0661***	**0.2313***	**0.1059***	**0.1096***	**0.0834***	**0.2547***	**0.1206***	**0.1050** *	**0.0993***	**0.2483***	**0.1494***
		Imp	15.18%	20.62%	8.80%	13.50%	14.76%	15.19%	12.90%	14.20%	6.71%	5.98%	7.16%	6.64%
	$\mathcal{U}_{g_2}$	无知识蒸馏	0.0434	0.0286	0.1411	0.0570	0.0555	0.0427	0.1389	0.0649	0.0728	0.0676	0.1808	0.1058
		DCD	0.0567	0.0370	0.1706	0.0699	0.0722	0.0517	0.1785	0.0805	0.0738	0.0717	0.1839	0.1087
		HetComp	**0.0668***	**0.0465***	**0.1841***	**0.0788***	**0.0801***	**0.0558***	**0.2008***	**0.0876***	**0.0880***	**0.0829***	**0.2088***	**0.1227***
		Imp	17.81%	25.68%	7.91%	12.73%	10.94%	7.93%	12.49%	8.82%	19.24%	15.62%	13.54%	12.88%
学生模型 模型：ML	$\mathcal{U}_{g_1}$	无知识蒸馏	0.0442	0.0308	0.1518	0.0623	0.0209	0.0148	0.0856	0.0322	0.0201	0.0158	0.0860	0.0386
		DCD	0.0817	0.0585	0.2179	0.0970	0.0861	0.0618	0.2214	0.0974	0.0891	0.0799	0.2289	0.1288
		HetComp	**0.0908***	**0.0641***	**0.2284***	**0.1027***	**0.1110***	**0.0831***	**0.2445***	**0.1173***	**0.1025***	**0.0960***	**0.2499***	**0.1475***
		Imp	11.14%	9.57%	4.82%	5.88%	28.92%	34.47%	10.43%	20.43%	15.04%	20.15%	9.17%	14.52%
	$\mathcal{U}_{g_2}$	无知识蒸馏	0.0468	0.0318	0.1537	0.0627	0.0208	0.0147	0.0869	0.0324	0.0201	0.0156	0.0861	0.0384
		DCD	0.0532	0.0368	0.1601	0.0680	0.0522	0.0371	0.1644	0.0665	0.0631	0.0562	0.1826	0.0976
		HetComp	**0.0651***	**0.0453***	**0.1906***	**0.0805***	**0.0806***	**0.0560***	**0.2012***	**0.0868***	**0.0709***	**0.0626***	**0.2052***	**0.1099***
		Imp	22.37%	23.10%	19.05%	18.38%	54.41%	50.94%	22.38%	30.53%	12.36%	11.39%	12.38%	12.60%
学生模型 模型：DNN	$\mathcal{U}_{g_1}$	无知识蒸馏	0.0463	0.0329	0.1398	0.0602	0.0408	0.0335	0.1088	0.0514	0.0691	0.0665	0.1606	0.0986
		DCD	0.0805	0.0579	0.2118	0.0947	0.0882	0.0675	0.2177	0.1010	0.0914	0.0854	0.2231	0.1314
		HetComp	**0.0902***	**0.0664***	**0.2308***	**0.1063***	**0.1106***	**0.0841***	**0.2527***	**0.1206***	**0.1014***	**0.0960***	**0.2472***	**0.1438***
		Imp	12.05%	14.68%	8.97%	12.25%	25.40%	24.59%	16.08%	19.41%	10.94%	12.41%	10.80%	9.44%
	$\mathcal{U}_{g_2}$	无知识蒸馏	0.0450	0.03034	0.1385	0.05795	0.0435	0.0352	0.1121	0.0532	0.0698	0.0671	0.1615	0.0993
		DCD	0.0501	0.0342	0.1551	0.0642	0.0586	0.0440	0.1516	0.0673	0.0670	0.0621	0.1686	0.0974
		HetComp	**0.0647***	**0.0448***	**0.1829***	**0.0751***	**0.0780***	**0.0572***	**0.1901***	**0.0833***	**0.0820***	**0.0778***	**0.2001***	**0.1188***
		Imp	29.14%	30.99%	17.92%	16.98%	33.11%	30.00%	25.40%	23.77%	22.39%	25.28%	18.68%	21.97%

∗ 表示相对于最佳基线的配对 t 检验（0.05 显著性水平）结果具有显著性。

表 4：DCD 与 HetComp 的差异比较。

	方法	Amusic		CiteULike		Foursquare	
		D@10	D@50	D@10	D@50	D@10	D@50
MF	无知识蒸馏	0.8984	0.8028	0.7668	0.6608	0.6211	0.4625
	DCD	0.6610	0.5286	0.4433	0.3184	0.2856	0.1690
	HetComp	0.5929	0.4709	0.3401	0.2359	0.2135	0.1210
	Imp	10.30%	10.92%	23.28%	25.91%	25.25%	28.40%
ML	无知识蒸馏	0.9072	0.8149	0.9583	0.8936	0.9532	0.8654
	DCD	0.6699	0.5283	0.5272	0.3659	0.3429	0.1967
	HetComp	0.5967	0.4756	0.3856	0.2693	0.2653	0.1481
	Imp	10.93%	9.98%	26.86%	26.40%	22.63%	24.71%
DNN	无知识蒸馏	0.9070	0.8232	0.8225	0.7330	0.6438	0.5119
	DCD	0.6862	0.5462	0.4836	0.3427	0.2377	0.1315
	HetComp	0.5338	0.4170	0.2942	0.2101	0.2014	0.1110
	Imp	22.21%	23.65%	39.16%	38.69%	15.27%	15.59%

表 5：准确率-效率权衡。时间（秒）表示为每位用户生成推荐所需的平均实际运行时间。我们使用 PyTorch 和 CUDA，在 RTX A5000 和 Xeon Gold 6226R 中央处理器上运行。

数据集	方法	准确率 R@10	准确率 N@10	效率 参数（嵌入维度）	效率 时间
Amusic	集成	0.1096	0.0820	5.79M (64)	10.57s
	HetComp	0.1102	0.0817	0.27M (18)	0.82秒
CiteULike	集成	0.1550	0.1156	11.72M (64)	22.10秒
	HetComp	0.1548	0.1150	0.45M (15)	1.10s
Foursquare	集成	0.1265	0.1213	18.52M (64)	35.47s
	HetComp	0.1263	0.1214	0.96M (20)	2.12秒

5.2 HetComp 研究

我们对 HetComp 进行深入分析，以加深对它的理解。补充结果（包括超参数研究和消融实验）见附录 A.4。

5.2.1 教师的训练轨迹为何有帮助？为了解答这一问题，我们分析教师中间状态（即 E1-E4，其中 E4 为收敛状态）所揭示的知识。所有报告结果均为各中间状态下所有教师模型的平均值。为便于直观理解，我们报告相对于 E4 数值的比率。

• 我们考察每个中间状态的排序列表中包含的物品。在所有用户的前 50 名排名列表的并集中，我们统计（1）唯一物品的数量，以及（2）冷门物品的数量。[7] 表 6 展示了各教师训练状态下唯一物品和冷门物品数量的比率。我们发现，推荐结果中包含的物品在训练过程中逐渐变得多样化。此外，冷门物品的占比也在逐步增加；与热门物品相比，冷门物品体现了更多个性化偏好。

• 我们分析每个状态捕捉群体层级偏好和用户层级偏好的能力。为了评估群体层级偏好，我们根据交互历史使用 k-means 确定 50 个用户组，然后找出每个组中最常被消费的前 50 个测试物品。我们使用这些群体层级测试物品来评估组内的每位用户。用户层级偏好对应原始推荐任务。我们使用 R@50 评估这两个层级的推荐准确率。图 3 汇总了各教师模型训练状态中群体层级偏好和用户层级偏好的比例。我们观察到，较早的状态捕捉到的用户层级偏好有限，但群体层级偏好相当准确。也就是说，尽管教师模型较早阶段的预测所包含的个性化（细粒度）偏好较少，但它们揭示了整体的群体层级（粗粒度）模式。

总而言之，在 HetComp 中，通过利用教师模型的中间状态，构成知识的物品会逐步变化，从而逐渐揭示更多样化且个性化的偏好。这与课程学习的理念十分契合：先关注整体概念，再逐渐学习具体模式 [2]。值得注意的是，我们所有的基线方法在学生模型的整个训练过程中传递的都是固定知识 。

5.2.2 动态知识构建（***DKC***）的研究。 我们对 DKC 进行了详细分析。这里，我们在 Ci-teULike 数据集上使用 MF（学生模型）。图 4（左）展示了用户平均知识选择（即 $\mathbb{E}_x[v_u[x]]$）的变化。在学生模型训练过程中，通过学习较早的预测，与最终排序的差异有所降低；HetComp 则沿着教师轨迹，逐渐转向更难的知识。此外，每位用户的知识选择各不相同，这反映了不同用户在学习各项知识时所面临的不同学习难度。图 4（右）比较了 DKC 的两个变体："DKC-g"使用通过对用户级选择取平均值计算得出的全局选择变量，"DKC-e"则使用一条简单规则，在经过一定的训练轮次间隔后转移到下一个状态。该间隔通过将 DKC 结束时的训练轮次均分为

[7]我们将交互次数处于最低 30% 的物品视为冷门物品。

表 6：教师模型的前 50 名排名列表中独特物品数量与冷门物品数量的相对比例。

教师训练状态	Amusic		CiteULike	
	独特	且不受欢迎	独特	且不受欢迎
E1	0.8386	0.7205	0.7055	0.7190
E2	0.8863	0.8618	0.8207	0.8366
E3	0.9479	0.9254	0.9230	0.9191
E4	1.0	1.0	1.0	1.0

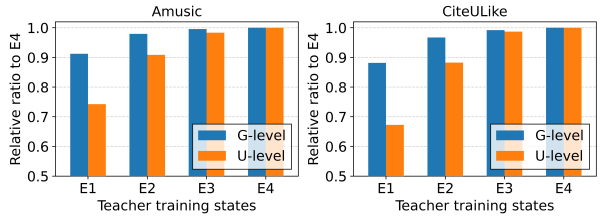

图 3：捕捉组层级和用户层级偏好能力的相对比例（指标：R@50）。

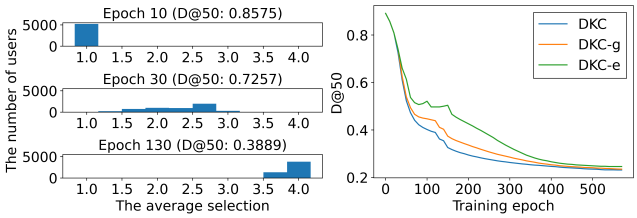

图4：（左）学生模型在第 10/30/130 训练轮次的平均选择分布。（右） $D@50$ 动态知识构建不同变体的曲线。差异是相对于收敛的教师模型的集成排序计算得出的。

3。与这些变体相比，DKC 为学生模型提供了更高效的课程。DKC-e 反而会增大差异（约在第 100 个训练轮次），因为它无法反映学生模型的学习状态。这些结果表明，在 DKC 中同时考虑用户级学习难度的差异以及学生模型的学习状态具有重要意义。

6 结论

我们提出了一种新的HetComp框架，将异构推荐模型的集成知识压缩到轻量级模型中，从而降低巨大的推理成本，同时保持较高的准确率。通过分析，我们表明，从异构教师模型中进行蒸馏尤其具有挑战性，而教师模型的训练轨迹有助于缓解这一较高的学习难度。基于由易到难学习的思想，HetComp利用动态知识构建 提供难度逐步增加的排序知识，并通过自适应知识迁移 逐步迁移更细粒度的排序信息。我们进行了广泛的实验，结果表明，HetComp显著提升了蒸馏效率和学生模型的泛化能力。鉴于其与现有模型的良好兼容性，我们预计HetComp框架能够解决推荐系统中的准确率-效率权衡问题。

致谢。 我们感谢Sang-Wook Kim、Minsu Cho、Dong- woo Kim和Chanyoung Park提出的宝贵意见。本研究得到微软亚洲研究院和IITP（No.2022- 00155958）的支持，并得到由MSIT资助的IITP项目（No.2018-0-00584, 2019-0- 01906）以及由MSIT资助的NRF项目（No.2020R1A2B5B03097210）的支持。**数字对象标识符**。https://doi.org/10.5281/zenodo.7594882

参考文献

[1] Samira Abnar、Mostafa Dehghani 和 Willem Zuidema。2020。通过知识蒸馏迁移归纳偏置。arXiv 预印本 arXiv:2006.00555（2020）。[2] Yoshua Bengio、Jérôme Louradour、Ronan Collobert 和 Jason Weston。2009。课程学习。载于 ICML。[3] George Cazenavette、Tongzhou Wang、Antonio Torralba、Alexei A Efros 和 Jun-Yan Zhu。2022。通过匹配训练轨迹进行数据集蒸馏。载于 CVPR。4750–4759。[4] Yankai Chen、Huifeng Guo、Yingxue Zhang、Chen Ma、Ruiming Tang、Jingjie Li 和 Irwin King。2022。通过多面量化强化学习二值化图表示，用于 Top-K 推荐。载于 KDD。[5] Yuntao Chen、Naiyan Wang 和 Zhaoxiang Zhang。2018。Darkrank：通过跨样本相似度迁移加速深度度量学习。载于 AAAI。[6] Yudong Chen、Xin Wang、Miao Fan、Jizhou Huang、Shengwen Yang 和 Wenwu Zhu。2021。下一兴趣点推荐的课程元学习。载于 KDD。[7] Jingtao Ding、Yuhan Quan、Quanming Yao、Yong Li 和 Depeng Jin。2020。简化并增强隐式协同过滤的负采样鲁棒性。载于 NeurIPS。[8] Takashi Fukuda、Masayuki Suzuki、Gakuto Kurata、Samuel Thomas、Jia Cui 和 Bhuvana Ramabhadran。2017。从教师集成中高效进行知识蒸馏。载于 Interspeech。3697–3701。[9] Xiangnan He、Kuan Deng、Xiang Wang、Yan Li、Yongdong Zhang 和 Meng Wang。2020。LightGCN：简化并增强用于推荐的图卷积网络。载于 SIGIR。[10] Xiangnan He、Lizi Liao、Hanwang Zhang、Liqiang Nie、Xia Hu 和 Tat-Seng Chua。2017。神经协同过滤。载于 WWW。[11] Geoffrey Hinton、Oriol Vinyals 和 Jeffrey Dean。2015。蒸馏神经网络中的知识。NIPS（2015）。[12] Cheng-Kang Hsieh、Longqi Yang、Yin Cui、Tsung-Yi Lin、Serge Belongie 和 Deborah Estrin。2017。协同度量学习。载于 WWW。[13] Lu Jiang、Deyu Meng、Qian Zhao、Shiguang Shan 和 Alexander G Hauptmann。2015。自步课程学习。载于 AAAI。[14] Xiao Jin、Baoyun Peng、Yichao Wu、Yu Liu、Jiaheng Liu、Ding Liang、Junjie Yan 和 Xiaolin Hu。2019。通过路径约束优化进行知识蒸馏。载于 CVPR。1345–1354。[15] SeongKu Kang、Junyoung Hwang、Wonbin Kweon 和 Hwanjo Yu。2020。DE-RRD：用于推荐系统的知识蒸馏框架。载于 CIKM。[16] SeongKu Kang、Junyoung Hwang、Wonbin Kweon 和 Hwanjo Yu。2021。面向物品侧排序正则化的推荐系统蒸馏。《Information Sciences》 580 2021 15 34 10 1016 2021 08 060 （），–。[17] https://doi.org/。/j.ins.。。SeongKu Kang、Junyoung Hwang、Wonbin Kweon 和 Hwanjo Yu。2021。面向推荐系统的拓扑蒸馏。载于 KDD。[18] SeongKu Kang、Junyoung Hwang、Dongha Lee 和 Hwanjo Yu。2019。面向冷启动用户跨域推荐的半监督学习。载于 CIKM。[19] SeongKu Kang、Dongha Lee、Wonbin Kweon 和 Hwanjo Yu。2022。面向单类协同过滤的异构目标共识学习。载于 WWW。[20] SeongKu Kang、Dongha Lee、Wonbin Kweon 和 Hwanjo Yu。2022。面向推荐系统的个性化知识蒸馏。《Knowledge-Based Systems》239（ 2022 107958 10 1016 2021 107958 ），。[21] https://doi.org/。/j.knosys.。。M Kumar、Benjamin Packer 和 Daphne Koller。。潜变量模型的自步学习。载于 NeurIPS。[22] Wonbin Kweon、SeongKu Kang 和 Hwanjo Yu。2021。面向 Top-K 推荐系统的双向蒸馏。载于 WWW。[23] xu lan、Xiatian Zhu 和 Shaogang Gong。2018。通过动态原生集成进行知识蒸馏。载于 NeurIPS。[24] Dongha Lee、SeongKu Kang、Hyunjun Ju、Chanyoung Park 和 Hwanjo Yu。2021。为单类协同过滤自举用户和物品表示。载于 SIGIR。[25] Youngjune Lee 和 Kee-Eung Kim。2021。Top-N 推荐系统中排序蒸馏的双重校正策略。载于 CIKM。[26] Dawen Liang、Rahul G. Krishnan、Matthew D. Hoffman 和 Tony Jebara。2018。用于协同过滤的变分自编码器。载于 WWW。[27] Weiwei Liu、Ivor W Tsang 和 Klaus-Robert Müller。2017。面向多类别和多标签的由易到难学习范式。《机器学习研究期刊》18（2017）。[28] Yuang Liu、Wei Zhang 和 Jun Wang。2020。自适应多教师多层知识蒸馏。《Neurocomputing》415（2020），106–113。[29] Sean MacAvaney、Franco Maria Nardini、Raffaele Perego、Nicola Tonellotto、Nazli Goharian 和 Ophir Frieder。2020。面向开放域答案重排序的训练课程。载于 SIGIR。529–538。[30] John I Marden。1996。《排序数据的分析与建模》。CRC出版社。[31] Seyed Iman Mirzadeh、Mehrdad Farajtabar、Ang Li、Nir Levine、Akihiro Matsukawa 和 Hassan Ghasemzadeh。2020。通过教师助理改进知识蒸馏。载于 AAAI。5191–5198。[32] Samuel EL Oliveira、Victor Diniz、Anisio Lacerda、Luiz Merschmanm 和 Gisele L Pappa。2020。排序聚合在推荐系统中是否有效？一项实验分析。《ACM智能系统与技术汇刊》（TIST）11卷2期（2020），1–26。[33] German I. Parisi、Ronald Kemker、Jose L. Part、Christopher Kanan 和 Stefan Wermter。2019。神经网络的持续终身学习：综述。《Neural Networks》113（2019）。https://doi.org/10.1016/j.neunet.2019.01.012[34] Sashank Reddi、Rama Kumar Pasumarthi、Aditya Menon、Ankit Singh Rawat、Felix Yu、Seungyeon Kim、Andreas Veit 和 Sanjiv Kumar。2021。Rankdistil：面向排序的知识蒸馏。载于 AISTATS。PMLR。[35] Steffen Rendle 和 Christoph Freudenthaler。2014。利用隐式反馈改进物品推荐的成对学习。载于 WSDM。[36] 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。[37] Adriana Romero、Nicolas Ballas、Samira Ebrahimi Kahou、Antoine Chassang、Carlo Gatta 和 Yoshua Bengio。2014。Fitnets：轻量深度网络的提示。arXiv 预印本 arXiv:1412.6550（2014）。[38] Suvash Sedhain、Aditya Krishna Menon、Scott Sanner 和 Lexing Xie。2015。Autorec：自编码器与协同过滤相结合。载于 WWW。111–112。[39] Wenxian Shi、Yuxuan Song、Hao Zhou、Bohan Li 和 Lei Li。2021。循着你的路径：一种知识蒸馏的渐进式方法。载于欧洲机器学习与数据库知识发现联合会议。Springer，596–611。[40] Jiaxi Tang 和 Ke Wang。2018。排序蒸馏：学习高性能紧凑型排序模型以用于推荐系统。载于 KDD。[41] Haoyu Wang、Defu Lian 和 Yong Ge。2019。通过蒸馏图卷积网络实现二值化协同过滤。载于 IJCAI。[42] Kai Wang、Yu Liu、Qian Ma 和 Quan Z Sheng。2021。Mulde：面向低维知识图谱嵌入的多教师知识蒸馏。载于 WWW。1716–1726。[43] X. Wang、Y. Chen 和 W. Zhu。2021。《课程学习综述》。《IEEE模式分析与机器智能汇刊》（2021）。https://doi.org/10.1109/TPAMI.2021.3069908[44] Markus Weimer、Alexandros Karatzoglou、Quoc Le 和 Alex Smola。2007。COFI RANK——用于协同排序的最大间隔矩阵分解。载于 NIPS。[45] Chuhan Wu、Fangzhao Wu 和 Yongfeng Huang。2021。一个教师就够了吗？来自多位教师的预训练语言模型蒸馏。载于计算语言学协会论文集：ACL-IJCNLP 2021 Findings。4408–4413。[46] Xiaoxia Wu、Ethan Dyer 和 Behnam Neyshabur。2021。课程学习在何时有效？ICLR（2021）。[47] Fen Xia、Tie-Yan Liu、Jue Wang、Wensheng Zhang 和 Hang Li。2008。学习排序的列表式方法：理论与算法。载于 ICML。[48] Xin Xia、Hongzhi Yin、Junliang Yu、Qinyong Wang、Guandong Xu 和 Quoc Viet Hung Nguyen。2022。利用自监督知识蒸馏实现设备端下一物品推荐。载于 SIGIR。[49] Liuyu Xiang、Guiguang Ding 和 Jungong Han。2020。从多位专家中学习：用于长尾分类的自步知识蒸馏。载于 ECCV。Springer，247–263。[50] Shan You、Chang Xu、Chao Xu 和 Dacheng Tao。2017。从多个教师网络中学习。载于 KDD。1285–1294。[51] Fei Yuan、Linjun Shou、Jian Pei、Wutao Lin、Ming Gong、Yan Fu 和 Daxin Jiang。2021。用于知识蒸馏的强化多教师选择。载于 AAAI。[52] Hansi Zeng、Hamed Zamani 和 Vishwa Vinay。2022。密集检索蒸馏的课程学习。载于 SIGIR。1979–1983。[53] Tianyi Zhou、Shengjie Wang 和 Jeff Bilmes。2020。用于半监督学习的时间一致性自监督。载于 ICML。[54] Jieming Zhu、Jinyang Liu、Weiqi Li、Jincai Lai、Xiuqiang He、Liang Chen 和 Zibin Zheng。2020。通过知识蒸馏实现集成CTR预测。载于 CIKM。[55] Qingqing Zhu、Xiuying Chen、Pengfei Wu、JunFei Liu 和 Dongyan Zhao。2021。结合课程学习与知识蒸馏进行对话生成。载于 EMNLP。[56] Ziwei Zhu、Jianling Wang 和 James Caverlee。2019。通过联合协同自编码器改进 Top-k 推荐。载于 WWW。

A 附录

A.1 实验设置

HetComp 的源代码可通过作者的 GitHub 仓库公开获取[8]。

A.1.1 数据集。 我们使用三个真实世界数据集：Amazon-music (Amusic)、CiteULike 和 Foursquare。这些数据集均可公开访问，也广泛应用于以往的研究 [15, 18, 22, 24, 25]。我们遵循 [24] 的预处理方法（CiteULike、Foursquare），并对 Amusic 应用 10核过滤。表 7 给出了数据统计。

表 7：数据集统计。

数据集	用户数	物品数	交互数	密度
Amazon-music	5,729	9,267	65,344	0.001231
CiteULike	5,219	25,181	125,580	0.000956
Foursquare	19,465	28,593	1,115,108	0.002004

A.1.2 实验细节。 对于所有实验和推理，我们使用 PyTorch，并通过 CUDA 在 RTX A5000 和 Xeon Gold 6226R 中央处理器上运行。我们报告五次独立运行的平均值。对于所有基线方法，我们使用作者提供的公开实现。然而，与 [52], 中的做法一样，我们发现其对排名靠前的未观测物品（即 P^-）进行的采样过程是不必要的，移除这些过程后，排序匹配知识蒸馏方法的性能得到了显著提升。因此，在我们的实验中，我们移除了所有排序匹配方法中的采样过程。在 D@K 中，控制指数函数尖锐度的 λ 被设为 10。对于动态知识构建，我们使用 D@50，将 p 设为 10，并将 α 设为 1。05。随着学生模型逐渐收敛，我们采用一种简单的退火方案，每经过 p 个训练轮次就降低 $\alpha = \alpha \times 0.995$ 的值。我们将 E 设为 4，分别对应于每位教师达到收敛训练轮次的 25%、50%、75% 和 100% 时的检查点。最后，将 $|P^-|$ 设为 50。对于基线方法特定的超参数，我们在原论文建议的范围内进行调优。

A.2 教师模型集成研究

A.2.1 教师推荐模型。 在本研究中，我们使用了六种具有不同架构和学习目标的推荐模型。这些模型是各模型类型的代表，并展现出了有竞争力的性能。

• **矩阵分解** (BPR [36])：一种基于矩阵分解、通过成对排序损失训练的模型。排序分数由用户和物品的潜在因子的内积定义。

• **机器学习** (CML [12])：一种基于度量学习、通过成对 hinge 损失训练的模型。排序分数由单位球度量空间中的欧氏距离定义。

• **深度神经网络** (NeuMF [10])：一种基于深度神经网络、通过二元交叉熵训练的模型。排序分数由多层感知机的非线性函数计算得出。

• **GNN**（LightGCN [9]）：一种基于图神经网络的模型，使用成对损失进行训练。排序分数通过聚合多个 GNN 层中的用户和物品表示来计算。• **自编码器** （VAE [26]）：一种基于变分自编码器的模型。排序分数由生成模块（即解码器）计算。

• **I-AE**：VAE 的一种变体，用于学习物品侧交互 [38, 56]。众所周知，物品侧自编码器捕捉的信息与其用户侧对应模型互为补充 [56]。

使用全部六个模型可获得最佳的集成性能，下一小节将提供支持这一配置的实证证据。

A.2.2 异构教师模型的必要性。 表 8 展示了针对教师模型配置的实证研究。“最佳教师”表示在每个数据集上展现出最佳性能的教师模型。“集成”表示六个异构模型的集成结果。“Ensemble-id”表示六个具有不同初始化的相同模型的集成[9]。“w/o model”表示排除“集成”中某个模型后的五个模型集成。

首先，集成的性能始终高于 Ensemble-id 和最佳教师模型。我们研究了集成与 Ensemble-id 的模型预测相关性[10]。与后者相比，前者的预测相关性分别低 23%（Amusic）、20%（CiteULike）和 38%（Foursquare）。众所周知，高多样性的模型能够提升整体集成准确率，而集成较低的相关性在一定程度上支持了其更高的性能。这一观察结果也与近期关于集成 [32] 和多教师知识蒸馏 [8, 45, 51] 的研究一致，后者表明模型的多样性是影响性能的关键因素。最后，集成通过整合全部六个模型实现了最佳性能；与集成相比，所有由五个模型组成的情况（即 w/o model）都表现有限。

A.2.3 集成技术。 下面介绍我们为 g 排序知识构建所选择的集成函数。由于每个教师模型对某些用户-物品交互的预测优于其他模型，因此有选择地将它们的知识反映到集成中至关重要。我们注意到，模型预测的一致性是揭示预测 [53]可靠性的关键因素。这一因素已成功应用于 RS，以获得可靠的负样本 [7] ，以及整合多任务学习模型的多个头 [19]。虽然可以考虑更复杂的技术，但根据经验，我们所选择的方法已获得令人满意的性能。我们在实验中将其与使用可训练重要性的技术进行了比较。

g 通过整合一组排列 $\Pi = [\pi^1, \pi^2, ..., \pi^M]$生成集成排序 πd。对于每个排列π^x的前-K项，每个排序预测 r（π^x， i）具有一个重要性分数 c_i^x：

$$c_i^x = \exp(-r(\pi^x, i)/\lambda) + \exp(-\operatorname{std}[r(\pi^x, i)]/\lambda) \tag{8}$$

第一项会为排序更高的物品赋予更高的分数，第二项则利用预测的方差，偏好模型能够做出一致预测的物品。按照 [7], , 我们根据最近 5 个训练轮次的方差定义一致性：

$$\begin{aligned} \operatorname{std}\left[r(\pi^x, i)\right] &= \sqrt{\sum_{s=t-4}^{t} \left([r(\pi^x, i)]_s - \operatorname{Mean}\left[r(\pi^x, i)\right]\right)^2 /5}, \\ \operatorname{Mean}\left[r(\pi^x, i)\right] &= \sum_{s=t-4}^{t} \left[r(\pi^x, i)\right]_s /5. \end{aligned} \tag{9}$$

[8] https://github.com/SeongKu-Kang/HetComp_WWW23

[9] 我们使用在每个数据集上表现最佳的模型。[10] 我们比较每个模型前 50 名排序中包含的测试交互。我们计算每个用户的所有模型对之间的相关性，并比较平均值。我们使用 sklearn 提供的马修斯相关系数（MCC）。

表 8：集成研究。使用六个异构模型可获得最佳性能。

方法	Amusic				CiteULike				Foursquare			
	R@10	N@10	R@50	N@50	R@10	N@10	R@50	N@50	R@10	N@10	R@50	N@50
最佳教师	0.0972	0.0706	0.2475	0.1139	0.1337	0.0994	0.2844	0.1392	0.1147	0.1085	0.2723	0.1635
集成	0.1096	0.0820	0.2719	0.1273	0.1550	0.1156	0.3144	0.1571	0.1265	0.1213	0.2910	0.1730
Ensemble-id	0.1013	0.0736	0.2569	0.1181	0.1511	0.1130	0.2952	0.1505	0.1215	0.1174	0.2853	0.1709
不含矩阵分解	0.1091	0.0826	0.2634	0.1266	0.1545	0.1161	0.3128	0.1564	0.1206	0.1124	0.2883	0.1708
不含机器学习	0.1050	0.0798	0.2574	0.1232	0.1541	0.1137	0.3085	0.1541	0.1222	0.1181	0.2854	0.1746
不含深度神经网络	0.1074	0.0819	0.2642	0.1263	0.1543	0.1147	0.3134	0.1560	0.1193	0.1099	0.2855	0.1678
不含图神经网络	0.1085	0.0823	0.2634	0.1259	0.1544	0.1150	0.3117	0.1557	0.1197	0.1124	0.2857	0.1703
不含自编码器	0.1032	0.0776	0.2520	0.1204	0.1482	0.1101	0.3046	0.1508	0.1177	0.1102	0.2775	0.1658
不含 I-AE	0.1040	0.0797	0.2613	0.1243	0.1519	0.1129	0.3046	0.1528	0.1204	0.1150	0.2812	0.1710

其中，$[r(\pi^x, i)]_s$表示训练轮次 s 的排序预测。最后，根据总体重要性对物品进行重新排序，即可生成集成排序，即 $\mathbb{E}_x[c_i^x]$。请注意，一致性仅在蒸馏之前预先计算一次，对于 HetComp 不会产生额外开销。

A.3 HetComp 的离线训练成本

HetComp 的额外空间和计算成本主要用于知识构建过程。对于教师模型的训练轨迹 T，我们存储排名靠前的 (K) 个未观测物品的排列。请注意，其余未观测物品的排序并非必要。此外，我们还使用已观测项目的排列。总之，HetComp 在每个教师模型上为用户使用 $(K \times E) + |P_u^+|$ 空间。K 和 $|P_u^+|$ 在许多推荐场景中通常取值较小；只要分布合理，当 E 约为 3 时，我们在经验上也能获得令人满意的结果（图5）。

表9：离线训练期间每个训练轮次的平均时间开销。

方法	Amusic	CiteULike	Foursquare
RRD	0.69秒	2.27秒	8.12秒
DCD	1.38秒	3.74秒	12.01秒
HetComp	1.15秒	3.08秒	11.81秒

表9展示了 HetComp 和排序匹配知识蒸馏基线方法的平均时间开销。我们使用 MF 作为学生模型，并且在其他基础模型上也观察到了类似结果。与采用基础列表式学习的 RRD 相比，DCD 和 HetComp 需要进行额外的蒸馏计算，因此会产生额外的训练开销。不过，与需要在整个训练过程中进行此类计算的 DCD 不同，我们的知识构建主要发生在学生模型训练的早期阶段。因此，与 DCD 相比，HetComp 的平均时间开销更低。此外，两者表现出相似的收敛行为。如第5节所示，HetComp 能够通过缩小与教师模型之间的差异，显著提升蒸馏质量。就此而言，HetComp 可视为一种有效方案：只需在离线训练中增加一些计算，即可降低在线推理成本。

A.4 补充实验结果

消融实验。 表10展示了与多种消融方案的比较。我们报告了矩阵分解（学生模型）在CiteULike 数据集上的结果。首先，(a-b)表明，我们为由易到难学习设计的两个组件（即 DKC 和 ADO）能够有效提升学生模型。此外，我们还比较了传递已观测项目知识的不同方式（P^+）；(c)表明，使用 P^+ 确实有助于提升学生模型；而(d)对应于直接传递整个项目排列的朴素方法，结果表明如第4.2.2节所述，未将 P^+ 和 P^- 分开进行的排序迁移会产生负面影响。最后，(e)表明，惩罚所有未观测物品，使其排序低于已观测项目，并不起作用。这些结果证明了我们分别迁移 P^+ 和 P^-这一策略的优越性。

表10：消融实验。DKC：动态知识构建（第4.1节），ADO：自适应蒸馏目标（第4.2.1节）。

消融实验 R@10N@10R@50 N@50 提升。R@10		
异构组合 0.13790.1031 0.2814 0.1396 -		
(a) 去除 DKC（即来自集成的知识蒸馏）	0.1264 0.0951 0.2711 0.1308	9.10%
(b) 去除 ADO（即仅 L_F）	0.1311 0.0992 0.2765 0.1360	5.19%
(c) 不含 $L_{KD}(P^+, N)$	0.1303 0.0994 0.2754 0.1349	5.83%
（d）$L_{KD}(\{P^+, P^-\}, N)$	0.1241 0.0903 0.2796 0.1314	11.12%
(e) $L_{KD}(P^+, \{P^-, N\}) + L_{KD}(P^-, N)$	0.1262 0.0950 0.2665 0.1310	9.27%

超参数研究。 图5展示了 HetComp（学生模型：MF）在不同 α 下的推荐性能，该参数控制动态知识构建中的转换速度。在两个数据集上，最佳性能均出现在约 1.03–1.05 处。

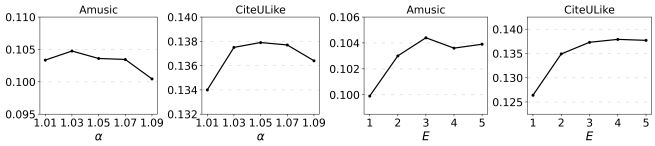

图5：不同 α 和 E下的 R@10 对比。

来自不同教师模型的知识蒸馏。 表11展示了从不同教师模型迁移知识时 DCD 和 HetComp 的性能。我们使用 MF（学生模型）和 CiteULike 数据集。我们观察到，从（a-b）同构模型蒸馏，到（c-d）跨模型蒸馏，再到（e）异构模型集成蒸馏，HetComp 在所有设置下都能有效提升蒸馏质量，并取得最佳推荐性能。

表11：与不同教师模型的性能比较。

	R@10	N@10	R@50	N@50	D@10
(a) 教师：矩阵分解	0.1249	0.0915	0.2604	0.1273	-
DCD	0.0937	0.0698	0.2206	0.1044	0.3831
HetComp	0.1092	0.0794	0.2373	0.1128	0.3247
(b) 教师：集成（矩阵分解）	0.1395	0.1037	0.2763	0.1395	-
DCD	0.1004	0.0749	0.2307	0.1088	0.4044
HetComp	0.1194	0.0878	0.2584	0.1237	0.3286
（c）教师：LightGCN	0.1337	0.0994	0.2844	0.1392	-
DCD	0.1041	0.0778	0.2367	0.1120	0.4164
HetComp	0.1228	0.0896	0.2510	0.1230	0.3590
（d）教师：集成（LightGCN）	0.1511	0.1130	0.2952	0.1505	-
DCD	0.1108	0.0838	0.2495	0.1186	0.4180
HetComp	0.1305	0.0973	0.2630	0.1255	0.3377
(e) 教师：集成	0.1550	0.1156	0.3144	0.1571	-
DCD	0.1106	0.0851	0.2640	0.1246	0.4433
HetComp	0.1379	0.1031	0.2814	0.1396	0.3401