南京航空航天大學
NANJING UNIVERSITY OF AERONAUTICS AND ASTRONAUTICS

组会汇报

2025年4月21日

论文

LLMEmb: Large Language Model Can Be a Good Embedding Generator for Sequential Recommendation

Qidong Liu[1, 2], **Xian Wu**[3 *], **Wanyu Wang**[2], **Yejing Wang**[2], **Yuanshao Zhu**[2], **Xiangyu Zhao**[2 *], **Feng Tian**[4 *], **Yefeng Zheng**[3, 5]

[1]School of Auto. Science & Engineering, MOEKLINNS Lab, Xi'an Jiaotong University
[2]City University of Hong Kong
[3]Jarvis Research Center, Tencent YouTu Lab
[4]School of Comp. Science & Technology, MOEKLINNS Lab, Xi'an Jiaotong University
[5]Medical Artificial Intelligence Lab, Westlake University
liuqidong@stu.xjtu.edu.com, {kevinxwu, yefengzheng}@tencent.com, {wanyuwang4-c, yejing.wang}@my.cityu.edu.hk
yuanshao@ieee.org, xianzhao@cityu.edu.hk, fengtian@mail.xjtu.edu.cn

AAAI 2025

序列推荐系统SRS

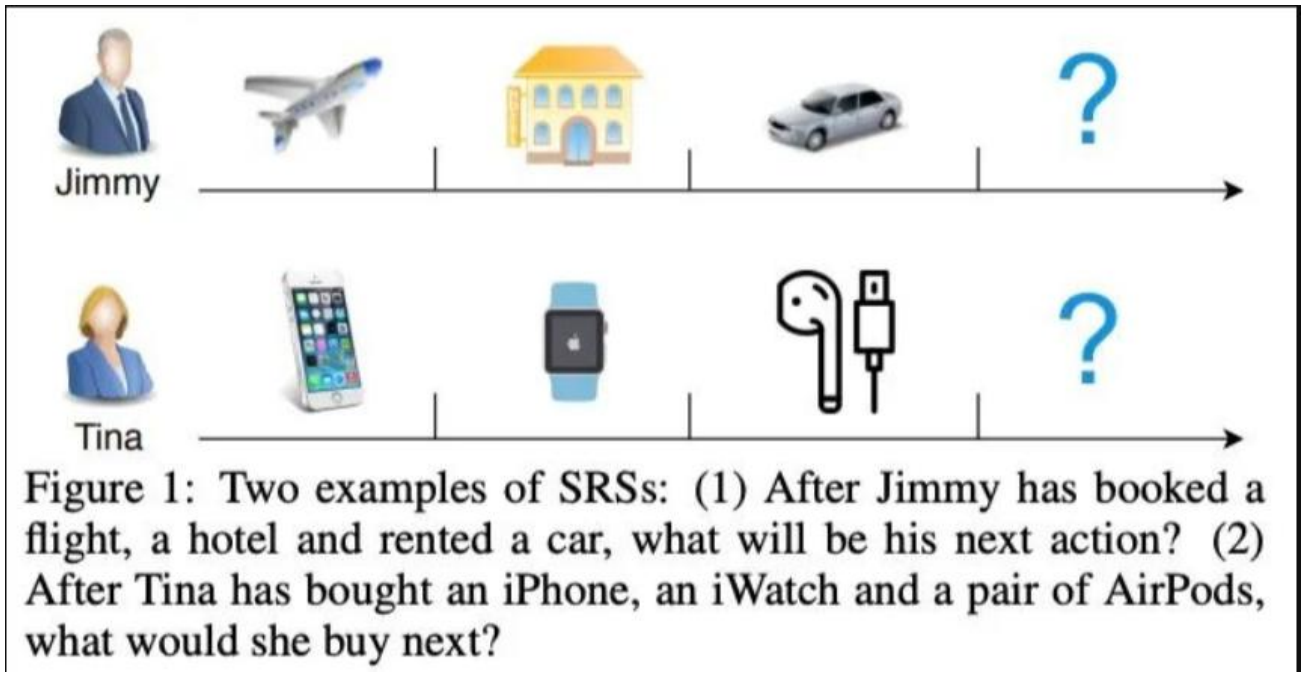


Figure 1: Two examples of SRSs: (1) After Jimmy has booked a flight, a hotel and rented a car, what will be his next action? (2) After Tina has bought an iPhone, an iWatch and a pair of AirPods, what would she buy next?

序列推荐系统与传统的推荐系统（协同过滤、基于内容的过滤）不同，传统的推荐系统,例如基于内容和协同过滤的推荐系统，以一种静态的方式建模用户和商品的交互并且只可以捕获用户广义的喜好。而相反地，SRSs则是将用户和商品的交互建模为一个动态的序列并且利用序列的依赖性来活捉当前和最近用户的喜好。

序列推荐系统SRS

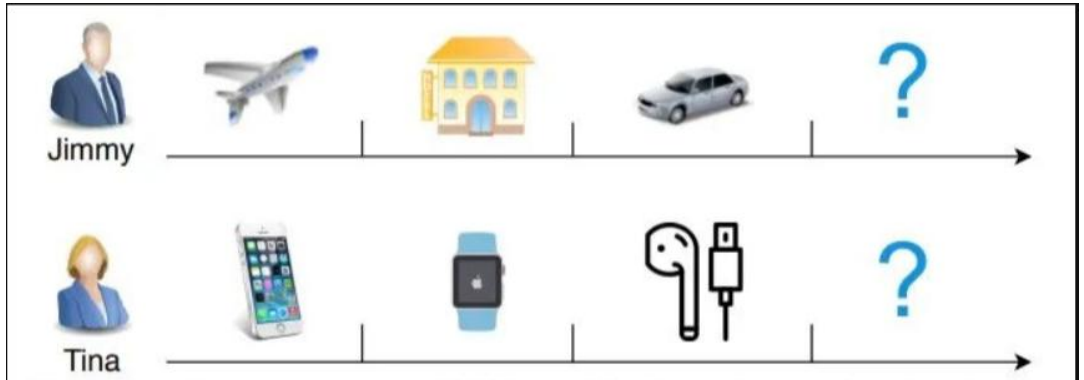


Figure 1: Two examples of SRSs: (1) After Jimmy has booked a flight, a hotel and rented a car, what will be his next action? (2) After Tina has bought an iPhone, an iWatch and a pair of AirPods, what would she buy next?

用户跟物品之间的互动是序列依赖的

比如在左图中，Jimmy首先买了机票，然后准备订酒店，这时他定酒店的行为基本和他买机票的行为相关，jimmy可能会选择一个离机场不远的酒店。订完酒店以后，他租车的行为就会受到订酒店的影响，Jimmy很有可能选择一个取车地点离酒店比较近的租车行。所以，在这一系列互动中，Jimmy每一个行为都依赖于之前产生的行为。这种依赖关系在交易数据中是非常常见的。

用户的兴趣与物品的流行程度都是动态变化的

这很容易理解，就比如用户以前喜欢用iPhone，现在喜欢用华为；诺基亚的手机以前非常流行，现在却已经很少见了……这些动态的变化只有依靠SRS才能够有效地捕捉。

用户-物品的互动通常在特定的序列上下文环境中产生

不同的上下文会催生出不同的用户行为。就比如说在刷抖音的时候，如果突然刷到一个很感兴趣的视频，我很有可能会给它点赞，但是如果我点赞了一些差不多的视频后，还继续给我推荐，那我很可能因为腻烦直接划走（未点赞或观看时间很短）。因此SRS在丰富推荐结果、避免同质化的任务上比传统推荐系统更容易做到。

序列推荐系统SRS

$$R = argmax f(S)$$

其中:

- f:表示效用函数(utility function),输出关于候选商品的排序分数，它可以是多种形式，例如条件概率等。
- $S = \{i_1, i_2, \ldots, i_{|S|}\}$:是用户和商品的交互序列，其中每一次交互$i_j = 是一个三元组，u表示用户，a表示用户的行为,例如点击,加购等,v$表示对应的商品。
- R:表示基于排序的分数阐述的商品序列;

LLMEmb——背景

序列推荐系统（SRS）的目标是根据用户的历史交互记录预测用户可能感兴趣的下一个项目。尽管SRS在准确性上取得了显著进展，但长尾问题仍然是一个关键挑战。长尾问题指的是大多数项目由于交互频率低而难以被推荐，这不仅降低了用户的体验，也减少了商家的利润。现有的SRS模型在处理低流行度项目时表现不佳，主要是因为这些项目的嵌入分布稀疏且与高流行度项目距离较远。

大型语言模型（LLM）具有捕捉项目之间语义关系的能力，不受项目流行度的影响，因此有潜力解决长尾问题。然而，直接将LLM应用于推荐任务存在两个主要问题：**语义鸿沟（Semantic Gap）和语义损失（Semantic Loss）**。语义鸿沟指的是LLM的通用性与推荐任务之间的差异，而语义损失则是由于在将LLM嵌入适应到SRS模型时，可能会丢失原始LLM嵌入中的语义信息。

长尾问题（流行度）

↓

LLM捕获语义关系的能力
不受项目流行度影响

↓

LLM应用于推荐的挑战
（本文解决的问题）

- **语义鸿沟（Semantic Gap）**
- **语义损失（Semantic Loss）**

LLMEmb

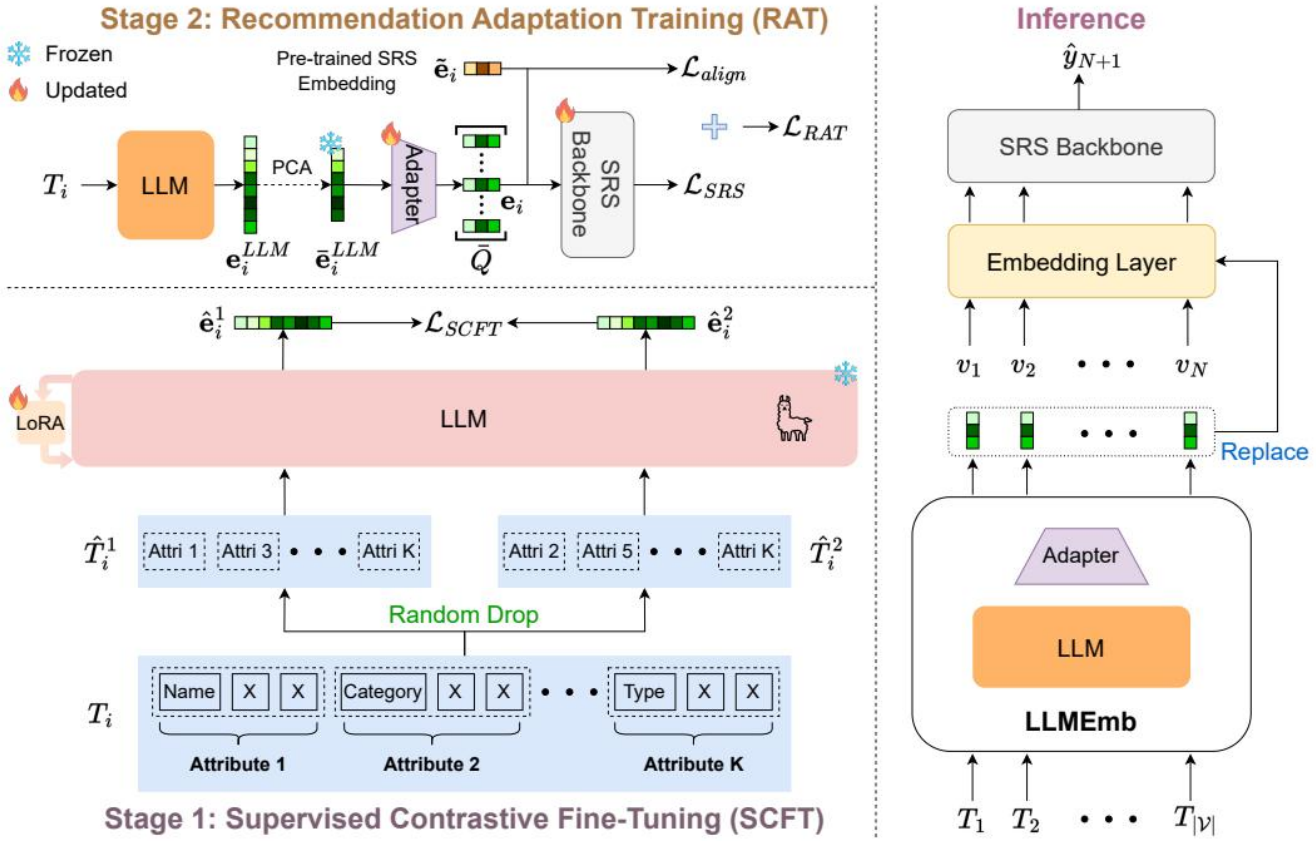
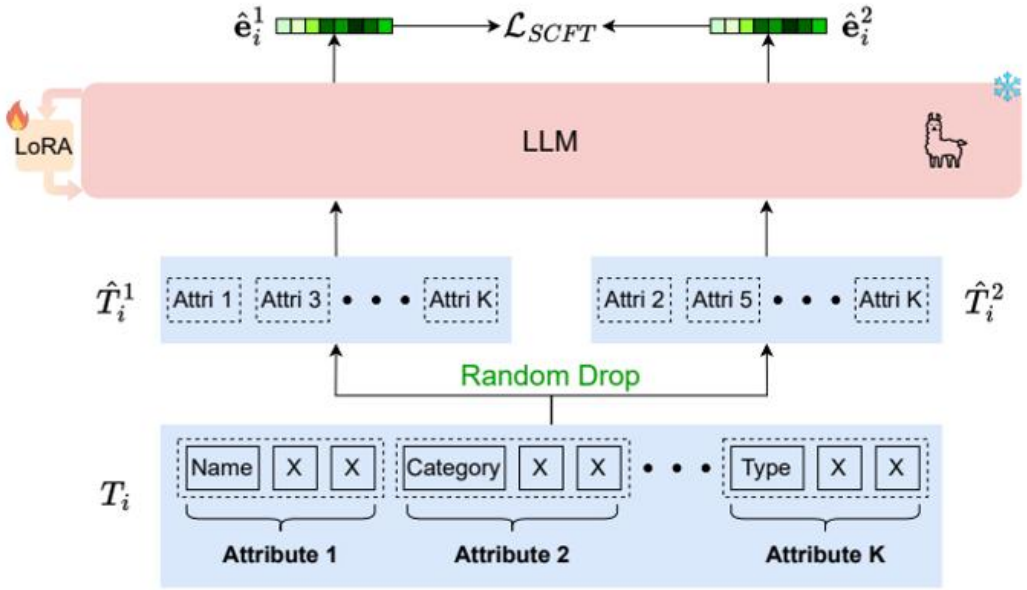


Figure 3: The overview of the proposed LLMEmb.

- **LLMEmb（LLM Embedding Generator）**：
  - 利用LLM生成高质量的项目嵌入。
  - 两阶段训练：
    - 监督对比微调（SCFT）：使LLM适应推荐任务。
    - 推荐适应训练（RAT）：将LLM嵌入与协同信号结合。
  - 无缝集成到任何SRS模型中。

LLMEmb——SCFT



Stage 1: Supervised Contrastive Fine-Tuning (SCFT)

阶段1：监督对比微调（SCFT）

目标： 调整LLM以更好地捕捉项目之间的语义关系，同时考虑到推荐领域的特定需求。

方法：

1. **提示构建（Prompt Construction）：** 根据项目的属性（如名称、类别等）构建文本提示，使LLM能够从语义角度理解项目。
2. **数据增强（Data Augmentation）：** 通过随机丢弃部分属性生成两个不同的提示副本，作为正样本对，强调属性对项目语义的影响。
3. **对比学习（Contrastive Learning）：** 使用对比损失函数优化LLM，使得同一项目的两个不同提示生成的嵌入更加接近，而不同项目的嵌入更加远离。这有助于提高嵌入的区分能力和均匀性。

效果： 通过SCFT，LLM能够生成更符合推荐任务需求的嵌入，这些嵌入不仅包含丰富的语义信息，而且能够更好地区分不同的项目。

$$\mathcal{L}_{CL}^1 = -\frac{1}{B}\sum_{i=1}^{B}\log\frac{\exp(\text{sim}(\hat{\mathbf{e}}_i^1, \hat{\mathbf{e}}_i^2)/\tau)}{\sum_{k=1}^{B}\mathbb{I}_{[i\neq k]}\exp(\text{sim}(\hat{\mathbf{e}}_i^1, \hat{\mathbf{e}}_k^2)/\tau)}$$

$$\mathcal{L}_{SCFT} = \mathcal{L}_{CL}^1 + \mathcal{L}_{CL}^2$$

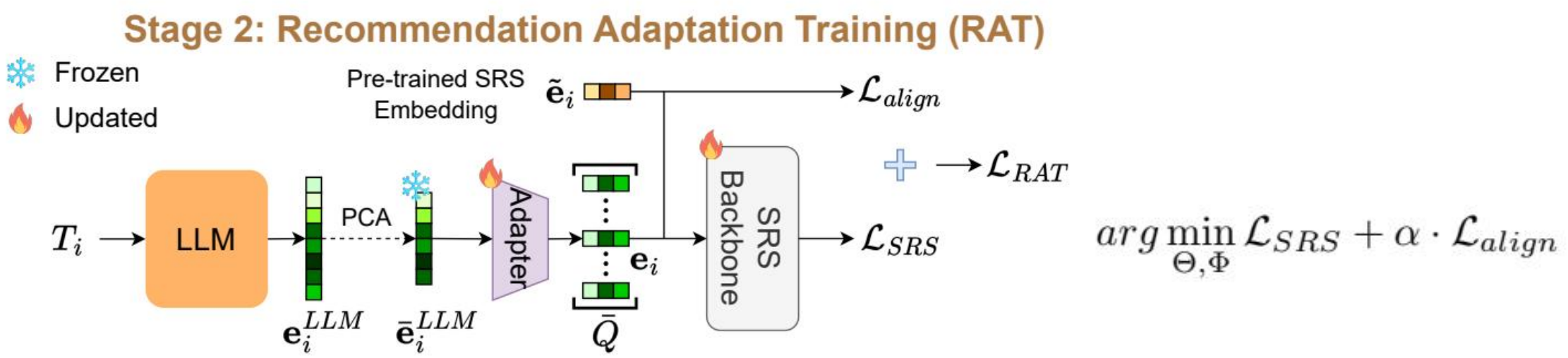
T_i：表示项目 i 的原始属性集合，包括名称（Name）、类别（Category）、类型（Type）等。

Random Drop：随机丢弃过程，从原始属性集合 T_i 中随机移除一些属性，以生成两个不同的提示 $\hat{T}_i^1$ 和 $\hat{T}_i^2$。这个过程用于数据增强，帮助模型学习到更加鲁棒的嵌入表示。

$\hat{T}_i^1$ **和** $\hat{T}_i^2$：经过随机丢弃后生成的两个不同的提示，它们代表了项目 i 的两种不同描述。

$\hat{e}_i^1$ **和** $\hat{e}_i^2$：LLM处理两个不同提示后生成的两个嵌入向量。这两个嵌入向量用于对比学习，目的是使同一项目的两个不同描述生成的嵌入更加接近。

LLMEmb——RAT



阶段2：推荐适应训练（RAT）

目标： 将LLM生成的嵌入与推荐系统中的协同信号相结合，解决维度不匹配问题，并优化嵌入以适应推荐模型。

方法：

1. **嵌入转换（Embedding Transformation）：** 使用PCA将LLM嵌入的维度降低到中间大小，然后通过一个可训练的适配器进一步转换为适合SRS模型的维度。
2. **适应训练（Adaptation）：** 将转换后的嵌入与SRS模型的主干网络结合，并通过SRS模型的损失函数进行优化。在这个过程中，LLM嵌入被冻结，以保留原始语义信息。
3. **协同对齐（Collaborative Alignment）：** 通过对比学习，将LLM嵌入与预训练的SRS嵌入对齐，从而引入协同信号并优化嵌入分布。

效果： 通过RAT，LLM生成的嵌入不仅保留了原始的语义信息，还融入了协同信号，从而在推荐任务中表现出更好的性能。此外，适配器的设计使得嵌入的维度与推荐模型兼容，进一步提升了嵌入的适用性。

T_i：表示第 i 个项目的文本提示（prompt）。

e_i^{LLM}：表示通过大型语言模型（LLM）生成的第 i 个项目的初始嵌入向量。

$\bar{e}_i^{LLM}$：表示经过主成分分析（PCA）降维处理后的第 i 个项目的LLM嵌入向量。

$\tilde{e}_i$：表示预训练的序列推荐系统（SRS）中第 i 个项目的嵌入向量。

e_i：表示经过适配器（Adapter）调整后的第 i 个项目的最终嵌入向量。

$\mathcal{L}_{align}$：表示对齐损失（Alignment Loss），用于使适配器生成的嵌入 e_i 与预训练的SRS嵌入 $\tilde{e}_i$ 对齐。

$\mathcal{L}_{SRS}$：表示序列推荐系统（SRS）的主损失函数，用于优化SRS模型。

$\mathcal{L}_{RAT}$：表示推荐适应训练（Recommendation Adaptation Training, RAT）的总损失函数，通常是对齐损失和SRS主损失的组合。

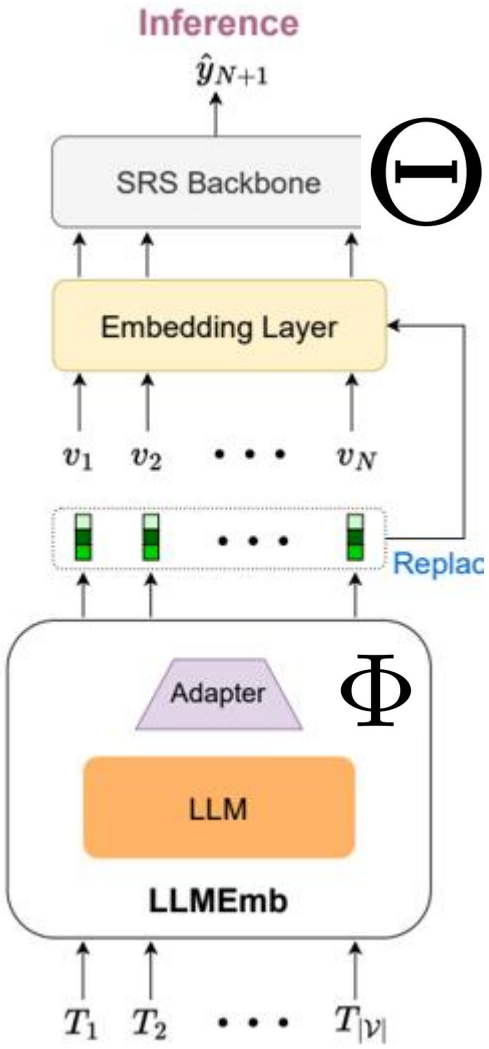
LLMEmb——Training&Interface

SCFT $\mathcal{L}_{SCFT} = \mathcal{L}_{CL}^{1} + \mathcal{L}_{CL}^{2}$ → $\{\mathbf{A}_i, \mathbf{B}_i\}_{i=1}^{M}$

LoRA伴随的层的低秩矩阵

RAT $\arg\min_{\Theta,\Phi} \mathcal{L}_{SRS} + \alpha \cdot \mathcal{L}_{align}$

其中Θ代表SRS（序列推荐系统）主干网络的参数，$\Phi = \{W_1, W_2, b_1, b_2\}$是适配器的参数。超参数$\alpha$控制对齐的强度。



推理：

一个通用的SRS由SRS主干网络和嵌入函数组成。

在推理过程中，使用从RAT阶段训练获得的SRS主干网络（即参数Θ）进行序列处理。

对于嵌入函数，我们为所有项目生成LLM嵌入，使用它们的文本提示T_i，然后将降维后的e^{-}LLM输入适配器来生成嵌入e_i，将这些生成的嵌入替换了嵌入层的权重，有效地充当嵌入组件。

这种方法在推理过程中与传统SRS模型相比没有引入额外的计算。

LLMEmb——Experiment

1. 数据集

- 使用了三个真实世界的数据集：**Yelp**、**Amazon Beauty** 和 **Amazon Fashion**。
  - **Yelp**：包含大量的签到记录，用于兴趣点推荐。
  - **Amazon**：从电子商务平台收集，Beauty 和 Fashion 是其两个子类别。
  - 数据预处理遵循了之前序列推荐系统（SRS）工作的标准方法（Kang and McAuley, 2018）

2. 序列推荐模型

- 提出的 LLMEmb 方法是模型无关的（model-agnostic），可以集成到多种 SRS 模型中。
- 为了验证其通用性，实验中选择了以下三种经典的 SRS 模型：
  - **GRU4Rec**（Hidasi et al., 2016）：基于 GRU 的序列推荐模型。
  - **Bert4Rec**（Sun et al., 2019）：基于 BERT 的序列推荐模型。
  - **SASRec**（Kang and McAuley, 2018）：基于自注意力机制的序列推荐模型。

3. 基线方法

- 为了验证 LLMEmb 的有效性，实验中选择了以下基线方法：
  - **MELT**（Kim et al., 2023）：一种最新的长尾序列推荐方法。
  - **LLM2X**（Harte et al., 2023）、**SAID**（Hu et al., 2024）和 **TSLRec**（Liu et al., 2024a）：三种基于大型语言模型（LLM）增强的序列推荐方法。

4. 实验环境

- 实验平台：
  - CPU：Intel Xeon Gold 6133
  - GPU：Tesla V100
- 编程环境：
  - Python 3.9.5
  - PyTorch 1.12.0
- 基础模型：
  - 所有基于 LLM 的方法（包括 TSLRec、SAID 和 LLMEmb）均使用 LLaMA-7B 2023）作为基础模型。

LLMEmb——Experiment

Backbone	Model	Yelp				Fashion				Beauty			
		Overall		Tail		Overall		Tail		Overall		Tail	
		H@10	N@10	H@10	N@10	H@10	N@10	H@10	N@10	H@10	N@10	H@10	N@10
GRU4Rec	- None	0.4879	0.2751	0.0171	0.0059	0.4798	0.3809	0.0257	0.0101	0.3683	0.2276	0.0796	0.0567
	- MELT	0.4985	0.2825	0.0201	0.0079	0.4884	0.3975	0.0291	0.0112	0.3702	0.2161	0.0009	0.0003
	- LLM2X	0.4872	0.2749	0.0201	0.0072	0.4881	0.4100	0.0264	0.0109	0.4151	0.2713	0.0896	0.0637
	- SAID	0.4891	0.2764	0.0180	0.0062	0.4920	0.4168	0.0347	0.0151	0.4193	0.2621	0.0936	0.0661
	- TSLRec	0.4528	0.2509	0.0255	0.0095	0.4814	0.4042	0.0149	0.0071	0.3119	0.1865	0.0750	0.0474
	- **LLMEmb**	**0.5270***	**0.2980***	**0.1116***	**0.0471***	**0.5062***	**0.4329***	**0.1046***	**0.0477***	**0.4445***	**0.2726**	**0.3183***	**0.1793***
Bert4Rec	- None	0.5307	0.3035	0.0115	0.0044	0.4668	0.3613	0.0142	0.0067	0.3984	0.2367	0.0101	0.0038
	- MELT	0.6206	0.3770	0.0429	0.0149	0.4897	0.3810	0.0059	0.0019	0.4716	0.2965	0.0709	0.0291
	- LLM2X	0.6199	0.3781	0.0874	0.0330	0.5109	0.4159	0.0377	0.0169	0.5029	0.3209	0.0927	0.0451
	- SAID	0.6156	0.3732	0.0973	0.0382	0.5135	0.4124	0.0694	0.0433	0.5127	0.3360	0.1124	0.0664
	- TSLRec	0.6069	0.3680	0.0969	0.0388	0.5078	0.4143	0.0418	0.0182	0.4936	0.3178	0.1013	0.0589
	- **LLMEmb**	**0.6294***	**0.3881***	**0.1876***	**0.1094***	**0.5244***	**0.4238***	**0.1485***	**0.0764***	**0.5247***	**0.3485***	**0.2430***	**0.1224***
SASRec	- None	0.5940	0.3597	0.1142	0.0495	0.4956	0.4429	0.0454	0.0235	0.4388	0.3030	0.0870	0.0649
	- MELT	0.6257	0.3791	0.1015	0.0371	0.4875	0.4150	0.0368	0.0144	0.4334	0.2775	0.0460	0.0172
	- LLM2X	0.6415	0.3997	0.1760	0.0789	0.5210	0.4486	0.0768	0.0473	0.5043	0.3319	0.1608	0.0940
	- SAID	0.6277	0.3841	0.1548	0.0669	0.5316	0.4619	0.0901	0.0540	0.5097	0.3343	0.1549	0.0906
	- TSLRec	0.6152	0.3795	0.1383	0.0620	0.5125	0.4594	0.0652	0.0382	0.4977	0.3366	0.1211	0.0789
	- **LLMEmb**	**0.6647***	**0.4113***	**0.2951***	**0.1456***	**0.5521***	**0.4730***	**0.1513***	**0.0826***	**0.5277***	**0.3460***	**0.4194***	**0.2595***

LLMEmb——Experiment

Dataset	Model	Overall		Tail	
		H@10	N@10	H@10	N@10
Yelp	**LLMEmb**	**0.6647**	**0.4113**	**0.2951**	**0.1456**
	- *w/o* SCFT	0.6538	0.4031	0.2474	0.1218
	- *w/o* adapter	0.6414	0.3968	0.2196	0.1055
	- *w/o* freeze	0.6257	0.3800	0.1710	0.0740
	- *w/o* align	0.6598	0.4060	0.2793	0.1310

LLMEmb——Experiment

Can the proposed LLMEmb alleviate the long-tail problem in SRS?

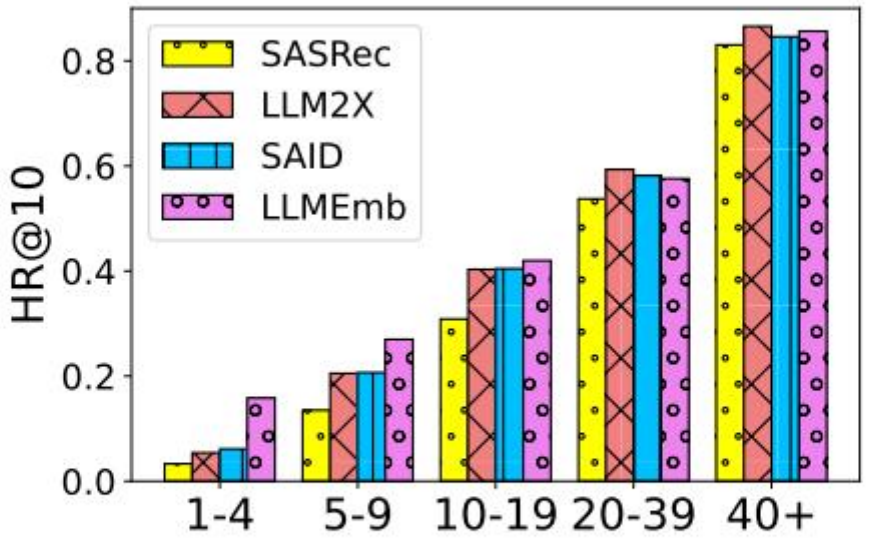


Figure 5: The experimental results of group analysis based on Yelp dataset and SASRec backbone.

LLMEmb——Experiment

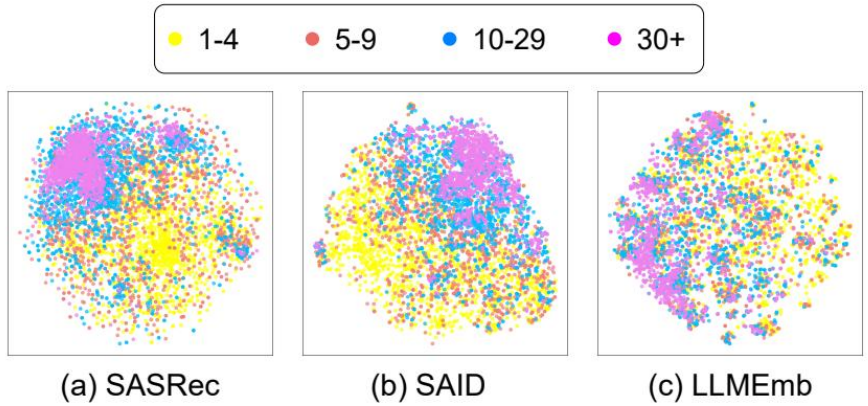
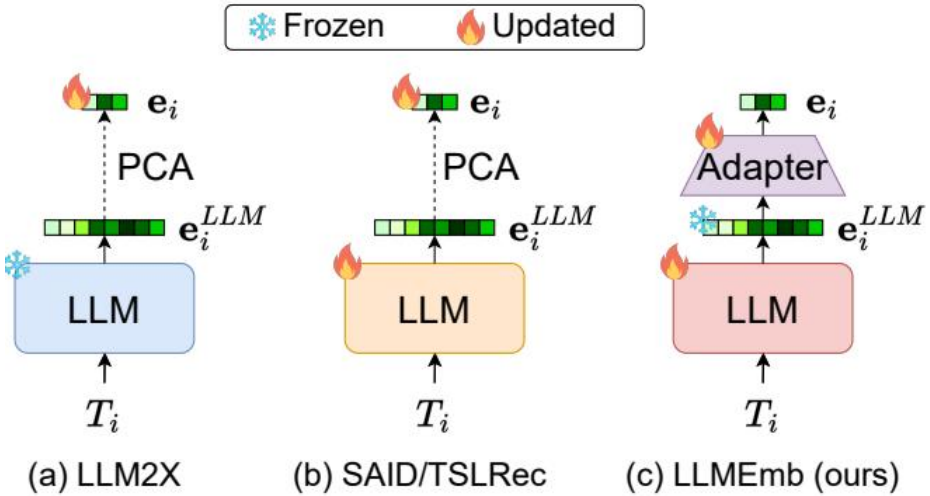


Figure 6: The visualization of embeddings. The LLMEmb and baselines are based on SASRec and the Yelp dataset.



研究问题5:LLMEmb能否纠正嵌入分布的偏斜?

可视化(研究问题5)

为了探究所提出的LLMEmb方法是否能够纠正嵌入分布的偏斜，本文通过t-SNE对分布进行了可视化如图所示。SAID通过引入LLM嵌入能够获得更均匀的分布。然而，由于语义损失问题，它仍然受到项目流行度的影响而聚集。相比之下，本文的LLMEmb获得了更好的嵌入，其分布更加均匀。结果回应了研究问题5，并揭示了LLMEmb的优越性。

南京航空航天大學
NANJING UNIVERSITY OF AERONAUTICS AND ASTRONAUTICS

Thanks