Semantic Convergence: Harmonizing Recommender Systems via Two-Stage Alignment and Behavioral Semantic Tokenization

Guanghan Li, Xun Zhang, Yufei Zhang, Yifan Yin, Guojun Yin*, Wei Lin

Meituan, Beijing, China

{liguanghan02, zhangxun12, zhangyufei08, yinyifan, yinguojun02, linwei31}@meituan.com

Abstract

Large language models (LLMs), endowed with exceptional reasoning capabilities, are adept at discerning profound user interests from historical behaviors, thereby presenting a promising avenue for the advancement of recommendation systems. However, a notable discrepancy persists between the sparse collaborative semantics typically found in recommendation systems and the dense token representations within LLMs. In our study, we propose a novel framework that harmoniously merges traditional recommendation models with the prowess of LLMs. We initiate this integration by transforming ItemIDs into sequences that align semantically with the LLMs' space, through the proposed *Alignment Tokenization* module. Additionally, we design a series of specialized supervised learning tasks aimed at aligning collaborative signals with the subtleties of natural language semantics. To ensure practical applicability, we optimize online inference by pre-caching the top-K results for each user, reducing latency and improving efficiency. Extensive experimental evidence indicates that our model markedly improves recall metrics and displays remarkable scalability of recommendation systems.

Introduction

Recent advancements in Large Language Models (LLMs) technologies, particularly those exemplified by trailblazing models such as GPT-4, have signified a substantial progression in the sphere. The intersection of these advancements with recommendation technologies has engendered significant intrigue due to the renowned proficiency of LLMs in the area of advanced natural language processing. This confluence suggests a promising trajectory for augmenting semantic comprehension and behavioral inference within recommendation systems.

Nevertheless, the fusion of LLMs with recommendation systems engenders a unique set of complexities. The modality of language token representation in LLMs is fundamentally divergent from the myriad of sparse identifiers employed by traditional recommendation algorithms. This semantic representation disparity precipitates considerable alignment and scalability challenges (Zheng et al. 2023), elements that are crucial for the efficacious implementation of these hybrid systems. Existing methodologies, inclusive of fine-tuning LLMs on user behavior sequences (Zhang et al. 2023; Li et al. 2023) and the integration of supplementary item identifiers into the LLMs (Zhu et al. 2024; Ren et al. 2024), has several limitations as depicted in Figure. 1. Firstly, in existing methods, as shown in (a), the LLM is either used merely as a feature extractor (LLM as feature), where textual information is fed into the LLM to extract semantic features to enhance existing recommendation models. However, this approach may be limited by the incompatibility between the semantic information generated by the LLM and the recommendation models based on behavior information. It also overlooks the LLM's ability to understand user behavior sequences. Alternatively, as illustrated in (b), large-scale item ID sequences are trained into

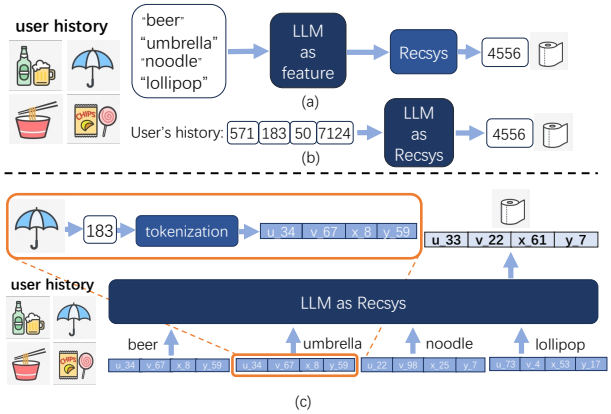

Figure 1: Illustration of LLM-Based Recommendation Methods. (a) and (b) illustrates existing methods, where either textual information is input into the LLM to provide features for existing recommendation models (Ren et al. 2024) in (a), or large-scale item IDs are directly fed into the LLM through instruction fine-tuning, enabling the LLM to directly output recommended item IDs (Li et al. 2023) in (b). (c) represents our method, which can compress large-scale item IDs into a small-scale token representation that is easier for the LLM to process. Furthermore, our method allows the LLM to simultaneously handle both item IDs and textual information, directly outputting the recommended item IDs.

*Corresponding author.

the LLM through instruction fine-tuning. While this method attempts to leverage the LLM's understanding of user behavior sequences, it lacks important textual semantic signals. Moreover, the large-scale item IDs in real industrial environments increase the difficulty of fine-tuning the LLM.

In response to these challenges, as shown in (c), we propose an innovative two-stage alignment framework, conceived to synchronize recommendation models with LLMs via a process of semantic convergence. Initially, the proposed *Alignment Tokenization* Module is entrusted with the translation of item embeddings from recommendation systems into sequences that are semantically congruent with the representations utilized by LLMs. This pivotal step serves to bridge the divide between sparse and dense representations, thereby augmenting the comprehension of user interests. Subsequent to this, we have engineered a series of *Alignment Tasks*, meticulously crafted to further hone the semantic calibration of LLMs. These tasks strategically leverage the inherent features of recommendation systems, thus enabling LLMs to more astutely discern user interests across a variety of domains. Our framework, which integrates harmoniously with extant systems, has achieved substantial progress in improving recommendation metrics and scalability.

Our contributions can be encapsulated as follows:

1) A two-stage recommendation system is designed with LLMs, effectively aligning the semantics of both behavior and language signals.

2) A novel methodology has been devised to map the product ID representation, derived from the recommendation system, to a sequential representation that can undergo further training with any LLM structure or traditional recommendation model.

3) We propose several fine-tuning tasks for LLM, which includes Sequential alignment, text alignment, and negative sampling strategies. This approach effectively models user interests and enhances the robustness of the task.

Related Works

Sequential Recommendation. The field of Sequential Recommendation is widely applied, with ongoing research dedicated to forecasting consumer preferences based on their historical interactions. LSTM (Wu et al. 2017) and GRU4Rec (Hidasi et al. 2015) have demonstrated their proficiency in capturing both long-term dependencies and immediate associations. Current endeavors are increasingly adopting sophisticated graph convolutional neural network models within recommendation systems like NGCF (Wang et al. 2019) and lightGCN (He et al. 2020). Additionally, Transformer-based models, *e.g.* SASRec (Kang and McAuley 2018), BERT4Rec (Sun et al. 2019), S3-Rec (Zhou et al. 2020) have garnered attention for their ability to leverage self-attention mechanisms.

LLM in Recsys. Incorporating Large Language Models (LLMs) into Recommendation Systems (RS) has become a hot topic in recent research, thanks to the vast knowledge base and superior reasoning abilities of LLMs. By enhancing the representation of IDs (Hou et al. 2022; Hua et al. 2023; Hou et al. 2023), or tweaking the design of training tasks and the foundational model structure (Li et al. 2023; Zhang et al. 2023; Lin et al. 2023; Zhu et al. 2024; Wang et al. 2024a,b), LLMs are given recommendation capabilities. Some studies also use LLM as a feature enhancer (Lyu et al. 2023; Xi et al. 2023; Di Palma 2023; Wei et al. 2024), but the main recommendation task is still managed by traditional models, which may not fully utilize the reasoning abilities of LLMs. Solutions such as NoteLLM (Zhang et al. 2024a,b) and others (Ren et al. 2024), which are based on representation learning, or system integration methods (Luo et al. 2024), have been proposed, but they have complex processes and are not easily scalable. Our method uses LLM as a recommendation system, fully aligning semantic information with collaborative semantics, and is also compatible with traditional recommendation models, making the framework easily adaptable to any recommendation scheme.

Method

In this section, we present our two-stage alignment framework for Large Language Models (LLMs) in recommendation systems, as depicted in Fig. 2. The first stage, **Alignment Tokenization**, mitigates the inefficiency of LLM training due to the vast scale of items. This is achieved by mapping items onto a discrete vector set (tokenization). During this process, we introduce an alignment module to better synchronize the tokenization with the LLM's input semantic space. The second stage, **Alignment Task**, enhances the LLM's ability to predict user interests by incorporating training data that is beneficial to recommendation task into LLM's training process. Concurrently, We pre-cache predictions of LLM to facilitate feasible online usage. These modules will be elaborated further in the subsequent sections.

Alignment Tokenization

To enable LLM to comprehend items, it is necessary to represent items as tokens in the LLM's vocabulary. However, in recommendation scenarios, the number of items is typically vast. Utilizing the original item IDs as LLM tokens would result in increased training sparsity and cost. To address this issue, we propose a mapping method that transforms large-scale item space into smaller discrete space. Our approach involves constructing a small-scale discrete index library, that is, the $CodeBook_1$ to $CodeBook_4$ on the left side of Fig. 2, where each item is represented by four indices from the CodeBooks. These indices can be shared among items, with more related items sharing greater number of indices.

To create the CodeBooks, we draw inspiration from the concept of RQ-VAE (Rajput et al. 2024). Firstly, we define a cascaded CodeBooks with N levels, ranging from coarse to fine-grained, with each layer containing C codes. Each item selects an optimal code from each layer, and is thereby represented by N codes.

In training stage, for the first layer of the CodeBooks, We initially cluster the codes into C centers based on a batch of input item embeddings, which serve as the initial code embeddings of the first layer. It is worth noting that LLM have strong semantic understanding capabilities but lack an understanding of user behavior. Based on this, we utilize em-

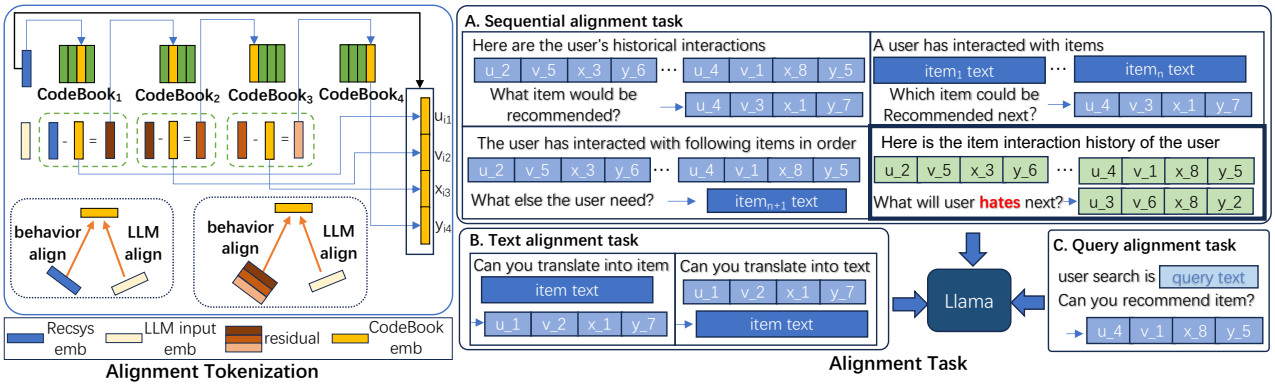

Figure 2: The pipeline of ours two-stage alignment for recommendation. During the Alignment Tokenization phase, our objective is to obtain four token indices—u, v, x, and y for each item. Subsequently, in the Alignment Task process, we utilize these item indices to introduce negative sampling tasks (highlighted by black boxes in the Alignment Task) alongside various positive prediction tasks. These tasks collectively fine-tune the LLM, integrating both behavioral and semantic signals into the model.

beddings derived from DCCF (Ren et al. 2023) trained exclusively on behavior as the item embeddings. Subsequently, the item embeddings will serve as the input for the first layer, identifying the nearest code in the first layer of the CodeBooks to form *embedding pairs* for calculating the distance loss. For the subsequent CodeBooks layers, we use the residual of *embedding pairs* in the previous layer as input of current layer. The C initial cluster centers and *embedding pairs* of the current layer are both generated based on this input. Clearly, the distance loss for the current layer is also calculated from the *embedding pairs*. Consequently, N distance losses are summed. This process demonstrates that the CodeBook approximates the item's embedding by iteratively approximating the residuals generated from previous layers. This is shown as *behavior alignment* on the left side of Fig. 2

This process which can be understood as the tokenization of LLM leverages user behavior to map item IDs into a smaller space. This space, however, is independent of LLM, potentially resulting in misalignments between the semantic space of tokenization and input of LLM. To address this challenge, we have implemented an *LLM alignment* loss, which is depicted in the Fig. 2 on the left. This *LLM alignment* mechanism operates by imposing penalties on the discrepancies between LLM embedding and the codes, thereby ensuring that each code layer is effectively synchronized with the semantic space by the LLM. The LLM embedding are represented by the average pooling of the LLM input layer embedding through the titles and descriptions of items.

The whole training process of this cascaded CodeBooks can be described as follows:

$$E_i^n = \left| E_i^{(n-1)} - B_{c^*}^{(n-1)} \right| \quad \text{if } n \geq 1, E_i^0 = \text{Emb}(\text{r}_i) \quad (1)$$

$$c^* = \arg\min_c \text{Dist}(E_i^n, B_c^n) \quad (2)$$

$$L_i^B = \sum_{n=0}^{N} \text{Dist}(E_i^n, B_{c^*}^n), L_i^L = \sum_{n=0}^{N} \text{Dist}(B_{c^*}^n, E_i^{\text{LLM}}) \quad (3)$$

$$L_i = L_i^B + L_i^L \quad (4)$$

where $Emb(r_i)$ denotes the i_{th} item embedding from DCCF (Ren et al. 2023), i.e. *Recsys emb* in Fig. 2. E_i^n is the input to the n_{th} layer of CodeBooks, and B_c^n refers to the representation of c_{th} code in n_{th} layer of CodeBooks. $Dist()$ denotes a vector distance function, such as cosine distance, and E_i^{LLM} represents the embedding of the i_{th} item at the LLM input layer. In our experiments, we chose llama-7B (Touvron et al. 2023) as the LLM. The loss function consists of *Behavior Alignment* loss L_i^B and *LLM Alignment* loss L_i^L as depicted on the left side of Fig. 2

After the CodeBooks has been trained, each item searches for the nearest code in each layer of CodeBooks according to the training process, ultimately representing each item with N codes. In our experiments, $N = 4$ and $C = 256$. These codes are subsequently incorporated into the LLM's vocabulary to represent the items during the fine-tuning process.

Note that, unlike previous work (Rajput et al. 2024), our method overlook the auto-encoding process. This distinction arises from our objective of constructing discrete index library, whereas the auto-regressive encoder is primarily designed for generation tasks. Pursuing an unrelated task objective would compromise the final accuracy. We conducted a quantitative analysis in Tab. 3.

Alignment Task

After obtaining the item quantization representation for each item through our Alignment Tokenization, the subsequent task involves fine-tuning the LLM using user interaction and text descriptions. To prevent training instability arising from substantial dimensional differences of embeddings between CodeBooks and LLM, We utilize only the code indices of each item rather than their embeddings. Consequently, the newly introduced tokens representing items remain in an untrained state until the LLM undergoes fine-tuning. Following prior work (Zheng et al. 2023), as depicted on the right side of Fig. 2, several fine-tuning tasks are defined including *Se-*

quential alignment task, *Text alignment task*, *Query alignment task*. The following are examples for each task:

A. Sequential alignment task
[prompt]: Here is the item interaction history of the user: $<item_i>$,..., what to recommend to the user next?
[label]: $<item_j>$

B. Text alignment task
[prompt]: An item is called $title$ and described as $description$, can you tell me which item it is?
[label]: $<item_j>$

C. Query alignment task
[prompt]: You meet a user's query: $query$. Please respond to this user by selecting an appropriate item
[label]: $<item_j>$

where the $<item_i>$ and $<item_j>$ refer to token index of the item derived from Alignment Tokenization, the $title$ and $description$ represent the item's title and textual description, respectively, and the $query$ denotes user's review. For all tasks, [prompt] are utilized as the training inputs for LLM, while [label] serves as the training targets. These tasks are integrated into training set to fine-tuning LLM with cross-entropy loss, a widely used loss for training LLMs. Llama-7B (Touvron et al. 2023) is selected as our LLM.

The user interaction item sequence is a vital component of the training data. However, the number of interacted items per user is typically limited. Utilizing only interacted items as training data can lead to over-fitting and introduce sample selection bias (Ma et al. 2018) due to insufficient data. To address this challenge, as depicted in the Fig. 2, the green area is enclosed by black box, we introduced negative sampling strategy for user behavior to align with the behavior generalization capabilities of traditional recommendation models. For scenarios without negative samples, we randomly sample items that the user has not interacted with. Based on the negative samples of user behaviors, we have added a "negative behavior task" in the Sequential alignment task. This task is used to represent items that users are unlikely to interact with in the current context. The negative samples are incorporated into a new set of negative prompt expressions, as a example illustrated below:

Negative sampling task
[prompt]: Here is the item interaction history of the user: $<item_i>$,..., what will user **hate** next?
[label]: $<item_j>$

The data corresponding to these negative prompt expressions are used along with other tasks as training data for LLM fine-tuning. The efficacy of our negative sampling strategy is illustrated in Tab. 3.

It is noteworthy that in industrial recommendation systems, various training acceleration techniques, such as Low-Rank Adaptation (LoRA) (Hu et al. 2021) and FlashAttention (Dao et al. 2022), can be utilized to expedite the training process. Furthermore, incremental training can help in reducing training data. Nevertheless, it is important to acknowledge that despite the implementation of these acceleration methods, the training speed of LLM remains slower than that of traditional recommendation models. This issue is a prevalent challenge faced across the this field.

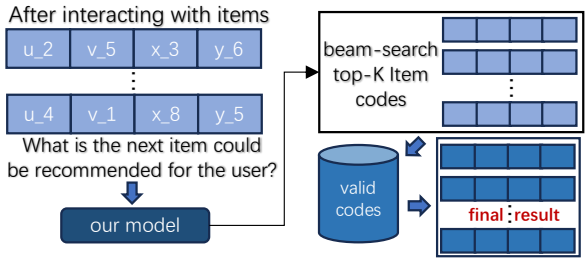

Figure 3: Inference Stage. We cache valid codes in predictions for each user during the online inference phase.

Inference

Following supervised fine-tuning, The prompt "*The user has interacted with* $<item_i>$*,... in chronological order. Can you predict the next possible item that the user may expect?*" which mentioned in LC-Rec (Zheng et al. 2023) is used for inference. However, due to the large number of parameters in the LLM, the inference process incurs significant computational costs and exhibits slow inference speed, rendering it unsuitable for real industrial system. To address this limitation, as depicted on the Fig. 3, we pre-cache the top-K item codes for each user, generated through beam search inference within the LLM. The *valid codes* refers to the collection of item codes corresponding to all possible items obtained from the CodeBooks. We only cache valid item codes. During the online inference phase, retrieval of the relevant user's cached data suffices.

It is important to note that when new items are introduced into the item pool, it is unnecessary to retrain LLM and CodeBooks. The token representation of items can be derived through inference during our alignment tokenization stage. Despite the fact that new items have not been trained in the LLM, these items have acquired meaningful representations during CodeBook's inference, thereby mitigating the cold start problem for new items.

Experiment

Experiment Setup

Datasets We evaluated our proposed method by utilizing the "Games", "Arts" and "Instruments" datasets from the Amazon review dataset (Ni, Li, and McAuley 2019). These datasets encompass user interactions with items, with each item accompanied by a title and description. To mitigate the impact of long-tail data on training, we define the maximum sequence length for user interactions as 20. Furthermore, samples with a concatenated prompt length exceeding 2048

Dataset	Metrics	SASRec	BERT4Rec	FMLP	DCCF	P5	LC-Rec	Ours	Improv.
Games	HR@1	0.0013	0.0136	0.0154	0.0197	0.0158	0.0279	**0.0323**	+15.77%
	HR@5	0.0296	0.0468	0.0496	0.0578	0.0495	0.0755	**0.0858**	+13.64%
	HR@10	0.0576	0.0773	0.0806	0.0882	0.0789	0.1167	**0.1274**	+9.17%
	NDCG@5	0.0153	0.0302	0.0325	0.0389	0.0329	0.0518	**0.0591**	+14.09%
	NDCG@10	0.0243	0.0400	0.0424	0.0486	0.0423	0.0651	**0.0724**	+11.21%
Arts	HR@1	0.0015	0.0184	0.0372	0.0339	0.0487	0.0639	**0.0676**	+5.79%
	HR@5	0.0423	0.0467	0.0689	0.0696	0.0705	0.0999	**0.1041**	+4.20%
	HR@10	0.0643	0.0655	0.0922	0.0954	0.0876	0.1237	**0.1289**	+4.20%
	NDCG@5	0.0225	0.0329	0.0534	0.0520	0.0597	0.0822	**0.0863**	+4.99%
	NDCG@10	0.0295	0.0389	0.0608	0.0602	0.0652	0.0899	**0.0942**	+4.78%
Instruments	HR@1	0.0002	0.0396	0.0527	0.0440	0.0595	0.0708	**0.0731**	+3.25%
	HR@5	0.0523	0.0625	0.0794	0.0742	0.0834	0.0985	**0.1022**	+3.76%
	HR@10	0.0714	0.0797	0.1016	0.0929	0.1023	0.1225	**0.1268**	+3.51%
	NDCG@5	0.0276	0.0516	0.0663	0.0595	0.0717	0.0848	**0.0879**	+3.66%
	NDCG@10	0.0338	0.0571	0.0734	0.0656	0.0777	0.0925	**0.0958**	+3.57%

Table 1: Comparative analysis of metrics across various models and datasets, with the best results emphasized in bold. As shown in the table, the "LC-Rec" is the second-best method after our approach. Therefore, "Improv." indicates the relative improvement of our method compared to "LC-Rec".

Dataset	Users	Items	Len avg	Long user
Games	50546	16859	8.962	4.80%
Arts	45141	20956	8.658	4.34%
Instruments	24772	9922	8.322	3.48%

Table 2: Statistics of the datasets. In addition to the number of "Users" and "Items", "Len avg" denotes the average length of user sequences, while "Long user" denotes the proportion of users whose sequence length exceeds 20, relative to the total number of users.

were excluded. The statistical details of these publicly available datasets are presented in Table. 2.

Baselines For comparison purposes, we chose several representative recommendation models as baselines:

- **SASRec (Kang and McAuley 2018)**: A self-attention driven sequence recommendation model captures long-term dependencies within user behavior sequences.
- **BERT4Rec (Sun et al. 2019)**: The sequence recommendation method captures patterns in users' behavior by utilizing BERT's bidirectional encoding.
- **FMLP (Zhou et al. 2022)**: The model employs learnable filters to enhance the sequence data and utilizes the Fast Fourier Transform to automatically filter out noise.
- **DCCF (Ren et al. 2023)**: Self-supervised contrastive learning was employed to achieve intention decoupling and enhances the representations of users and items.
- **P5 (Geng et al. 2022)**: Various recommendation tasks were integrated into a unified model. We implement it based on this code (Wenyueh 2023).
- **LC-Rec (Zheng et al. 2023)**: A quantization method for item indexing was designed, which enhances recommendation capabilities of the LLM through fine-tuning.

In all methods, we extract the first S-1 actions from user behavior sequence of length S to serve as input prompt and label in training set. In the evaluation process, the user's last interacted item is designated as the label, with the preceding actions as input prompts.

Metrics We employed two widely recognized metrics to assess the performances: top-K Hit Ratio (HR) and top-K Normalized Discounted Cumulative Gain (NDCG). In our paper, we utilize HR@K and NDCG@K to represent these metrics, and the parameter K is set to 1, 5, and 10.

Experiment Results

We conducted a comparison between our method and several baseline models. The results are presented in Tab. 1. It is evident that the LLM-based approach outperforms traditional methods across all three datasets. This superiority can be attributed to the LLMs, which encompasses a wealth of world knowledge and enhances recommendation effectiveness by incorporating information such as behavior or text. Moreover, our proposed method exhibits improvements over the LLM-based approach, LC-Rec (Zheng et al. 2023) on all three datasets, as shown in the last column of Tab. 1. This primarily owe to the two-stage alignment that we specifically designed for the recommendation task.

Our results indicate that the improvement achieved by our method on the *Games* dataset is more significant compared to the *Arts* and *Instruments* datasets. This discrepancy may be attributed to longer average sequence length and higher proportion of long-sequence users in the *Games* dataset, as illustrated in Table. 2. The richer the user behavior, the more significant performance enhancement of LLM. It suggests that LLMs with robust semantic parsing capabilities possess a superior ability to learn from multi-modal behavior.

Additionally, we found that the gains from comparing fewer recommendation results are greater than those from comparing more results, *e.g.*, the gain in HR@5 is greater

	HR@5	HR@10	NG@5	NG@10
ED	0.0817	0.1197	0.0570	0.0693
w/o LA	0.0836	0.1264	0.0586	0.0723
w/o NS	0.0791	0.1195	0.0548	0.0677
Ours	0.0858	0.1274	0.0591	0.0724

Table 3: Ablation study in the Games dataset. "NG" denotes NDCG. It showcases the effect of every modification we made based on LC-Rec (Zheng et al. 2023).

than that in HR@10. This maybe because, the LLM only has one positive label token per sample during training, which results in only the top-1 token being penalized by loss function. For this reason, the top-ranked recommendation results benefiting more from our approach.

Ablation Study

To demonstrate the benefits of each feature of our proposed method, considering that LC-Rec (Zheng et al. 2023) is the best-performing method among baselines, we compare the results with the variants by removing each newly added cue compared to LC-Rec (Zheng et al. 2023), *i.e.*, 1) "ED", our tokenization incorporates an Encoder-Decoder similar to RQ-VAE (Rajput et al. 2024), 2) "w/o LA", our method without the *LLM alignment* loss L^L in eq. (3), 3) "w/o NS", our method without negtive sampling, as shown in Tab. 3.

Our method shows significant improvement over "ED", indicating that the encoder-decoder module used for generative tasks is not suitable for our clustering representation task. The penalty function designed for generative capabilities can weaken our representation ability. Furthermore, our method shows a slight improvement compared to "w/o AM", this is because the alignment loss helps pre-align with LLM input space in our tokenization stage, facilitating more efficient convergence of the LLM during fine-tuning phase. The slight improvement in performance may be attributed to LLM's rich general knowledge, which eases convergence challenges. Lastly, our method shows significant improvement over "w/o NS", thanks to our negative sampling strategy that adds a large amount of effective data, enhances generalizability, and alleviates sample selection bias.

Different negative sampling ratios. Table. 4 illustrates the evaluation of different negative sampling ratios and their performance. It is evident that as the negative sampling ratio increases, the performance gradually improves, albeit at a diminishing rate when the negative sampling ratio reaches 1:4. This is attributed to our negative sampling strategy, which has enhanced the LLM's generalization ability regarding behaviors and alleviated sample selection bias.

Varying numbers of CodeBooks. Table. 5 presents the performance for varying numbers of CodeBooks. We observe that when the number of CodeBooks is 2, the collision rate exceeds 10%. However, when C is 3 or greater, the collision rate becomes negligible. Token sharing among items facilitates more comprehensive token training. Consequently, an excessive number of new words in LLM can result in sparse token training, while too few new words may lead to fewer token representations for individual items, potentially causing inadequate expression. As shown in Table. 5, the optimal performance is achieved when the number of CodeBooks is 4. This scenario represents a balanced state between the number of CodeBooks and tokens, prompting us to select $C = 4$ as the number of CodeBooks for our experiments.

NSR	HR@5	HR@10	NDCG@5	NDCG@10
1:0	0.0791	0.1195	0.0548	0.0677
1:1	0.0817	0.1239	0.0559	0.0695
1:2	0.0830	0.1256	0.0574	0.0711
1:3	0.0858	0.1274	0.0591	0.0724
1:4	0.0871	0.1277	0.0606	0.0737
1:5	0.0856	0.1270	0.0597	0.0729

Table 4: Comparison of results with different negative sampling ratios in Games Dataset, where "NSR" represents the negative sampling ratio. The left and right sides of the colon represent positive and negative samples, respectively

Scaling law of recommendation ability. To verify whether larger and more advanced LLM exhibit superior performance in our recommendation task, we conducted experimental comparisons across various parameter scales and model versions. The experimental results are presented in Table. 6. The first two models represent different versions with the same parameter scale, whereas the last model feature larger parameter scales. We observed that when utilizing LLMs with similar parameter sizes, more advanced model versions do not necessarily exhibit superior performance. This phenomenon may be attributed to the fact that the advanced models primarily incorporate additional training corpora, which might be significantly different from the recommendation domain. Furthermore, in recommendation tasks, the semantic information tends to be relatively simple, lower-version models are adequate for handling the simple semantic information. However, larger parameter sizes yield improvements in performance. This could be attributed to the fact that models with larger parameters are more effective at learning behavior patterns. Furthermore, the superior performance of larger parameter LLMs implies a scaling law for LLMs in recommendation capabilities. This not only indicates that we can improve recommendation performance by simply increasing the model parameters, but also suggests that LLM-based recommendation models can benefit from the rapid advancements in the LLM domain.

Fine-tuning LLMs with different embedding initialization methods. When obtaining token representation for each item, we only utilized the "index" results produced during alignment tokenization phase, omitting the "embedding" results from this phase. Consequently, the tokens newly added to the LLM were not pre-initialized before fine-tuning. However, the embeddings produced by the CodeBooks during alignment tokenization phase possess semantic significance. Intuitively, initializing LLM's token embeddings with these CodeBooks embeddings might yield better performance. To test this hypothesis, we projected the 32-dimensional embeddings obtained from the DCCF model to the LLM's dimensionality (4096) using a specific mapping module, and then used this projection as the initialization for the LLM's

C	CR	T	HR@5	HR@10	NG@5	NG@10
1	——	16859	0.0700	0.1111	0.0470	0.0602
2	10.8%	512	0.0789	0.1198	0.0537	0.0668
3	0.05%	768	0.0804	0.1225	0.0555	0.0690
4	0.0%	1024	0.0858	0.1274	0.0591	0.0724
5	0.0%	1280	0.0804	0.1215	0.0554	0.0687

Table 5: Performance with varying numbers of CodeBooks in Games Dataset. "C" denotes the number of CodeBooks, indicating that each item is represented by "C" tokens. "CR" stands for the collision rate, representing the proportion of items with duplicate token representations among all items. "T" denotes the number of new words added to the LLM. "NG" is the abbreviation for NDCG. When C=1, the original item IDs are directly used as token representations, thereby eliminating the possibility of collisions.

model	HR@5	HR@10	NG@5	NG@10
Llama-7B	0.0858	0.1274	0.0591	0.0724
Llama-2-7B	0.0817	0.1242	0.0562	0.0699
Llama-13B	0.0910	0.1329	0.0636	0.0771

Table 6: Performance Evaluation Using Different LLMs in Games Dataset. The "NG" stands for NDCG.

token embeddings. Given the substantial dimensional differences, we omitted experiments with a 1-layer MLP mapping module and instead employed 2-layer and 3-layer MLPs. As illustrated in Table. 7, our findings indicate that the performances of these mapping modules has declined. This may be attributed to the dimensional and semantic gap between code embeddings and the input embeddings of LLM, which hinders the convergence of the LLM.

Performance comparison using only behavior sequence training. In our experimental setup, the first five baseline methods were trained using only user behavior. However, both LC-Rec and our method utilized not only user behavior but also textual information. To exclude the influence of textual information, we compared the performance of all models using only user behavior sequences. We selected the best three methods from baselines, DCCF and P5, as well as LC-Rec which also utilizes textual information, as our baseline

Method	HR@5	HR@10	NG@5	NG@10
2-layer	0.0721	0.1152	0.0475	0.0614
3-layer	0.0747	0.1168	0.0501	0.0636
Ours	0.0858	0.1274	0.0591	0.0724

Table 7: Comparison of token embedding initialization methods for LLM in Games Dataset. "NG" denotes NDCG. "2-layer" and "3-layer" refer to different modules that map the CodeBook embeddings with 32 dimensions to 4096 dimensions. Specifically, these modules are "MLP-Relu-MLP", and "MLP-Relu-MLP-Relu-MLP", respectively. The gradients of these modules and the CodeBook embeddings will be activated during the fine-tuning phase.

Method	HR@5	HR@10	NG@5	NG@10
DCCF	0.0578	0.0882	0.0389	0.0486
P5	0.0495	0.0789	0.0329	0.0423
LC-Rec	0.0708	0.1044	0.0494	0.0602
Ours	0.0773	0.1132	0.0537	0.0652
Ours w/ T	0.0858	0.1274	0.0591	0.0724

Table 8: Performance Comparison Using Only User Behavior Sequences in Games Dataset. "NG" stands for the NDCG metric, and "Ours w/ T" indicates that our method uses complete information. We selected three well-performing methods as our baselines.

comparisons. We conducted experiments using the Games dataset. The experimental results are shown in Table. 8. As can be observed, using only user behavior will result in a significant loss of performance. However, our method with using only user behavior significantly outperforms the first two baselines, thanks to the LLM's powerful learning capabilities. Furthermore, our method shows substantial improvement over LC-Rec, which can be attributed to the superior performance of our two-stage alignment method.

Limitation and Future Work

While our work has demonstrated outstanding performance, caching results during inference phase may be constrained by storage space and is inadequate for managing frequently changing item pool. Pre-caching vectors reveal an alternative method. However, the beam search decoding method incurs high costs for vector retrieval. Consequently, one of our future endeavors is to enhance efficiency of online inference. Furthermore, the large-scale data inherent in the recommendation domain renders the training costs of LLMs prohibitively high. Thus, improving training efficiency in the recommendation domain is our another focus. Additional, our work only involves text and behavior, we plan to incorporate additional modalities such as images. Lastly, in ablation study on scaling law of recommendation capability, we were constrained by computational resources and thus only conducted experiments with 13B model and below. We plan to allocate more computational resources to the experiments.

Conclusion

In this paper, we propose a method via Alignment Tokenization and Alignment Task to enhance the recommendation system based LLM. Specifically, for Alignment Tokenization, to alleviate the issues of increased training costs and sparsity caused by large-scale items, we present a method for mapping large-scale items to a smaller-scale index library. Additionally, we introduce LLM alignment loss to pre-align during the tokenization phase, addressing the misalignment between tokenization and input space of LLM. For Alignment Task, to increase data volume and mitigate sample selection bias, we incorporate a negative sampling strategy in the training data of LLM, aligning the traditional recommendation model's ability of generalize user interests. Our method's effectiveness is validated on three datasets.

References

Dao, T.; Fu, D.; Ermon, S.; Rudra, A.; and Ré, C. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35: 16344–16359.

Di Palma, D. 2023. Retrieval-augmented recommender system: Enhancing recommender systems with large language models. In *Proceedings of the 17th ACM Conference on Recommender Systems*, 1369–1373.

Geng, S.; Liu, S.; Fu, Z.; Ge, Y.; and Zhang, Y. 2022. Recommendation as language processing (rlp): A unified pretrain, personalized prompt & predict paradigm (p5). In *RecSys*.

He, X.; Deng, K.; Wang, X.; Li, Y.; Zhang, Y.; and Wang, M. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, 639–648.

Hidasi, B.; Karatzoglou, A.; Baltrunas, L.; and Tikk, D. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939*.

Hou, Y.; He, Z.; McAuley, J.; and Zhao, W. X. 2023. Learning vector-quantized item representation for transferable sequential recommenders. In *WWW*.

Hou, Y.; Mu, S.; Zhao, W. X.; Li, Y.; Ding, B.; and Wen, J.-R. 2022. Towards universal sequence representation learning for recommender systems. In *SIGKDD*.

Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; and Chen, W. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Hua, W.; Xu, S.; Ge, Y.; and Zhang, Y. 2023. How to Index Item IDs for Recommendation Foundation Models. *SIGIR-AP*.

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

Li, X.; Chen, C.; Zhao, X.; Zhang, Y.; and Xing, C. 2023. E4SRec: An elegant effective efficient extensible solution of large language models for sequential recommendation. *arXiv preprint arXiv:2312.02443*.

Lin, X.; Wang, W.; Li, Y.; Feng, F.; Ng, S.-K.; and Chua, T.-S. 2023. A Multi-facet Paradigm to Bridge Large Language Model and Recommendation. *arXiv preprint arXiv:2310.06491*.

Luo, S.; Yao, Y.; He, B.; Huang, Y.; Zhou, A.; Zhang, X.; Xiao, Y.; Zhan, M.; and Song, L. 2024. Integrating Large Language Models into Recommendation via Mutual Augmentation and Adaptive Aggregation. *arXiv preprint arXiv:2401.13870*.

Lyu, H.; Jiang, S.; Zeng, H.; Xia, Y.; and Luo, J. 2023. Llmrec: Personalized recommendation via prompting large language models. *arXiv preprint arXiv:2307.15780*.

Ma, X.; Zhao, L.; Huang, G.; Wang, Z.; Hu, Z.; Zhu, X.; and Gai, K. 2018. Entire space multi-task model: An effective approach for estimating post-click conversion rate. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, 1137–1140.

Ni, J.; Li, J.; and McAuley, J. 2019. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *EMNLP-IJCNLP*.

Rajput, S.; Mehta, N.; Singh, A.; Hulikal Keshavan, R.; Vu, T.; Heldt, L.; Hong, L.; Tay, Y.; Tran, V.; Samost, J.; et al. 2024. Recommender systems with generative retrieval. *Advances in Neural Information Processing Systems*, 36.

Ren, X.; Wei, W.; Xia, L.; Su, L.; Cheng, S.; Wang, J.; Yin, D.; and Huang, C. 2024. Representation learning with large language models for recommendation. In *Proceedings of the ACM on Web Conference 2024*, 3464–3475.

Ren, X.; Xia, L.; Zhao, J.; Yin, D.; and Huang, C. 2023. Disentangled contrastive collaborative filtering. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1137–1146.

Sun, F.; Liu, J.; Wu, J.; Pei, C.; Lin, X.; Ou, W.; and Jiang, P. 2019. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *CIKM*.

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

Wang, H.; Liu, X.; Fan, W.; Zhao, X.; Kini, V.; Yadav, D.; Wang, F.; Wen, Z.; Tang, J.; and Liu, H. 2024a. Rethinking Large Language Model Architectures for Sequential Recommendations. *arXiv preprint arXiv:2402.09543*.

Wang, X.; He, X.; Wang, M.; Feng, F.; and Chua, T.-S. 2019. Neural graph collaborative filtering. In *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*, 165–174.

Wang, X.; Wu, L.; Hong, L.; Liu, H.; and Fu, Y. 2024b. LLM-Enhanced User-Item Interactions: Leveraging Edge Information for Optimized Recommendations. *arXiv preprint arXiv:2402.09617*.

Wei, W.; Ren, X.; Tang, J.; Wang, Q.; Su, L.; Cheng, S.; Wang, J.; Yin, D.; and Huang, C. 2024. Llmrec: Large language models with graph augmentation for recommendation. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, 806–815.

Wenyueh. 2023. LLM-RecSys-ID. https://github.com/Wenyueh/LLM-RecSys-ID/, Last accessed on 08-15-2024.

Wu, C.-Y.; Ahmed, A.; Beutel, A.; Smola, A. J.; and Jing, H. 2017. Recurrent Recommender Networks. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, 495–503.

Xi, Y.; Liu, W.; Lin, J.; Zhu, J.; Chen, B.; Tang, R.; Zhang, W.; Zhang, R.; and Yu, Y. 2023. Towards open-world recommendation with knowledge augmentation from large language models. *arXiv preprint arXiv:2306.10933*.

Zhang, C.; Wu, S.; Zhang, H.; Xu, T.; Gao, Y.; Hu, Y.; and Chen, E. 2024a. NoteLLM: A Retrievable Large Language Model for Note Recommendation. In *Companion Proceedings of the ACM on Web Conference 2024*, 170–179.

Zhang, C.; Zhang, H.; Wu, S.; Wu, D.; Xu, T.; Gao, Y.; Hu, Y.; and Chen, E. 2024b. NoteLLM-2: Multimodal Large Representation Models for Recommendation. *arXiv preprint arXiv:2405.16789*.

Zhang, Y.; Feng, F.; Zhang, J.; Bao, K.; Wang, Q.; and He, X. 2023. CoLLM: Integrating Collaborative Embeddings into Large Language Models for Recommendation. *arXiv preprint arXiv:2310.19488*.

Zheng, B.; Hou, Y.; Lu, H.; Chen, Y.; Zhao, W. X.; and Wen, J.-R. 2023. Adapting large language models by integrating collaborative semantics for recommendation. *arXiv preprint arXiv:2311.09049*.

Zhou, K.; Wang, H.; Zhao, W. X.; Zhu, Y.; Wang, S.; Zhang, F.; Wang, Z.; and Wen, J.-R. 2020. S3-rec: Self-supervised learning for sequential recommendation with mutual information maximization. In *CIKM*.

Zhou, K.; Yu, H.; Zhao, W. X.; and Wen, J.-R. 2022. Filter-enhanced MLP is all you need for sequential recommendation. In *WWW*.

Zhu, Y.; Wu, L.; Guo, Q.; Hong, L.; and Li, J. 2024. Collaborative large language model for recommender systems. In *Proceedings of the ACM on Web Conference 2024*, 3162–3172.