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]

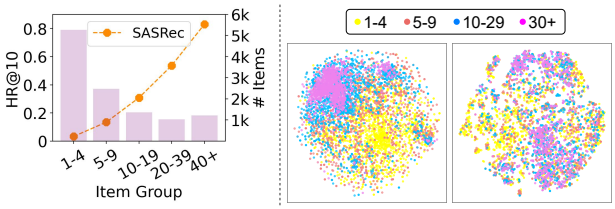
[1]School of Auto. Science & Engineering, MOEKLINNS Lab, Xi'an Jiaotong University
[2]City University of Hong Kong
[3]Jarvis Research Center, Tencent YouTu Lab
[4]School of Comp. Science & Technology, MOEKLINNS Lab, Xi'an Jiaotong University
[5]Medical Artificial Intelligence Lab, Westlake University
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

Abstract

Sequential Recommender Systems (SRS), which model a user's interaction history to predict the next item of interest, are widely used in various applications. However, existing SRS often struggle with low-popularity items, a challenge known as the long-tail problem. This issue leads to reduced serendipity for users and diminished profits for sellers, ultimately harming the overall system. Large Language Model (LLM) has the ability to capture semantic relationships between items, independent of their popularity, making it a promising solution to this problem. In this paper, we introduce **LLMEmb**, a novel method leveraging LLM to generate item embeddings that enhance SRS performance. To bridge the gap between general-purpose LLM and the recommendation domain, we propose a Supervised Contrastive Fine-Tuning (SCFT) approach. This approach includes attribute-level data augmentation and a tailored contrastive loss to make LLM more recommendation-friendly. Additionally, we emphasize the importance of integrating collaborative signals into LLM-generated embeddings, for which we propose Recommendation Adaptation Training (RAT). This further refines the embeddings for optimal use in SRS. The LLMEmb-derived embeddings can be seamlessly integrated with any SRS models, underscoring the practical value. Comprehensive experiments conducted on three real-world datasets demonstrate that LLMEmb significantly outperforms existing methods across multiple SRS models.

Introduction

Sequential recommender systems (SRS) have been extensively applied across various practical scenarios, such as e-commerce (Zhou et al. 2018) and short video (Pan et al. 2023). The primary objective of SRS is to capture users' preferences based on their historical interactions and predict the next most possible item (Fang et al. 2020). To achieve this, many research studies have committed to developing neural network architectures for better modeling user interaction history, *e.g.,* SASRec (Kang and McAuley 2018) and Bert4Rec (Sun et al. 2019).



(a) Long-tail Problem (b) SASRec (c) LLM

Figure 1: The preliminary experiments are conducted for SASRec on the Yelp dataset.

Although the accuracy of SRS has seen continuous improvement, the long-tail problem remains a critical challenge that can undermine the overall user experience. To illustrate this issue, we trained a popular SRS model, SASRec, on the Yelp dataset and grouped the items based on their interaction frequencies. As depicted in Figure 1(a), the histogram reveals that the majority of items have fewer than 5 records, while the corresponding line graph indicates their relatively low performance. This phenomenon highlights the difficulty in effectively recommending long-tail items, which can result in reduced serendipity for users and diminished profits for sellers. Our analysis suggests that the long-tail problem in SRS primarily stems from the skewed distribution of item embeddings. To further investigate this, we visualized the item embedding distribution of SASRec using t-SNE in Figure 1(b). The result confirms that the embeddings of low-popularity items (i.e., 1-4) are sparsely distributed and distant from those of more popular items, indicating the poor quality of these embeddings. In contrast, the Large Language Model (LLM) shows promise in capturing semantic relationships between items through textual features, such as titles. Figure 1(c) shows the item embeddings generated by LLaMA (Touvron et al. 2023) are more uniformly distributed, which motivates the development of an LLM-based generator for producing higher-quality embeddings.

*Corresponding authors: Xian Wu, Xiangyu Zhao, Feng Tian

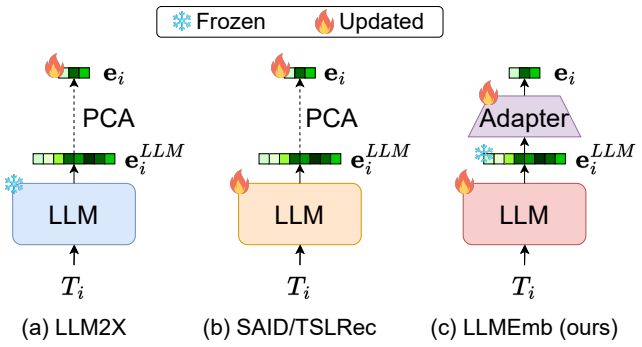


Figure 2: The comparison between the existing LLM enhanced SRS methods and our LLMEmb.

Some recent studies have explored the potential of leveraging LLM to enhance SRS (Harte et al. 2023; Hu et al. 2024; Liu et al. 2024a). However, they encounter two significant challenges. (i) **Semantic Gap**: LLM2X (Harte et al. 2023) adopts the general-purpose LLM to generate item embeddings. While these embeddings can contain the semantics, they are not tailored to the recommendation field. In an effort to address this, methods like SAID (Hu et al. 2024) and TSLRec (Liu et al. 2024a) propose fine-tuning open-sourced LLM to better align with recommendation tasks. However, these approaches remain confined to language modeling or category prediction, overlooking the crucial role of item attributes in distinguishing items within the recommendation field (Hou et al. 2019). (ii) **Semantic loss**: As shown in Figure 2, to further adapt the LLM embeddings to collaborative SRS models, existing methods reduce the dimension and update the embeddings with SRS models directly. However, drastic dimensionality reduction and continual training can result in a significant loss of the original semantic richness contained in LLM embeddings, thereby limiting their effectiveness, particularly for long-tail items.

To address the above challenges, we propose an LLM-based item embedding generator (**LLMEmb**) specified for SRS. The proposed LLMEmb involves a two-stage training. For the first stage, we design a Supervised Contrastive Fine-Tuning (SCFT) to bridge the semantic gap between general and recommendation domains. In detail, attribute-level data augmentation is designed to construct the training pairs for enhancing the distinguishing abilities of LLM. The fine-tuned LLM can derive recommendation-friendly embeddings. The second stage, *i.e.,* Recommendation Adaptation Training (RAT), focuses on injecting the collaborative signals into LLM embeddings. To prevent semantic loss, we design a trainable adapter that allows for dimension transformation while keeping the LLM embeddings frozen. During inference, the generated embeddings can be cached into the embedding layer of SRS models, ensuring that no additional computational burden is introduced. The contributions of this paper are concluded as follows:

- We design a novel LLM-based item embedding generator, which can help alleviate the long-tail problem for the sequential recommendation.
- To fill the semantic gap between general and recommendation domains, we propose an attribute-level contrastive fine-tuning method. To avoid semantic loss, we fabricate a recommendation adaptation strategy.
- We have conducted comprehensive experiments on three public datasets and verified the superior performance of LLMEmb combined with three popular SRS models.

Preliminary

Problem Definition. Let $v_i \in \mathcal{V}$ denotes the item in an item set, then the input sequence of user u can be represented as $\mathcal{Q}^{(u)} = \{v_1^{(u)}, v_2^{(u)}, \ldots, v_{N_u}^{(u)}\}$, which is ordered by timeline. N_u is the length of the interaction sequence. For simplicity, we omit the user-specific superscript (u) in subsequent notations. The task of recommending the next item can thus be formulated as:

$$\arg\max_{v_j \in \mathcal{V}} P(v_{N+1} = v_j | \mathcal{Q}) \tag{1}$$

General SRS Framework. Most existing SRS models can be broadly concluded into a two-step framework known as ***Embedding-Sequence***. In the first step, the item identities are transformed into dense embeddings to represent them in a high-dimension space, capturing the collaborative relationships among items. Here, we denote item i as v_i. The ***Embedding*** procedure is formalized as $\mathbf{e}_i = \mathrm{Emb}(v_i)$. $\mathrm{Emb}(\cdot)$ denotes the **embedding function**, and the resulting $\mathbf{e}_i \in \mathbb{R}^d$ represents the high-dimensional embedding of the item i, with d being the dimension size. After the first step, the input sequence is transformed into an embedding sequence $\bar{\mathcal{Q}} = \{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_N\}$. The next step is the ***Sequence*** procedure, which aims to extract the user preference from interaction histories. Thus, it absorbs $\bar{\mathcal{Q}}$ and outputs the representation of the user $\mathbf{u} \in \mathbb{R}^d$. The process can be represented as $\mathbf{u} = \mathrm{Seq}(\{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_N\})$. $\mathrm{Seq}(\cdot)$ is the sequence modeling function, referred to as **SRS backbone** in this paper. Finally, the recommending probability of each item is calculated by taking the inner product of the user and item representations, *i.e.,* $P(v_{N+1} = v_i | \mathcal{Q}) = \mathbf{u}^T \mathbf{e}_i$. Let $\hat{\mathbf{y}}$ denote the probability vector of all items. The framework is then optimized using a loss function, such as Binary Cross-Entropy calculated based on $\hat{\mathbf{y}}$.

To model user preferences more precisely, various neural architectures have been fabricated for **SRS backbone** $\mathrm{Seq}(\cdot)$, such as recurrent neural networks (Cho et al. 2014) for GRU4Rec (Hidasi et al. 2016) and self-attention (Vaswani et al. 2017) for SASRec (Kang and McAuley 2018). However, the embedding function is often simply designed as a randomly initialized embedding layer and trained from scratch. In this paper, we focus on **utilizing the LLM to generate better embedding function**, *i.e.,* $\mathrm{Emb}(\cdot)$, which can be integrated into most SRS models.

Method

In this section, we will introduce the details of the proposed LLMEmb. Firstly, we will give an overview of our method. Then, the supervised contrastive fine-tuning will be addressed to illustrate how we fine-tune a general LLM into

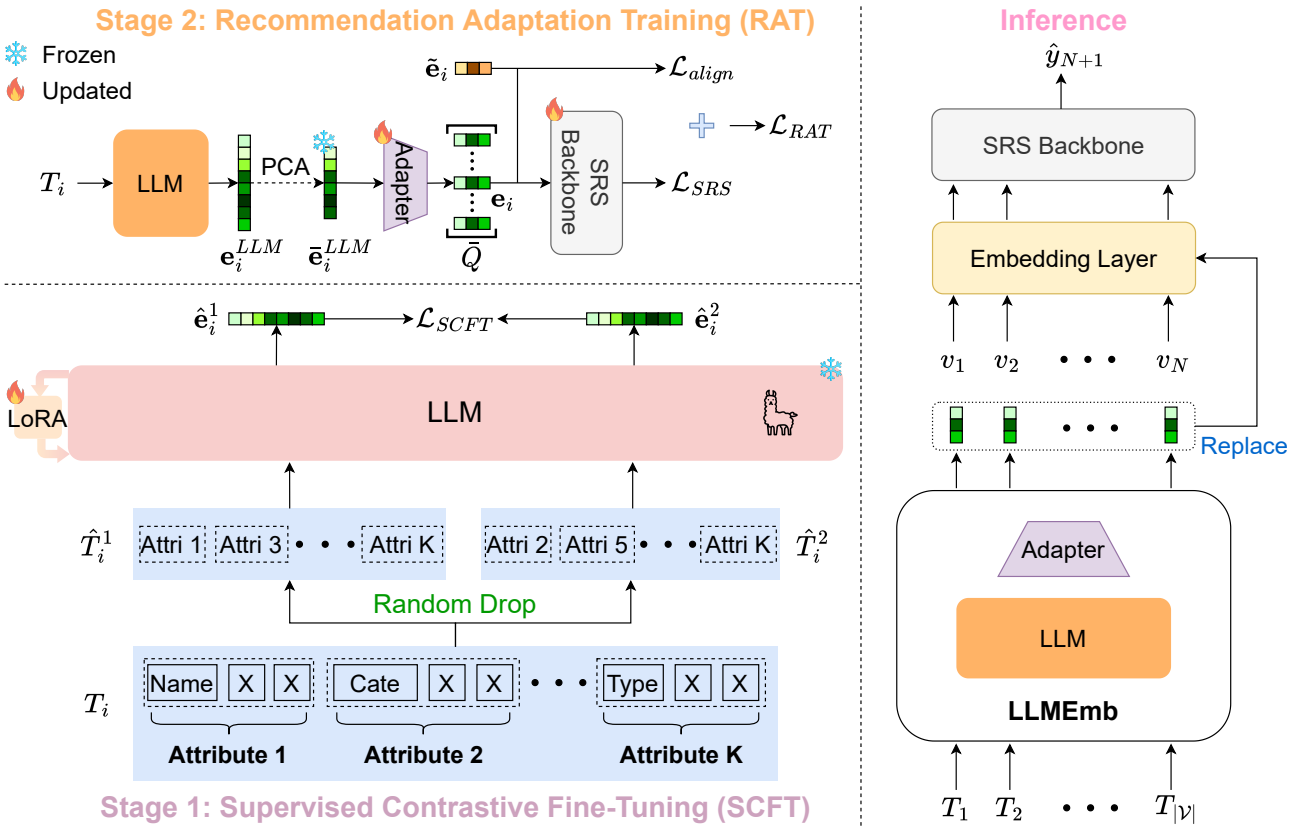


Figure 3: The overview of the proposed LLMEmb.

a recommendation-friendly one. Next, to further combine the collaborative signals with the LLM embeddings, we devise the recommendation adaptation training for LLMEmb.

Overview

Figure 3 shows the training and inference process of LLMEmb, composed of an LLM and an adapter. For the training process, there are two stages specialized for the LLM and adapter, respectively. In the first stage, known as **Supervised Contrastive Fine-tuning (SCFT)**, the objective is to fine-tune the general-purpose LLM to enhance its ability to distinguish items based on their various attributes. Specifically, the textual prompt of one item, composed of its attributes, will be augmented into two copies by randomly dropping a certain ratio of attributes. Then, we will fine-tune the LLM by contrasting its embedding of distinct items. After that, it can derive recommendation-friendly LLM embeddings, containing the semantic information of items. The second stage, termed **Recommendation Adaptation Training (RAT)**, involves training the adapter designed to transform the LLM embeddings into the final item embeddings. These item embeddings are then fed into the SRS backbone and optimized using general recommendation loss. During the inference phase, the LLMEmb will generate all item embeddings in advance. These precomputed embeddings substitute the original embedding layer of the SRS model.

Supervised Contrastive Fine-Tuning (SCFT)

The LLM has demonstrated exceptional semantic understanding capabilities across various natural language processing tasks (Zhao et al. 2023; Chang et al. 2024), suggesting the potential to enhance SRS by extracting rich semantic information from the item's textual attributes. However, most LLMs are trained for general purposes and may struggle to perceive subtle distinctions between items with varying attributes. To address this semantic gap, we design a supervised contrastive fine-tuning for the LLM (LLaMA in this paper). The key idea is enabling the LLM to distinguish between items by contrasting their attributes.

Prompt Construction. To encourage the LLM to understand the item from a semantic perspective, we construct textual prompts based on its attributes, *e.g.,* name, categories, and others. The designed prompt consists of two parts. One is a domain-related instruction, denoted as I, to inform the LLM about the type of recommendation task. For example, the instruction can be "*The point of interest has the following attributes:* " for a POI recommendation (Long et al. 2023). The other part includes all the attributes of the item, where each attribute is structured in the format "<*Attri*> *is* <*Content*>". Here, <*Attri*> and <*Content*> will be replaced by the attribute name and actual attribute values. Let A_j denote the atomic prompt for each attribute, then the prompt of item i can be formulated as follows:

$$T_i = [I, A_1, A_2, \ldots, A_K] \tag{2}$$

where $[\cdot]$ represents the concatenation operation for strings and K is the number of attributes.

Data Augmentation. As previously discussed, our goal is to fine-tune the LLM to equip it with the capacity to distinguish the items with different attributes. Fundamentally, each item can be considered a negative sample relative to other items, as they represent distinct semantics within the recommendation. By fine-tuning the LLM to push the distance between different items, we improve the uniformity of semantic representations (Ou et al. 2024), which can subsequently enhance recommendation adaptability. Then, to emphasize the fine-grained impact of item attributes, we propose to randomly drop a certain ratio of the item's attributes to get two copies of one item. These two copies serve as a pair of positive samples. Specifically, the augmentation process is as follows:

$$\begin{aligned} \hat{T}_i^1 &= [I, \mathrm{RandomDrop}(\{A_j\}_{j=1}^K, r)] \\ \hat{T}_i^2 &= [I, \mathrm{RandomDrop}(\{A_j\}_{j=1}^K, r)] \end{aligned} \tag{3}$$

where $\mathrm{RandomDrop}(\cdot)$ denotes the operation of randomly dropping and r is the ratio for dropping.

Contrastive Fine-tuning. Recent research has demonstrated that LLM can effectively generate high-quality embeddings for text, which are useful for tasks such as retrieval and matching (Lee et al. 2024; Wang et al. 2023a). Inspired by these works, we propose to utilize the LLM embeddings as the semantic representation of items. In detail, for each item i, we input prompt T_i into the LLM and then average the corresponding word token embeddings from the final transformer layer to produce the LLM embedding, mark as $\mathbf{e}_i^{LLM} \in \mathbb{R}^{d_{token}}$. d_{token} represents the dimension of token embedding in the LLM. We then apply in-batch contrastive learning (Yang et al. 2023) directly to these LLM embeddings. In detail, the augmented textual prompts, $\hat{T}_i^1$ and $\hat{T}_2^1$, are fed into the LLM, producing the corresponding embeddings $\hat{\mathbf{e}}_i^1$ and $\hat{\mathbf{e}}_i^2$ for each item i. After that, the contrastive loss for one side augmentation can be expressed as follows:

$$\mathcal{L}_{CL}^1 = -\frac{1}{B}\sum_{i=1}^{B} \log \frac{\exp(\mathrm{sim}(\hat{\mathbf{e}}_i^1, \hat{\mathbf{e}}_i^2)/\tau)}{\sum_{k=1}^{B} \mathbb{I}_{[i\neq k]} \exp(\mathrm{sim}(\hat{\mathbf{e}}_i^1, \hat{\mathbf{e}}_k^2)/\tau)} \tag{4}$$

where $\mathbb{I}_{[i\neq k]} \in \{0, 1\}$ is an indicator function and B is the batch size. $\mathrm{sim}(\cdot)$ is a similarity measuring function, which is the inner product in this paper. τ is a trainable temperature coefficient. In the same way, we can get the other side of contrastive loss $\mathcal{L}_{CL}^2$ by exchanging the positions of $\hat{\mathbf{e}}^1$ and $\hat{\mathbf{e}}^2$ in the Equation (4). The final loss function used for fine-tuning the LLM is given by:

$$\mathcal{L}_{SCFT} = \mathcal{L}_{CL}^1 + \mathcal{L}_{CL}^2 \tag{5}$$

Recommendation Adaptation Training (RAT)

While the fine-tuned LLM can generate embeddings that better suit recommendation tasks, two key challenges remain when integrating these embeddings into SRS. The first challenge is the lack of collaborative signals, which are crucial for the effectiveness of SRS models (Cai et al. 2021). The second challenge is dimension incompatibility, as the LLM embeddings often largely differ in size from the embeddings typically used in SRS models. To address these challenges, we introduce a Recommendation Adaptation Training (RAT) designed to transform LLM-generated embeddings into final item embeddings suitable for SRS models. The RAT framework consists of three key components. The first component is **Embedding Transformation**, which integrates a trainable adapter to adjust the dimensionality of the LLM embeddings. The second component, **Adaptation**, involves injecting collaborative signals into the adapter by training it alongside the SRS backbone. Finally, **Collaborative Alignment** is devised to assist the optimization.

Embedding Transformation. Previous works (Harte et al. 2023; Hu et al. 2024) have proposed using PCA (Pearson 1901) to reduce the dimension of LLM embeddings, but they face semantic loss. To alleviate this problem, we propose a two-level transformation strategy. At the first level, we also apply PCA to reduce the embedding size, facilitating optimization (Goodfellow, Vinyals, and Saxe 2014). However, to preserve the semantics contained in LLM embeddings, we limit the reduction to an intermediate size (*e.g.,* 1536), which remains significantly larger than the typical dimensionality of SRS embeddings (usually 128). This process can be formatted as $\mathbf{e}^{LLM} \xrightarrow{PCA} \bar{\mathbf{e}}^{LLM}$, where $\mathbf{e}^{LLM}$ is the LLM embedding derived from the fine-tuned LLM, and $\bar{\mathbf{e}}^{LLM} \in \mathbb{R}^{d_m}$ denotes the downsized LLM embedding with d_m being the intermediate size. Following this, we design an adapter to generate the final item embedding, ensuring compatibility with SRS models. For example, the final embedding of item i can be computed as

$$\mathbf{e}_i = \mathbf{W}_1(\mathbf{W}_2\bar{\mathbf{e}}_i^{LLM}) + \mathbf{b}_2) + \mathbf{b}_1 \tag{6}$$

where $\mathbf{W}_1 \in \mathbb{R}^{d\times\frac{d_m}{2}}, \mathbf{W}_2 \in \mathbb{R}^{\frac{d_m}{2}\times d_m}$ and $\mathbf{b}_1 \in \mathbb{R}^{d\times 1}, \mathbf{b}_2 \in \mathbb{R}^{\frac{d_m}{2}\times 1}$ are parameters of the two-layer adapter. By this transformation process, we can get the final item embeddings from LLM embeddings.

Adaptation. Although the semantic relationships captured by LLM embeddings can significantly benefit long-tail items, the incorporation of collaborative signals remains essential for effective recommendation tasks (Cai et al. 2021). Thus, we design an adaptation process to train the derived embeddings. Specifically, we treat the LLM embeddings $\bar{\mathbf{e}}^{LLM}$, along with the proposed adapter, as the embedding function. These embeddings are then combined with an SRS backbone to complete the sequential recommendation process. To learn collaborative signals, we update the randomly initialized SRS backbone and the adapter using the loss function specific to the corresponding SRS model, denoted as $\mathcal{L}_{SRS}$. For example, SASRec (Kang and McAuley 2018) adopts the Binary Cross-Entropy loss. It is worth noting that we freeze the parameters of the LLM embeddings $\bar{\mathbf{e}}^{LLM}$ during training, because the update of it will destroy the original semantic relationships. Consequently, during the RAT stage, only the parameters of the SRS backbone and the adapter are updated.

Collaborative Alignment. As mentioned earlier, only the adapter is trained to transform the semantic LLM embeddings into the final item embeddings. However, this ap-

proach may lead to overfitting, as only a small proportion of parameters (*i.e.,* those of the adapter) are updated. To mitigate this problem, we propose to align the derived item embeddings with the well-trained collaborative embeddings. Such an alignment will assist the optimization process by learning coarse collaborative relationships between items. Specifically, we first train an SRS model and take out its embedding layer. Let $\tilde{\mathbf{e}}_i$ denote item i's embedding of the well-trained SRS model. Then, we design an in-batch contrastive loss to align $\mathbf{e}_i$ with $\tilde{\mathbf{e}}_i$:

$$\mathcal{L}_{align}^{1} = -\frac{1}{S}\sum_{i=1}^{S}\log\frac{\exp(\text{sim}(\mathbf{e}_i,\tilde{\mathbf{e}}_i)/\gamma)}{\sum_{k=1}^{S}\mathbb{I}_{[i\neq k]}\exp(\text{sim}(\mathbf{e}_i,\tilde{\mathbf{e}}_k)/\gamma)} \tag{7}$$

where S and γ denote the sum of sequence lengths of one batch and the temperature for contrastive learning, respectively. Similarly, we can compute the contrastive loss $\mathcal{L}_{align}^{2}$ that aligns $\tilde{\mathbf{e}}_i$ with $\mathbf{e}_i$. The sum of these two losses is denoted as $\mathcal{L}_{align}$, used for training the adapter and SRS backbone together with $\mathcal{L}_{SRS}$.

Training and Inference

In this section, we will detail the training and inference process.

Training. During the SCFT stage, we adopt the LoRA (Hu et al. 2022) technique to fine-tune the LLM, allowing us to save computational resources. Consequently, only the low-rank matrics $\{\mathbf{A}_i, \mathbf{B}_i\}_{i=1}^{M}$ are trained by $\mathcal{L}_{SCFT}$, where M is number of layers accompanied by LoRA. In the RAT stage, the optimization process is formulated as:

$$arg\min_{\Theta,\Phi}\mathcal{L}_{SRS} + \alpha\cdot\mathcal{L}_{align} \tag{8}$$

where Θ represents the parameters of SRS backbone and $\Phi = \{\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}_1, \mathbf{b}_2\}$ is the ones of the adapter. The hyperparameter α controls the strength of the alignment.

Inference. As previously described, a general SRS consists of an SRS backbone and an embedding function. During inference, the well-trained SRS backbone (*i.e.,* parameter Θ) obtained from the RAT stage is used for the ***Sequence*** procedure. For the embedding function, we generate LLM embeddings for all items using their textual prompts T_i and then feed the dimension-reduced $\bar{\mathbf{e}}^{LLM}$ to the adapter. As a result, the final embeddings $\mathbf{e}_i \in \mathbb{R}^d$ are precomputed and cached in advance. These generated embeddings replace the weights of the embedding layer, effectively serving as the ***Embedding*** component. In conclusion, this approach introduces no additional computational burden during inference compared to traditional SRS models.

Experiment

In this section, we will show the experimental results to respond to the following Research Questions (**RQ**).

- **RQ1**: How does the proposed LLMEmb perform, compared with LLM-based baselines? Can the LLMEmb enhance various SRS models?
- **RQ2**: Do all designs for LLMEmb take effect?
- **RQ3**: How do hyper-parameters affect the performance of our LLMEmb?
- **RQ4**: Can the proposed LLMEmb alleviate the long-tail problem in SRS?
- **RQ5**: Can LLMEmb correct embedding distributions?

Experimental Settings

Dataset. In the experiments, we adopt three real-world datasets for verification, *i.e.,* Yelp, Amazon Beauty, and Amazon Fashion. Yelp includes amounts of check-in records, which can be used for point-of-interest recommendation. Amazon is collected from an e-commerce platform. Beauty and Fashion are two sub-categories of this dataset. We follow the preprocessing of the previous SRS works (Kang and McAuley 2018).

Sequential Recommendation Backbones. Since LLMEmb is a model-agnostic method, it can be integrated with many SRS models. To verify the generality, we test it on GRU4Rec (Hidasi et al. 2016), Bert4Rec (Sun et al. 2019), and SASRec (Kang and McAuley 2018).

Baselines. To verify the effectiveness of our LLMEmb, we compare one state-of-the-art **Long-tail Sequential Recommendation** baseline, *i.e.,* MELT (Kim et al. 2023), and three up-to-date **LLM-enhanced Sequential Recommendation** baselines, including LLM2X (Harte et al. 2023), SAID (Hu et al. 2024), TSLRec (Liu et al. 2024a).

Implementation Details. All the experiments in this paper are conducted on an Intel Xeon Gold 6133 platform, equipped with Tesla V100 GPUs. The code is built on Python 3.9.5 with PyTorch 1.12.0. For a fair comparison, the LLM used for baselines and our LLMEmb is LLaMA-7B (Touvron et al. 2023). Our code is available online[1].

Evaluation Metrics. Following the previous works (Sun et al. 2019; Kang and McAuley 2018), we adopt common used Top-10 *Normalized Discounted Cumulative Gain* (**N@10**) and *Hit Rate* (**H@10**) as the metrics. Each positive item in the test set will be paired with 100 randomly sampled uninteracted items to calculate the metrics. Besides, for the robustness of the results, we repeatedly conduct each experiment three times with random seeds 42, 43, 44 and report the average values in the following tables and figures.

Overall Performance (RQ1)

To respond to the **RQ1**, we show the overall and long-tail performance on three datasets in Table 1. Specifically, according to the Pareto principle, we divide the items with the popularity ranked at the last 80% into the **Tail** group. The results indicate that our LLMEmb can achieve superior performance compared with all competitors. Especially, our method benefits the long-tail items with a large margin. For a more detailed analysis, we find that LLM-based methods often outperform MELT, a collaborative method for the long-tail problem. Such a phenomenon verifies the effectiveness of introducing semantics by the LLM. Comparing these three LLM-based methods, TSLRec often lags behind, because it only adopts the identities instead of textual

[1]https://github.com/Applied-Machine-Learning-Lab/LLMEmb

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***

Table 1: The overall results of competing methods and **LLMEmb** on three datasets. The boldface refers to the highest score, and the underline indicates the best result of the baselines. "*" indicates the statistically significant improvements (*i.e.,* two-sided t-test with $p < 0.05$) over the best baseline.

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

Table 2: The ablation study conducted on the Yelp dataset and based on the SASRec backbone. The boldface refers to the highest score.

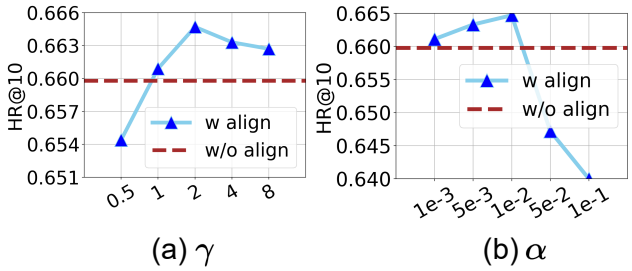


Figure 4: The results of experiments for the temperature γ and the weight α of alignment loss based on the Yelp dataset and SASRec backbone.

information of items when using the LLM. Though SAID and LLM2X can also bring a large performance elevation to all SRS models, they are still inferior to LLMEmb, especially for the long-tail items. This comparison indicates our LLMEmb can better maintain the semantic relationship in the original LLM embeddings. In conclusion, due to our design of LLM fine-tuning and recommendation adaptation, LLMEmb can enhance the three SRS models consistently.

Ablation Study (RQ2)

For **RQ2**, we have conducted the ablation study and show the results in Table 2. To investigate the effect of the proposed SCFT, we evaluate adopting the LLaMA without fine-tuning to derive LLM embeddings, denoted as *w/o SCFT*. The performance of this variant drops under both overall and tail metrics, highlighting the necessity to fill the **semantic gap** between general LLM and recommendation tasks. Then, we evaluate three variants to verify the designs of RAT. *w/o adapter* means removing the trainable adapter directly, which shows sub-optimal performance. It indicates the effectiveness of transformation. The variant without freezing the LLM embeddings during training, marked as *w/o freeze*, severely harms the performance, suggesting the optimization difficulty in training large-size embedding. *w/o align* eliminating the alignment loss directly illustrates the effectiveness of the collaborative alignment by the performance decrease.

Hyper-parameter Analysis (RQ3)

The temperature γ and scale α of the collaborative loss are two vital hyper-parameters in our LLMEmb. We show trends with their changes in Figure 4 to answer the **RQ3**. As the temperature γ changes from 0.5 to 8, the overall HR@10 values of LLMEmb rise first and drop then. The reason lies

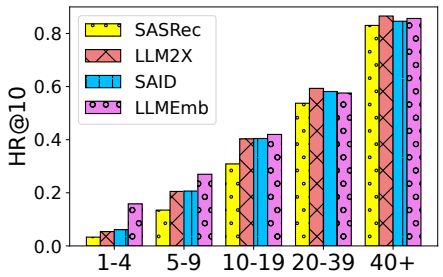


Figure 5: The experimental results of group analysis based on Yelp dataset and SASRec backbone.

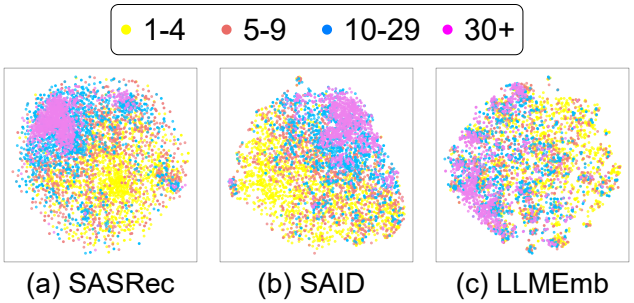


Figure 6: The visualization of embeddings. The LLMEmb and baselines are based on SASRec and the Yelp dataset.

in that the proper uniformity brought by contrastive learning can assist the optimization. In terms of the scale, the performance gets elevated with α rise from $1e^{-3}$ to $1e^{-2}$, which indicates the effectiveness of the designed alignment. However, larger α downgrades the performance, because the higher intensity of contrastive loss will lead to a convergence dilemma. Besides, the intermediate size d_m is also important to the designed adapter.

Group Analysis (RQ4)

To explore the long-tail problem more carefully and answer the **RQ4**, we cluster the items by their popularity in 5 groups and show the results in Figure 5. Observing the figure, we find that the LLM-based methods can benefit the items with any popularity because of the semantic relationships. Compared with LLM2X and SAID, our LLMEmb brings more performance elevation to long-tail items, especially for the 1-4 group. Such a phenomenon validates that our method can better integrate semantics from the LLM into recommendation. However, the LLMEmb underperforms LLM2X for those popular items (*e.g.,* 40+ group) slightly, indicating a seesaw problem between popular and long-tail items.

Visualization (RQ5)

To investigate whether the proposed LLMEmb can correct the skewed distribution of the embeddings, we visualize the distributions by t-SNE in Figure 6. The figure shows that SAID can get a more even distribution by introducing the LLM embeddings. However, it is still congregated by the item's popularity due to the **semantic loss** issue. In contrast, our LLMEmb gets better embeddings, which are distributed more uniformly. The results respond to **RQ5** and reveal the superiority of our LLMEmb intrinsically.

Related Works

Sequential Recommendation. The sequential recommendation aims to capture the user's preference from his or her historical interactions and then predict the next most possible item (Liu et al. 2023b,c, 2024g; Li et al. 2023a; Liu et al. 2024h, 2023a; Li et al. 2023b; Liang et al. 2023; Liu et al. 2023d, 2024b; Wang et al. 2023b). Many existing SRS works focus on fabricating the neural architecture to get the preference and dynamics more accurately. For example, Caser (Tang and Wang 2018) adopts CNN for sequence modeling, while SASRec (Kang and McAuley 2018) firstly integrates self-attention (Vaswani et al. 2017) layers. Later, for higher efficiency, some research studies (Zhou et al. 2022) propose the MLP-based structure. On the other hand, the loss function for training SRS models has also been highlighted in recent years. Bert4Rec (Sun et al. 2019) propose the cloze task to derive the training loss, while CLS4Rec (Liu et al. 2021) further designs the contrastive loss for training the SRS models. However, most existing works have ignored the importance of item embeddings, which suffer from skewed distribution. In this paper, we propose an LLM-based method to construct better embeddings.

Large Language Model for Recommendation. Many efforts have been made to utilize the powerful LLM for recommendation (Liu et al. 2024e,c,f). A branch of research studies proposes to utilize the LLM for recommendation directly. For instance, TALLRec (Bao et al. 2023) designs the textual prompt for recommendation tasks, which motivates the LLM to generate the predicted item name. Besides, to combine the collaborative signals into the LLM, E4SRec (Li et al. 2023c) and LLaRA (Liao et al. 2023) design a trainable adapter to project the pre-trained item embeddings to language space and impose parameter-efficient fine-tuning (Liu et al. 2024d). Despite the brilliant performance of these models, the direct utilization of the LLM is resource-consuming, which is intolerant to real-time recommendation. For this issue, LLM2X (Harte et al. 2023), SAID (Hu et al. 2024) and TSLRec (Liu et al. 2024a) propose to adopt the LLM embeddings to enhance SRS models. However, they still face semantic gap and loss, leading to sub-optimal performance.

Conclusion

In this paper, we propose a novel LLM-based generator, *i.e.,* LLMEmb, to derive item embeddings for the sequential recommendation. Specifically, to equip the LLM with the capacity to identify the items for recommendation tasks, we devise a supervised contrastive fine-tuning. Then, to avoid semantic loss and inject collaborative signals, we propose the recommendation adaptation training to update a trainable adapter. In the end, the well-trained LLM and adapter constitute the LLMEmb and can generate the final item embeddings. We conduct experiments on three real-world datasets and verify the effectiveness of LLMEmb.

Acknowledgements

This research was partially supported by National Key Research and Development Program of China (2022ZD0117100), National Natural Science Foundation of China (No.62192781, No.62177038, No.62293551, No.62277042, No.62137002, No.61721002, No.61937001, No.62377038), Project of China Knowledge Centre for Engineering Science and Technology, "LENOVO-XJTU" Intelligent Industry Joint Laboratory Project, Research Impact Fund (No.R1015-23), Collaborative Research Fund (No.C1043-24GF), APRC - CityU New Research Initiatives (No.9610565, Start-up Grant for New Faculty of CityU), CityU - HKIDS Early Career Research Grant (No.9360163), Hong Kong ITC Innovation and Technology Fund Midstream Research Programme for Universities Project (No.ITS/034/22MS), Hong Kong Environmental and Conservation Fund (No. 88/2022), and SIRG - CityU Strategic Interdisciplinary Research Grant (No.7020046), Tencent (CCF-Tencent Open Fund, Tencent Rhino-Bird Focused Research Program).

References

Bao, K.; Zhang, J.; Zhang, Y.; Wang, W.; Feng, F.; and He, X. 2023. Tallrec: An effective and efficient tuning framework to align large language model with recommendation. In *Proceedings of the 17th ACM Conference on Recommender Systems*, 1007–1014.

Cai, R.; Wu, J.; San, A.; Wang, C.; and Wang, H. 2021. Category-aware collaborative sequential recommendation. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*, 388–397.

Chang, Y.; Wang, X.; Wang, J.; Wu, Y.; Yang, L.; Zhu, K.; Chen, H.; Yi, X.; Wang, C.; Wang, Y.; et al. 2024. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*, 15(3): 1–45.

Cho, K.; van Merriënboer, B.; Gulçehre, Ç.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1724–1734.

Fang, H.; Zhang, D.; Shu, Y.; and Guo, G. 2020. Deep learning for sequential recommendation: Algorithms, influential factors, and evaluations. *ACM Transactions on Information Systems (TOIS)*, 39(1): 1–42.

Goodfellow, I. J.; Vinyals, O.; and Saxe, A. M. 2014. Qualitatively characterizing neural network optimization problems. *arXiv preprint arXiv:1412.6544*.

Harte, J.; Zorgdrager, W.; Louridas, P.; Katsifodimos, A.; Jannach, D.; and Fragkoulis, M. 2023. Leveraging large language models for sequential recommendation. In *Proceedings of the 17th ACM Conference on Recommender Systems*, 1096–1102.

Hidasi, B.; Karatzoglou, A.; Baltrunas, L.; and Tikk, D. 2016. Session-based recommendations with recurrent neural networks. In *The International Conference on Learning Representations*.

Hou, M.; Wu, L.; Chen, E.; Li, Z.; Zheng, V. W.; and Liu, Q. 2019. Explainable fashion recommendation: a semantic attribute region guided approach. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, 4681–4688.

Hu, E. J.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; Chen, W.; et al. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*.

Hu, J.; Xia, W.; Zhang, X.; Fu, C.; Wu, W.; Huan, Z.; Li, A.; Tang, Z.; and Zhou, J. 2024. Enhancing sequential recommendation via llm-based semantic embedding learning. In *Companion Proceedings of the ACM on Web Conference 2024*, 103–111.

Kang, W.-C.; and McAuley, J. 2018. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*, 197–206. IEEE.

Kim, K.; Hyun, D.; Yun, S.; and Park, C. 2023. MELT: Mutual Enhancement of Long-Tailed User and Item for Sequential Recommendation. In *Proceedings of the 46th international ACM SIGIR conference on Research and development in information retrieval*, 68–77.

Lee, C.; Roy, R.; Xu, M.; Raiman, J.; Shoeybi, M.; Catanzaro, B.; and Ping, W. 2024. NV-Embed: Improved Techniques for Training LLMs as Generalist Embedding Models. *arXiv preprint arXiv:2405.17428*.

Li, C.; Wang, Y.; Liu, Q.; Zhao, X.; Wang, W.; Wang, Y.; Zou, L.; Fan, W.; and Li, Q. 2023a. STRec: Sparse Transformer for Sequential Recommendations. In *Proceedings of the 17th ACM Conference on Recommender Systems*, 101–111.

Li, M.; Zhang, Z.; Zhao, X.; Wang, W.; Zhao, M.; Wu, R.; and Guo, R. 2023b. Automlp: Automated mlp for sequential recommendations. In *Proceedings of the ACM Web Conference 2023*, 1190–1198.

Li, X.; Chen, C.; Zhao, X.; Zhang, Y.; and Xing, C. 2023c. E4srec: An elegant effective efficient extensible solution of large language models for sequential recommendation. *arXiv preprint arXiv:2312.02443*.

Liang, J.; Zhao, X.; Li, M.; Zhang, Z.; Wang, W.; Liu, H.; and Liu, Z. 2023. Mmmlp: Multi-modal multilayer perceptron for sequential recommendations. In *Proceedings of the ACM Web Conference 2023*, 1109–1117.

Liao, J.; Li, S.; Yang, Z.; Wu, J.; Yuan, Y.; Wang, X.; and He, X. 2023. Llara: Aligning large language models with sequential recommenders. *arXiv preprint arXiv:2312.02445*.

Liu, D.; Xian, S.; Lin, X.; Zhang, X.; Zhu, H.; Fang, Y.; Chen, Z.; and Ming, Z. 2024a. A Practice-Friendly Two-Stage LLM-Enhanced Paradigm in Sequential Recommendation. *arXiv preprint arXiv:2406.00333*.

Liu, L.; Cai, L.; Zhang, C.; Zhao, X.; Gao, J.; Wang, W.; Lv, Y.; Fan, W.; Wang, Y.; He, M.; et al. 2023a. Linrec: Linear attention mechanism for long-term sequential recommender

systems. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 289–299.

Liu, Q.; Hu, J.; Xiao, Y.; Zhao, X.; Gao, J.; Wang, W.; Li, Q.; and Tang, J. 2024b. Multimodal recommender systems: A survey. *ACM Computing Surveys*, 57(2): 1–17.

Liu, Q.; Tian, F.; Zheng, Q.; and Wang, Q. 2023b. Disentangling interest and conformity for eliminating popularity bias in session-based recommendation. *Knowledge and Information Systems*, 65(6): 2645–2664.

Liu, Q.; Wu, X.; Wang, Y.; Zhang, Z.; Tian, F.; Zheng, Y.; and Zhao, X. 2024c. LLM-ESR: Large Language Models Enhancement for Long-tailed Sequential Recommendation. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Liu, Q.; Wu, X.; Zhao, X.; Zhu, Y.; Xu, D.; Tian, F.; and Zheng, Y. 2024d. When MOE Meets LLMs: Parameter Efficient Fine-tuning for Multi-task Medical Applications. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1104–1114.

Liu, Q.; Wu, X.; Zhao, X.; Zhu, Y.; Zhang, Z.; Tian, F.; and Zheng, Y. 2024e. Large Language Model Distilling Medication Recommendation Model. *arXiv preprint arXiv:2402.02803*.

Liu, Q.; Yan, F.; Zhao, X.; Du, Z.; Guo, H.; Tang, R.; and Tian, F. 2023c. Diffusion augmentation for sequential recommendation. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, 1576–1586.

Liu, Q.; Zhao, X.; Wang, Y.; Wang, Y.; Zhang, Z.; Sun, Y.; Li, X.; Wang, M.; Jia, P.; Chen, C.; Huang, W.; and Tian, F. 2024f. Large Language Model Enhanced Recommender Systems: Taxonomy, Trend, Application and Future. *arXiv preprint arXiv:2412.13432*.

Liu, Z.; Chen, Y.; Li, J.; Yu, P. S.; McAuley, J.; and Xiong, C. 2021. Contrastive self-supervised sequential recommendation with robust augmentation. *arXiv preprint arXiv:2108.06479*.

Liu, Z.; Liu, Q.; Wang, Y.; Wang, W.; Jia, P.; Wang, M.; Liu, Z.; Chang, Y.; and Zhao, X. 2024g. Bidirectional Gated Mamba for Sequential Recommendation. *arXiv preprint arXiv:2408.11451*.

Liu, Z.; Liu, S.; Zhang, Z.; Cai, Q.; Zhao, X.; Zhao, K.; Hu, L.; Jiang, P.; and Gai, K. 2024h. Sequential Recommendation for Optimizing Both Immediate Feedback and Long-term Retention. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1872–1882.

Liu, Z.; Tian, J.; Cai, Q.; Zhao, X.; Gao, J.; Liu, S.; Chen, D.; He, T.; Zheng, D.; Jiang, P.; et al. 2023d. Multi-task recommendations with reinforcement learning. In *Proceedings of the ACM Web Conference 2023*, 1273–1282.

Long, J.; Chen, T.; Nguyen, Q. V. H.; and Yin, H. 2023. Decentralized collaborative learning framework for next POI recommendation. *ACM Transactions on Information Systems*, 41(3): 1–25.

Ou, Y.; Chen, L.; Pan, F.; and Wu, Y. 2024. Prototypical contrastive learning through alignment and uniformity for recommendation. *arXiv preprint arXiv:2402.02079*.

Pan, Y.; Gao, C.; Chang, J.; Niu, Y.; Song, Y.; Gai, K.; Jin, D.; and Li, Y. 2023. Understanding and modeling passive-negative feedback for short-video sequential recommendation. In *Proceedings of the 17th ACM conference on recommender systems*, 540–550.

Pearson, K. 1901. LIII. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, 2(11): 559–572.

Sun, F.; Liu, J.; Wu, J.; Pei, C.; Lin, X.; Ou, W.; and Jiang, P. 2019. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM international conference on information and knowledge management*, 1441–1450.

Tang, J.; and Wang, K. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the eleventh ACM international conference on web search and data mining*, 565–573.

Touvron, H.; Lavril, T.; Izacard, G.; Martinet, X.; Lachaux, M.-A.; Lacroix, T.; Rozière, B.; Goyal, N.; Hambro, E.; Azhar, F.; et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Wang, L.; Yang, N.; Huang, X.; Yang, L.; Majumder, R.; and Wei, F. 2023a. Improving text embeddings with large language models. *arXiv preprint arXiv:2401.00368*.

Wang, Y.; Zhao, X.; Chen, B.; Liu, Q.; Guo, H.; Liu, H.; Wang, Y.; Zhang, R.; and Tang, R. 2023b. PLATE: A prompt-enhanced paradigm for multi-scenario recommendations. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1498–1507.

Yang, Z.; Huang, T.; Ding, M.; Dong, Y.; Ying, R.; Cen, Y.; Geng, Y.; and Tang, J. 2023. Batchsampler: Sampling mini-batches for contrastive learning in vision, language, and graphs. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 3057–3069.

Zhao, W. X.; Zhou, K.; Li, J.; Tang, T.; Wang, X.; Hou, Y.; Min, Y.; Zhang, B.; Zhang, J.; Dong, Z.; et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

Zhou, G.; Zhu, X.; Song, C.; Fan, Y.; Zhu, H.; Ma, X.; Yan, Y.; Jin, J.; Li, H.; and Gai, K. 2018. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 1059–1068.

Zhou, K.; Yu, H.; Zhao, W. X.; and Wen, J.-R. 2022. Filter-enhanced MLP is all you need for sequential recommendation. In *Proceedings of the ACM web conference 2022*, 2388–2399.