CoVE: Compressed Vocabulary Expansion Makes Better LLM-based Recommender Systems

Haochen Zhang[1], Tianyi Zhang[1], Junze Yin[1], Oren Gal[2], Anshumali Shrivastava[1], Vladimir Braverman[1,3]

[1]Rice University, [2]Swarm & AI, University of Haifa, [3]John Hopkins University

Correspondence: hz112@rice.edu

Abstract

Recommender systems play a pivotal role in providing relevant content to users. With the rapid development of large language models (LLMs), researchers have begun utilizing LLMs to build more powerful recommender systems. However, existing approaches that focus on aligning LLMs with recommendation tasks do not fully leverage their sequential information processing capabilities, leading to suboptimal performance.

In this paper, we propose a novel system called *compressed vocabulary expansion (CoVE)*. In CoVE, each item is assigned a unique ID within the expanded vocabulary. Our framework effectively capitalizes on sequence understanding abilities of LLMs, significantly enhancing their performance on recommendation tasks. Additionally, we compress the embedding layer, making CoVE practical for large-scale industrial applications. The effectiveness and performance of CoVE are demonstrated through comprehensive experiments on multiple recommendation datasets and comparisons with prior works. Our code can be found at https://github.com/HaochenZhang717/CoVE-official-Repo.

1 Introduction

Large language models (LLMs), like GPT series (Radford, 2018; Radford et al., 2019; Brown et al., 2020; Achiam et al., 2023a), LLaMA (Touvron et al., 2023a,c), and Deepseek (Guo et al., 2025), have demonstrated strong capabilities for reasoning and problem-solving (Achiam et al., 2023b; Touvron et al., 2023b; Almazrouei et al., 2023). LLMs are typically pretrained on a large corpus and then finetuned on smaller datasets for downstream tasks (Xiong et al., 2024; Rafailov et al., 2024; Wei et al., 2021a). The pretrain-and-finetune paradigm achieves the state-of-the-art performance in many tasks (Wang, 2024; Huang et al., 2022).

On the other hand, in current big data era, people increasingly face challenges related to information overload (Boka et al., 2024). This brings sequential recommendation problems, where a recommender system analyzes a user's historical interaction data to predict the next item that is likely to interest them (Boz et al., 2024). With the rise of LLMs, a large body of research (Harte et al., 2023; Liu et al., 2024b; Hu et al., 2024; Wang et al., 2024; Boz et al., 2024; Yuan et al., 2024) has been dedicated to developing proper frameworks to effectively leverage LLMs for sequential recommender systems.

Among existing works, there are two main ways to utilize LLMs for recommender systems. One approach is to use embeddings provided by LLMs to initialize non-LLM recommender systems. For example, Yuan et al. (2023); Boz et al. (2024) show that LLM embeddings improve the performance of several non-LLM deep learning sequential recommender models (Sun et al., 2019; Kang and McAuley, 2018; Hidasi, 2015). Despite the positive results from this pipeline, using LLMs solely as embedding providers takes advantage of their embedding ability but does not effectively leverage their content comprehension capability. This can potentially result in suboptimal performance.

The other approach is finetuning LLMs to directly provide recommendations (Bao et al., 2023a; Boz et al., 2024) which use the pipeline consisting of 1) finetuning LLMs on a recommendation dataset, 2) using finetuned LLMs to generate the next item's title, and 3) conducting embedding retrieval to avoid hallucinated item titles. Bao et al. (2023a) shows that this pipeline produces more accurate recommendations than prior works: it leverages LLMs' token prediction ability, which may address the limitations of the first line of work. However, there are issues in the design of this framework. 1) Under this framework, LLM needs to accurately predict the next item's title, which consists of multiple tokens, and this is a difficult

Findings of the Association for Computational Linguistics: ACL 2025, pages 12575–12591
July 27 - August 1, 2025

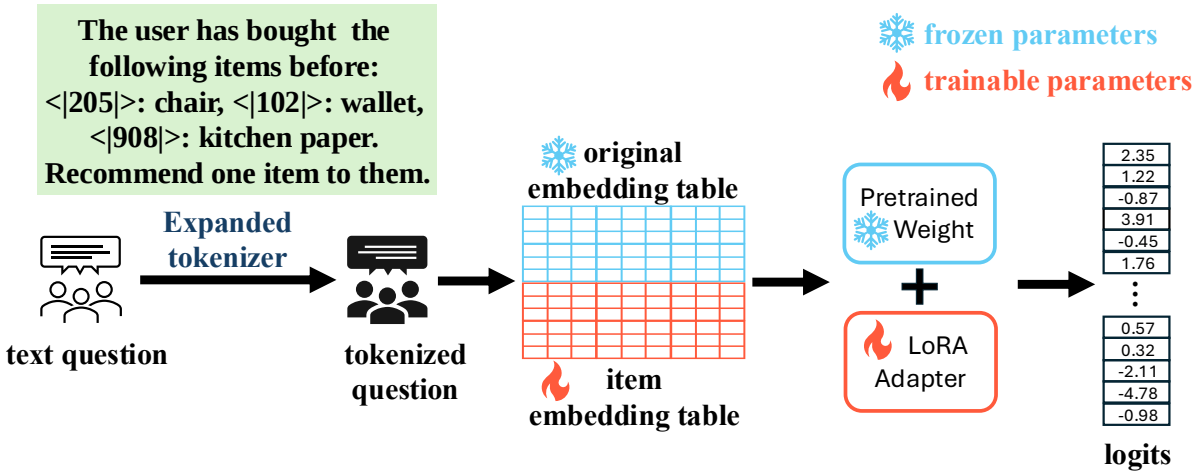


Figure 1: An overview of our CoVE framework. Each item is assigned a unique ID in the tokenizer. It is then mapped to a distinct embedding. After that, our framework finetunes item embeddings, transformer weights, and the lm_head to align LLMs with sequential recommendation tasks.

task. 2) The recommendation hallucination problem is inevitable because the generated item's title may not exist in the item space. Though embedding retrieval can be a remedy, this solution is far from ideal. 3) Generating text with LLM is very time-consuming.

As an alternative to the two broad lines of LLM-based recommendation, we propose a novel system herein called *compressed vocabulary expansion (CoVE)* which is significantly different from the existing pipeline. CoVE first expands the tokenizer's vocabulary and embedding table by assigning each item in the item space a unique token in the tokenizer and a corresponding unique embedding in the embedding table. During finetuning, the item embedding table is updated together with the LoRA adapter (Hu et al., 2021) to align the pretrained LLM with the recommendation task, as shown in Figure 1. During inference, CoVE can provide a recommendation by generating one token representing the unique ID of the next item, instead of generating the full title. The advantage of CoVE over existing frameworks is three-fold:

1. The expanded vocabulary enables CoVE to directly leverage the next-token prediction capability of the LLM, eliminating any chance of hallucination.

2. CoVE naturally enables leveraging temporal correlations by providing the entire context to the LLM as a sequence and utilizing next-token prediction.

3. During inference, CoVE operates in a non-generative manner, outputting logits and making recommendations based on them. This significantly speeds up inference compared to the aforementioned framework.

The inspiration for CoVE comes from vocabulary expansion for domain-specific LLM alignment (Cui et al., 2023; Wang et al., 2023; Liu et al., 2024a), where it has been shown that expanding the vocabulary in a downstream domain improves an LLM's ability in content comprehension and generation. However, in CoVE the memory cost of an additional embedding table is prohibitive in large-scale industrial scenarios, where the item space may contain millions of products. For instance, The newest version of the Amazon Review dataset contains 48.19 million items in total[1]. When finetuning on such a large dataset, the item embedding table itself takes up approximately 96GB in GPU memory, which makes training infeasible for even the most powerful GPUs. Therefore, a key research question that we need to address in this paper is:

How can we make CoVE memory-efficient while preserving its superior performance compared to the state-of-the-art results?

The solution to this problem is using embedding layer compression methods which have been explored by prior works (Desai et al., 2022a; Guan et al., 2019; Pansare et al., 2022) to alleviate the memory problem brought by large embedding tables. Inspired by them, we propose to use a hash function to compress the item embedding table to

[1] https://amazon-reviews-2023.github.io.

make CoVE more memory efficient. In summary, the contribution of this study is as follows:

- We propose a novel finetuning framework, CoVE, to align pretrained LLMs with sequential recommendation tasks. CoVE fully leverages LLMs' sequential information processing capabilities, achieving up to a 62% improvement in recommendation accuracy over existing methods.
- CoVE achieves approximately 100 times faster inference speed than the existing finetune-and-retrieval framework due to the expanded vocabulary.
- Embedding layer compression ensures the memory efficiency of CoVE. Experiments show that we can compress the embedding table by up to 16 times while still achieving better results than prior methods.

2 Proposed Framework

Finetuning Task Formulation Our goal is to leverage an LLM to construct a recommender system. To achieve this, we finetune the LLM on a recommendation dataset. The key difference between our finetuning task and common finetuning tasks is that we also tune the embedding table and lm_head. Table 1 provides a sample from the finetuning dataset. During training, the Task Instruction, Task Input, and Task Output are fed into the LLM, which is trained to minimize the next-token prediction loss. At inference, we provide only the Task Instruction and Task Input to the LLM and expect the first token it generates to be the unique ID of the predicted item. In this way, CoVE does not require the LLM to generate the complete name of an item as in Bao et al. (2023a). Instead, obtaining the output of lm_head, i.e., logits, is sufficient for making predictions. The logits form a vector whose dimensionality equals the sum of the number of tokens in the original vocabulary and the cardinality of the item space, denoted as $|\mathcal{I}|$. At inference, we only need to consider the last $|\mathcal{I}|$ entries of the logits for item prediction.

CoVE As shown in Figure 1, CoVE involves tokenizer vocabulary expansion, item embedding table expansion, and a low-rank weight adapter. The core idea of CoVE is to assign each item a unique embedding. To achieve this, we expand the tokenizer's vocabulary by adding item IDs, such as <|205|>, <|102|>, and <|908|> in Figure 1. These item IDs are then encoded as distinct embeddings. CoVE finetunes item embeddings, weights in transformer blocks, and weights in lm_head to align pretrained LLMs with sequential recommendation tasks.

Hashing-based Embedding Compression Due to the large item space, item embeddings must be compressed to enable efficient GPU training. We propose a hashing-based embedding compression technique, inspired by Shi et al. (2020a). Let $\mathcal{I}$ denote the item space and $\mathcal{S}$ the latent shared item space, where $|\mathcal{S}| \ll |\mathcal{I}|$. We define $\mathcal{H}$ as a universal hash family, where each hash function $h : \mathcal{I} \to \mathcal{S}$ maps items to the shared item space. We sample k such hash functions, denoted as $h_1, \ldots, h_k$. Each shared item $s \in \mathcal{S}$ is assigned a high-dimensional embedding, and an item's embedding is obtained by averaging the embeddings of its hashed counterparts in the shared space. Specifically, the embedding for an item i is given by

$$\frac{1}{k} \sum_{j=1}^{k} \mathbf{e}_{h_j(i)},$$

where $\mathbf{e}_s$ represents the high-dimensional embedding of the latent shared item s.

We select a universal hash function based on simple arithmetic operations–addition, multiplication, and modulo–which can be efficiently computed on GPUs. Specifically, given an item's index i, the hash code for its shared embedding is computed as

$$h(i) = ((ai + b) \bmod p) \bmod |\mathcal{S}|,$$

where p is a large prime number, and $a \in \{1, \ldots, p\}$ and $b \in \{0, \ldots, p\}$ are randomly chosen parameters.

3 Experiments

3.1 Experiment Settings

Datasets We conduct experiments on four public real-world benchmarks: Amazon Video Games (McAuley, 2024), Amazon Beauty (He and McAuley, 2016), Amazon Toys & Games (He and McAuley, 2016), and Amazon Sports & Outdoors (He and McAuley, 2016). The statistics of these datasets are shown in Table 2. For Amazon Beauty, Amazon Toys & Games, and Amazon Sports & Outdoors, we compare CoVE with the results reported in (Rajput et al., 2023); therefore, we adopt the same train-test split as in (Rajput et al., 2023).

Task Instruction:	Given a list of video games the user has played before, please recommend a new video game that the user likes to the user.
Task Input:	Here is the list of video games that the user has played before: <|16659|>: "Homeworld - PC", <|5641|>: "Starcraft", <|4375|>: "StarCraft Expansion Pack: Brood War - PC", <|14747|>: "Sid Meier's Alpha Centauri - PC"
Task Output:	<|16673|>: "Total Annihilation Gold (Mac)"

Table 1: A tuning sample for the recommendation system. <|·|> denotes the unique ID for each item in item space.

For Amazon Video Games, we compare CoVE with the results reported in (Bao et al., 2023a); hence, we adopt the same train-test split as in (Bao et al., 2023a).

Architecture, Hyperparameters, and Implementation For experiments on Amazon Beauty, Toys & Games, and Sports & Outdoors, we use LLaMA-3.2-3B (Dubey et al., 2024) as the backbone model and finetune it for up to 10 epochs. We set the learning rate to 10^{-4}, the batch size to 32, the LoRA rank to 8, and the LoRA alpha to 16. For experiments on Amazon Video Games, we maintain the same hyperparameters but change our backbone model to LLaMA-2-7B (Touvron et al., 2023c) and apply 4-bit QLoRA for a fair comparison with BIGRec (Bao et al., 2023a). All experiments are conducted on an NVIDIA A100 GPU.

Datasets	# items	# sequences
Video Games	17,408	149,796
Beauty	12,101	22,363
Toys & Games	11,924	19,412
Sports & Outdoors	18,357	35,598

Table 2: Statistics of datasets.

Baselines To verify the efficacy of the proposed framework, we compare it with various prior methods, including GRU4Rec (Hidasi, 2015), BERT4Rec (Sun et al., 2019), Caser (Tang and Wang, 2018), SASRec (Kang and McAuley, 2018), P5 (Geng et al., 2022), HGN (Ma et al., 2019), S^3-Rec (Zhou et al., 2020), FDSA (Zhang et al., 2019), TIGER (Rajput et al., 2023), DROS (Yang et al., 2023), GPT4Rec (Li et al., 2023a), and BIGRec (Bao et al., 2023a). A brief description of these methods is provided in Appendix A. BIGRec and CoVE finetune LLMs to build recommender systems, whereas BIGRec adopts a finetune-and-retrieval framework. The rest of the above work do not directly finetune LLMs.

Metrics Like many existing works, we adopt normalized discounted cumulative gain at rank K (NG@K, or NGK) (Wang et al., 2013; Gienapp et al., 2020; Jayashree and Christy, 2015) and hit rate at rank K (HR@K, or HRK) (Elahi and Zirak, 2024). These metrics help quantify how effectively a system ranks relevant items for a given user. Specifically, NG@K measures the ranking quality of recommended items by considering both the relevance of items and their positions in the ranked list and is defined as:

$$NG@K = \frac{\sum_{i=1}^{K} \frac{\mathrm{rel}_i}{\log_2(i+1)}}{\sum_{i=1}^{K} \frac{\mathrm{rel}_i^*}{\log_2(i+1)}},$$

where rel_i is the relevance score of the item at position i and rel_i^* represents the relevance scores of the items sorted in optimal order. Because of this construction, we can see that $NG@K \in [0, 1]$, where 1 denotes the best possible ranking.

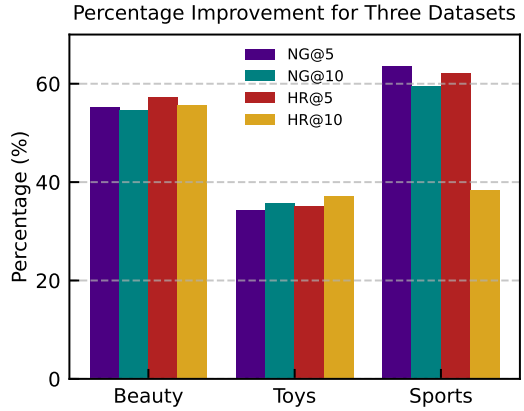
HR@K, on the other hand, measures the percentage of users for whom at least one relevant item appears in the top K recommendations and is defined as:

$$HR@K = \frac{\sum_{u=1}^{U} \mathbb{I}(\exists i \in \{1, \ldots, K\}, \mathrm{rel}_{u,i} > 0)}{U},$$

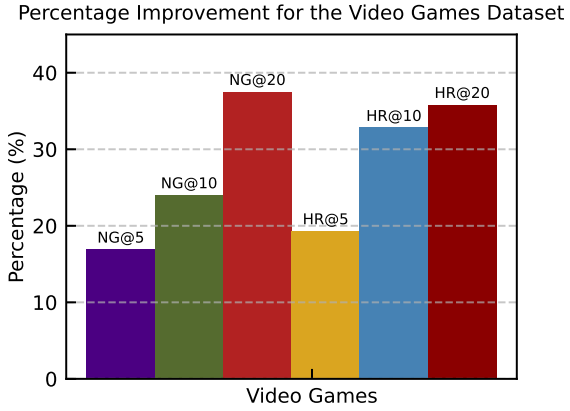
where U is the total number of users, $\mathbb{I}(\cdot)$ is an indicator function, and $\mathrm{rel}_{u,i}$ is the relevance of item i for user u. In our experiments, we have shown that our pipeline CoVE results in higher NG@K and HR@K for various choices of K.

3.2 Main Results

Recommendation Accuracy Table 3 compares CoVE with prior works on the Beauty, Toys & Games, and Sports & Outdoors datasets. The results show that CoVE provides recommendations more consistent with users' interests. The percentage improvement of CoVE over prior works



(a) The percentage improvement of CoVE compared with the state-of-the-art results in NG@5, NG@10, HR@5, and HR@10.



(b) The percentage improvement of CoVE compared to BigRec on the Video Games dataset. BigRec uses finetune-and-retrieval framework.

Figure 2: The percentage improvement of CoVE over the best metric achieved by TIGER (Rajput et al., 2023) (left) BIGRec (Bao et al., 2023a) (right) on four different datasets: "Beauty", "Toys", "Sports", and "Video Games". The metric values for CoVE and prior works on these datasets are presented in Tables 3 and 4.

Dataset	Metric	GRU 4 Rec	BERT 4 Rec	Caser	SAS Rec	P5	HGN	S^3-Rec	FDSA	TIGER	CoVE
Beauty	**NG5**	0.0099	0.0124	0.0131	0.0249	0.0107	0.0206	0.0244	0.0163	0.0321	**0.0498**
	NG10	0.0137	0.0170	0.0176	0.0318	0.0136	0.0266	0.0327	0.0208	0.0384	**0.0593**
	HR5	0.0164	0.0203	0.0205	0.0387	0.0163	0.0325	0.0387	0.0267	0.0454	**0.0714**
	HR10	0.0283	0.0347	0.0347	0.0605	0.0254	0.0512	0.0647	0.0407	0.0648	**0.1009**
Toys	**NG5**	0.0059	0.0071	0.0107	0.0306	0.0050	0.0221	0.0294	0.0140	0.0371	**0.0509**
	NG10	0.0084	0.0099	0.0141	0.0374	0.0066	0.0277	0.0376	0.0189	0.0432	**0.0595**
	HR5	0.0097	0.0116	0.0166	0.0463	0.0070	0.0321	0.0443	0.0228	0.0521	**0.0719**
	HR10	0.0176	0.0203	0.0270	0.0675	0.0121	0.0497	0.0700	0.0381	0.0712	**0.0986**
Sports	**NG5**	0.0086	0.0075	0.0072	0.0192	0.0041	0.0120	0.0204	0.0156	0.0181	**0.0296**
	NG10	0.0110	0.0099	0.0097	0.0249	0.0052	0.0159	0.0240	0.0156	0.0225	**0.0359**
	HR5	0.0129	0.0115	0.0116	0.0233	0.0061	0.0189	0.0251	0.0182	0.0264	**0.0428**
	HR10	0.0204	0.0191	0.0194	0.0350	0.0095	0.0313	0.0385	0.0288	0.0400	**0.0624**

Table 3: Performance comparison of different recommendation models across three datasets: Beauty, Toys and Games, and Sports and Outdoors. The results of CoVE shown in the table use a compression rate of 2. The best metrics are always achieved by using CoVE and are all in bold font in this table, and the best metric achieved by prior works (see the **Baselines** paragraph in Section 3.1) is underlined.

on these three datasets ranges between 30% and 62%, as shown in Figure 2 (a). In Table 4, we compare CoVE with prior works on the Video Games dataset, where BIGRec achieves the second-best performance. The comparison between CoVE and BIGRec effectively contrasts two different approaches for finetuning LLMs in sequential recommendation tasks. Therefore, we conclude that the CoVE framework is more suitable than the finetune-and-retrieval framework for aligning LLMs with sequential recommendation problems. Figure 2 (b) shows that CoVE consistently outperforms BIGRec in NG@k and HR@k, and as k increases, the percentage improvements become larger.

Datasets	Model	NG5	NG10	NG20	HR5	HR10	HR20
Video Games	GRU4Rec	0.0018	0.0024	0.0030	0.0024	0.0041	0.0069
	Caser	0.0019	0.0024	0.0035	0.0032	0.0048	0.0092
	SASRec	0.0015	0.0024	0.0035	0.0021	0.0037	0.0057
	P5	0.0007	0.0010	0.0017	0.0012	0.0023	0.0049
	DROS	0.0013	0.0016	0.0022	0.0019	0.0027	0.0052
	GPT4Rec-LLaMA	0.0000	0.0001	0.0001	0.0000	0.0002	0.0002
	BIGRec	0.0189	0.0216	0.0248	0.0243	0.0329	0.0457
	CoVE	**0.0221**	**0.0268**	**0.0341**	**0.0290**	**0.0437**	**0.0621**

Table 4: Performance comparison of different recommender systems on Video Game dataset. The best metrics are always achieved by using CoVE and are all in bold font in this table, while the best metric achieved by prior works (see the **Baselines** paragraph in Section 3.1) is underlined.

Inference Time Another advantage of CoVE is that it avoids generating multiple tokens during inference and instead provides recommendations based on the logits output. The inference speeds of BIGRec and CoVE are 0.066 and 6.5 samples per second respectively. CoVE achieves an almost 100× speed-up compared to the finetune-and-retrieval framework.

3.3 Ablation Study

Datasets	I./E.	NG5	NG10	HR5	HR10
Beauty	**E.**	0.045	0.0519	0.0622	0.0836
	I.	0.0057	0.0084	0.0094	0.0178
	Both	0.0498	0.0593	0.0714	0.1009
Toys	**E.**	0.0429	0.0482	0.0566	0.0732
	I.	0.0058	0.0085	0.0095	0.018
	Both	0.0509	0.0595	0.0719	0.0986
Sports	**E.**	0.0213	0.0255	0.0308	0.0439
	I.	0.0053	0.0069	0.0079	0.0128
	Both	0.0296	0.0359	0.0428	0.0624

Table 5: Performance metrics of CoVE under different settings. **I.** represents using prompts with the item's title information and setting the item embedding table as frozen. **E.** represents using prompts without item's title information and setting the item embedding table as trainable. **Both** represents using prompts with item's title information and setting the item embedding table as trainable, which is the default setting of CoVE. We present three datasets across four metrics.

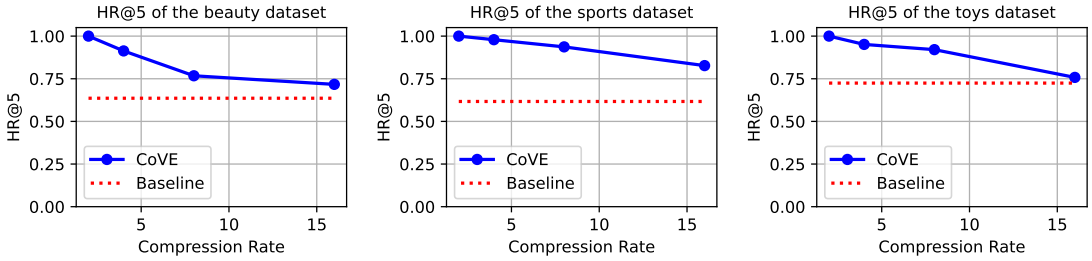
Importance of Item Title Our prompt includes item titles as important textual information, as shown in Table 1. Here, we evaluate the importance of such text information in CoVE. For comparison, we finetune the same model with the same hyperparameters but use a different prompt format, where item titles are removed. Below is a tuning sample without items' titles:

- **Task Instruction:** Given a list of video games the user has played, please recommend a new video game that the user likes to the user.
- **Task Input:** Here is the list of video games that the user has played: <|16659|>, <|5641|>, <|4375|>, <|14747|>, where <|·|> denotes the unique ID for each item in item space.
- **Task Output:** <|16673|>.

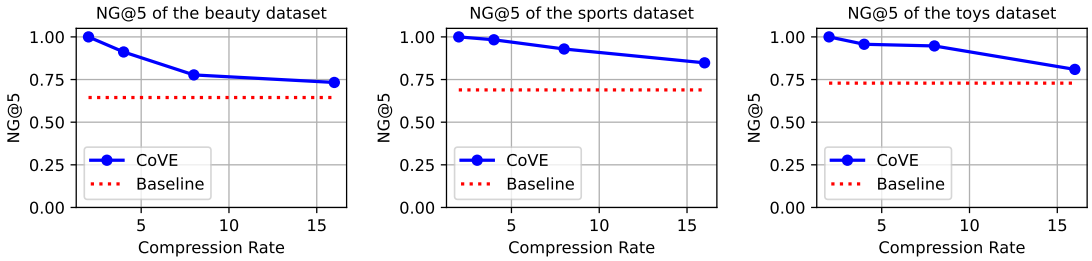
In Table 5, we present the evaluation metrics of CoVE with the two different prompts. We observe that CoVE without title information does not provide recommendations as effectively as CoVE with title information. However, CoVE does not completely fail in the absence of title information, which can be attributed to the strong sequence understanding ability of LLMs.

Importance of Tuning Embedding Table CoVE tunes both the item embedding table and transformer weights. We analyze the importance of tuning the embedding table by comparing the performance of CoVE with a trainable item embedding table to its performance with a frozen item embedding table. As in Table 5, the results indicate that CoVE with a frozen item embedding table performs very poorly. This suggests that learning high-quality embeddings for item IDs is crucial.

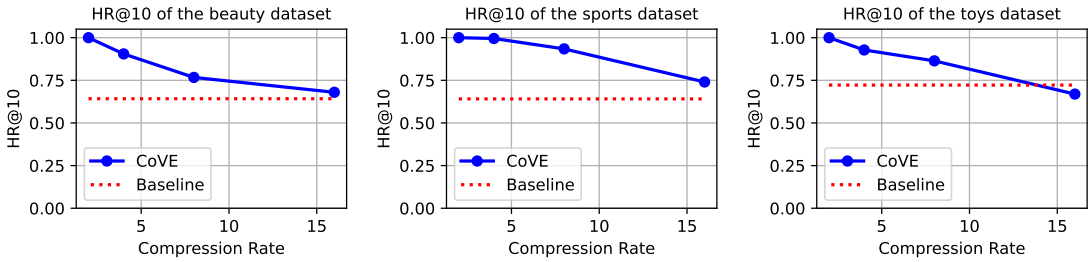
Different Embedding Layer Compression Rate Embedding layer compression plays a crucial role



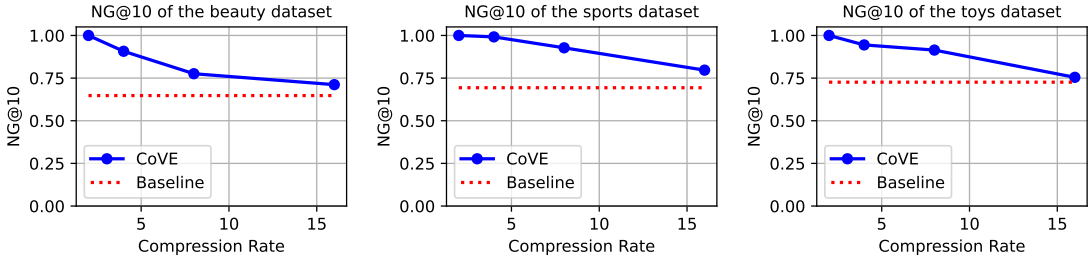
(a) HR@5 comparison between CoVE and baseline methods on the Beauty, Sports, and Toys datasets.



(b) NG@5 comparison between CoVE and baseline methods on the Beauty, Sports, and Toys datasets.



(c) HR@10 comparison between CoVE and baseline methods on the Beauty, Sports, and Toys datasets.



(d) NG@10 comparison between CoVE and baseline methods on the Beauty, Sports, and Toys datasets.

Figure 3: Normalized metrics, HR@5, NG@5, HR@10, and NG@10, of CoVE under different compression rates $(2, 4, 8, 16)$, where the original embedding layer is compressed to $1/2, 1/4, 1/8$, and $1/16$ of its original size, respectively. We compare four metrics across three datasets, resulting in twelve figures, showcasing CoVE alongside the state-of-the-art baseline. Notably, even when the embedding layer is compressed 16-fold, CoVE still outperforms the state-of-the-art baseline with only one exception (HR@10 of the toys dataset).

in CoVE, especially in large-scale industrial scenarios. Through a comparison of CoVE with different compression rates ranging from 2 to 16, we find that the performance of CoVE degrades as the compression rate increases, as shown in Figure 3. However, even with a compression rate of 16, CoVE still outperforms the baseline in HR@5 and NG@5. We also observe that the robustness of CoVE to compression rates varies across datasets. For instance, CoVE remains highly robust to compression rates up to 8 on the Sports & Outdoors and Toys & Games datasets. However, CoVE is more sensitive to higher compression rates on the Beauty dataset, where normalized NG@5 and HR@5 degrade to 0.75 at a compression rate of 8.

To visualize the performance of CoVE with

higher compression rate, we provide performance of CoVE on the Beauty dataset with compression beyond 16, as shown in Figure 4.

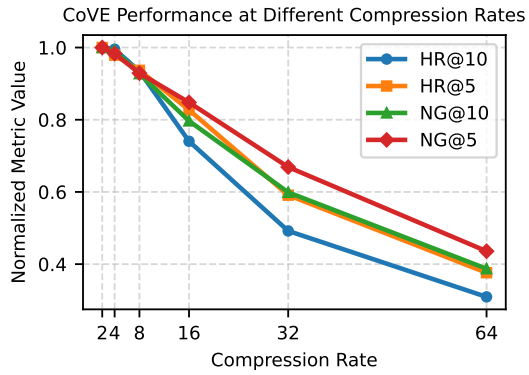


Figure 4: Performance of CoVE on the Beauty dataset under increasing embedding compression rates (2× to 64×).

4 A Case Study

In this section, we analyze what an LLM can learn after finetuning under the CoVE framework. To show this, we give an example of the input and the generated output of finetuned LLM by CoVE:

- **Input:** Instruction: Given a list of items the user has bought before, please recommend a new item that the user likes to the user. Here is the list of items that the user has bought: <|1|>: "Hasbro Electronic Catch Phrase", <|2|>: "Gloom", <|3|>: "Cards Against Humanity", <|4|>: "Carcassonne Basic Game", <|5|>: "Asmodee 7 Wonders Wonder Pack".

- **Generated Output:** <|5|>: "Asmodee 7 Wonders Wonder Pack", <|1441|>: "Cards Against Humanity: First Expansion".

The generated content strictly follows the format of the user's interaction history, i.e., each item's ID is followed by its title. We find that the finetuned LLM consistently outputs the correct item title corresponding to its ID. This indicates that under the CoVE framework, the finetuned LLM learns the mapping between an item's ID and its title. We argue that this is crucial for high-quality recommendations. One key advantage of CoVE is its ability to convert the item title prediction task into an item ID prediction task. While the latter is easier than the former—since the number of tokens to predict is significantly smaller—item ID prediction is meaningful only when the LLM correctly understands the relationship between IDs and titles.

5 Related Work

LLMs in Sequential Recommendation Currently, LLMs are primarily utilized in recommendation systems in two ways: providing embeddings for items (Zhang et al., 2021; Wu et al., 2021a; Liu et al., 2021b) and directly generating recommendations (Bao et al., 2023a,b; Hegselmann et al., 2023; Li et al., 2023a; Ngo and Nguyen, 2024; Zhang et al., 2024).

Our work is more related to generative recommendation systems. For instance, Bao et al. (2023b) proposes a finetuning framework to align LLMs with recommendations, where the task is defined as a Yes-No question-answering problem. However, under this framework, generating the next item for users is challenging. GPT4Rec (Li et al., 2023a) integrates LLMs with a search engine, where the LLM generates multiple queries representing the user's interests, and the search engine outputs the final recommendation. The existing work most similar to our approach is the finetune-retrieval framwork adopted in (Bao et al., 2023a; Boz et al., 2024). During training, BIGRec finetunes an LLM on a recommendation dataset. During inference, BIGRec computes the distances between the embedding of the sentence generated by the finetuned LLM and the embeddings of all items in the item space. Based on these distances, BIGRec determines the next recommended item for the user. Our approach is similar to BIGRec in the training stage; however, in the inference stage, we do not need to generate content using LLMs. Instead, we provide recommendations directly based on logits of items. This approach reduces inference time and eliminates hallucinations. The trade-off is a larger embedding layer, which we mitigate through embedding layer compression. Liu et al. (2025) studies the long-tail problem in sequential recommendation by enhancing item embeddings for rare items. It uses a pre-trained LLM (LLaMA-7B) and adapts it to the recommendation domain via contrastive fine-tuning (SCFT) and recommendation adaptation, and focuses on preserving semantic richness while integrating collaborative signals.

Vocabulary Expansion of LLMs Our study extends the vocabulary of pretrained LLMs to assign each item a unique embedding, thereby enhancing

the performance of LLM-based recommendation systems. This technique is closely related to vocabulary expansion in domain-specific LLMs. Pretrained LLMs are typically trained on a large corpus that differs in distribution from domain-specific corpora. Thus, vocabulary expansion has become a key strategy to adapt LLMs pretrained on general datasets to domain-specific applications, improving decoding efficiency and semantic comprehension ability (Liu et al., 2024a; Kajiura et al., 2023).

For instance, Cui et al. (2023) continue training LLaMA models on a Chinese corpus with additional tokens, leading to improved proficiency in understanding and generating Chinese content. Liu et al. (2023b) propose a task-specific tokenization approach to adapt the generation pipeline for the mental health domain. Similarly, Liu et al. (2024a) introduce an adaptive method for vocabulary expansion in domain-specific LLMs. Liu et al. (2023a) train a new tokenizer on domain-specific data and expand the existing general-purpose tokenizer by incorporating newly identified tokens to adapt LLMs for chip design.

Embedding Layer Compression One branch of the embedding layer compression technique is low-precision methods which focus on reducing the number of bits used to represent each weight in the embedding table. Due to variations in bit width and their unique advantages, low-precision methods can be further categorized into binarization and quantization.

Binarization methods compress full-precision weights into binary codes (1-bit representation) (Liu et al., 2014; Zhang et al., 2014; Lian et al., 2017; Zhang et al., 2017; Liu et al., 2018). Other notable works include discrete collaborative filtering (DCF) (Zhang et al., 2016), which learns binary embeddings for collaborative filtering; candidate generation and re-ranking (CIGAR) (Kang and McAuley, 2019), which utilizes the scaled $\tanh$ function for binary approximation; hashing with graph neural networks (HashGNN) (Tan et al., 2020); and low-loss quantized graph convolutional networks (L^2Q-GCN) (Chen et al., 2021b), which propose variations using straight-through estimators for gradient computation during training.

Quantization methods map 32-bit floating-point weights to lower-precision representations using multiple bits rather than just one (Xu et al., 2021). Moreover, Guan et al. (2019); Yang et al. (2020) study both uniform and non-uniform quantization techniques. To better preserve model accuracy, Li et al. (2023b) proposes adaptive low-precision training (ALPT), which adaptively adjusts quantization step sizes during training.

Generating Item Embeddings The use of LLMs in recommendation systems marks a significant shift from traditional embedding approaches. While earlier techniques primarily relied on collaborative filtering (Schafer et al., 2007) or simple word embeddings (Musto et al., 2015), LLMs offer a more sophisticated understanding of items through their pre-trained knowledge and contextual processing capabilities.

Several studies have demonstrated the effectiveness of LLM-generated embeddings in capturing complex item characteristics and relationships. For instance, Zhang et al. (2021) and Wu et al. (2021a) both examine the application of pre-trained language models to news recommendations. Specifically, Zhang et al. (2021) propose a BERT-based user-news matching model that not only leverages the extensive language knowledge of the pre-trained model to enhance textual representation but also captures multi-grained user-news matching signals at both the word and news levels. In contrast, Wu et al. (2021a) explore different variations of the BERT model. Additionally, Liu et al. (2021b) investigate a recent state-of-the-art Chinese pre-trained language model called Enhanced Representation through Knowledge Integration (ERNIE).

6 Conclusion

We propose a novel framework, CoVE, to better align LLMs with sequential recommendation problems. The core idea of CoVE is to assign each item a unique ID that can be encoded as a token and given a unique embedding. The item embedding table is finetuned along with the weights in transformer blocks during training. CoVE outperforms baselines by up to 62% in NG@5, NG@10, HR@5, and HR@10 on Amazon review datasets. Additionally, CoVE achieves approximately 100$\times$ speed-up in inference time compared to the existing finetune-and-retrieval framework. Through comprehensive ablation experiments, we find that including an item's textual information enhances CoVE's performance. From case studies, we observe that CoVE effectively memorizes ID-title pairs for different items, which is crucial for providing high-quality recommendations. In this paper, we used AI tools solely for language assistance.

Limitations

Embedding layer compression is an important part of CoVE. In this study, we adopt hash function to compress embedding layers. There are more advanced compression method that can be further explored. Besides, from the memory efficiency perspective, more works can be done on the robustness of CoVE to other memory-efficient methods, such as memory-efficient optimizers, quantization, low-rank approximation, etc.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023a. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023b. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Mérouane Debbah, Étienne Goffinet, Daniel Hesslow, Julien Launay, Quentin Malartic, et al. 2023. The falcon series of open language models. *arXiv preprint arXiv:2311.16867*.

Keqin Bao, Jizhi Zhang, Wenjie Wang, Yang Zhang, Zhengyi Yang, Yancheng Luo, Chong Chen, Fuli Feng, and Qi Tian. 2023a. A bi-step grounding paradigm for large language models in recommendation systems. *arXiv preprint arXiv:2308.08434*.

Keqin Bao, Jizhi Zhang, Yang Zhang, Wenjie Wang, Fuli Feng, and Xiangnan He. 2023b. 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.

Tesfaye Fenta Boka, Zhendong Niu, and Rama Bastola Neupane. 2024. A survey of sequential recommendation systems: Techniques, evaluation, and future directions. *Information Systems*, page 102427.

Artun Boz, Wouter Zorgdrager, Zoe Kotti, Jesse Harte, Panos Louridas, Vassilios Karakoidas, Dietmar Jannach, and Marios Fragkoulis. 2024. Improving sequential recommendations with llms. *ACM Transactions on Recommender Systems*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Tong Chen, Hongzhi Yin, Yujia Zheng, Zi Huang, Yang Wang, and Meng Wang. 2021a. Learning elastic embeddings for customizing on-device recommenders. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 138–147.

Yankai Chen, Yifei Zhang, Yingxue Zhang, Huifeng Guo, Jingjie Li, Ruiming Tang, Xiuqiang He, and Irwin King. 2021b. Towards low-loss 1-bit quantization of user-item representations for top-k recommendation. *arXiv preprint arXiv:2112.01944*.

Yifang Chen, Jiayan Huo, Xiaoyu Li, Yingyu Liang, Zhenmei Shi, and Zhao Song. 2024. Fast gradient computation for rope attention in almost linear time. *arXiv preprint arXiv:2412.17316*.

Yifang Chen, Xiaoyu Li, Yingyu Liang, Zhenmei Shi, and Zhao Song. 2025a. The computational limits of state-space models and mamba via the lens of circuit complexity. In *The Second Conference on Parsimony and Learning (Proceedings Track)*.

Yifang Chen, Xiaoyu Li, Yingyu Liang, Zhenmei Shi, and Zhao Song. 2025b. Universal approximation of visual autoregressive transformers. *arXiv preprint arXiv:2502.06167*.

Yifang Chen, Xiaoyu Li, Yingyu Liang, Zhenmei Shi, Zhao Song, and Yu Tian. 2025c. Time and memory trade-off of kv-cache compression in tensor transformer decoding. *arXiv preprint arXiv:2503.11108*.

Weiyu Cheng, Yanyan Shen, and Linpeng Huang. 2020. Differentiable neural input search for recommender systems. *arXiv preprint arXiv:2006.04466*.

Yiming Cui, Ziqing Yang, and Xin Yao. 2023. Efficient and effective text encoding for chinese llama and alpaca. *arXiv preprint arXiv:2304.08177*.

Wei Deng, Junwei Pan, Tian Zhou, Deguang Kong, Aaron Flores, and Guang Lin. 2021. Deeplight: Deep lightweight feature interactions for accelerating ctr predictions in ad serving. In *Proceedings of the 14th ACM international conference on Web search and data mining*, pages 922–930.

Aditya Desai, Li Chou, and Anshumali Shrivastava. 2022a. Random offset block embedding (robe) for compressed embedding tables in deep learning recommendation systems. *Proceedings of Machine Learning and Systems*, 4:762–778.

Aditya Desai, Li Chou, and Anshumali Shrivastava. 2022b. Random offset block embedding (robe) for compressed embedding tables in deep learning recommendation systems. *Proceedings of Machine Learning and Systems*, 4:762–778.

Aditya Desai, Yanzhou Pan, Kuangyuan Sun, Li Chou, and Anshumali Shrivastava. 2021. Semantically constrained memory allocation (scma) for embedding in efficient recommendation systems. *arXiv preprint arXiv:2103.06124*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Ali Elahi and Armin Zirak. 2024. Online and offline evaluations of collaborative filtering and content based recommender systems. *arXiv preprint arXiv:2411.01354*.

Yeqi Gao, Zhao Song, Weixin Wang, and Junze Yin. 2025a. A fast optimization view: Reformulating single layer attention in llm based on tensor and svm trick, and solving it in matrix multiplication time. In *The 41st Conference on Uncertainty in Artificial Intelligence*.

Yeqi Gao, Zhao Song, and Junze Yin. 2023. Gradientcoin: A peer-to-peer decentralized large language models. *arXiv preprint arXiv:2308.10502*.

Yeqi Gao, Zhao Song, and Junze Yin. 2025b. An iterative algorithm for rescaled hyperbolic functions regression. In *The 28th International Conference on Artificial Intelligence and Statistics*.

Shijie Geng, Shuchang Liu, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. 2022. Recommendation as language processing (rlp): A unified pretrain, personalized prompt & predict paradigm (p5). In *Proceedings of the 16th ACM Conference on Recommender Systems*, pages 299–315.

Lukas Gienapp, Benno Stein, Matthias Hagen, and Martin Potthast. 2020. Estimating topic difficulty using normalized discounted cumulated gain. In *Proceedings of the 29th ACM international conference on information & knowledge management*, pages 2033–2036.

Antonio A Ginart, Maxim Naumov, Dheevatsa Mudigere, Jiyan Yang, and James Zou. 2021. Mixed dimension embeddings with application to memory-efficient recommendation systems. In *2021 IEEE International Symposium on Information Theory (ISIT)*, pages 2786–2791. IEEE.

Hui Guan, Andrey Malevich, Jiyan Yang, Jongsoo Park, and Hector Yuen. 2019. Post-training 4-bit quantization on embedding tables. *arXiv preprint arXiv:1911.02079*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Jesse Harte, Wouter Zorgdrager, Panos Louridas, Asterios Katsifodimos, Dietmar Jannach, and Marios Fragkoulis. 2023. Leveraging large language models for sequential recommendation. In *Proceedings of the 17th ACM Conference on Recommender Systems*, pages 1096–1102.

Ruining He and Julian McAuley. 2016. 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.

Stefan Hegselmann, Alejandro Buendia, Hunter Lang, Monica Agrawal, Xiaoyi Jiang, and David Sontag. 2023. Tabllm: Few-shot classification of tabular data with large language models. In *International Conference on Artificial Intelligence and Statistics*, pages 5549–5581. PMLR.

B Hidasi. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Jun Hu, Wenwen Xia, Xiaolu Zhang, Chilin Fu, Weichang Wu, Zhaoxin Huan, Ang Li, Zuoli Tang, and Jun Zhou. 2024. Enhancing sequential recommendation via llm-based semantic embedding learning. In *Companion Proceedings of the ACM on Web Conference 2024*, pages 103–111.

Shaoyi Huang, Dongkuan Xu, Ian EH Yen, Yijue Wang, Sung-En Chang, Bingbing Li, Shiyang Chen, Mimi Xie, Sanguthevar Rajasekaran, Hang Liu, et al. 2022. Sparse progressive distillation: Resolving overfitting under pretrain-and-finetune paradigm. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*.

R Jayashree and A Christy. 2015. Improving the enhanced recommended system using bayesian approximation method and normalized discounted cumulative gain. *Procedia Computer Science*, 50:216–222.

Gangwei Jiang, Hao Wang, Jin Chen, Haoyu Wang, Defu Lian, and Enhong Chen. 2021. xlightfm: Extremely memory-efficient factorization machine. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 337–346.

Manas R Joglekar, Cong Li, Mei Chen, Taibai Xu, Xiaoming Wang, Jay K Adams, Pranav Khaitan, Jiahui Liu, and Quoc V Le. 2020. Neural input search for large scale recommendation models. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2387–2397.

Teruno Kajiura, Shiho Takano, Tatsuya Hiraoka, and Kimio Kuramitsu. 2023. Vocabulary replacement in sentencepiece for domain adaptation. In *Proceedings of the 37th Pacific Asia Conference on Language, Information and Computation*, pages 645–652.

Wang-Cheng Kang, Derek Zhiyuan Cheng, Ting Chen, Xinyang Yi, Dong Lin, Lichan Hong, and Ed H Chi. 2020. Learning multi-granular quantized embeddings for large-vocab categorical features in recommender systems. In *Companion Proceedings of the Web Conference 2020*, pages 562–566.

Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*, pages 197–206. IEEE.

Wang-Cheng Kang and Julian McAuley. 2019. Candidate generation with binary codes for large-scale top-n recommendation. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pages 1523–1532.

Jinming Li, Wentao Zhang, Tian Wang, Guanglei Xiong, Alan Lu, and Gerard Medioni. 2023a. Gpt4rec: A generative framework for personalized recommendation and user interests interpretation. *arXiv preprint arXiv:2304.03879*.

Shiwei Li, Huifeng Guo, Lu Hou, Wei Zhang, Xing Tang, Ruiming Tang, Rui Zhang, and Ruixuan Li. 2023b. Adaptive low-precision training for embeddings in click-through rate prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 4435–4443.

Shiwei Li, Huifeng Guo, Xing Tang, Ruiming Tang, Lu Hou, Ruixuan Li, and Rui Zhang. 2024. Embedding compression in recommender systems: A survey. *ACM Computing Surveys*, 56(5):1–21.

Zhihang Li, Zhao Song, Zifan Wang, and Junze Yin. 2023c. Local convergence of approximate newton method for two layer nonlinear regression. *arXiv preprint arXiv:2311.15390*.

Defu Lian, Rui Liu, Yong Ge, Kai Zheng, Xing Xie, and Longbing Cao. 2017. Discrete content-aware matrix factorization. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 325–334.

Defu Lian, Haoyu Wang, Zheng Liu, Jianxun Lian, Enhong Chen, and Xing Xie. 2020. Lightrec: A memory and search-efficient recommender system. In *Proceedings of The Web Conference 2020*, pages 695–705.

Jiehao Liang, Zhao Song, Zhaozhuo Xu, Junze Yin, and Danyang Zhuo. 2025. Dynamic maintenance of kernel density estimation data structure: From practice to theory. In *The 41st Conference on Uncertainty in Artificial Intelligence*.

Yingyu Liang, Heshan Liu, Zhenmei Shi, Zhao Song, Zhuoyan Xu, and Junze Yin. 2024. Conv-basis: A new paradigm for efficient attention inference and gradient computation in transformers. *arXiv preprint arXiv:2405.05219*.

Chengyuan Liu, Shihang Wang, Lizhi Qing, Kun Kuang, Yangyang Kang, Changlong Sun, and Fei Wu. 2024a. Gold panning in vocabulary: An adaptive method for vocabulary expansion of domain-specific llms. *arXiv preprint arXiv:2410.01188*.

Han Liu, Xiangnan He, Fuli Feng, Liqiang Nie, Rui Liu, and Hanwang Zhang. 2018. Discrete factorization machines for fast feature-based recommendation. *arXiv preprint arXiv:1805.02232*.

Haochen Liu, Xiangyu Zhao, Chong Wang, Xiaobing Liu, and Jiliang Tang. 2020. Automated embedding size search in deep recommender systems. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2307–2316.

Mingjie Liu, Teodor-Dumitru Ene, Robert Kirby, Chris Cheng, Nathaniel Pinckney, Rongjian Liang, Jonah Alben, Himyanshu Anand, Sanmitra Banerjee, Ismet Bayraktaroglu, et al. 2023a. Chipnemo: Domain-adapted llms for chip design. *arXiv preprint arXiv:2311.00176*.

Qidong Liu, Xian Wu, Wanyu Wang, Yejing Wang, Yuanshao Zhu, Xiangyu Zhao, Feng Tian, and Yefeng Zheng. 2025. Llmemb: Large language model can be a good embedding generator for sequential recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 12183–12191.

Qidong Liu, Xian Wu, Yejing Wang, Zijian Zhang, Feng Tian, Yefeng Zheng, and Xiangyu Zhao. 2024b. Llm-esr: Large language models enhancement for long-tailed sequential recommendation. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Siyang Liu, Naihao Deng, Sahand Sabour, Yilin Jia, Minlie Huang, and Rada Mihalcea. 2023b. Task-adaptive tokenization: Enhancing long-form text generation efficacy in mental health and beyond. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 15264–15281.

Siyi Liu, Chen Gao, Yihong Chen, Depeng Jin, and Yong Li. 2021a. Learnable embedding sizes for recommender systems. *arXiv preprint arXiv:2101.07577*.

Xianglong Liu, Junfeng He, Cheng Deng, and Bo Lang. 2014. Collaborative hashing. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2139–2146.

Yiding Liu, Weixue Lu, Suqi Cheng, Daiting Shi, Shuaiqiang Wang, Zhicong Cheng, and Dawei Yin. 2021b. Pre-trained language model for web-scale retrieval in baidu search. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 3365–3375.

Fuyuan Lyu, Xing Tang, Hong Zhu, Huifeng Guo, Yingxue Zhang, Ruiming Tang, and Xue Liu. 2022. Optembed: Learning optimal embedding table for click-through rate prediction. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 1399–1409.

Chen Ma, Peng Kang, and Xue Liu. 2019. Hierarchical gating networks for sequential recommendation. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 825–833.

Julian McAuley. 2024. Amazon product data. Accessed: 2025-02-15.

Cataldo Musto, Giovanni Semeraro, Marco De Gemmis, Pasquale Lops, et al. 2015. Word embedding techniques for content-based recommender systems: An empirical evaluation. *Recsys posters*, 1441.

Hoang Ngo and Dat Quoc Nguyen. 2024. Recgpt: Generative pre-training for text-based recommendation. *arXiv preprint arXiv:2405.12715*.

Niketan Pansare, Jay Katukuri, Aditya Arora, Frank Cipollone, Riyaaz Shaik, Noyan Tokgozoglu, and Chandru Venkataraman. 2022. Learning compressed embeddings for on-device inference. *Proceedings of Machine Learning and Systems*, 4:382–397.

Liang Qu, Yonghong Ye, Ningzhi Tang, Lixin Zhang, Yuhui Shi, and Hongzhi Yin. 2022. Single-shot embedding dimension search in recommender system. In *Proceedings of the 45th International ACM SIGIR conference on research and development in Information Retrieval*, pages 513–522.

Alec Radford. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36.

Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan Hulikal Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Tran, Jonah Samost, et al. 2023. Recommender systems with generative retrieval. *Advances in Neural Information Processing Systems*, 36:10299–10315.

J Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. 2007. Collaborative filtering recommender systems. In *The adaptive web: methods and strategies of web personalization*, pages 291–324. Springer.

Hao-Jun Michael Shi, Dheevatsa Mudigere, Maxim Naumov, and Jiyan Yang. 2020a. Compositional embeddings using complementary partitions for memory-efficient recommendation systems. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 165–175.

Hao-Jun Michael Shi, Dheevatsa Mudigere, Maxim Naumov, and Jiyan Yang. 2020b. Compositional embeddings using complementary partitions for memory-efficient recommendation systems. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 165–175.

Zhao Song, Chongxi Wang, Guangyi Xu, and Junze Yin. 2025. The expressibility of polynomial based attention scheme. In *The 31st SIGKDD Conference on Knowledge Discovery and Data Mining*.

Zhao Song, Weixin Wang, and Junze Yin. 2023. A unified scheme of resnet and softmax. *arXiv preprint arXiv:2309.13482*.

Zhao Song, Junze Yin, and Lichen Zhang. 2024. Solving attention kernel regression problem via preconditioner. In *International Conference on Artificial Intelligence and Statistics*, pages 208–216. PMLR.

Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pages 1441–1450.

Yang Sun, Fajie Yuan, Min Yang, Guoao Wei, Zhou Zhao, and Duo Liu. 2020. A generic network compression framework for sequential recommender systems. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1299–1308.

Qiaoyu Tan, Ninghao Liu, Xing Zhao, Hongxia Yang, Jingren Zhou, and Xia Hu. 2020. Learning to hash with graph neural networks for recommender systems. In *Proceedings of The Web Conference 2020*, pages 1988–1998.

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

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023a. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal

Azhar, et al. 2023b. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023c. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Haochun Wang, Chi Liu, Nuwa Xi, Zewen Qiang, Sendong Zhao, Bing Qin, and Ting Liu. 2023. Huatuo: Tuning llama model with chinese medical knowledge. *arXiv preprint arXiv:2304.06975*.

Yining Wang, Liwei Wang, Yuanzhi Li, D He, W Chen, and TY Liu. 2013. A theoretical analysis of normalized discounted cumulative gain (ndcg) ranking measures. In *Proc. 26th Annual Conference on Learning Theory (COLT 2013)*. Citeseer.

Yu Wang. 2024. On finetuning large language models. *Political Analysis*, 32(3):379–383.

Yuxiang Wang, Xin Shi, and Xueqing Zhao. 2024. Mllm4rec: multimodal information enhancing llm for sequential recommendation. *Journal of Intelligent Information Systems*, pages 1–17.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021a. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

Zhikun Wei, Xin Wang, and Wenwu Zhu. 2021b. Autoias: Automatic integrated architecture searcher for click-trough rate prediction. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 2101–2110.

Chuhan Wu, Fangzhao Wu, Tao Qi, and Yongfeng Huang. 2021a. Empowering news recommendation with pre-trained language models. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*, pages 1652–1656.

Xiaorui Wu, Hong Xu, Honglin Zhang, Huaming Chen, and Jian Wang. 2020. Saec: Similarity-aware embedding compression in recommendation systems. In *Proceedings of the 11th ACM SIGOPS Asia-Pacific Workshop on Systems*, pages 82–89.

Yongji Wu, Defu Lian, Neil Zhenqiang Gong, Lu Yin, Mingyang Yin, Jingren Zhou, and Hongxia Yang. 2021b. Linear-time self attention with codeword histogram for efficient recommendation. In *Proceedings of the Web Conference 2021*, pages 1262–1273.

Wei Xiong, Hanze Dong, Chenlu Ye, Ziqi Wang, Han Zhong, Heng Ji, Nan Jiang, and Tong Zhang. 2024. Iterative preference learning from human feedback: Bridging theory and practice for rlhf under kl-constraint. In *Forty-first International Conference on Machine Learning*.

Zhiqiang Xu, Dong Li, Weijie Zhao, Xing Shen, Tianbo Huang, Xiaoyun Li, and Ping Li. 2021. Agile and accurate ctr prediction model training for massive-scale online advertising systems. In *Proceedings of the 2021 international conference on management of data*, pages 2404–2409.

Bencheng Yan, Pengjie Wang, Jinquan Liu, Wei Lin, Kuang-Chih Lee, Jian Xu, and Bo Zheng. 2021a. Binary code based hash embedding for web-scale applications. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 3563–3567.

Bencheng Yan, Pengjie Wang, Kai Zhang, Wei Lin, Kuang-Chih Lee, Jian Xu, and Bo Zheng. 2021b. Learning effective and efficient embedding via an adaptively-masked twins-based layer. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 3568–3572.

Jie Amy Yang, Jianyu Huang, Jongsoo Park, Ping Tak Peter Tang, and Andrew Tulloch. 2020. Mixed-precision embedding using a cache. *arXiv preprint arXiv:2010.11305*.

Zhengyi Yang, Xiangnan He, Jizhi Zhang, Jiancan Wu, Xin Xin, Jiawei Chen, and Xiang Wang. 2023. A generic learning framework for sequential recommendation with distribution shifts. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 331–340.

Wei Yuan, Chaoqun Yang, Guanhua Ye, Tong Chen, Nguyen Quoc Viet Hung, and Hongzhi Yin. 2024. Fellas: Enhancing federated sequential recommendation with llm as external services. *ACM Transactions on Information Systems*.

Zheng Yuan, Fajie Yuan, Yu Song, Youhua Li, Junchen Fu, Fei Yang, Yunzhu Pan, and Yongxin Ni. 2023. Where to go next for recommender systems? id-vs. modality-based recommender models revisited. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2639–2649.

Caojin Zhang, Yicun Liu, Yuanpu Xie, Sofia Ira Ktena, Alykhan Tejani, Akshay Gupta, Pranay Kumar Myana, Deepak Dilipkumar, Suvadip Paul, Ikuhiro Ihara, et al. 2020. Model size reduction using frequency based double hashing for recommender systems. In *Proceedings of the 14th ACM Conference on Recommender Systems*, pages 521–526.

Hanwang Zhang, Fumin Shen, Wei Liu, Xiangnan He, Huanbo Luan, and Tat-Seng Chua. 2016. Discrete collaborative filtering. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 325–334.

Haochen Zhang, Xi Chen, and Lin F Yang. 2023a. Adaptive liquidity provision in uniswap v3 with

deep reinforcement learning. *arXiv preprint arXiv:2309.10129*.

Haochen Zhang, Xingyu Lin, Sui Peng, Junjie Tang, Antonello Monti, et al. 2023b. Surrogate-model-based sequential algorithm for weather-dependent probabilistic power flow with high calculation efficiency. *Authorea Preprints*.

Haochen Zhang, Zhiyun Peng, Junjie Tang, Ming Dong, Ke Wang, and Wenyuan Li. 2022. A multi-layer extreme learning machine refined by sparrow search algorithm and weighted mean filter for short-term multi-step wind speed forecasting. *Sustainable Energy Technologies and Assessments*, 50:101698.

Haochen Zhang, Junze Yin, Guanchu Wang, Zirui Liu, Tianyi Zhang, Anshumali Shrivastava, Lin Yang, and Vladimir Braverman. 2025a. I3s: Importance sampling subspace selection for low-rank optimization in llm pretraining. *arXiv preprint arXiv:2502.05790*.

Qi Zhang, Jingjie Li, Qinglin Jia, Chuyuan Wang, Jieming Zhu, Zhaowei Wang, and Xiuqiang He. 2021. Unbert: User-news matching bert for news recommendation. In *IJCAI*, volume 21, pages 3356–3362.

Tingting Zhang, Pengpeng Zhao, Yanchi Liu, Victor S Sheng, Jiajie Xu, Deqing Wang, Guanfeng Liu, Xiaofang Zhou, et al. 2019. Feature-level deeper self-attention network for sequential recommendation. In *IJCAI*, pages 4320–4326.

Xiaokun Zhang, Bo Xu, Youlin Wu, Yuan Zhong, Hongfei Lin, and Fenglong Ma. 2024. Finerec: Exploring fine-grained sequential recommendation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1599–1608.

Yan Zhang, Defu Lian, and Guowu Yang. 2017. Discrete personalized ranking for fast collaborative filtering from implicit feedback. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31.

Zhi Zhang, Chris Chow, Yasi Zhang, Yanchao Sun, Haochen Zhang, Eric Hanchen Jiang, Han Liu, Furong Huang, Yuchen Cui, and Oscar Hernan Madrid Padilla. 2025b. Statistical guarantees for lifelong reinforcement learning using pac-bayesian theory. In *The 28th International Conference on Artificial Intelligence and Statistics*.

Zhiwei Zhang, Qifan Wang, Lingyun Ruan, and Luo Si. 2014. Preference preserving hashing for efficient recommendation. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 183–192.

Xiangyu Zhao, Haochen Liu, Hui Liu, Jiliang Tang, Weiwei Guo, Jun Shi, Sida Wang, Huiji Gao, and Bo Long. 2020. Memory-efficient embedding for recommendations. *arXiv preprint arXiv:2006.14827*.

Xiangyu Zhaok, Haochen Liu, Wenqi Fan, Hui Liu, Jiliang Tang, Chong Wang, Ming Chen, Xudong Zheng, Xiaobing Liu, and Xiwang Yang. 2021. Autoemb: Automated embedding dimensionality search in streaming recommendations. In *2021 IEEE International Conference on Data Mining (ICDM)*, pages 896–905. IEEE.

Kun Zhou, Hui Wang, Wayne Xin Zhao, Yutao Zhu, Sirui Wang, Fuzheng Zhang, Zhongyuan Wang, and Ji-Rong Wen. 2020. S3-rec: Self-supervised learning for sequential recommendation with mutual information maximization. In *Proceedings of the 29th ACM international conference on information & knowledge management*, pages 1893–1902.

A Description of Prior Methods

GRU4Rec (Hidasi, 2015) applies a GRU-based RNN for sequential recommendations. BERT4Rec (Sun et al., 2019) is a bidirectional self-attention model for the sequential recommendation problem. Caser (Tang and Wang, 2018) embeds recent items in the iteration sessions into latent spaces and uses convolutional filters to learn sequential patterns. SASRec (Kang and McAuley, 2018) proposes a self-attention-based sequential model to balance the advantage of Markov Chains and the advantage of RNN. P5 (Geng et al., 2022) converts various data formats into natural language and builds on pretrained T5 to learn sequential interaction patterns. HGN (Ma et al., 2019) adopts gating modules to control the flow of item features and learn item relations. S^3-Rec (Zhou et al., 2020) applies self-supervised learning to learn correlations among data. FDSA (Zhang et al., 2019) integrates different features of items into feature sequences and then models item transition patterns and feature transition patterns. TIGER (Rajput et al., 2023) first learns semantic IDs of different items and then trains a generative recommendation system based on the semantic IDs. DROS (Yang et al., 2023) is a framework to enhance the generalization of recommendation systems that is more suitable for streaming data. GPT4Rec (Li et al., 2023a) introduces a novel generative framework for personalized recommendation systems that addresses several limitations of existing NLP-based recommender systems, such as treating items as only IDs and using discriminative modeling. BIGRec (Bao et al., 2023a) uses a finetune-and-retrieval framework to align LLM with sequential recommendation tasks.

B More Related Work

B.1 Embedding Layer Compression in Recommendation Systems

Embedding layer compression techniques for recommender systems can be broadly categorized into three main areas, low-precision methods, mixed-dimension methods, and weight-sharing methods (Li et al., 2024). Each category addresses different aspects of reducing memory usage while maintaining model performance.

Mixed Dimension Second, mixed-dimension techniques optimize memory usage by assigning different embedding dimensions to different features. Similarly, due to the different characteristics that prior works focus on, mixed-dimension techniques can be further categorized into rule-based approaches, neural architecture search (NAS) based approaches, and pruning.

Rule-based approaches, such as those proposed by Ginart et al. (2021); Sun et al. (2020), determine embedding dimensions using predefined heuristics based on feature frequency or field size. The advantage of rule-based approaches is their computational efficiency; however, they may not achieve optimal compression. For example, the compressed sequential recommendation framework (CpRec) proposed by Sun et al. (2020) divides a group of items into subgroups, each of which is assigned a predefined dimension based on feature frequencies. Similarly, Ginart et al. (2021) assigns a predefined dimension based on the number of features included in a particular field.

NAS-based approaches automatically determine optimal embedding dimensions. Unlike rule-based approaches, which use predefined dimensions, NAS-based approaches learn the optimal embedding dimensions (Liu et al., 2020; Wei et al., 2021b; Zhao et al., 2020). To achieve this, Joglekar et al. (2020) present neural input search (NIS), which utilizes reinforcement learning; Zhaok et al. (2021) analyze automated embedding dimensionality search (AutoEmb), which employs differentiable architecture search; and Chen et al. (2021a); Lyu et al. (2022) respectively propose recommendation with universally learned elastic embeddings (RULE) and the optimal embedding table learning framework (OptEmbed), both of which leverage evolutionary search strategies.

Pruning techniques optimize memory usage by identifying and removing less important weights, thereby creating sparse embedding tables (Liu et al., 2021a; Qu et al., 2022). Deng et al. (2021) use iterative pruning with retraining, whereas Cheng et al. (2020); Yan et al. (2021b) employ learnable masks to determine which weights to retain.

Weight Sharing Weight-sharing approaches reduce memory usage by allowing multiple features to share embedding parameters, which can be categorized into two sub-types: hashing and vector quantization.

Hashing functions map features to shared embeddings (Pansare et al., 2022; Zhang et al., 2020; Desai et al., 2021). For example, Shi et al. (2020b) uses multiple hash functions to generate two index vectors and maintain two meta-tables. Furthermore, (Yan et al., 2021a) develops binary code based hash (BCH) that operates features at the bit level. Random offset block embedding (ROBE) (Desai et al., 2022b) uses hash functions so that all elements of the embedding table can be mapped to a shared memory.

Vector quantization methods cluster similar embeddings together (Lian et al., 2020). Similarity-aware embedding compression (Saec) (Wu et al., 2020) uses traditional clustering, while subsequent works like multi-granular quantized embeddings (MGQE) (Kang et al., 2020) and extremely memory-efficient factorization machine (xLightFM) (Jiang et al., 2021) employ product quantization. Linear-time self attention (LISA) (Wu et al., 2021b) specifically addresses quantization for self-attention mechanisms in recommender systems.

B.2 Recommender System and LLM at a Broader Scale

To ensure efficient training or deploying CoVE at scale, we also need LLM optimization to reduce memory or computation (Zhang et al., 2025a; Liang et al., 2025; Gao et al., 2025a,b; Song et al., 2025; Liang et al., 2024; Gao et al., 2023; Song et al., 2024, 2023; Li et al., 2023c; Chen et al., 2025a, 2024, 2025b,c). In particular, (Zhang et al., 2025a) proposes the selection of subspaces of importance sampling (I3S) for low-rank optimization to enable memory-efficient training of LLMs. (Gao et al., 2025a,b, 2023; Song et al., 2024, 2023; Li et al., 2023c) analyze the attention inspired regression problem to reduce the computational complexity. (Song et al., 2025) builds upon a linear time variant of the softmax attention problem. (Liang et al.,

2025) analyzes the dynamic maintenance of kernel density estimation. (Liang et al., 2024) develops the convolution basis to reduce the computational cost of masked attention. (Chen et al., 2024, 2025b) analyze the efficiency of RoPE attention and visual autoregressive transformers. (Chen et al., 2025a) makes use of the concept of circuit complexity to study the computational limitation of Mamba.

Additionally, recommender systems also exhibit a strong connection with reinforcement learning, as both aim to model sequential decision-making processes where the system learns optimal strategies (e.g., item recommendations) by interacting with users and observing feedback over time (Zhang et al., 2023a, 2025b). From the perspective of LLMs, our work can be inspirational for and related to other possible application where LLMs are not very closely involved yet (Zhang et al., 2022, 2023b).