CoLLM-FaissRet: Integrating Collaborative Information as a Modality with Efficient Faiss Retrieval for Recommendation

Chuhong Zheng[1*]

[1]*Jacobs School of Engineering, UC San Diego, La Jolla, USA*
**Corresponding Author. Email: c5zheng@ucsd.edu*

Abstract: The integration of Large Language Models (LLMs) into recommender systems (LLMRec) has emerged as a promising direction for addressing cold-start scenarios. However, existing approaches that leverage LLMs for warm-start recommendations can be categorized into three paradigms, each with critical weaknesses: direct application of general LLMs leads to accuracy issues; fine-tuned LLMs overfit to semantic information and lack collaborative signals; and methods integrating collaborative filtering suffer from either limited synergy or prohibitive computational costs. To overcome these challenges, we propose CoLLM-FaissRet, an improved approach that treats collaborative information as a separate modality and integrates it into LLMs through direct mapping, incorporating the Faiss library for efficient retrieval to better handle large-scale cold-start and warm-start recommendation scenarios. Our experimental evaluation diverges from prior work through systematic architecture parameter optimization and validation on novel datasets. These results confirm that appropriate learning rate configuration for different components of our framework can substantially improve recommendation accuracy. Furthermore, our empirical findings validate the generalization capability of our approach across diverse recommendation scenarios. The implementation of our method is publicly available at: *https://github.com/ChuhongZheng/CoLLM-FaissRet*.

Keywords: Recommender System, Large Language Model, Collaborative Information.

1. Introduction

Large Language Models (LLMs) have demonstrated remarkable capabilities, spurring interest in their use for recommender systems and leading to a new paradigm: LLM as Recommender (LLMRec) [1, 18].

Current LLM-based recommendation methods fall into three categories, each with key limitations. *Direct LLM application* [3, 6, 14] faces accuracy issues from the mismatch between general LLMs and recommendation needs. *Fine-tuned LLMs* [1, 5, 10, 19] tend to over-rely on semantic information and lack collaborative signals. *Collaborative filtering integrated LLMs* [2, 12, 16, 20-21, 23] still face challenges in accuracy and efficiency due to limited synergy or the considerable computational and memory costs of full candidate scoring.

To effectively enhance LLMs with collaborative signals in a lightweight and flexible framework, we propose **CoLLM-FaissRet**, a novel approach that treats collaborative information as an independent modality and integrates it into language models through direct mapping from pretrained conventional recommendation models. Our method employs a Multilayer Perceptron

(MLP) as the mapping function and incorporates the Faiss library [4] for efficient vector similarity search, enabling selective focus on the most relevant candidate items during inference.

The proposed framework effectively integrates collaborative information from conventional models with the reasoning capabilities of LLMs, combining the strengths of both approaches: LLMs excel in cold-start scenarios with rich semantic understanding, while conventional collaborative models leverage historical interactions for warm-start recommendations. By incorporating Faiss into the mapping module, our method achieves enhanced accuracy through more precise similarity computations and improved efficiency via optimized top-K retrieval, making it particularly suitable for large-scale recommendation scenarios where both performance and computational overhead are critical considerations.

The main contributions of this work are summarized as follows:

- **Novel Framework:** We propose **CoLLM-FaissRet**, an improved approach that treats collaborative information as a separate modality and integrates it into LLMs through direct mapping, incorporating Faiss for efficient retrieval to better handle large-scale cold-start and warm-start recommendation scenarios.
- **Parameter Optimization:** Through extensive parameter optimization of the CoLLM-FaissRet, we achieve significant improvements in recommendation accuracy, demonstrating the importance of careful parameter selection in multi-modal recommendation systems.
- **Datasets Evaluation:** Comprehensive experiments conducted on three real-world datasets consistently validate the CoLLM-FaissRet ' s superior effectiveness and enhanced robustness across diverse evaluation scenarios, providing strong empirical evidence for its practical applicability.

2. Related Works

In this section, we review related works in three main categories: (1) *direct LLM application for recommendation* [3, 6, 14], which leverages LLMs' natural language abilities without parameter updates; (2) *fine-tuned LLMs for recommendation* [1, 5, 10, 19], which employ fine-tuning and external tools to enhance recommendation capabilities; and (3) *collaborative filtering integrated LLMs for recommendation* [2, 12, 16, 20-21, 23], which focuses on incorporating collaborative signals through various integration strategies.

2.1. Direct LLM Application for Recommendation

Research has explored In-context Learning (ICL) for recommendation, leveraging LLMs' natural language abilities without updating model parameters. For example, Chat-Rec [6] uses ChatGPT's conversational framework to guide recommendations, while other studies elicit recommendations through carefully designed ICL prompts [3, 14].

Drawback: The accuracy limitations in these methods stem from the fundamental mismatch between general-purpose LLMs and recommendation-specific requirements. Since general-purpose LLMs are not pre-trained for modeling user-item interactions, they often show a performance gap compared to dedicated recommendation models, resulting in suboptimal accuracy in practice.

2.2. Fine-tuned LLMs for Recommendation

To address the limitations of direct LLM application, researchers have adopted Instruction Tuning, performing supervised fine-tuning (SFT) on recommendation-specific datasets to enhance LLMs' recommendation abilities [1, 19]. Another direction augments LLMs with external tools (e.g.,

databases and APIs) through fine-tuning or advanced prompting, enabling more interactive conversational recommendation systems [5, 10].

Drawback: The accuracy challenges in these approaches arise from LLMs' inherent bias toward semantic information. Empirical studies show that even after tuning, they can underperform traditional models in certain scenarios [13], primarily because they overly rely on pre-trained semantic priors (e.g., text similarity) while neglecting crucial collaborative information essential for collaborative filtering [17].

2.3. Collaborative Filtering Integrated LLMs for Recommendation

To mitigate LLMs' bias toward semantic priors, researchers have developed various strategies to incorporate collaborative signals into LLM-based recommendation systems. Some approaches combine LLMs with traditional collaborative filtering models through ensemble techniques [2], while others learn ID-based collaborative embeddings directly within the LLM's representation space [21, 23]. Another line of work uses LLMs as feature extractors to provide semantic features for collaborative models [12, 16], and some methods introduce mapping mechanisms to project pre-trained collaborative embeddings into the LLM's token space [20].

Drawback: These integration approaches face challenges in both accuracy and efficiency during the fusion of collaborative and semantic information. Simply combining LLMs with traditional models through ensemble techniques often results in shallow integration that fails to achieve effective synergy between the two components. Learning collaborative embeddings directly in the LLM's space may distort the original structural properties essential for capturing user-item interactions. Using LLMs merely as feature extractors loses their powerful semantic reasoning capabilities in the final recommendation process. Additionally, methods that map collaborative embeddings to the token space encounter significant efficiency challenges: they require scoring all candidate items during inference, leading to computational costs that scale with the catalog size; the full candidate pool introduces considerable noise that may interfere with semantic alignment; and caching all item embeddings consumes substantial GPU memory, which can limit the deployable model capacity.

3. Methodology

3.1. Overview

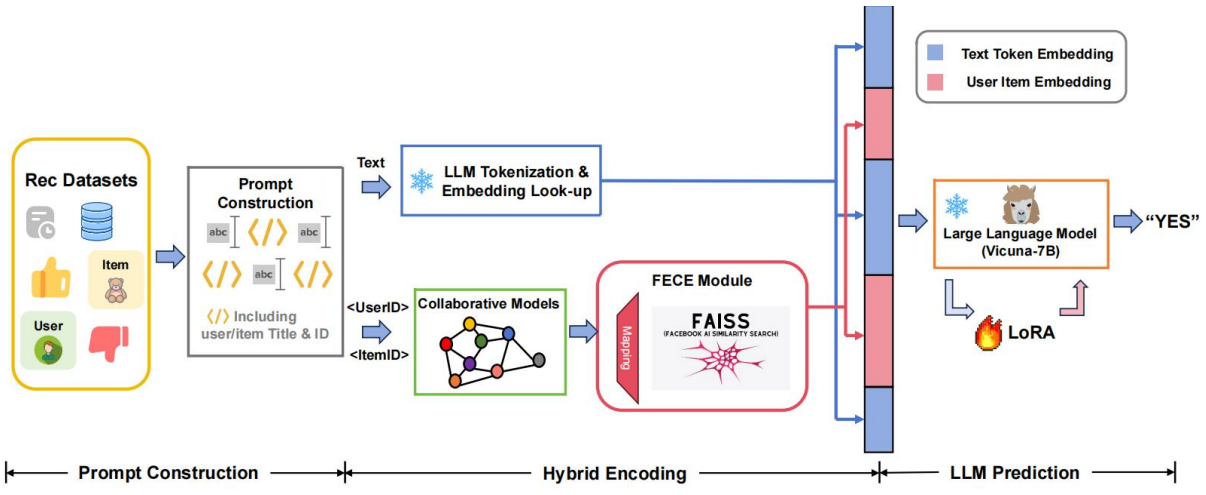


Figure 1: Overall architecture of the proposed method

Figure 1 illustrates the comprehensive architecture of our proposed framework, which consists of three sequential components: *Prompt Construction* (section 3.2), *Hybrid Encoding* (section 3.3), and *LLM Prediction* (section 3.4).

The workflow begins with transforming recommendation data into structured language prompts through the prompt construction module. These prompts subsequently flow into the hybrid encoding component, where they are converted into latent representations compatible with LLM processing. Finally, the encoded representations are fed into the LLM to generate recommendation predictions.

Our framework's key innovation lies in effectively integrating collaborative information to enhance LLM's recommendation capabilities, making it suitable for both cold-start and warm-start scenarios. This is achieved through specific designs in the first two components: during prompt construction, we incorporate user and item ID fields alongside textual descriptions to represent collaborative signals; during hybrid encoding, we introduce the Faiss Enhanced Collaborative Encoding (FECE) module to extract and map collaborative representations into the LLM's embedding space. The incorporation of Faiss during inference further ensures computational efficiency in large-scale applications.

3.2. Prompt Construction

The prompt construction process takes the recommendation dataset as input and generates structured prompts as output. Following the approach of TALLRec [1], we employ fixed templates to transform recommendation instances into textual prompts. Specifically, user profiles are represented by the titles of historically interacted items, while candidate items are described by their respective titles.

To incorporate collaborative filtering signals, we introduce unique identifier fields that serve as semantic-agnostic placeholders for capturing collaborative information. The complete prompt template is structured as follows:

```
#Question: A user has given high ratings to the following items:
<HisItemTitleList>. Additionally, we have information about the
user's preferences encoded in the feature <UserID>. Using all
available information, make a prediction about whether the user
would enjoy the item titled <TargetItemTitle> with the feature
<TargetItemID>? Answer with "Yes" or "No". #Answer:
```

In this template, <HisItemTitleList> denotes the chronologically ordered sequence of item titles from the user ' s interaction history, providing explicit preference signals. <TargetItemTitle> represents the title of the candidate item under evaluation. The <UserID> and <TargetItemID> fields function as collaborative information injectors, where user and item identifiers are embedded as special features despite lacking semantic meaning. This design preserves textual coherence while enabling the model to leverage identity-based collaborative signals through the underlined feature descriptors.

For each recommendation instance, we dynamically populate the four template fields with corresponding dataset values to generate instance-specific prompts. It is noteworthy that during pre-training phases, we employ an alternative template configuration that excludes identity-related fields, relying solely on textual content for preference modeling.

3.3. Hybrid Encoding

The hybrid encoding component is designed to transform input prompts into latent representations compatible with LLM processing. As illustrated in Figure 1, our approach employs dual encoding pathways to handle different data types. For textual content, we utilize the LLM's inherent tokenization and embedding mechanisms to generate token embeddings. In contrast, for structured identifiers such as <UserID> and <TargetItemID>, we introduce a FECE Module to extract and encode collaborative signals for LLM utilization.

Formally, given a prompt corresponding to sample $(u, i, y) \in \mathcal{D}$, we first tokenize its textual content using the LLM Tokenizer, resulting in a token sequence $P = [t_1, t_2, \ldots, t_k, u, t_{k+1}, \ldots, i, \ldots, t_K]$, where t_k represents a text token and \$u/i denotes user/item identifiers within their respective fields. This sequence is subsequently encoded into embedding representations:

$$E = \left[e_{t_1}, \ldots, e_{t_k}, e_u, e_{t_{k+1}}, \ldots, e_i, \ldots, e_{t_K}\right] \tag{1}$$

where $e_{t_k} \in \mathbb{R}^{1\times d_2}$ signifies the token embedding for t_k obtained via the LLM's embedding layer, i.e., $e_{t_k} = Embedding_{LLM}(t_k)$; whereas $e_u/e_i \in \mathbb{R}^{1\times d_2}$ represents collaborative embeddings for user u and item i, generated through our collaborative encoding module.

FECE Module: This module comprises a conventional collaborative model $f_\psi(\cdot)$ and a mapping layer $g_\phi(\cdot)$ parameterized by ϕ. Given user u and item i, the collaborative model generates preliminary representations $\mathbf{u} = f_\psi(u; \mathcal{D})$ and $\mathbf{i} = f_\psi(i; \mathcal{D})$ encapsulating collaborative information. The mapping layer then projects these representations into the LLM's embedding space:

$$e_u = g_\phi(\mathbf{u}), \quad \mathbf{u} = f_\psi(u; \mathcal{D}) \tag{2}$$

$$e_i = g_\phi(\mathbf{i}), \quad \mathbf{i} = f_\psi(i; \mathcal{D}) \tag{3}$$

where $\mathbf{u}, \mathbf{i} \in \mathbb{R}^{1\times d_1}$ denote user and item representations from f_ψ, and the mapping layer g_ϕ is implemented as a MLP with input dimension d_1 and output dimension d_2 (typically $d_1 < d_2$).

During the training phase, the module operates without Faiss integration to ensure optimal accuracy and gradient propagation for learning high-quality collaborative representations. However, during inference, we leverage Faiss for efficient approximate nearest neighbor search, significantly accelerating retrieval speed in large-scale scenarios where processing the full item catalog becomes computationally prohibitive. This design achieves an effective balance between training accuracy and inference efficiency.

3.4. LLM Prediction

After converting the input prompt into embedding sequence E (see Equation 1), the LLM employs these representations for prediction generation. Since general-purpose LLMs lack specialized training for recommendation scenarios, we augment the base model with a Low-Rank Adaptation (LoRA) component [9] to facilitate recommendation-specific predictions, as illustrated in Figure 1. This integration involves incorporating pairs of rank-decomposition matrices into the original LLM parameters through a plug-in mechanism, enabling effective adaptation to new tasks (recommendation) with minimal parameter overhead. The prediction process is formally defined as:

$$\hat{y} = h_{\widehat{\Theta+\Theta'}}(E) \tag{4}$$

where Θ represents the frozen parameters of the pre-trained LLM $h(\cdot)$, Θ' denotes the trainable LoRA parameters dedicated to recommendation tasks, and $\hat{y}$ indicates the predicted probability of

positive interaction (i.e., the likelihood of the LLM responding "Yes" to the recommendation prompt). The LoRA implementation ensures parameter-efficient learning by updating only the adapter weights during task-specific optimization.

Our tuning methodology employs a two-stage strategy: initially optimizing the LoRA module for recommendation task adaptation, followed by FECE module fine-tuning for collaborative information extraction, ensuring stable knowledge integration in both cold-start and warm-start scenarios.

4. Experiments

4.1. Experimental Settings

All experiments are conducted on a Linux server with Ubuntu 20.04, using Miniconda3. The hardware configuration includes two GPUs (24GB VRAM each) and a CPU with 32 vCPUs, with CUDA 11.6 for GPU acceleration. Our experimental setup generally follows the configurations described in CoLLM [20].

4.1.1. Datasets

We evaluate our method on three publicly available real-world datasets: ML-1M, Amazon-Book, and BookCrossing.

ML-1M [7] is a widely-used movie recommendation dataset containing approximately 1 million ratings collected between 2000 and 2003. **Amazon-Book** [8] is derived from the Amazon Product Review dataset's book category, containing user reviews from 1996 to 2018. **BookCrossing** [24] is a book recommendation dataset containing both numerical ratings (1-10 scale) and textual metadata including book authors and titles. We follow [20] for ML-1M and Amazon-Book, and [1] for BookCrossing.

4.1.2. Algorithms

We evaluate the following methods in our experiments:

- **CoLLM-FaissRet (Ours)**: Our proposed approach treats collaborative information as a separate modality and integrates it into LLMs through direct mapping, incorporating Faiss for efficient retrieval to better handle large-scale cold-start and warm-start recommendation scenarios.
- **MF** [11]: A widely-used latent factor-based collaborative filtering technique based on Matrix Factorization.
- **CTRL (DIN)** [12]: A method that combines language and collaborative models through knowledge distillation, adopting DIN [22] as the collaborative model.
- **TALLRec** [1]: An LLM-based recommendation method that aligns large language models with recommendation tasks via instruction tuning, implemented using the Vicuna-7B model.

4.1.3. Experimental Factors

Our experimental design focuses on the following key factors:
*(1)*we perform extensive parameter optimization for CoLLM-FaissRet on the ML-1M and Amazon-Book datasets, examining embedding dimensions of $\{64,128,256\}$; *(2)* we conduct fine-grained learning rate adjustments for LoRA fine-tuning within the range of $\{1e-4,1e-3\}$; *(3)* we conduct fine-grained learning rate adjustments for FECE module mapping within the range of $\{1e-4,1e-3\}$; and *(4)* we conduct additional experiments on the BookCrossing dataset to validate model robustness.

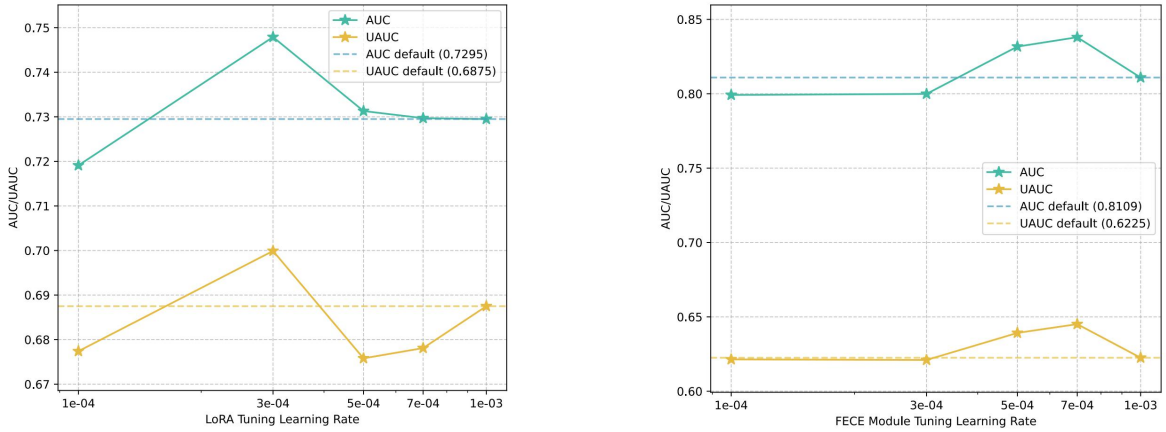
4.1.4. Measurement

To evaluate the performance of the studied methods, we employ two widely-used metrics for explicit recommendation:

- *Area Under the ROC Curve (AUC)*: This metric measures overall ranking accuracy by quantifying the area under the receiver operating characteristic curve.
- *User-based AUC (UAUC)* [15]: This metric is calculated by first computing AUC scores individually for each user based on their exposed items, then averaging these values across all users.

4.2. Results

4.2.1. Parameter Optimization



(a) AUC and UAUC on ML-1M as the LoRA tuning rate is swept from 1e-4 to 1e-3

(b) AUC and UAUC on Amazon-Book as the FECE module learning rate is varied over the same range

Figure 2: Impact of learning rate on CoLLM-FaissRet performance, all other parameters were fixed

Our parameter optimization methodology employs a two-stage strategy: initially optimizing the LoRA module for recommendation task adaptation, followed by FECE module fine-tuning for collaborative information extraction. We perform extensive parameter searches on the ML-1M and Amazon-Book datasets, examining embedding dimensions of {64,128,256}. For learning rates, we conduct fine-grained adjustments: for LoRA fine-tuning, we explore rates in $\{1e-3, 7e-4, 5e-4, 3e-4, 1e-4\}$ with a default of 1e-3; similarly, for the FECE module, we vary the learning rate over the same range. Due to space limitations, Figure 2 specifically presents the impact of two key learning rates: the LoRA fine-tuning rate on ML-1M (a) and the FECE module learing rate on Amazon-Book (b). Detailed results for other parameter optimization experiments can be found in *Appendix A.1*. It is worth noting that we did not explore larger learning rates beyond 1e-3 because preliminary experiments revealed that higher learning rates (e.g., 5e-3) caused training instability and significant performance degradation. Similarly, we constrained the embedding dimensions to a maximum of 256 due to GPU memory limitations, as larger dimensions would require substantially more computational resources that exceed our hardware capacity.

The experimental results demonstrate the strong performance the CoLLM-FaissRet framework across multiple datasets. As shown in Figure 2, careful tuning of learning rates significantly enhances model accuracy. Specifically, when setting the LoRA tuning rate to 3×10^{-4} on ML-1M

dataset, we achieve impressive AUC and UAUC scores of **0.7479** and **0.6999** respectively. Similarly, optimizing the FECE module learning rate to 7×10^{-4} yields even better performance with AUC reaching **0.8380** and UAUC at **0.6451**. These results confirm that appropriate learning rate configuration for different components of our framework can substantially improve recommendation accuracy.

4.2.2. Datasets Evaluation

We conduct additional experiments on the BookCrossing dataset to validate model robustness. The **BookCrossing** [24] dataset contains book ratings on a 1-10 scale and textual metadata including book authors and titles. We preprocess the data by binarizing ratings using a threshold of 5 to convert them into binary labels. For each user, we construct recommendation samples by randomly selecting one interacted item as the prediction target and 10 historical interactions as context. The dataset is randomly partitioned into training, validation, and testing sets with an 8:1:1 ratio to ensure fair evaluation.

Table 1 provides a comprehensive comparison with other algorithms, where CoLLM-FaissRet consistently demonstrates competitive performance. Particularly noteworthy is its robustness on the BookCrossing dataset, where our method achieves AUC of **0.6947** and UAUC of **0.6362**, outperforming several strong baselines. This validates the generalization capability of our approach across diverse recommendation scenarios. It is noteworthy that bold numbers in Table 1 indicate the best performance, and underlined numbers indicate the second best performance.

Table 1: Performance comparison between algorithms on three real-world datasets

Method	ML-1M		Amazon-Book		**BookCrossing**	
	AUC	UAUC	AUC	UAUC	AUC	UAUC
MF	0.6482	0.6361	0.7134	0.5565	0.6173	0.5478
CTRL	<u>0.7159</u>	0.6492	**0.8202**	<u>0.5996</u>	**0.7129**	0.5923
TallRec	0.7097	<u>0.6818</u>	0.7375	0.5983	0.6347	<u>0.6017</u>
CoLLM-FaissRet	**0.7295**	**0.6875**	<u>0.8109</u>	**0.6225**	<u>0.6947</u>	**0.6362**

4.2.3. Summary

The CoLLM-FaissRet framework achieves significant improvements in accuracy and robustness, with parameter optimization showing AUC gains of 2.52% for LoRA and 3.34% for FECE learning rate, and UAUC gains of 1.80% and 3.63%, respectively. On the BookCrossing dataset, CoLLM-FaissRet's AUC is competitive with the best method, while its UAUC surpasses the second-best by 5.73%.

5. Conclusion

This work proposes **CoLLM-FaissRet**, an novel framework that treats collaborative information as an independent modality for LLMRec systems. Our approach integrates direct mapping techniques with Faiss for efficient vector retrieval, enabling effective handling of both cold-start and warm-start recommendation scenarios.

Future research will explore alternative large language models and validate the inference efficiency advantages of CoLLM-FaissRet on larger real-world datasets. These investigations will further demonstrate the method's practical applicability in production environments.

[1] *Keqin Bao, Jizhi Zhang, Yang Zhang, Wenjie Wang, Fuli Feng, and Xiangnan He. Tallrec: An effective and efficient tuning framework to align large language model with recommendation. In Proceedings of the 17th ACM conference on recommender systems, pages 1007–1014, 2023.*

[2] *Keqin Bao, Jizhi Zhang, Wenjie Wang, Yang Zhang, Zhengyi Yang, Yanchen Luo, Chong Chen, Fuli Feng, and Qi Tian. A bi-step grounding paradigm for large language models in recommendation systems. ACM Transactions on Recommender Systems, 3(4):1–27, 2025.*

[3] *Sunhao Dai, Ninglu Shao, Haiyuan Zhao, Weijie Yu, Zihua Si, Chen Xu, Zhongxiang Sun, Xiao Zhang, and Jun Xu. Uncovering chatgpt's capabilities in recommender systems. In Proceedings of the 17th ACM Conference on Recommender Systems, pages 1126–1132, 2023.*

[4] *Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, PierreEmmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The faiss library. arXiv preprint arXiv:2401.08281, 2024.*

[5] *Yue Feng, Shuchang Liu, Zhenghai Xue, Qingpeng Cai, Lantao Hu, Peng Jiang, Kun Gai, and Fei Sun. A large language model enhanced conversational recommender system. arXiv preprint arXiv:2308.06212, 2023.*

[6] *Yunfan Gao, Tao Sheng, Youlin Xiang, Yun Xiong, Haofen Wang, and Jiawei Zhang. Chat-rec: Towards interactive and explainable llms-augmented recommender system. arXiv preprint arXiv:2303.14524, 2023.*

[7] *F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. Acm transactions on interactive intelligent systems (tiis), 5(4):1–19, 2015.*

[8] *Ruining He and Julian McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In proceedings of the 25th international conference on world wide web, pages 507–517, 2016.*

[9] *Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. ICLR, 1 (2):3, 2022.*

[10] *Xu Huang, Jianxun Lian, Yuxuan Lei, Jing Yao, Defu Lian, and Xing Xie. Recommender ai agent: Integrating large language models for interactive recommendations. ACM Transactions on Information Systems, 43(4):1–33, 2025.*

[11] *Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. Computer, 42(8):30–37, 2009.*

[12] *Xiangyang Li, Bo Chen, Lu Hou, and Ruiming Tang. Ctrl: Connect collaborative and language model for ctr prediction. ACM Transactions on Recommender Systems, 2023.*

[13] *Jianghao Lin, Rong Shan, Chenxu Zhu, Kounianhua Du, Bo Chen, Shigang Quan, Ruiming Tang, Yong Yu, and Weinan Zhang. Rella: Retrieval-enhanced large language models for lifelong sequential behavior comprehension in recommendation. In Proceedings of the ACM Web Conference 2024, pages 3497–3508, 2024.*

[14] *Junling Liu, Chao Liu, Peilin Zhou, Renjie Lv, Kang Zhou, and Yan Zhang. Is chatgpt a good recommender? a preliminary study. arXiv preprint arXiv:2304.10149, 2023.*

[15] *Yiyu Liu, Qian Liu, Yu Tian, Changping Wang, Yanan Niu, Yang Song, and Chenliang Li. Concept-aware denoising graph neural network for micro-video recommendation. In Proceedings of the 30th ACM international conference on information & knowledge management, pages 1099–1108, 2021.*

[16] *Aashiq Muhamed, Iman Keivanloo, Sujan Perera, James Mracek, Yi Xu, Qingjun Cui, Santosh Rajagopalan, Belinda Zeng, and Trishul Chilimbi. Ctr-bert: Cost-effective knowledge distillation for billion-parameter teacher models. In NeurIPS Efficient Natural Language and Speech Processing Workshop, 2021.*

[17] *Jerry Wei, Jason Wei, Yi Tay, Dustin Tran, Albert Webson, Yifeng Lu, Xinyun Chen, Hanxiao Liu, Da Huang, Denny Zhou, et al. Larger language models do in-context learning differently. arXiv preprint arXiv:2303.03846, 2023.*

[18] *Likang Wu, Zhi Zheng, Zhaopeng Qiu, Hao Wang, Hongchao Gu, Tingjia Shen, Chuan Qin, Chen Zhu, Hengshu Zhu, Qi Liu, et al. A survey on large language models for recommendation. World Wide Web, 27(5):60, 2024.*

[19] *Junjie Zhang, Ruobing Xie, Yupeng Hou, Xin Zhao, Leyu Lin, and Ji-Rong Wen. Recommendation as instruction following: A large language model empowered recommendation approach. ACM Transactions on Information Systems, 43(5):1–37, 2025.*

[20] *Yang Zhang, Fuli Feng, Jizhi Zhang, Keqin Bao, Qifan Wang, and Xiangnan He. Collm: Integrating collaborative embeddings into large language models for recommendation. IEEE Transactions on Knowledge and Data Engineering, 2025.*

[21] *Bowen Zheng, Yupeng Hou, Hongyu Lu, Yu Chen, Wayne Xin Zhao, Ming Chen, and Ji-Rong Wen. Adapting large language models by integrating collaborative semantics for recommendation. In 2024 IEEE 40th International Conference on Data Engineering (ICDE), pages 1435–1448. IEEE, 2024.*

[22] *Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. Deep interest network for click-through rate prediction. In Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining, pages 1059–1068, 2018.*

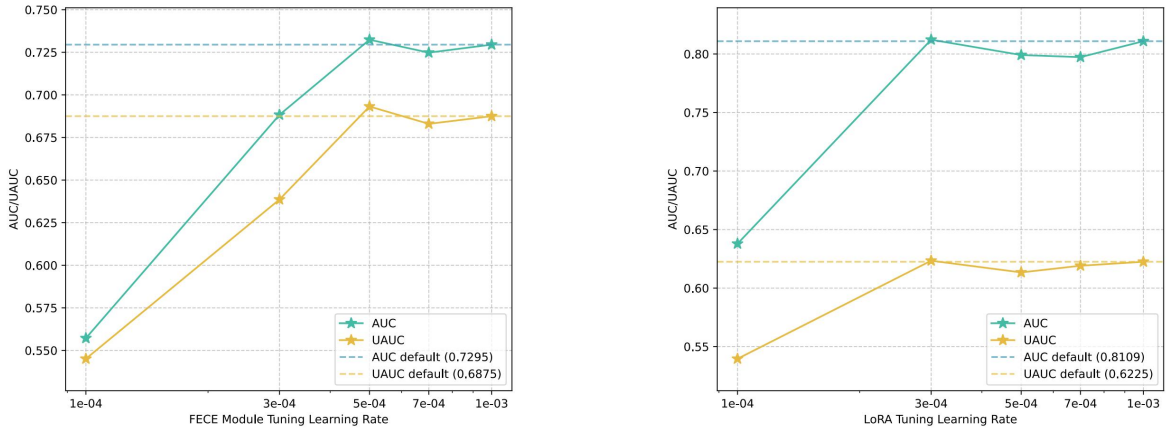
[23] *Yaochen Zhu, Liang Wu, Qi Guo, Liangjie Hong, and Jundong Li. Collaborative large language model for recommender systems. In Proceedings of the ACM Web Conference 2024, pages 3162–3172, 2024.*

[24] *Cai-Nicolas Ziegler, Sean M McNee, Joseph A Konstan, and Georg Lausen. Improving recommendation lists through topic diversification. In Proceedings of the 14th international conference on World Wide Web, pages 22–32, 2005.*

Appendix

A. Additional Experimental Results

A.1 Parameter Optimization



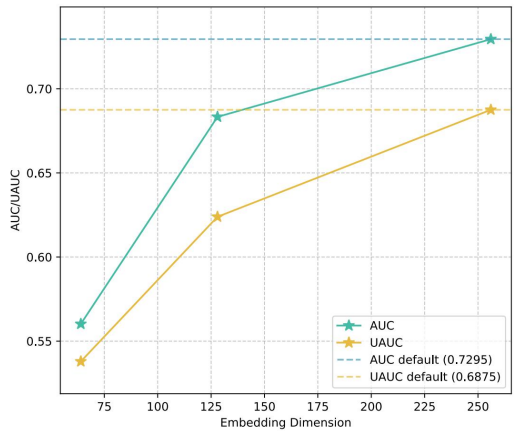
(a) AUC and UAUC on ML-1M as the FECE module learning rate is swept from 1e-4 to 1e-3

(b) AUC and UAUC on Amazon-Book as the LoRA tuning rate is varied over the same range

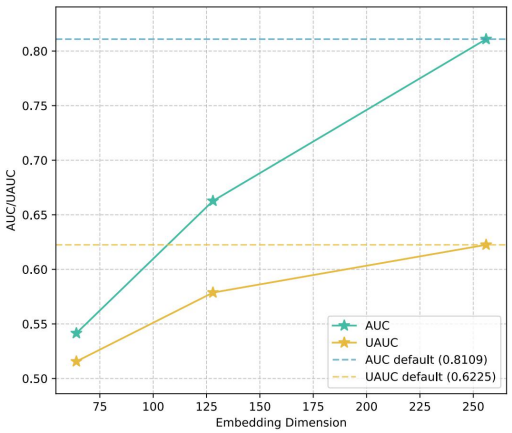
Figure A.1: Impact of learning rate on CoLLM-FaissRet performance, all other parameters were fixed

Our parameter optimization methodology employs a two-stage strategy: initially optimizing the LoRA module for recommendation task adaptation, followed by FECE module fine-tuning for collaborative information extraction. We perform extensive parameter searches on the ML-1M and Amazon-Book datasets, examining embedding dimensions of $\{64,128,256\}$. For learning rates, we conduct fine-grained adjustments: for LoRA fine-tuning, we explore rates in $\{1e-3,7e-4,5e-4,3e-4,1e-4\}$ with a default of 1e-3; similarly, for the FECE module, we vary the learning rate over the same range. In the appendix, we provide additional experimental results: Figure A.1 presents the impact of two key learning rates: the FECE module learning rate on ML-1M (a) and the LoRA fine-tuning rate on Amazon-Book (b). Additionally, Figure A.2 shows the impact of embedding dimensions on CoLLM-FaissRet performance, with the left panel displaying AUC and UAUC on ML-1M for embedding dimensions $\{64,128,256\}$ and the right panel showing the same for Amazon-Book, with all other parameters fixed.

We observe that the parameter optimization, including both learning rate and embedding dimensions, does not lead to significant performance improvements. The experimental results demonstrate only minor fluctuations in AUC and UAUC metrics across different parameter configurations. It is worth noting that we did not explore larger learning rates beyond 1e-3 because preliminary experiments revealed that higher learning rates (e.g., 5e-3) caused training instability and significant performance degradation. Similarly, we constrained the embedding dimensions to a maximum of 256 due to GPU memory limitations, as larger dimensions would require substantially more computational resources that exceed our hardware capacity.



(a) AUC and UAUC on ML-1M examining embedding dimensions of {64,128,256}



(b) AUC and UAUC on Amazon-Book examining embedding dimensions of {64,128,256}

Figure A.1: Impact of embedding dimensions on CoLLM-FaissRet performance, all other parameters were fixed