LLMs For Warm and Cold Next-Item Recommendation: A Comparative Study across Zero-Shot Prompting, In-Context Learning and Fine-Tuning

Haya Halimeh
Paderborn University
Paderborn, Germany *haya.halimeh@uni-paderborn.de*

Florian Freese
University of Wuppertal
Wuppertal, Germany *freese@uni-wuppertal.de*

Oliver Müller
Paderborn University
Paderborn, Germany *oliver.mueller@uni-paderborn.de*

Abstract

Recommendation systems are essential for delivering personalized content across e-commerce and streaming services. However, traditional methods often fail in cold-start scenarios where new items lack prior interactions. Recent advances in large language models (LLMs) offer a promising alternative. In this paper, we adopt the retrieve-and-recommend framework and propose to fine-tune the LLM jointly on warm- and cold-start next-item recommendation tasks, thus, mitigating the need for separate models for both item types. We computationally compare zero-shot prompting, in-context learning, and fine-tuning using the same LLM backbone, and benchmark them against strong PLM-based baselines. Our findings provide practical insights into the trade-offs between accuracy and computational cost of these methods for next-item recommendation. To enhance reproducibility, we release the source code under `https://github.com/HayaHalimeh/LLMs-For-Next-Item-Recommendation.git`.

Keywords: LLM for recommendations, next-item recommendations, cold-start problem.

1. Introduction

Serving as vital conduits for delivering engaging content to users, recommendation systems (RS) have become indispensable across a broad spectrum of e-commerce applications, such as online shopping [45], video [6], music [30], and movie platforms [13]. Traditional RS base on collaborative filtering, content-based methods, or a mix of both. Collaborative filtering assumes users will continue to prefer items similar to those they liked before, while content-based methods match user profiles with item attributes [28].

In challenging scenarios, as in the cold-start item problem, where RS have to recommend items that lack any interaction history, these models often struggle [28]. The cold-start item problem frequently arises in real-world streaming media services [5]. For instance, how can a movie platform recommend a newly released movie that no one has watched yet?

Collaborative filtering fails in cold-start settings because it relies on interaction histories. Content-based methods can make use of item descriptions to match cold-start items to users. However, they ignore the relational dynamics between users and their historically selected items [35]. As a result, they may lose valuable insights as they do not preserve the sequential nature of user consumption over time. Overcoming this challenge requires solutions that can generalize well to improve user experience and drive greater engagement [12]. Pre-trained language models (PLMs) offer promising avenues in this regard, due to their strong capabilities in sequential understanding and reasoning over text [3]. Building on this line of thought, researchers have framed the next-item recommendation task as a language modeling problem [23], [31], [44]. More recently, LLMs have gained attention for next-item recommendation [36], with the retrieve-and-recommend framework emerging as a promising approach [9]. Here, the LLM receives a prompt including a user's interaction history along with a set of candidate items. The LLM is then tasked with inferring the most relevant next items from the given candidate set based on that user history [12], [29]. This framework mirrors conventional RS pipelines, where external candidate sets are created first and the recommendation model functions as a ranking [41] or selection mechanism [24]. Prior work within the retrieve-and-recommend framework has mainly focused on directly prompting the LLM in a zero-shot or in-context learning (ICL) manner [7], [16], [34]. However, since LLMs are not inherently optimized for recommendation tasks, their performance can suffer unless adapted or fine-tuned appropriately [1], [35], [42]. We see a major advantage of the retrieve-and-recommend framework, as it can be purposed to handle both warm-start items (with prior user interactions) and cold-start items (with none) using the same system and, thus, mitigating the need for separate models for different item types.

As such, in this paper, (i) we propose to fine-tune the LLM on both warm- and cold-start next-item recommendation tasks before prompting. By tuning on both scenarios simultaneously, we hypothesize that the model can learn to accommodate both item types. Additionally, to build the training data, we combine the standard leave-one-out strategy from next-item recommendation literature [39] with a disjoint user split, ensuring that both users and cold-start items in the test set are entirely unseen during training. This setup compels the model to rely on generalizable patterns to generate recommendations to new items and users that it had not encountered before to minimize the necessity of repetitive fine-tuning. (ii) Using the same open-source LLM backbone, we evaluate and compare zero-shot prompting, ICL, and fine-tuning approaches, and benchmark them against state-of-the-art PLM-based baselines. We also assess the computational costs of each method and explore how their performance varies based on factors such as the size of the candidate set and the number of training examples used during fine-tuning.

The remainder of this paper provides in Section 2 a brief background on traditional and LLM-based methods for next-item recommendation, details the methodology in Section 3, illustrates the experimental setting in Section 4, presents the results in Section 5 and discuss them with limitations and future directions in Section 6. Finally, Section 7 concludes this paper.

2. Background

The next-item recommendation task considers the sequence of items a user has interacted with in the past to generate personalized recommendations that adapt to her evolving interests [33]. Early methods used Markov Chains to capture item transitions [27], while later work adopted deep neural networks to model sequential patterns more effectively [20], as well as, data augmentation and contrastive learning to improve user and item representations [38], [46]. However, these methods rely on historical user interactions and, therefore, fail when encountering cold-start items that lack prior interactions. One of the primary cold-start solutions is to incorporate auxiliary user or item information, such as social networks, genres, and tags [2], [4] with the goal of learning a latent space that connects users to cold-start items. Helpful as that is, this content-based solution does not preserve the interaction order in which users select their historical items and may therefore lose relevant information for next-item recommendation [35].

Recent advances in natural language processing have led researchers to frame recommendation as a language modeling task. For instance, by leveraging the reasoning capabilities and sequential understanding of PLMs, early approaches reformulated item-based recommendation into cloze-style tasks—where models predict the next-item a user might like by either predicting masked tokens in a text sequence [44] or by assigning relevance scores to items [31].

With the emergence of LLMs, more attention has shifted toward their application in RS. Broadly speaking, LLM-based recommendation methods fall into one of two paradigms: non-tuning and tuning [11], [36]. In the non-tuning paradigm, the LLM's internal weights remain fixed and the model is guided via prompts. A popular strategy in this space is the retrieve-and-recommend framework [7], [9], [16], [34]. Here, instead of letting the model generate items from an open vocabulary, the model is restricted to choose from a fixed list of candidate items. Specifically, the LLM is given a prompt that includes the user's interaction history and a relatively small set of possible next items (including the ground-truth next-item and distractor items). The model is then task with selecting the most likely next-item based on that user's history [7], [9], [12]. Techniques in this area include zero-shot prompting [7], [34], where the prompt includes a task-specific instruction to guide the model, and ICL [16], where the prompt also includes one or a few interaction examples to help the model better understand the recommendation task [36], [11]. This formulation serves three practical purposes: (1) An unconstrained recommendation space complicates practical usage of LLMs [43]. (2) It aligns well with retrieval-based recommendation systems, where an external module pre-selects items and the LLM functions as a ranking [41] or selection mechanism [24]. (3) It mitigates the hallucination problem, as it conditions LLMs on retrieved items that match those in the item database [9].

However, although non-tuning methods offer convenience for deployment, empirical evidence shows that they often underperform because LLMs are not inherently optimized for recommendation tasks out of the box [1], [42]. The tuning paradigm addresses this by adjusting model parameters—partially or fully—to better suit recommendation tasks, consequently improving performance [42]. LLM-based next-item recommenders are often fine-tuned, as in traditional RS, on past interactions [39] and then used to directly generate the most likely next-item without relying on a candidate set [19], making them unsuitable for cold-start items.

Instead, we propose to leverage the retrieve-and-recommend framework and to fine-tune the LLM on both warm- and cold-start on disjoint users for next-item recommendation tasks prior to prompting. We hypothesize that this joint fine-tuning step enables the model to better generalize across both item types to select relevant items—even when the target item is entirely new.

3. Methodology

The following subsections provide a detailed overview of the methodology, including data construction setup, candidate set generation, and the implementation of each LLM approach. A visual summary of the overall methodology is presented in Figure 1.

3.1. Dataset Construction

To reproduces the standard next-item recommendation task for warm and cold-start items simultaneously in a dataset, let

$$U = \{u_1, u_2, \dots, u_z\}, \qquad I = \{i_1, i_2, \dots, i_k\} \tag{1}$$

denote the sets of users and items in that dataset, respectively. We sample $p\%$ of the items uniformly at random and mark them as cold-start items, obtaining

$$J = \{j_1, j_2, \dots, j_\ell\} \subseteq I. \tag{2}$$

For every user u, the interaction history is represented by a time-ordered sequence

$$H_u = \langle h_1, h_2, \dots, h_{T_u} \rangle, \qquad where\ T_u = |H_u|. \tag{3}$$

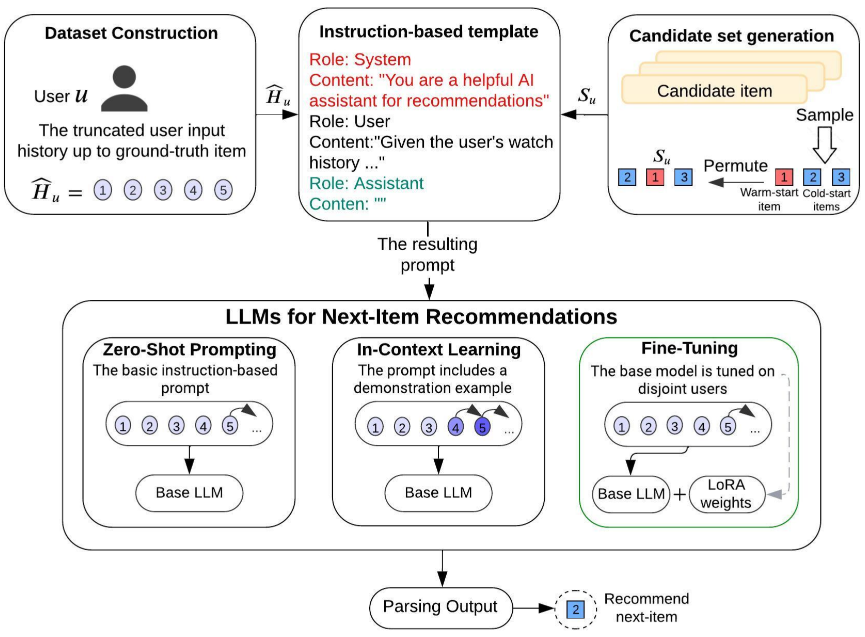
Dataset Construction
User u
The truncated user input history up to ground-truth item
$\widehat{H}_u$ = 1 2 3 4 5
$\widehat{H}_u$
Instruction-based template
Role: System
Content: "You are a helpful AI assistant for recommendations"
Role: User
Content:"Given the user's watch history ..."
Role: Assistant
Conten: ""
S_u
Candidate set generation
Candidate item
Sample
S_u
Permute
Warm-start item
Cold-start items
The resulting prompt
LLMs for Next-Item Recommendations
Zero-Shot Prompting
The basic instruction-based prompt
Base LLM
In-Context Learning
The prompt includes a demonstration example
Base LLM
Fine-Tuning
The base model is tuned on disjoint users
Base LLM
LoRA weights
Parsing Output
Recommend next-item

Fig. 1. LLMs for warm and cold-start next-item recommendations across zero-shot prompting, in-context learning and fine-tuning approaches.

We construct for every user u an evaluation pair $(\widehat{H}_u, g_u)$, where g_u is the ground-truth item that has to be predicted and $\widehat{H}_u$ the truncated user history that does not include the withheld ground-truth item. Taking the chronologically ordered nature of histories into consideration, we identify the positions at which user u has interacted with a cold-start item

$$P_u = \{\, t \mid h_t \in J \}. \tag{4}$$

Two mutually exclusive cases arise:

(i) Warm-start item case ($P_u = \varnothing$): In this case, user u has never encountered a cold-start item from J. Therefore, we define the ground-truth item as the final interaction and the prefix as the truncated user history

$$g_u = h_{T_u}, \qquad \widehat{H}_u = \langle h_1, h_2, \ldots, h_{T_u - 1} \rangle. \tag{5}$$

(ii) Cold-start item case ($P_u \neq \varnothing$): Here, user u has interacted with at least one cold-start item from J. Let $t_u^* = \min P_u$ be the index of the first cold-start item that user u has interacted with. We flag this item as the prediction target. To avoid information leakage from future interaction, only the interactions strictly preceding this item form the truncated user history

$$g_u = h_{t_u^*}, \qquad \widehat{H}_u = \langle h_1, h_2, \ldots, h_{t_u^* - 1} \rangle. \tag{6}$$

3.2. Candidate set generation

To generate the candidate set for each user, we sample uniformly at random from both warm- and cold-start items o distractor items that are neither in the truncated user history nor equal to the ground-truth

$$S_u = \{s_1, s_2, \ldots, s_o\}, \qquad S_u \cap \left(\widehat{H}_u \cup \{g_u\}\right) = \varnothing, \tag{7}$$

and then insert g_u into this set, resulting in a candidate set size of $o + 1$. The candidate set is randomly permuted before being integrated into the prompts. As discussed earlier, the model receives the pair $(\widehat{H}_u, S_u)$, where the task is to select the item from the given candidate set S_u, that user u is most likely to interact with next based on her/his truncated user history $\widehat{H}_u$.

3.3. Zero-Shot Prompting

In line with previous work [1], [10], [42], we use an instruction-based template and the LLaMA model family [32]. LLaMA models support different roles, including system, user and assistant. The system role defines the context for the interaction, while the user role provides the input to the model (e.g. instruction and task-specific input). The assistant role then processes this input and generates responses accordingly. The prompt template is structured as follows:
Role: System
Content: "You are a helpful AI assistant for recommendations"
Role: User
Content:"Given the user's watch history: $\{\widehat{H}_u\}$, and a set of $o + 1$ candidate movies: $\{S_u\}$. Recommend the next movies from the candidate set that the user would most likely watch next. Return only the titles of the movies in ranked order without explanations."
Role: Assistant
Content: ""

3.4. In-Context Learning (ICL)

ICL additionally incorporates demonstration examples in the prompt. In the case of personalized recommendations, introducing examples from other users may introduce noise, as different users typically have distinct preferences. To avoid this, we follow [16] and adapt ICL by taking the user's own interaction sequence as the source of the demonstration example. Specifically, we extract the second last item h_{T-1} as the ground-truth item, and use the prefix of the interaction sequence up to it $\widehat{H}_u \setminus h_{T-1}$ as the truncated input history. We also provide a different candidate set $\widehat{S}_u$ that is independently created from S_u. The prompt template is structured as follows:
Role: System
Content: "You are a helpful AI assistant for recommendations"
Role: User
Content: "Given the user's watch history: $\{\widehat{H}_u \setminus h_{T-1}\}$, and a set of $o + 1$ candidate movies: $\{\widehat{S}_u\}$. The user chose to watch $\{h_{T-1}\}$. Recommend the next movies from the new candidate set $\{S_u\}$ that the user would most likely watch next. Return only the titles of the movies in ranked order without explanations"
Role: Assistant
Content: ""

3.5. Fine-Tuning

We fine-tune the LLM jointly on warm- and cold-start item recommendation tasks prior to prompting. Once the dataset creation step is complete, we apply an 80/10/10 user split for training, validation, and test. Users differ based on whether their ground-truth item is cold-start (never seen during fine-tuning) or warm-start (seen during fine-tuning). We hypothesize, that this setup activates the model's ability to generalize by generating recommendations for disjoint users with (i) unseen cold-start items and (ii) warm-start items it was trained on. Using the same prompt template as in zero-shot prompting, the model learns during training to select the most likely next-item from the candidate set regardless of item type. At test time, we assess whether this fine-tuning strategy improves task adaptation and reduces the need for repeated retraining.

4. Experimental setting

4.1. Dataset

We use the MovieLens-1M (ML-1M) dataset [14] for evaluation. This dataset contains three main components: (1) users, each identified by a unique ID; (2) movies, each with metadata such as titles; and (3) interactions, which are time-stamped ratings that indicate user preferences and reflect the sequence in which users rated movies. In our experiments, we represent each user by their chronological interaction history, where each interaction is expressed in terms of the movie title they interacted with and rated. The goal of the next-item recommendation task is to predict the item a user is likely to interact with next.

We selected this dataset for its long user interaction histories, which allow us to simulate the cold-start next-item recommendation scenario without significantly reducing the user base. Following standard practice, we retain users with at least five ratings (5-Core) [16] and randomly designate 10% of the items as cold-start. Users are split into disjoint training, validation, and test sets with no overlap, and cold-start items in validation and test remain unseen during training. To ensure balanced evaluation, we maintain a roughly 1:1 ratio of cold-start to warm-start item cases through random sampling. Table 1 summarizes the descriptive statistics after processing.

Table 1. Descriptive statistics of the ML-1M after processing: *#Users (CGI)* and *#Users (WGI)* denote the number of users with a cold or warm-start ground-truth item, respectively.

	Training	**Validation**	**Test**
#Users (WGI)	2692	315	357
#Users (CGI)	2019	331	326
Ratio CGI/(WGI + CGI)	0.43	0.51	0.48

4.2. Baseline Methods

We group the baseline methods into two categories. The first includes rule-based methods like Random, which simply recommends items at random. The second includes strong PLM-based approaches, commonly used for next-item recommendation [31], [37]. EmbSim [44] compares the user's history and item titles using embeddings and selects the item with the highest cosine similarity. We use the *stella_en_1.5B_v5* model for this, as it is currently the top-performing model on the MTEB benchmark [26]. PairNSP [21] uses BERT's next sentence prediction to estimate how likely a candidate item logically follows the user's history. Items are ranked by this probability. ItemCLS [40] uses zero-shot classification with *bart-large-mnli*, treating the user history as a "premise" and each candidate item as a "hypothesis" to rank items by likelihood of being the next relevant choice.

4.3. Evaluation Metrics

We report the top-1 hit ratio (HR@1), which measures how often the model's top recommendation matches the ground-truth item. Additionally, since LLMs may generate invalid outputs—such as hallucinated items—we also report the inverse compliance rate (ICR) [7]. ICR is defined as the proportion of invalid predictions relative to the total number of test samples, where a lower ICR indicates better adherence to the task. Each prediction falls accordingly into one of the following three types: Correct prediction, where the top-ranked generated item matches the ground-truth item. False prediction, where the top-ranked generated item appears in the candidate set but does not match the ground-truth item. Hallucinated prediction, where the generated item does not appear in the candidate set. To evaluate predictions, we first clean the LLM outputs (e.g., by removing line breaks) and compare them to the candidate set using fuzzy string matching, where we retain only high-confidence matches (similarity score ≥ 90).

4.4. Implementation Details

The experiments are conducted locally on a system equipped with two NVIDIA TITAN RTX GPUs, each with 24 GB of VRAM, running NVIDIA-SMI 535.171.04 and CUDA version 12.2.

We use *"Llama-3.1-8B-Instruct"* from the LLaMA family [32] as the base LLM model.

Fine-tuning is carried out using the parameter-efficient LoRA technique [17] with 4-bit quantization [8] and cross-entropy [32] as loss. Given the varying lengths of user histories, we set the maximum sequence length to 1024 and apply right-side padding to ensure uniform sequence lengths. Unless stated otherwise, all experiments use a training setup with three epochs, AdamW 8-bit optimizer [25], constant learning rate 1e-4, weight decay 0.01, warmup ratio 0.1, micro batch size 1, and an early stopping mechanism to prevent overfitting. For text generation, we employ nucleus sampling [15], setting top-p at 0.9 to ensure a balance between diversity and coherence, and using a temperature of 0.6 to regulate randomness in the output.

4.5. Sensitivity Analysis

In real-world systems, the size of the candidate set can vary. Therefore, we evaluate the performance of the LLM-based approaches across different candidate set sizes. In addition, to test fine-tuning efficiency in low-data scenarios, we adopt a k-shot training setup, where only a limited number of samples from the training set are selected for model training. By setting a relatively small value for k, we assess whether the LLM can efficiently learn recommendation capabilities under severely limited training data.

5. Results

Table 2 presents the ICR@1 and HR@1 results aggregated over five independent runs, with a candidate set size of 10. The key observations are as follows: The results demonstrate notable performance disparities between warm and cold next-item recommendations, suggesting that the cold-start item problem is also challenging for the fine-tuned LLM model.

In line with existing research [16], [29], ICL outperforms zero-shot prompting. However, it falls behind the simpler more efficient PairNSP approach, while the proposed fine-tuning strategy achieves the best performance for both warm and cold-start next-item recommendation.

The low ICR observed in the LLaMA model across all approaches, highlights the effectiveness of adopting a system role and instruction template when aligning LLMs to the recommendation task. While previous research suggests that longer inputs may degrade the reasoning capabilities of LLMs [22], our findings suggest that ICL remains task-aligned and fine-tuning significantly enhances compliance behavior, achieving ICR values that are at zero.

Table 2. Performance results aggregated over 5 independent runs.

Strategy	**WGI (HR@1)**	**CGI (HR@1)**	**Avg (HR@1)**
Random	0.103 ± 0.029	0.080 ± 0.032	0.092 ± 0.031
EmbSim (Setlla)	0.289 ± 0.013	0.122 ± 0.013	0.206 ± 0.013
PairNSP (BERT)	0.267 ± 0.012	<u>0.182 ± 0.017</u>	<u>0.225 ± 0.015</u>
ItemCLS (BART)	0.112 ± 0.022	0.121 ± 0.012	0.117 ± 0.017
Zero-Shot (LLaMA)	0.289 ± 0.007	0.112 ± 0.020	0.201 ± 0.014
ICL (LLaMA)	<u>0.343 ± 0.022</u>	0.088 ± 0.008	0.216 ± 0.015
Fine-tuned (LLaMA)	**0.401 ± 0.053**	**0.210 ± 0.020**	**0.306 ± 0.037**
	WGI (ICR@1)	**CGI (ICR@1)**	**Avg (ICR@1)**
Zero-Shot (LLaMA)	0.06	0.05	0.07
ICL (LLaMA)	<u>0.01</u>	<u>0.04</u>	<u>0.02</u>
Fine-tuned (LLaMA)	**0.0**	**0.0**	**0.0**

Table 3 reports the computational cost of the models. Due to their small size, the baselines offer minimal computational overhead and faster inference compared to the LLM-based approaches. ICL is slightly more expensive than zero-shot prompting, potentially due to longer prompts. Fine-tuning is resource-intensive but its computational cost during inference is comparable to that of zero-shot and ICL.

Table 3. Computational cost of approaches. The base model is LLaMA after applying 4-bit quantization. *Tuned* refers to fine-tuned version merged with LoRa weights. *GPU Memory* is reported as the peak usage, while *GPU APD* denotes the average power demand.

Strategy	**Model Size (GB)**	**Time Inference (Min)**	**GPU Memory (GB)**	**GPU APD (W)**
		Inference		
EmbSim	1.63	2.17	≈5.19	≈130
PairNSP	0.49	1.34	≈0.76	≈153
ItemCls	1.50	2.17	≈5.66	≈124
Zero-Shot	8.46	30.7	≈11.8	≈282
ICL	8.46	34.0	≈11.0	≈286
Fine-tuned	8.46	21.5	≈14.7	≈259
		Fine-Tuning		
LoRa Weights	4.54	154.5	≈14.7	≈316

The results in Figure 2 demonstrate that as the candidate set size increases, the performance declines across all LLM-based approaches, reflecting the increased difficulty of the task. However, fine-tuning consistently achieves the highest performance in all scenarios, but starts converging toward the lower performance levels of ICL and zero-shot as the candidate size grows.

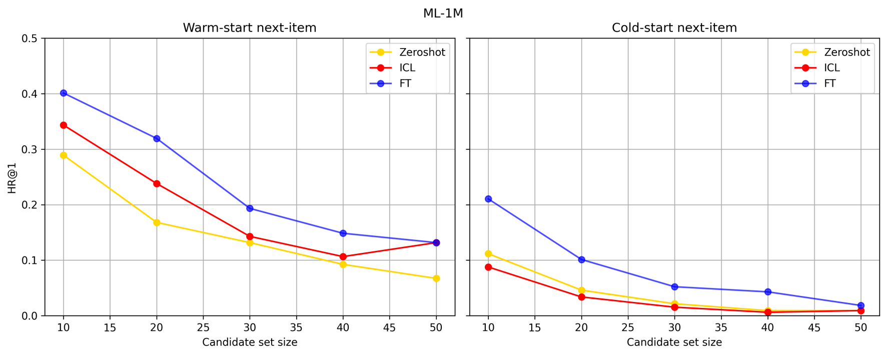

Fig. 2. Performance using different candidate set sizes.

Figure 3 shows the performance of the fined-tuned models under the k-shot training setting, aggregated over three independent runs. The results are benchmarked against zero-shot and ICL, represented by the dotted lines. The performance trend of k-shot training lags behind the simpler approaches, especially when k is small, but start to improve at higher k values for cold-start cases. The results suggest that, while the performance of the fine-tuned models improves as more data becomes available, simpler approaches remain more effective under limited data.

6. Discussion, Limitations and Future Work

The results show that the cold-start problem remains a persistent challenge—even for fine-tuned LLMs. This is likely because warm-start items can cluster with semantically or behaviorally similar items, while cold-start items lack signals the model can latch onto.

Consistent with prior work [1], fine-tuning LLMs improves performance for warm-start recommendations. Our findings show that this effect is also evident in cold-start scenarios.

By fine-tuning on both warm and cold-start next-item cases simultaneously, the model accommodates both scenarios and eliminates the need for separate models for different item types. Moreover, unlike traditional RS, which must be retrained from scratch to incorporate new data, this strategy offers a more scalable solution as it allows the model to generalize to completely

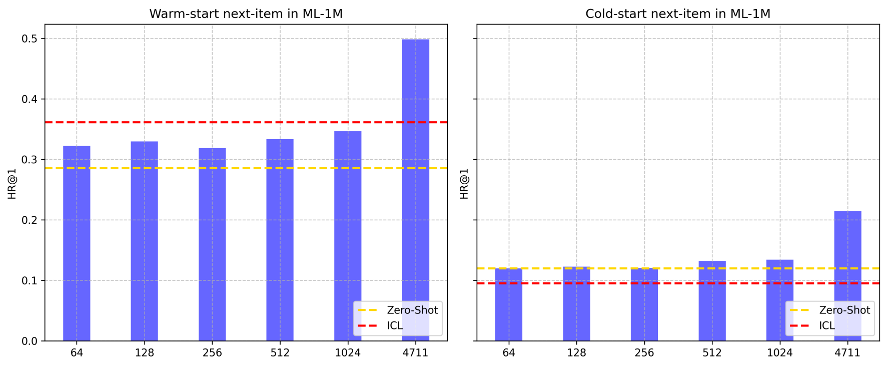

Fig. 3. Performance of fine-tuned models under different k-shot settings aggregated over 3 independent runs. The last bar in each diagram represents fine-tuning on the whole dataset.

new users and items that it had not encountered before. Notably, fine-tuning also improves task compliance, mitigating typical LLM issues such as hallucination [24].

Fine-tuning LLMs requires, however, considerable computational resources, whereas simpler PLM-based methods like PairNSP, built on a BERT backbone, offer strong results with low overhead and faster inference. We also observe that the performance trend of k-shot training lags behind the simpler approaches when k is still small, and that the relative advantage of fine-tuned models narrows as the candidate space grows.

The findings suggest that although fine-tuning delivers the best overall performance, it comes with substantial GPU and energy requirements. Therefore, it is important to weigh the trade-off between performance gains and operational cost. In scenarios like cold-starts, the accuracy improvements achieved through fine-tuning might justify the higher resource investment, as improved generalization can potentially translate into tangible business value. Still, in settings with limited budgets or only modest performance requirements, lightweight alternatives might offer a more efficient and cost-effective solution.

Regarding limitations, first, we use the MovieLens dataset, which features rich semantic content in item titles within a well-curated domain. More complex domains like e-commerce, especially under cold-start conditions might yield different results. The dataset's chronological structure also aligns well with the sequential modeling abilities of LLMs. When user behaviors are non-sequential, the advantages of sequence-aware models may diminish, or the prediction task might become less challenging as temporal ordering is no longer critical. Second, relying on a single model architecture limits the broader applicability of our findings. Thus, while our results demonstrate promising trends, exploring these dimensions across diverse datasets and model architectures is necessary to fully assess the generalizability of those approaches.

Future work could investigate decoupling item representation learning from text generation, as proposed in [18]. Replacing raw titles with semantically aligned ID embeddings could provide the model with more compact inputs, potentially improving scalability as the candidate set grows. Another promising direction is the "item tree" approach as suggested in [43], where the LLM traverses a hierarchical item catalogue to narrow down candidates during inference. Comparing such LLM-driven retrieval strategies with the retrieve-and-recommend paradigm is interesting for uncovering trade-offs in efficiency and effectiveness.

7. Conclusion

This paper investigates LLM-based approaches for next-item recommendation within a retrieve-and-recommend framework across zero-shot prompting, in-context learning, and fine-tuning, and provides insights into the practical trade-offs in terms of effectiveness and efficiency.

References

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

[2] Barjasteh, I., Forsati, R., Masrour, F., Esfahanian, A.H., Radha, H.: Cold-start item and user recommendation with decoupled completion and transduction. In: Proceedings of the 9th ACM Conference on Recommender Systems. pp. 91–98 (2015)

[3] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al.: Language models are few-shot learners. Advances in neural information processing systems 33, pp. 1877–1901 (2020)

[4] Chen, Y.F., Zhao, X., Liu, J.Y., Ge, B., Zhang, W.M.: Item cold-start recommendation with personalized feature selection. Journal of Computer Science and Technology 35, pp. 1217–1230 (2020)

[5] Chou, S.Y., Yang, Y.H., Jang, J.S.R., Lin, Y.C.: Addressing cold start for next-song recommendation. In: Proceedings of the 10th ACM Conference on Recommender Systems. pp. 115–118 (2016)

[6] Covington, P., Adams, J., Sargin, E.: Deep neural networks for youtube recommendations. In: Proceedings of the 10th ACM conference on recommender systems. pp. 191–198 (2016)

[7] Dai, S., Shao, N., Zhao, H., Yu, W., Si, Z., Xu, C., Sun, Z., Zhang, X., Xu, J.: Uncovering chatgpt's capabilities in recommender systems. In: Proceedings of the 17th ACM Conference on Recommender Systems. pp. 1126–1132 (2023)

[8] Dettmers, T., Pagnoni, A., Holtzman, A., Zettlemoyer, L.: Qlora: Efficient finetuning of quantized llms. Advances in Neural Information Processing Systems 36 (2024)

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

[10] Di Palma, D., Biancofiore, G.M., Anelli, V.W., Narducci, F., Di Noia, T., Di Sciascio, E.: Evaluating chatgpt as a recommender system: A rigorous approach. arXiv preprint arXiv:2309.03613 (2023)

[11] Fan, W., Zhao, Z., Li, J., Liu, Y., Mei, X., Wang, Y., Tang, J., Li, Q.: Recommender systems in the era of large language models (llms). arXiv e-prints pp. arXiv–2307 (2023)

[12] Gao, Y., Sheng, T., Xiang, Y., Xiong, Y., Wang, H., Zhang, J.: Chat-rec: Towards interactive and explainable llms-augmented recommender system. arXiv e-prints pp. arXiv–2303 (2023)

[13] Gomez-Uribe, C.A., Hunt, N.: The netflix recommender system: Algorithms, business value, and innovation. ACM Transactions on Management Information Systems (TMIS) 6(4), pp. 1–19 (2015)

[14] Harper, F.M., Konstan, J.A.: The movielens datasets: History and context. Acm transactions on interactive intelligent systems (tiis) 5(4), pp. 1–19 (2015)

[15] Holtzman, A., Buys, J., Du, L., Forbes, M., Choi, Y.: The curious case of neural text degeneration. arXiv preprint arXiv:1904.09751 (2019)

[16] Hou, Y., Zhang, J., Lin, Z., Lu, H., Xie, R., McAuley, J., Zhao, W.X.: Large language models are zero-shot rankers for recommender systems. In: European Conference on Information Retrieval. pp. 364–381 (2024)

[17] Hu, E.J., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W., et al.: Lora: Low-rank adaptation of large language models. In: International Conference on Learning Representations

[18] Hu, J., Xia, W., Zhang, X., Fu, C., Wu, W., Huan, Z., Li, A., Tang, Z., Zhou, J.: Enhancing sequential recommendation via llm-based semantic embedding learning. In: Companion Proceedings of the ACM Web Conference 2024. pp. 103–111 (2024)

[19] Ji, J., Li, Z., Xu, S., Hua, W., Ge, Y., Tan, J., Zhang, Y.: Genrec: Large language model for generative recommendation. In: European Conference on Information Retrieval. pp. 494–502. Springer (2024)

[20] Kang, W.C., McAuley, J.: Self-attentive sequential recommendation. In: 2018 IEEE international conference on data mining (ICDM). pp. 197–206. IEEE (2018)

[21] Kenton, J.D.M.W.C., Toutanova, L.K.: Bert: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of naacL-HLT. vol. 1, p. 2. Minneapolis, Minnesota (2019)

[22] Levy, M., Jacoby, A., Goldberg, Y.: Same task, more tokens: the impact of input length on the reasoning performance of large language models. arXiv preprint arXiv: 2402.14848 (2024)

[23] Li, J., Jing, M., Lu, K., Zhu, L., Yang, Y., Huang, Z.: From zero-shot learning to cold-start recommendation. In: Proceedings of the AAAI conference on artificial intelligence. vol. 33, pp. 4189–4196 (2019)

[24] Li, L., Zhang, Y., Liu, D., Chen, L.: Large language models for generative recommendation: A survey and visionary discussions. arXiv preprint arXiv:2309.01157 (2023)

[25] Loshchilov, I.: Decoupled weight decay regularization. arXiv preprint arXiv: 1711.05101 (2017)

[26] Muennighoff, N., Tazi, N., Magne, L., Reimers, N.: Mteb: Massive text embedding benchmark. arXiv preprint arXiv:2210.07316 (2022), `https://arxiv.org/abs/2210.07316`

[27] Rendle, S., Freudenthaler, C., Schmidt-Thieme, L.: Factorizing personalized markov chains for next-basket recommendation. In: Proceedings of the 19th international conference on World wide web. pp. 811–820 (2010)

[28] Ricci, F., Rokach, L., Shapira, B., Kantor, P.B.: Recommender Systems Handbook. Springer-Verlag, Berlin, Heidelberg, 3st edn. (2021)

[29] Sanner, S., Balog, K., Radlinski, F., Wedin, B., Dixon, L.: Large language models are competitive near cold-start recommenders for language-and item-based preferences. In: Proceedings of the 17th ACM conference on recommender systems. pp. 890–896 (2023)

[30] Semerci, O., Gruson, A., Edwards, C., Lacker, B., Gibson, C., Radosavljevic, V.: Homepage personalization at spotify. In: Proceedings of the 13th ACM conference on recommender systems. pp. 527–527 (2019)

[31] Sileo, D., Vossen, W., Raymaekers, R.: Zero-shot recommendation as language modeling. In: European Conference on Information Retrieval. pp. 223–230. Springer (2022)

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

[33] Vats, A., Jain, V., Raja, R., Chadha, A.: Exploring the impact of large language models on recommender systems: An extensive review. arXiv preprint arXiv: 2402.18590 (2024)

[34] Wang, L., Lim, E.P.: Zero-shot next-item recommendation using large pretrained language models. arXiv preprint arXiv:2304.03153 (2023)

[35] Wu, H., Wong, C.W., Zhang, J., Yan, Y., Yu, D., Long, J., Ng, M.K.: Cold-start next-item recommendation by user-item matching and auto-encoders. IEEE Transactions on Services Computing 16(4), pp. 2477–2489 (2023)

[36] Wu, L., Zheng, Z., Qiu, Z., Wang, H., Gu, H., Shen, T., Qin, C., Zhu, C., Zhu, H., Liu, Q., et al.: A survey on large language models for recommendation. World Wide Web 27(5), pp. 60 (2024)

[37] Wu, X., Zhou, H., Shi, Y., Yao, W., Huang, X., Liu, N.: Could small language models serve as recommenders? towards data-centric cold-start recommendation. In: Proceedings of the ACM on Web Conference 2024. pp. 3566–3575 (2024)

[38] Xie, X., Sun, F., Liu, Z., Wu, S., Gao, J., Zhang, J., Ding, B., Cui, B.: Contrastive learning for sequential recommendation. In: 2022 IEEE 38th international conference on data engineering (ICDE). pp. 1259–1273. IEEE (2022)

[39] Yang, F., Chen, Z., Jiang, Z., Cho, E., Huang, X., Lu, Y.: Palr: Personalization aware llms for recommendation. arXiv preprint arXiv:2305.07622 (2023)

[40] Yin, W., Hay, J., Roth, D.: Benchmarking zero-shot text classification: Datasets, evaluation and entailment approach. arXiv preprint arXiv:1909.00161 (2019)

[41] Yue, Z., Rabhi, S., Moreira, G.d.S.P., Wang, D., Oldridge, E.: Llamarec: Two-stage recommendation using large language models for ranking. arXiv preprint arXiv: 2311.02089 (2023)

[42] Zhang, J., Xie, R., Hou, Y., Zhao, W.X., Lin, L., Wen, J.R.: Recommendation as instruction following: A large language model empowered recommendation approach. arXiv preprint arXiv:2305.07001 (2023)

[43] Zhang, W., Wu, C., Li, X., Wang, Y., Dong, K., Wang, Y., Dai, X., Zhao, X., Guo, H., Tang, R.: Llmtreerec: Unleashing the power of large language models for cold-start recommendations. arXiv preprint arXiv:2404.00702 (2024)

[44] Zhang, Y., Ding, H., Shui, Z., Ma, Y., Zou, J., Deoras, A., Wang, H.: Language models as recommender systems: Evaluations and limitations (2021)

[45] Zhao, H., Yao, Q., Li, J., Song, Y., Lee, D.L.: Meta-graph based recommendation fusion over heterogeneous information networks. In: Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining. pp. 635–644 (2017)

[46] Zhou, K., Wang, H., Zhao, W.X., Zhu, Y., Wang, S., Zhang, F., Wang, Z., Wen, J.R.: S3-rec: Self-supervised learning for sequential recommendation with mutual information maximization. In: Proceedings of the 29th ACM international conference on information & knowledge management. pp. 1893–1902 (2020)