Customizing In-context Learning for Dynamic Interest Adaption in LLM-based Recommendation

Anonymous ACL submission

Abstract

Periodically updating Large Language Model (LLM)-based recommender systems to adapt to dynamic user interests—as is done for traditional ones—is impractical due to high training costs, even with acceleration methods. This work explores the possibility of achieving interest adaptation without any model-level updates via *In-context Learning (ICL)*, which enables adaptation through few-shot examples within input prompts. Using new-interest interactions as ICL few-shot examples, LLMs can directly learn the new interest in prompt without needing model updates. However, existing LLM-based recommenders often lose the ICL ability during the recommendation tuning stage, while the original LLM's ICL lacks recommendation-specific focus. To address this, we introduce RecICL, a framework that establishes recommendation-oriented ICL. RecICL operates by performing the recommendation tuning stage in an ICL training manner. It consistently structures new-interest interactions as ICL few-shot examples to enable the use of ICL's ability to capture dynamic interests during data fitting. Extensive experiments across multiple benchmarks demonstrate RecICL's superior performance, achieving better results without model updates. Our implementation is publicly available at `https://anonymous.4open.science/r/RecICL-8003`.

1 Introduction

In recent years, LLM-based recommendation has emerged as a rapidly evolving field, demonstrating the substantial potential to transform recommender systems across diverse scenarios (Wu et al., 2023; Harte et al., 2023). Substantial efforts have been dedicated to this area, creating various mechanisms to align LLM capabilities with recommendation tasks. Among these approaches, instruction tuning on recommendation data has gained significant popularity as it addresses the fundamental limitation that recommendation-specific tasks are inherently absent from the original pretraining objectives of LLMs (Zhang et al., 2023a; Bao et al., 2023a; Zheng et al., 2024).

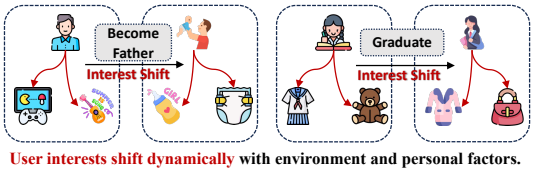

Figure 1: An illustration of user interest shift in real-world scenarios.

Despite progress, existing developments have overlooked dynamic interests, a crucial aspect for real-world applications. As shown in Figure 1, in the real world, user interest can shift rapidly due to the varying instant interest in dynamic environments (Chang et al., 2017; Papagelis et al., 2005; Wang et al., 2018). Updating the model periodically with incoming data is typically used to capture timely user interests (Chandramouli et al., 2011; Das et al., 2007). However, due to the massive number of parameters in LLMs, such model-level updates incur substantial computational and time costs for LLM-based recommendations, making them impractical for real-world applications. Recognizing the persistent issue of high update costs, this study explores the possibility of adapting the model to dynamic user interests without any model-level updates after initial training.

Among existing techniques, *In-context Learning* (ICL) seems to be a promising choice for timely learning user shift interests without model updates. Through ICL, LLMs can learn tasks from few-shot task examples being incorporated into their input context, without any updates to their parameters (Brown, 2020; Zhao et al., 2023). By incorporating recent user interaction data as few-shot task examples in the input, we can expect that ICL could effectively capture the interest shift from these ex-

amples. In this way, the adaption to user real-time interests without necessitating model updates.

However, directly applying ICL to LLM-based recommendations is not feasible. Intuitively, there seem to be two directed approaches: 1) employing ICL with general LLMs and 2) employing ICL with LLMs tuned for recommendations. Both approaches encounter challenges — the first lacks alignment with recommendation tasks, while the second often suffers from diminished ICL capabilities. These challenges would undoubtedly hinder the practical application of these methods. Given these challenges, we consider possessing recommendation-specific ICL capabilities — aligning the model with recommendation tasks while preserving and even enhancing its ICL abilities.

To achieve this, we propose RecICL, a method designed to customize recommendation-specific ICL in LLMs for dynamic interest adaptation. RecICL optimizes the recommendation instruction tuning phase to preserve ICL capabilities, following the general ICL-based tuning method: during instruction tuning for task alignment, we incorporate a few relevant training samples as the few-shot examples within the input, rather than providing just the single training sample to tune the model. However, unlike methods in other domains (Shi et al., 2022; Gu et al., 2023) that customize ICL by selecting semantically similar samples as few-shot examples, we strategically choose new-interest interactions relative to each training sample. During inference, these few-shot examples are replaced with real-time user data, allowing the model to adapt to evolving user interests in a few-shot manner. Extensive experiments on real-world datasets demonstrate the effectiveness of RecICL in enabling capturing dynamic user interests for LLM-based recommender systems.

The main contributions of our work are:

- To our knowledge, we are the first to explore adapting LLM-based recommenders to dynamic user interests without requiring any further model-level updates.
- We propose **RecICL**, a novel framework for adapting ICL in LLMs to recommendation tasks, enabling quick alignment with users' latest interests and personalized recommendation delivery.
- Experimental results demonstrate that our method significantly outperforms existing approaches and can also maintain robust performance over extended periods.

2 Related Work

In this section, we introduce two key topics central to our work: Streaming Recommender Systems and LLM-based Recommender Systems. The relevance of streaming recommender systems lies in their focus on addressing dynamic user interest shifts and leveraging real-time user interactions, which aligns with our goal of adapting to evolving user preferences. Meanwhile, LLM-based Recommender Systems build upon prior works that explore the application of LLMs in recommender systems.

2.1 Streaming Recommender System

Streaming recommender systems have gained significant attention in the past years due to their ability to handle dynamic user interest and item catalogs (Das et al., 2007; Song et al., 2008; Wang et al., 2018). Unlike traditional batch-based recommender systems which only train and test on static fixed datasets, streaming approaches can efficiently process continuous streams of data and provide up-to-date recommendations (Chandramouli et al., 2011; Papagelis et al., 2005; Vinagre et al., 2014).

Early work in this area focused on adapting traditional collaborative filtering techniques to streaming environments (Vinagre et al., 2014). More recently, researchers tend to apply continual graph learning techniques for streaming recommender systems (Wang et al., 2020, 2022; Xu et al., 2020a; He et al., 2023a). Additionally, Graphpro (Yang et al., 2024) leverages pre-training and fine-tuning techniques on the graph to address streaming recommendation tasks on dynamic data. However, they primarily rely on timely updates and iterations of the model parameters, which can be challenging for LLMs due to their high cost. In this paper, we draw inspiration from these previous studies and propose to tackle the challenge of streaming recommendations in the domain of LLMs for recommendation.

2.2 LLM-based Recommender System

Recently, inspired by the powerful and comprehensive capabilities of LLMs, an increasing number of researchers have been exploring various ways to leverage LLMs for recommendation (Wu et al., 2023; Lin et al., 2023a; Bao et al., 2024; Fan et al., 2023). Some researchers have attempted to effectively transfer the knowledge and capabilities of LLMs to traditional recommendation models (Liu et al., 2024; Cui et al., 2024; Xi et al., 2023; Yuan

et al., 2023; Wei et al., 2024). While these methods can address the issue of user interest shift through the iterative process of traditional models, such methods often require fine-tuning LLMs on static data to optimize embeddings, which can significantly impact the model's representational capabilities when data is updated or user interests change.

Another group of people focuses on harnessing the generative power of LLMs to produce recommendations directly (Lin et al., 2023b; Liao et al., 2024; Wang et al., 2023; Zhang et al., 2023b; Tan et al., 2024; Yang et al., 2023; He et al., 2023b). In detail, those researchers have noted that LLMs are exposed to limited recommendation data during their training phase, necessitating an alignment approach to learning recommendation tasks. Although achieving great success, these methods focus on static and fixed datasets where user interest is stable. However, in real-world scenarios, data is constantly updated, user interests are changing, and new user feedback is continually generated (Chang et al., 2017; Papagelis et al., 2005; Wang et al., 2018). Therefore, in this paper, we will delve into discussing how LLM recommendations perform in scenarios with user interest shift and explore methods to mitigate this issue by utilizing newly generated user feedback.

3 Preliminary

In this section, we first present the problem formulation (§ 3.1) for the studied user interest shift problem. Next, we present preliminary studies (§ 3.2) to illustrate the importance of timely adapting LLM-based recommenders to new user interests and analyze the potential of applying ICL to address this problem (§ 3.3).

3.1 Problem Defination

In recommendation, users are expected to continuously interact with the system, making the interaction data streamingly coming. We represent the streaming data T as $\{D_0, \ldots, D_t, \ldots, \}$, where D denotes a set of recommendation data points. and D_t denotes the data collected at t-time period. During the process, their interest could evolve. To ensure the recommendation performance, we usually update the recommenders using the newly collected data to capture the new interest. However, updating LLM-based recommender models is costly. Therefore, we usually need to train a model with $\{D_0, \ldots, D_T\}$ (denoting the trained model as f_T), but need it to serve for many periods, e.g., K periods, form D_{T+1} to D_{T+K}. In this work, we consider developing a method that could capture the user's new interest from the new data **without requiring further model updates after the initial trained**[1].

3.2 Importance of Adapting to New Interests

We conduct preliminary experiments using two representative LLM-based models, TALLRec (Bao et al., 2023b) and BinLLM (Zhang et al., 2024), to verify the importance of adapting LLM-based recommenders to users' evolving interests. Our analysis is based on Amazon-Books and Amazon-Movies datasets. Specifically, we uniformly divide the data according to the timestamp and use $\{D_0, \ldots, D_T\}$ to train TALLRec and BinLLM, obtaining the trained model f_T. Additionally, we train the model on $\{D_0, \ldots, D_{T+K}\}$ to obtain more updated model, f_{T+K}. We then compare the performance of the models on D_{T+1} and D_{T+K+1}. We define two metrics according to the performance difference between models to demonstrate the importance of capturing users' new interests:

- PDT: This metric evaluates the performance gap of the same model across two distinct testing periods, where smaller differences indicate stronger robustness to user interest shift. We compute PDT as:

$$PDT = AUC(f_T; D_{T+1}) - AUC(f_T; D_{T+K+1}). \quad (1)$$

- PDM: This metric measures the gap between the model and its upper bound (obtained by retraining with all the data before the test period) using a fixed test set. It aligns with real-world scenarios where retraining models with new data is a common practice (Yang et al., 2024; Xu et al., 2020b). Performance differences before and after retraining provide the potential benefits of model updates. A smaller gap suggests strong adaptability to new test environments, reducing the need for frequent updates. We compute PDM as:

$$PDM = AUC(f_{T+K}; D_{T+K+1}) - AUC(f_T; D_{T+K+1}), \quad (2)$$

where $AUC(f; D)$ represents the AUC evaluated for model f on dataset D.

Results. Figure 2 summarizes the results, with a traditional recommender model (HashGNN) included as a reference. From the figure, we can

[1] In all our experiments, we set T = 4 and K = 4.

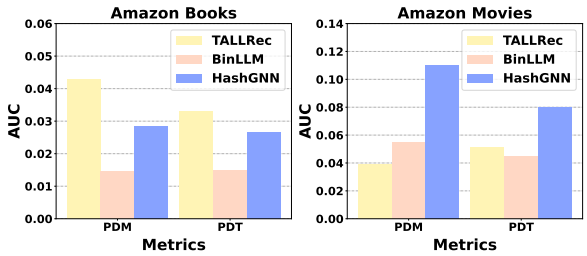

Figure 2: A comparative analysis of TALLRec, BinLLM, and HashGNN models using both Amazon-Book and Amazon-Movie datasets for performance assessment. A higher "PDM" indicates a greater benefit of updating the model. A higher "PDT" signifies a more significant impact of shifts in user interests on the model's performance.

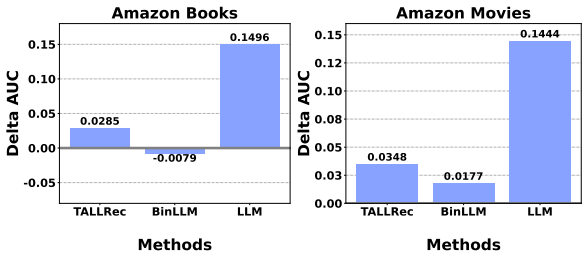

Figure 3: Performance improvement brought by using in-context learning of TALLRec, BinLLM, and General LLM on each dataset, where in-context learning refers to selecting the four most recent interactions for each user to formulate the few-shot examples.

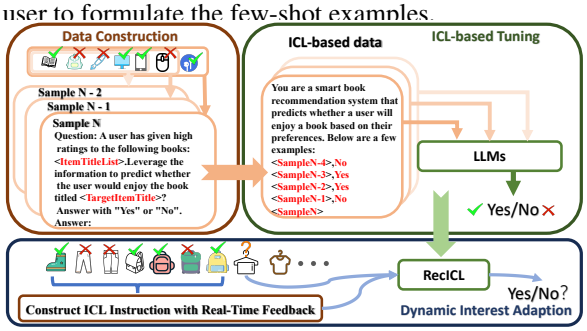

Figure 4: Overview of our RecICL pipeline, primarily consists of three stages: Data Construction, Model Training, and Dynamic Interest Adaption. Here we define the few-shot number as 4.

make the following observations: 1) Both TALLRec and BinLLM exhibit positive values on metric PDT, indicating that as the testing data shifts from D_{T+1} to D_{T+K+1}, the model f_T experiences a noticeable performance decline. 2) TALLRec and BinLLM also show positive values on metric PDM, meaning that the more updated model (f_{T+K}) outperforms the less updated model (f_T) when tested on D_{T+K+1}. This demonstrates that updating the model leads to improved results. The results of LLM-based recommenders are generally consistent with those of traditional models. All the results confirm that LLM-based recommenders also need to adapt to users' evolving interests; otherwise, they risk sub-optimal performance.

3.3 The Potential of ICL

ICL is promising for addressing user interest shift. It enables LLMs to quickly adapting to new tasks using few-shot examples without changing their parameters. Thus, we can expect integrating user feedback as few-shot examples can enhance the model's ability to recommend based on new interests.

However, applying ICL to existing LLM-based recommenders may not work well, as they can lose ICL capabilities during tuning. We compare ICL capabilities between these models and general LLMs by evaluating the performance improvements that ICL brings. Specifically, following the setting in §3.2, we train BinLLM and TALLRec on$\{D_0, \ldots, D_T\}$, comparing their performance (AUC) with and without ICL to measure improvements, noted as Delta AUC. A higher value of this metric indicates greater improvement brought by ICL to the model. Figure 3 summarizes the results on D_{T+K+1}. The findings indicate that TALLRec and BinLLM gain little to no performance boost with ICL, indicating a loss of ICL capabilities in LLM-based recommenders. Moreover, we cannot rely on the ICL abilities of general LLMs alone for recommendations as they lack sufficient capability (Bao et al., 2023b). This motivates us to preserve the ICL abilities when equipping LLMs with recommendation capabilities.

4 RecICL

4.1 Overview

With the insight of keeping the ICL abilities when tuning LLMs to recommendation tasks, we proposed RecICL for achieving adaptive personalized recommendation. The core lies in directly organizing the training example in an ICL format, making the ICL usable during tuning while capturing users' dynamic interests. As Figure 4 shows, our total RecICL framework includes three main components: (1) Data Construction (2) ICL-based Tuning, and (3) Dynamic Interest Adaption.

4.2 Data Construction

In our framework, each user interaction would be formatted as an instruction, as shown on the left side of Figure 4, where the task is described using

language text. Let (h_u, i, y) denote an interaction between a user u and item i in the dataset, where y is the interaction label, and h_u denotes the user's interaction history (including both interacted items and their labels). Next, we use the prompt template outlined in the "Data Construction" part of Figure 4 — feeding h_u and i into the '<ItemTitleList>' field and '<TargetItemTitle>' field in the prompt template, respectively. Let x represent the prompt generated for the interaction (h_u, i, y); it can finally be expressed as (x, y).

For each user, we convert her/his all interactions as above, obtaining a series of transformed data points:

$$[(x_0, y_0), \ldots, (x_n, y_n), \ldots, (x_N, y_N)], \quad (3)$$

where (x_n, y_n) represents the n-th interaction and N is the total number of interactions for the user. Here, the interaction are organized in chronological order based on the interaction timestamps.

4.3 ICL-based Tuning

To maintain the model's ICL abilities while tuning it on recommendation data, we propose a new ICL-based tuning method. Instead of tuning the model to predict y_n based solely on x_n for each interaction (x_n, y_n), we incorporate several of the most recent interactions for each user to help the predictions. These data points are integrated in an ICL format. Specifically, for each (x_n, y_n), we concatenate x_n with the M [2] most recent interactions (denoted as $\{(x_{n-1}, y_{n-1}), \ldots, (x_{n-M}, y_{n-M})\}$) as few-shot examples to construct the ICL instruction data, following the prompt template shown in the "ICL-based Tuning" section of Figure 4. Let x'_n represent the resulting prompt. Formally,

$$x'_n = P_{ICL}(\{(x_{n-1}, y_{n-1}), \ldots, (x_{n-M}, y_{n-M})\}; x_n), \quad (4)$$

where $P_{ICL}(\cdot)$ denotes the process for constructing the ICL instruction. Then each training data point can be represented by (x'_n, y_n).

After generating all the ICL instruction data, we use it to tune the model, ensuring that it retains its ICL capabilities while learning the recommendation task. Specifically, we fine-tune the LLM by minimizing the following optimization objective:

$$\underset{\theta}{minimize} \sum_{(x'_k, y_k)} \ell(f(x'_k; \theta), y_k), \quad (5)$$

where θ represents the LLM's parameters, $f(x'_k; \theta)$ denotes the model's prediction for x'_k, and ℓ is the commonly used cross-entropy loss.

By adopting this approach, we can achieve two goals simultaneously. On the one hand, we can align the LLM with recommendation scenarios by training it on user interaction data. On the other hand, we can prevent the LLM from experiencing catastrophic forgetting of ICL ability. More importantly, this method teaches the model how to leverage the users' most recent interests from the few-shot examples during the training process, enabling it to capture users' real-time interests at the inference stage.

4.4 Dynamic Interest Adaption

During recommendation period, the tuned LLM, with its ICL abilities preserved, can capture a user's new interests without requiring model updates. During the decoding time, we use the most-recent user feedback data as the few-shot example in ICL, allowing the model to access the user's newest interests. The process of constructing ICL instruction data for inference is identical to that used in training. Let x' represent a test data point represented in ICL instruction format; the final prediction is made as

$$\hat{y} = f(x', \theta^*), \quad (6)$$

where θ^* are the LLM model parameters tuned according to Equation (5).

5 Experiments

In this section, we will introduce the experiment setting and answer the following research questions: **RQ1**: How does RecICL perform under the user interest shift setting, and does it have any advantages compared to models updated with the full datasets? **RQ2**:Whether RecICL performance more stable across all time periods when comparing with other baselines? **RQ3**: What's the effect of few-shot selection [3] of RecICL?**RQ4**: How does RecICL perform in the most extreme user interest shift scenarios (User is not present in the training set)?

5.1 Experimental Settings

5.1.1 Datasets

We conduct our experiments on the following two real-world datasets[4]: (1) **Amazon-Books** refers

[2]Without explicit statement, we set M=4 in our experiments.

[3]We also analysis the impact of few-shot number in Appendix §C

[4]We also conduct experiments on the additional two datasets in Appendix §D

Table 1: Overall performance comparison on the Amazon-Book and Amazon-Movie datasets on the D_9. ↑ indicates higher values are better, while ↓ indicates lower values are better. "Rel Imp" denotes the relative improvement of RecICL compared to baselines on the AUC metric. "Collab." refers to the traditional collaborative methods. "LLMRec (ICL)" refers to the use of in-context learning with corresponding methods. Note that, ICL-LLM does not have PDM value since the general LLM is no need to update. All LLM methods employing Llama3.1-8B are indicated with a footnote; otherwise, Qwen1.5-0.5B is used. The best performance for each metric is bolded.

Dataset		Amazon-Books			Amazon-Movies		
Methods		AUC(↑)	Rel Imp(↑)	PDM(↓)	AUC(↑)	Rel Imp(↑)	PDM(↓)
Collab.	MF	0.6193	35.65%	0.0882	0.5901	57.30%	0.1565
	SASRec	0.5734	46.51%	0.1125	0.6554	41.62%	0.1029
	HashGNN	0.7396	13.59%	0.0285	0.6628	40.04%	0.1100
LLMRec (ICL)	ICL-LLM	0.7133	17.77%	-	0.7434	24.58%	-
	ICL-LLM_{Llama}	0.5821	44.32%	0.0214	0.7101	30.56%	0.0285
	ICL-TALLRec	0.7290	15.24%	0.0214	0.7763	19.56%	0.0285
	ICL-BinLLM	0.7708	8.99%	0.0053	0.7835	18.64%	0.0604
LLMRec	TALLRec	0.7005	19.92%	0.0428	0.7415	25.18%	0.0392
	$TALLRec_{Llama}$	0.6905	21.67%	0.0127	0.7648	21.37%	0.0328
	BinLLM	0.7787	7.88%	0.0145	0.7658	21.21%	0.0547
	$BinLLM_{Llama}$	0.7809	7.58%	0.0191	0.7639	21.51%	0.0737
Ours	**RecICL-TALLRec**	**0.8401**	-	**0.0031**	0.9145	-	**0.0057**
	RecICL-$TALLRec_{Llama}$	0.8399	-	0.0030	**0.9282**	-	0.0043
	RecICL-BinLLM	0.8353	-	0.0104	0.9055	-	0.0143
	RecICL-$BinLLM_{Llama}$	0.8197	-	0.0164	0.9212	-	0.0028

to the "book" subset of Amazon Review datasets [5]. This dataset consists of user reviews of book products from the Amazon platform between 1996 and 2018, with rating scores ranging from 1 to 5. We chose 4 as the threshold. Those with scores higher than 4 are labeled as "Yes"; otherwise, they are labeled "No". (2) **Amazon-Movies** refers to the "movie" subset of Amazon Review datasets. Similar to the Amazon-Book datasets, we choose the threshold as 4.

To better simulate real-world scenarios that prevent data leakage (Ji et al., 2023) while modeling user interest shifts, we divided the dataset into 10 parts similar to §3.2. Consistent with our setup in the preliminary experiments, by default, we set $T = 4$ and $K = 4$, which means we use $D_{train} = \{D_0, \ldots, D_4\}$ as the training set. The last 5,000 samples from D_4 are separated to form the validation set and we randomly select 5,000 samples from D_9 to serve as the test set for user interest shift. Specifically, for Amazon-Books, we preserved user interactions from the year 2017, and for Amazon-Movies, we preserved user interactions from the year 2014 to 2016. Besides, following the setting in BinLLM (Zhang et al., 2024), we filtered out users and items with fewer than 20 interactions to ensure data quality. [6]

[5] https://cseweb.ucsd.edu/~jmcauley/datasets/amazon_v2/.

[6] The detailed data statistics are shown in Table 2.

5.1.2 Evaluation and Metrics

To demonstrate the effectiveness of our approach, we compared it with a range of methods, including traditional recommendation models (MF (Koren et al., 2009), SASRec (Kang and McAuley, 2018), and HashGNN (Tan et al., 2020)) and current LLM-based recommendation models (TALLRec (Bao et al., 2023b) and BinLLM (Zhang et al., 2024)). [7] For evaluation metrics, we use AUC, a common metric in recommender systems quantifying the overall prediction accuracy, and PDM, as we defined in Equation (2), to evaluate our model's performance. As for PDM, similar to §3.2, we define it as the model's performance between the fully updated model and the less updated model testing on D_9, which indicates how close we are to the upper bound of performance.

5.2 Main Results (RQ1)

Table 1 presents the overall performance of our method on two datasets following significant user interest shifts. From this table, we can draw the following conclusions:

- Compared to all other methods, RecICL significantly improves the AUC metric compared to other methods, enhancing model performance

[7] The default approach for all LLM methods is to use Qwen1.5-0.5B with full-finetuning and use Llama3.1-8B combined with LoRA for training. More details and implementations for all baselines are described in the Appendix §A and §B.

during user interest shifts. Moreover, when examining PDM performance, we observe that the benefit of updating the RecICL model is minimal (0.0031 and 0.0057 for each dataset). It demonstrates stability and maintains high performance over extended periods without updates.

- When comparing traditional recommender systems with LLM-based recommender systems, a notable performance gap in their ability to adapt to changing user interests. In detail, traditional models are more sensitive to user interest shift, with PDM metrics near 0.1 (except for the hashGNN model on the Amazon-book dataset), indicating a need for frequent updates. In contrast, LLMs show robustness against such shifts, suggesting their potential as a solution to the challenge of evolving user interests.

- In our comparison of two datasets, we observe that baseline models are more adversely affected by the Amazon-Movie dataset. This effect is particularly pronounced in HashGNN, where performance metrics significantly change due to the dataset's broader time span and the resulting user interest shift. Addressing this shift is vital for enhancing the effectiveness of recommendation systems. Furthermore, when evaluating the two LLM-based methods across both datasets, we find that BinLLM is less impacted on the Amazon-Books dataset. We speculate that this is because BinLLM is heavily influenced by its collaborative models.

- When comparing RecICL-TALLRec and RecICL-BinLLM, we observe that their performance is remarkably similar. Contrary to expectations, the advantage of BinLLM's use of collaborative information is not clearly evident within the RecICL framework. This unexpected outcome may be attributed to two factors: (1) The global collaborative information provided by the collaborative model may not accurately reflect user interests as effectively as the user's most recent interactions. (2) There might be an ongoing issue with the collaborative model's performance degradation. Despite these observations, it's important to note that when comparing RecICL-BinLLM with the standalone BinLLM, we still see a significant performance improvement. This contrast underscores the effectiveness and high adaptability of our proposed RecICL method.

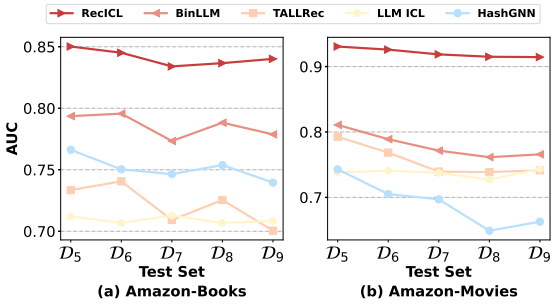

Figure 5: The performance of the model on different test sets after training on $\mathcal{D}_{train}$. The x-axis represents practical data partitions, with larger subscripts indicating a greater shift in user interests compared to the training set. The y-axis shows the corresponding AUC metric for each data partition.

Next, we will further analyze the RecICL method, our subsequent experiments will be based on RecICL-TALLRec since it shows the best performance in our main experiment.

5.3 Robust Analysis (RQ2)

To show the robustness of RecICL, validate its performance across different time periods, we present the results of several methods trained on D_{train} and evaluated on datasets D_5, D_6, D_7, D_8, and D_9 in Figure 5, respectively. For the sake of convenience, we utilize Qwen1.5-0.5B in our experimental analysis. The main findings are as follows:

- In terms of comparative performance, RecICL demonstrates consistent advantages across all periods, further validating the effectiveness of our approach. Even more encouraging is that when tested on D_5, where user interests have not undergone dramatic changes, our method still shows significant improvements over baseline approaches. We attribute this to the fact that while employing ICL, we essentially provide more personalized user input (each user's most recent interaction and its feedback), enabling the model to better model the user. This effectively personalizes the input prompt, leading to substantial improvements.

- When comparing all other LLM-based methods, we found that ICL performance remains relatively stable. Although ICL yields the lowest performance, it does not overfit user preferences from specific periods due to the absence of domain-specific fine-tuning. This further explains the stable performance of RecICL.

- Furthermore, by examining the performance of HashGNN and BinLLM in both datasets, we ob-

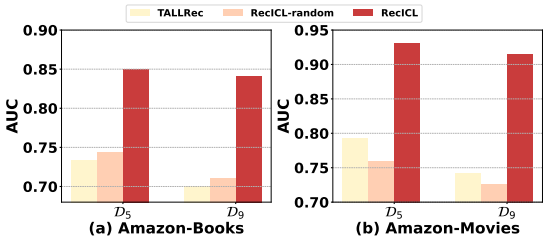

Figure 6: Performance comparison of TALLRec, RecICL, and RecICL using random select few-shot samples on D_5 and D_9.

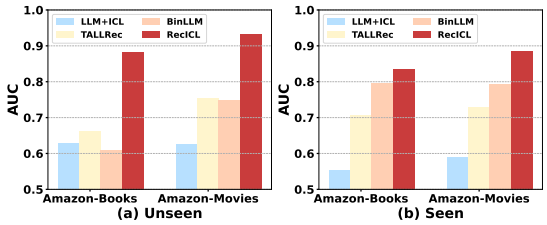

Figure 7: Performance comparison on seen (right) and unseen (left) users on Amazon-Books and Amazon-Movies datasets.

serve that BinLLM is more susceptible to the influence of the collaborative models it relies on, i.e., HashGNN. When the collaborative models are significantly affected by shifts in user interests, BinLLM is also substantially impacted. Nevertheless, BinLLM still demonstrates greater stability compared to HashGNN, underscoring the robustness of LLM-based approaches.

5.4 In-depth Analysis (RQ3)

Few-shot Selection. Wse first delve into the few-shot selection strategy of RecICL, and aim to answer the following question: *How impactful is leveraging users' recent interactions and feedback?* Specifically, we fist present a ablation study, which compare the performance ammong TALLRec, RecICL-random using four randomly selected interactions as few-shot examples, and RecICL. As shown in the figure 6, we observe that with random interactions, RecICL-random does not always show performance improvement compared with TALLRec, demonstrating the importance of choosing the most recent interaction as few-shot examples. This further verified that when using RecICL, we need to perform instance-level personalization to capture the user's dynamic interest thereby improving the recommendation accuracy.

Performance on Unseen Users. Apart from the ablation study, we also consider the most extreme scenario of user interest shift is when models have never seen a user during their training phase. Consequently, this user's interests are entirely unknown to the model, and we can only learn about the user's preferences through their real-time feedback. To validate that RecICL is also effective on this scenario, we divided the interactions in the test set into two categories based on whether the user has appeared in the training set. We then calculated the recommendation performance for each category separately. The results are illustrated in Figure 7.

Our findings indicate that RecICL demonstrates significant performance improvements compared to the baseline for users not present in the training set. This can be attributed to RecICL's ability to effectively leverage recent interactions from new users to model their interests. Besides, for RecICL, we also observed that in addition to smaller performance gains among user groups previously encountered by the model, the overall performance was also slightly lower compared to unseen users. This phenomenon may be caused by the fact that new users tend to have more focused interest preferences closely related to their recent interactions, while old users might have more complex, long-term interests that are not fully captured by our ICL input.

6 Conclusion and Future Work

In this paper, we highlight the challenges faced by LLMs in recommender systems when dealing with user interest shift. Unlike traditional models, LLMs cannot timely update their parameters due to high training costs. To address this issue, we propose RecICL, which ensures that the LLM aligns with the recommendation scenario while preserving and enhancing its in-context learning capabilities in the recommendation context. During deployment, it can utilize the user's most recent feedback by inputting this feedback as few-shot examples to the model, allowing it to capture the user's dynamic interests. Extensive experimental results also illustrate the effectiveness and adaptability of RecICL, successfully adapting to dynamic user interest without any model-level updates. In the future, we aim to delve deeper into this research direction. We'd like to explore ways to enable LLMs to better utilize collaborative information from updated traditional models, aligning with existing incremental learning methods.

Limitations

This paper primarily focuses on the issues and challenges that arise when deploying LLMs for dynamic user interest adaption, particularly focusing on strengthening the ICL capabilities of LLMs in recommendation scenarios.

However, our study has several limitations: 1) The experiments conducted in this study are solely based on the Qwen1.5-0.5B and Llama3.1-8B model, lacking validation with a broader range of models. In the future, we aim to expand experiments accordingly. 2) While the method proposed in this paper provides parameter options that can balance inference time and performance[8], it still encounters efficiency challenges when applied to real-world recommendation scenarios. A potential solution to address the inference efficiency issue is to leverage techniques such as employing prefill (Kwon et al., 2023) methods to mitigate these challenges. However, due to the limitations of our experimental setup, further exploration of this approach remains constrained.

Ethical Considerations

In this paper, we introduce RecICL to enhance the recommendation-specific ICL capability to timely capture the user's dynamic interest. We utilize publicly accessible data while diligently steering clear of sensitive information. Additionally, the implementation of LLMs could unintentionally reinforce hidden societal biases. We advise conducting thorough risk assessments and caution users about the possible risks involved in deploying the model.

References

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. 2023. Qwen technical report. arXiv preprint arXiv:2309.16609.

Keqin Bao, Jizhi Zhang, Xinyu Lin, Yang Zhang, Wenjie Wang, and Fuli Feng. 2024. Large language models for recommendation: Past, present, and future. In Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 2993–2996.

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.

[8]See Appendix §C

Keqin Bao et al. 2023b. Tallrec: An effective and efficient tuning framework to align large language model with recommendation. In RecSys, pages 1007–1014. ACM.

Tom B Brown. 2020. Language models are few-shot learners. arXiv preprint arXiv:2005.14165.

Badrish Chandramouli, Justin J. Levandoski, Ahmed Eldawy, and Mohamed F. Mokbel. 2011. Streamrec: a real-time recommender system. In Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2011, Athens, Greece, June 12-16, 2011, pages 1243–1246. ACM.

Shiyu Chang, Yang Zhang, Jiliang Tang, Dawei Yin, Yi Chang, Mark A. Hasegawa-Johnson, and Thomas S. Huang. 2017. Streaming recommender systems. In Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017, pages 381–389. ACM.

Yu Cui, Feng Liu, Pengbo Wang, Bohao Wang, Heng Tang, Yi Wan, Jun Wang, and Jiawei Chen. 2024. Distillation matters: Empowering sequential recommenders to match the performance of large language models. In Proceedings of the 18th ACM Conference on Recommender Systems, pages 507–517.

Abhinandan S Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. 2007. Google news personalization: scalable online collaborative filtering. In Proceedings of the 16th international conference on World Wide Web, pages 271–280.

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.

Wenqi Fan, Zihuai Zhao, Jiatong Li, Yunqing Liu, Xiaowei Mei, Yiqi Wang, Jiliang Tang, and Qing Li. 2023. Recommender systems in the era of large language models (llms). CoRR, abs/2307.02046.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. 2023. Pre-training to learn in context. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023, pages 4849–4870. Association for Computational Linguistics.

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.

Bowei He, Xu He, Yingxue Zhang, Ruiming Tang, and Chen Ma. 2023a. Dynamically expandable graph convolution for streaming recommendation.

In Proceedings of the ACM Web Conference 2023, pages 1457–1467.

Zhankui He, Zhouhang Xie, Rahul Jha, Harald Steck, Dawen Liang, Yesu Feng, Bodhisattwa Prasad Majumder, Nathan Kallus, and Julian McAuley. 2023b. Large language models as zero-shot conversational recommenders. In Proceedings of the 32nd ACM international conference on information and knowledge management, pages 720–730.

Yitong Ji, Aixin Sun, Jie Zhang, and Chenliang Li. 2023. A critical study on data leakage in recommender system offline evaluation. ACM Trans. Inf. Syst., 41(3):75:1–75:27.

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

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings.

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

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles.

Jiayi Liao, Sihang Li, Zhengyi Yang, Jiancan Wu, Yancheng Yuan, Xiang Wang, and Xiangnan He. 2024. Llara: Large language-recommendation assistant. SIGIR 2024.

Jianghao Lin et al. 2023a. How can recommender systems benefit from large language models: A survey. arXiv preprint arXiv:2306.05817.

Xinyu Lin et al. 2023b. A multi-facet paradigm to bridge large language model and recommendation. arXiv preprint arXiv:2310.06491.

Qidong Liu, Xian Wu, Xiangyu Zhao, Yejing Wang, Zijian Zhang, Feng Tian, and Yefeng Zheng. 2024. Large language models enhanced sequential recommendation for long-tail user and item. arXiv preprint arXiv:2405.20646.

Manos Papagelis, Ioannis Rousidis, Dimitris Plexousakis, and Elias Theoharopoulos. 2005. Incremental collaborative filtering for highly-scalable recommendation algorithms. In Foundations of Intelligent Systems, 15th International Symposium, ISMIS 2005, Saratoga Springs, NY, USA, May 25-28, 2005, Proceedings, volume 3488 of Lecture Notes in Computer Science, pages 553–561. Springer.

Weijia Shi, Julian Michael, Suchin Gururangan, and Luke Zettlemoyer. 2022. Nearest neighbor zero-shot inference. In Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022, pages 3254–3265. Association for Computational Linguistics.

Yang Song, Ziming Zhuang, Huajing Li, Qiankun Zhao, Jia Li, Wang-Chien Lee, and C Lee Giles. 2008. Real-time automatic tag recommendation. In Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval, pages 515–522.

Juntao Tan, Shuyuan Xu, Wenyue Hua, Yingqiang Ge, Zelong Li, and Yongfeng Zhang. 2024. Towards llm-recsys alignment with textual ID learning. SIGIR.

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 WWW '20: The Web Conference 2020, Taipei, Taiwan, April 20-24, 2020, pages 1988–1998. ACM / IW3C2.

João Vinagre, Alípio Mário Jorge, and João Gama. 2014. Fast incremental matrix factorization for recommendation with positive-only feedback. In User Modeling, Adaptation, and Personalization - 22nd International Conference, UMAP 2014, Aalborg, Denmark, July 7-11, 2014. Proceedings, volume 8538 of Lecture Notes in Computer Science, pages 459–470. Springer.

Junshan Wang, Guojie Song, Yi Wu, and Liang Wang. 2020. Streaming graph neural networks via continual learning. In Proceedings of the 29th ACM international conference on information & knowledge management, pages 1515–1524.

Junshan Wang, Wenhao Zhu, Guojie Song, and Liang Wang. 2022. Streaming graph neural networks with generative replay. In Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pages 1878–1888.

Weiqing Wang, Hongzhi Yin, Zi Huang, Qinyong Wang, Xingzhong Du, and Quoc Viet Hung Nguyen. 2018. Streaming ranking based recommender systems. In The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, pages 525–534.

Yancheng Wang et al. 2023. Recmind: Large language model powered agent for recommendation. arXiv preprint arXiv:2308.14296.

Wei Wei, Xubin Ren, Jiabin Tang, Qinyong Wang, Lixin Su, Suqi Cheng, Junfeng Wang, Dawei Yin, and Chao Huang. 2024. Llmrec: Large language models with graph augmentation for recommendation. In Proceedings of the 17th ACM International Conference on Web Search and Data Mining, pages 806–815.

Likang Wu et al. 2023. A survey on large language models for recommendation. arXiv preprint arXiv:2305.19860.

Yunjia Xi et al. 2023. Towards open-world recommendation with knowledge augmentation from large language models. CoRR, abs/2306.10933.

Yishi Xu, Yingxue Zhang, Wei Guo, Huifeng Guo, Ruiming Tang, and Mark Coates. 2020a. Graphsail: Graph structure aware incremental learning for recommender systems. In Proceedings of the 29th ACM International Conference on Information & Knowledge Management, pages 2861–2868.

Yishi Xu, Yingxue Zhang, Wei Guo, Huifeng Guo, Ruiming Tang, and Mark Coates. 2020b. Graphsail: Graph structure aware incremental learning for recommender systems. In CIKM '20: The 29th ACM International Conference on Information and Knowledge Management, Virtual Event, Ireland, October 19-23, 2020, pages 2861–2868. ACM.

Fan Yang, Zheng Chen, Ziyan Jiang, Eunah Cho, Xiaojiang Huang, and Yanbin Lu. 2023. Palr: Personalization aware llms for recommendation. arXiv preprint arXiv:2305.07622.

Yuhao Yang, Lianghao Xia, Da Luo, Kangyi Lin, and Chao Huang. 2024. Graphpro: Graph pre-training and prompt learning for recommendation. In Proceedings of the ACM on Web Conference 2024, WWW 2024, Singapore, May 13-17, 2024, pages 3690–3699. ACM.

Zheng Yuan et al. 2023. Where to go next for recommender systems? ID- vs. modality-based recommender models revisited. In SIGIR 2023, pages 2639–2649. ACM.

Junjie Zhang et al. 2023a. Recommendation as instruction following: A large language model empowered recommendation approach. arXiv preprint arXiv:2305.07001.

Yang Zhang, Keqin Bao, Ming Yan, Wenjie Wang, Fuli Feng, and Xiangnan He. 2024. Text-like encoding of collaborative information in large language models for recommendation. ACL 2024.

Yang Zhang, Fuli Feng, Jizhi Zhang, Keqin Bao, Qifan Wang, and Xiangnan He. 2023b. Collm: Integrating collaborative embeddings into large language models for recommendation. arXiv preprint arXiv:2310.19488.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. arXiv preprint arXiv:2303.18223.

Bowen Zheng et al. 2024. Adapting large language models by integrating collaborative semantics for recommendation. ICDE 2024.

A Baselines.

To fully investigate the performance of our RecICL, we mainly consider two types of baseline models, one is conventional recommender systems, and another is the recommender systems based on LLM. In detail, we select the following baselines:

- **MF** (Koren et al., 2009) refers to Matrix Factorization, which is a popular collaborative filtering technique, which works by decomposing the user-item interaction matrix and representing latent factors for users and items for predictions.
- **SASRec** (Kang and McAuley, 2018) refers to Self-Attentive Sequential Recommendation, which leverages the self-attention mechanism to capture long-term user preferences and item relationships, allowing it to model complex sequential patterns in user behavior.
- **HashGNN** (Tan et al., 2020) refers to Hashing with GNNs, which consists of a GNN encoder and a hash layer for encoding representations to hash codes. It can be viewed as a representation of the GNN-based method for collaborative filtering.
- **ICL** refers to how we directly apply the in-context learning techniques to prompt the LLM to determine whether the user will enjoy the item by giving the most recent interactions and the feedback of the user.
- **TALLRec** (Bao et al., 2023b) is a representation of LLM-based recommender systems that directly use instruction-tuning to finetune the LLM on recommendation data and achieve moderate performance.
- **BinLLM** (Zhang et al., 2024) is currently a state-of-the-art (SOTA) method for aligning LLMs with recommendation. It introduces collaborative information to LLMs by compressing the embedding from traditional recommender systems to 32-bit binary sequences and feeding it into the LLMs.

B Implementation Details

Similar to BinLLM (Zhang et al., 2024), for traditional recommender systems, we employ Binary Cross-Entropy (BCE) as the optimization loss and use the Adam optimizer (Kingma and Ba, 2015), unless otherwise specified by the original paper.

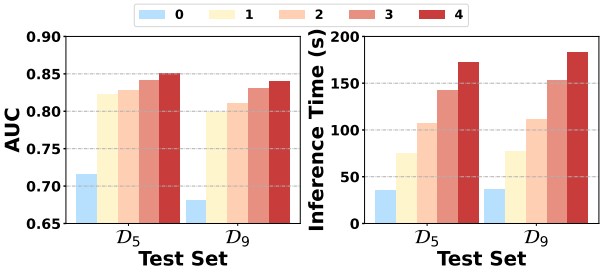

Figure 8: The performance of RecICL trained with varying numbers of few-shot samples (left) and its inference overhead on the entire test set (right). Note that when the number of few-shot samples is 0, it is equivalent to TALLRec.

For hyperparameter tuning, we explore the learning rate in $[1e{-}2, 1e{-}3, 1e{-}4]$ and tune the weight decay in the range of $[1e{-}2, 1e{-}3, \ldots, 1e{-}7]$. For embedding size, we perform tuning within the range of $[16, 32, 64, 128, 256, 512]$. For SASRec, we set the maximum length of historical interaction sequences according to the average user interaction count in the training data, as specified in the original paper. For all LLM-based methods, we employ the AdamW optimizer and adjust the learning rate within the range of $[1e{-}3, 1e{-}4, 1e{-}5]$. We set up 200 to the warm-up steps in our training process. Regarding the input interaction sequence length, we follow the TALLRec (Bao et al., 2023b) approach by setting the maximum sequence length to 10. For BinLLM, we utilize the optimal HashGNN and adapt it for binary sequence embedding. As for the backbone, we opt for Qwen1.5-0.5B (Bai et al., 2023), considering its convenience and efficiency and we also apply Llama3.1-8B (Dubey et al., 2024) with lora-tuning to further validate the method's effectiveness.

C Influnce of Few-shot Number

We first analyze the number of few-shot samples when applying RecICL, as shown in Figure 8. The figure illustrates the model's performance on D_5 and D_9 and the inference time changes for all 5000 samples. We can draw the following conclusions:

- When considering the overall performance, regardless of the number of few-shot samples chosen, the model's performance shows a qualitative improvement compared to zero samples (i.e., TALLRec). When considering Table 1, even RecICL with just one few-shot sample demonstrates a clear performance advantage over the previous SOTA method, BinLLM, which further demonstrates the effectiveness of our approach.
- Besides, we found that as the number of few-shot samples increases, the model continues to improve its recommendation performance. However, the most significant performance boost occurs when the number of few-shot samples increases from 0 to 1. This further indicate the importance of user's most recent interaction which directly reflect their current preference.
- When looking at the right figure, we observed that the inference time grows approximately linearly with the increase in few-shot samples, mainly due to the increased input length. This issue could potentially be addressed through prefill optimization.

In summary, there is a trade-off between the performance gains and the increased inference time brought by few-shot samples. When prioritizing performance, more few-shot samples can be used; when seeking balance, using 1 or 2 few-shot samples can bring noticeable performance improvements.

D Performance on Other Datasets

We conducted additional experiments on the Amazon-CDs and Amazon-Sports datasets using the Qwen 1.5-0.5B model. The data processing pipeline follows a similar pipeline to that of Amazon-Books and Amazon-Movies. For the CDs dataset, we utilize data from 2008, while for the Sports dataset, we use data starting from 2014, ensuring a comparable scale with the datasets in our main experiments. Following the approach described in Section §5, we employ $\{D_0, \ldots, D_4\}$ as the training set. The last 5,000 samples from D_4 are reserved for validation, and we randomly select 5,000 samples from D_9 to construct the test set for evaluating user interest shift. On these datasets, we compare two LLM-based methods, TALLRec and BinLLM, as baselines. Similar to our analysis experiments, we focus on RecICL-TALLRec due to its superior performance in the main experiments.

The results, summarized in Table 3, demonstrate that consistent with the findings on the Books and Movies datasets, TALLRec and BinLLM exhibit significant performance degradation as user interests shift, as reflected by higher PDM values. In contrast, our method not only maintains strong performance but also shows greater resilience to changes in user interest. This further validates that

Table 2: Data statistics of the datasets.

Dataset	#Interaction	#User	#Item
Amazon-Books	775,635	22,127	34,076
Amazon-Movies	378,329	11,799	14,632
Amazon-CDs	310,904	9,782	14,680
Amazon-Sports	253,972	10,968	24,731

Table 3: Performance Comparison on CDs and Sports Datasets

Methods	CDs		Sports	
	AUC(↑)	PDM(↓)	AUC(↑)	PDM(↓)
TALLRec	0.5948	0.0410	0.5725	0.0864
BinLLM	0.5815	0.1280	0.5781	0.0802
RecICL	**0.8162**	**0.0088**	**0.7592**	**0.0329**

our approach effectively adapts to dynamic user interests and provides personalized recommendations based on users' most recent preferences.