Information and Software Technology 192 (2026) 108021

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

Wise recommender: LLMs refined by iterative critics ☆

Zhisheng Yang [a], Xiaofei Xu [b], Ke Deng [c,*], Li Li [a,*]

[a] College of Computer and Information Science, Southwest University, Chongqing, 400715, China
[b] School of Information Technology, Murdoch University, Murdoch, WA 6150, Australia
[c] School of Computing Technologies, RMIT University, Melbourne, VIC 3000, Australia

ARTICLE INFO

Dataset link: https://github.com/yzsyzs478/Critic-LLM-RS

Keywords:
Large Language Model
Recommendation system
Interactive critic

ABSTRACT

Context: Large Language Models (LLMs) have been applied to recommendation tasks, giving rise to the new paradigm of LLM-as-Recommendation Systems (LLM-as-RS). Existing methods fall into two categories: tuning and non-tuning. While tuning strategies offer better task alignment, they are expensive and require specialized training. Non-tuning strategies are easier to deploy but often lack task-specific knowledge, limiting their effectiveness.

Objective: This study aims to enhance the recommendation quality of non-tuning LLM-based systems by addressing their lack of task awareness.

Method: We propose a novel approach, Critique-based LLMs as Recommendation Systems (Critic-LLM-RS), which introduces an independent machine learning model—the Recommendation Critic—to provide feedback on LLM-generated recommendations and guide the LLM toward improved recommendation strategies.

Results: Experiments on multiple real-world datasets demonstrate that Critic-LLM-RS significantly outperforms existing non-tuning approaches, regardless of whether open-source or proprietary LLMs are used.

Conclusion: Critic-LLM-RS enhances the task adaptability of non-tuning LLMs through a collaborative feedback mechanism, offering a new solution for building efficient and easily deployable recommendation systems.

1. Introduction

Large Language Models (LLMs) have been employed to craft recommendations that align with user preferences, a paradigm known as LLM-as-RS (see a survey [1]). *LLM-as-RS* can be implemented through either tuning or non-tuning strategies. The tuning strategy involves meticulously aligning LLMs with specific recommendation tasks via fine-tuning methods such as instruction tuning, content-based fine-tuning, or prompt adjustment. Examples of this approach include models like TALLRec [2], FLAN-T5 [3], fine-tuned LLMs for content recommendations [4], RecSysLLM [5], and POD [6,7].

In contrast, the non-tuning strategy leverages the inherent capabilities of pre-trained LLMs to generate recommendations directly through carefully designed prompts, bypassing the need for additional training or model adjustments [8–13]. This makes it ideal for zero-shot or few-shot learning scenarios. The non-tuning strategy avoids the time-intensive and expertise-demanding process of fine-tuning, but it often lacks task-specific knowledge, limiting its performance [14]. To improve performance, the existing non-tuning LLM-as-RS approaches focus on meticulous prompt design [8,15–17], breaking the task into steps [18], and retrieving information from a dataset [19].

To further enhance the non-tuning LLM-as-RS, we propose Critique-based LLMs as Recommendation Systems (Critic-LLM-RS) which introduces a new paradigm for recommendation systems by combining a separate machine-learning model called the *Recommendation Critic* with a large language model (LLM). The Recommendation Critic captures both collaborative and content-based filtering capabilities. It leverages user–item interactions to infer shared preferences (collaborative filtering) and uses textual item features to understand user interests (content-based filtering). Unlike previous methods that incorporate information before recommendation generation [8,15–19], Critic-LLM-RS applies it post-recommendation, providing targeted feedback to improve results iteratively.

The Recommendation Critic in Critic-LLM-RS can be viewed as a traditional recommender system. Why not use it to recommend items to users directly? Traditional systems depend entirely on their training data and can only recommend items within that dataset. They struggle with incomplete data coverage and rapidly evolving domains, where new items constantly emerge. In contrast, LLMs possess extensive factual knowledge learned from large text corpora, enabling Critic-LLM-RS to recommend items beyond the training dataset and across diverse

☆ This work was supported by the National Natural Science Foundation of China under grant number 61877051.
* Corresponding authors.
E-mail addresses: yzsheng@email.swu.edu.cn (Z. Yang), xiaofei.xu@murdoch.edu.au (X. Xu), ke.deng@rmit.edu.au (K. Deng), lily@swu.edu.cn (L. Li).

https://doi.org/10.1016/j.infsof.2026.108021
Received 26 July 2025; Received in revised form 9 December 2025; Accepted 5 January 2026
Available online 9 January 2026

domains. Critic-LLM-RS does not fine-tune LLMs. Why not directly fine-tune LLMs for recommendation? Direct fine-tuning of LLMs, though effective, is computationally expensive and impractical for proprietary (black-box) models such as GPT.

In the LLM-as-RS literature, an approach is typically classified as non-tuning when the parameters of LLMs remain frozen, and as tuning when the parameters of the LLM are updated [1,14]. Therefore, our Critic-LLM-RS is categorized as non-tuning since the parameters of the LLM are kept fixed throughout training. Although the proposed Recommendation Critic contains learnable parameters, it operates externally to the LLMs and thus does not alter the pretrained model. Moreover, the parameter scale of the Recommendation Critic (in the order of thousands) is negligible compared to that of the LLM (in the order of billions). Consequently, optimizing the Recommendation Critic is computationally tractable and far more efficient than fine-tuning the LLM itself.

The key contributions of this study are as follows:

- We propose Critic-LLM-RS, a new non-tuning LLM-as-RS approach, which employs a separate model to criticize the LLM's initial recommendations, prompting refinements for improved results.
- We justify the advantage of Critic-LLM-RS in the integration of recommendation training data with LLMs and disclose its capability of comprehensive collaborative information exploration.
- We validate the effectiveness of the proposed Critic-LLM-RS approach through extensive experiments and case studies on real-world datasets with open-source and proprietary LLMs, demonstrating its superiority over existing non-tuning LLM-as-RS approaches.

2. Related work

LLMs for recommendations

Large Language Models (LLMs) are causing a remarkable impact in the research field of recommendations. One line of research is Discriminative LLMs for Recommendation (DLLM4Rec) [1]. Approaches in this line typically leverage BERT and its variants for extracting and embedding high-quality textual features and comprehensive knowledge, which are then integrated with existing recommender systems to enhance recommendations [20–31].

Another line of research is Generative LLMs for Recommendation (GLLM4Rec) [1]. These approaches utilize the language generation capabilities of LLMs to generate recommendations directly. GLLM4Rec operates in two main modes: *LLM-Tokens+RS* and *LLM-as-RS* [32,33]. The LLM-Tokens+RS approach leverages semantic mining techniques to uncover potential user preferences by synthesizing tokens generated from item and user characteristics and then integrating these insights into an existing recommender system for the enhanced recommendations [4,22,34–39]. Beyond text description, the user–item graph has been explored to grasp structured knowledge of user–item interactions [40]. Specifically, the LLM learns to rewrite the user's (or item's) description considering the descriptions of other items (or users) that the user (or items) has engaged with, i.e., collaborative information. By taking hops on the graph, the final descriptions of users and items are rewritten iteratively. The LLM-as-RS approach directly leverages LLMs for crafting recommendations to achieve precise alignment with user preferences and elevate the quality of the recommendations [1]. The approaches of LLM-as-RS implement both non-tuning and tuning strategies.

Following the tuning strategies, models meticulously calibrate the alignment of LLMs with specific recommendation tasks through targeted training or fine-tuning via methods like instruction tuning, content-based fine-tuning, or prompt adjustment. This strategy is exemplified by models such as TALLRec [2], FLAN-T5 [3], LLMs fine-tuned expressly for content recommendations [4], RecSysLLM [5], and POD [6,7]. However, further training pre-trained LLMs on task-specific datasets is a costly, time-consuming, and expertise-requiring process.

Following the non-tuning strategy, models rely on the inherent, pre-trained prowess of LLMs to directly craft recommendations via meticulously designed prompts. The representative approaches include Llama4Rec [8], InteraRec [9], GENREC [10], FaiRLLM [11], and GPTRec [12,13]. This strategy bypasses LLM adjustments or training, rendering it perfectly suited for scenarios like zero-shot or few-shot learning [15–17], but it often lacks task-specific knowledge, limiting its performance [14]. To improve performance, the existing non-tuning LLM-as-RS approaches focus on meticulous prompt design [8,15–17]. Wang et al. [18] breaks complex recommendation tasks into manageable steps, each step involves thought, action, and observation, and the agent considers all previously explored paths for the next planning. Following the idea of retrieval-augmented generation (RAG) [41], Wang et al. [19] improves sequential recommendation by proposing a retriever to extract purchase behaviors from similar users in a dataset. The retrieved information is provided to LLMs as collaborative information to help identify the critical aspects users might consider when engaging with a specific item. Then, LLMs are prompted to recommend the next item and reflect on the recommendation to improve the critical aspects identified.

Unlike existing studies of non-tuning LLM-as-RS, we propose to build an external feedback model that explores collaborative and content-based filtering as in the traditional recommender systems to criticize the initial recommendations generated by LLMs, and then LLMs are prompted to refine recommendations. If needed, this criticize-and-enhance process can be repeated to ensure output enhancement.

LLMs with feedback

Recently, some researchers focused on providing textual feedback on the generated response and letting LLM refine the response [42–46]. Such approaches allow for the refinement of model response, improving the quality of response and reducing the likelihood of hallucinations. Early research on critic-style feedback directly asks humans for feedback. Since humans provide feedback, it would be expensive to collect it, so the generalization ability is limited. Another way is to utilize a feedback model to generate those critiques. Schick et al. [42] and Akyürek et al. [43] collect gold standard critiques from online forums or Wikipedia edit histories as training data and then use supervised fine-tuning or reinforcement learning to obtain a critique model, typically a small LLM. However, when applied to a specific domain, such datasets may not exist or be costly to obtain. In [47], an automated reasoning engine is used as a critic to enhance the logical reasoning capabilities of LLMs. In [48], LLMs are prompted to interact with external tools like search engines and code interpreters to verify the desired aspects of an initial output and subsequently amend the output based on the critiques from the verification.

This study proposes, to the best of our knowledge, the first feedback model for LLM-as-RS.

3. Methodology

The proposed *critique-based LLMs as a Recommendation System* (Critic-LLM-RS) is presented in Fig. 1. Recommendation Critic training is shown in the lower block (green background), and the recommendation with Critic-LLM-RS is presented in the upper block (blue background). For a specific recommendation task, like movie recommendation or book recommendation, one Recommendation Critic is trained separately using user–item interaction data. Note that the lock icon indicates frozen parameters, whereas the flame icon indicates parameters learnable during training.

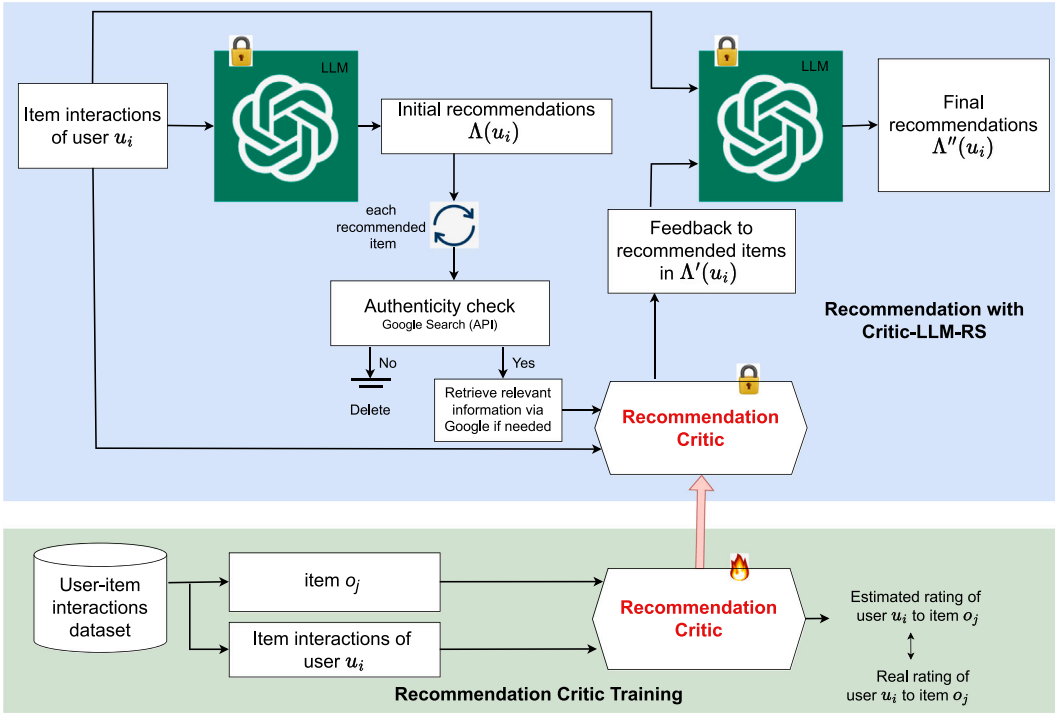


Fig. 1. Critic-LLM-RS overview: Recommendation with Critic-LLM-RS and Recommendation Critic training.

3.1. Recommendation with Critic-LLM-RS

We assume that Recommendation Critic has been trained. As illustrated in the upper block of Fig. 1, the LLM is used twice, where the left one refers to the first use and the right one is for the second use. The input to the LLM (left) consists of the interaction history of user u_i with items (such as a list of movies that the user has watched and rated previously) and a prompt indicating that the task is to recommend items that u_i likes. Each interaction entails the item information (e.g., movie name and other information like director, actors, and synopsis) and the rating given by the user to the item (e.g., 1–5). The LLM (left) outputs the initial recommendations to user u_i, denoted as $\Lambda(u_i)$. When LLM is prompted to recommend items preferred by the user based on its analysis, we also ask the LLM to provide a rating (e.g., 1–5) for each recommended item. It is essential for the proper functioning of Critic-LLM-RS, because it enables the LLM to effectively interpret and incorporate the feedback provided by the Recommendation Critic to each of the recommended items in the form of ratings.

Before using Recommendation Critic, the initial recommendations of LLM are verified to prevent LLM hallucination, i.e., LLM generates false, nonsensical, or fabricated information that is presented as factual. As shown in the upper block of Fig. 1, an authenticity check is conducted for each recommended item in $\Lambda(u_i)$, to determine whether it actually exists or is hallucinated by the LLM. Specifically, external sources like IMDb, Google, and others can be explored to verify the authenticity. If an item is determined to be hallucinated by the LLM, it is removed; otherwise, the item is confirmed to exist, and the relevant information is retrieved (e.g., the director and actors for movies; the URL, authors, language, year published, and description for books). We denote the recommended items after authenticity verification as $\Lambda'(u_i)$.

Then, Recommendation Critic evaluates each item $o \in \Lambda'(u_i)$. Beside o, the other input of Recommendation Critic is the interaction history of user u_i. A conventional *Critic* generally refers to a module that evaluates the generated outputs and provides feedback for improvement [43,46]. For our task, the Recommendation Critic provides recommendation-specific feedback by learning collaborative and content-based filtering capability from user–item interactions, addressing the lack of precise evaluation mechanisms in LLM-based recommendation [15,18]. Recommendation Critic estimates and outputs the rating that user u_i gives o. The rating is injected into the next-round prompt to refine the recommendations generated by the LLM [48].

This feedback injects supplementary evidence that helps the LLM refine its next-round generation, promoting high-scoring items and suppressing low-scoring ones in subsequent generations, thereby improving the accuracy and utility of the final recommendation list. The *final recommendation list* is denoted as $\Lambda''(u_i)$. If necessary, we can invoke the Recommendation Critic again to evaluate the recommended items in $\Lambda''(u_i)$ and prompt LLM to further refine the recommendations.

3.2. Recommendation Critic training

To evaluate each item in the initial recommendations by LLM, the Recommendation Critic, a multi-layer neural network with fully connected layers and ReLU activation, is trained to predict the user's rating of the item based on the user's interaction history. Training data incorporates the interactions between users and items, where each user–item interaction entails the item information and the rating given by the user to the item (e.g., 1–5). The item information differs for different recommendation tasks. For example, it can be *title*, *directedBy*, *starring*, *rating* for movie recommendation; *title*, *authors*, *lang*, *year*, *url*, *description*, and *rating* for book recommendation. As shown in the lower block of Fig. 1, given the item interaction history of user u_i and the information of item o_j (o_j is not in the interaction history of u_i but rated by user u_i), Recommendation Critic learns to estimate the rating of u_i to o_j. The training loss is the difference between the estimated rating and the real rating.

Specifically, the textual information of o_j and the textual information in the interaction history of u_i are encoded using pre-trained encoders like BERT.[1] Each item in the user's interaction history is encoded as an embedding of size d. If the user's interaction history contains n items, the resulting embedding is of dimension $n \times d$. The

[1] https://huggingface.co/google-bert/bert-base-uncased.

item o_j is also encoded as an embedding of size d. During training, the embeddings of all items in the interaction history of user u_i and the embedding of item o_j are concatenated as the input embedding. However, since different users may have varying numbers of interaction history items, this results in different input embedding dimensions, which could affect the stability and effectiveness of Recommendation Critic training.

To address this issue, we standardize the number of items in each user's interaction history to n_{max}. Suppose a user has fewer than n_{max} items in his/her interaction history. In that case, we insert dummy items (i.e., zero vectors) to the end of his/her interaction history until it has n_{max} items. The dummy items do not represent ratings and thus are masked out during model training (i.e., they do not contribute to the gradient). If a user has more than n_{max} items, we truncate the excess items by retaining n_{max} items selected randomly. This way, we standardize the input embedding dimension for each user to $(n_{max}+1)\times d$, which consists of embeddings for n_{max} items in interaction history and one embedding for the item to be predicted.

Recommendation Critic can be implemented by adapting existing recommendation algorithms to predict the rating that a user gives to an item, based on the interaction history of the user and the information of the item. After training, when the Recommendation Critic is used in the recommendation with Critic-LLM-RS as shown in Fig. 1, the number of items in the interaction history of user u_i needs to be standardized to n_{max} in the same way as in the training phase. Moreover, in situations where an item is not in the training data but returned by LLM as an initial recommendation, the trained Recommendation Critic can still work if the relevant information of the item can be retrieved via Google.

3.3. Why Critic-LLM-RS

Given the training for a recommendation task, one Recommendation Critic is trained. One may ask two questions: (1) Why not train a recommender system using one of the traditional recommendation technologies with the training data? (2) Why not directly fine-tune the LLM?

Regarding the first question, traditional recommender systems depend on the training data and only recommend the items present in the training data. In practice, it is hard to maintain a dataset containing all items for a particular recommendation task. Due to limitations in data coverage, such systems struggle to adapt to rapidly evolving domains such as news and e-commerce. Moreover, as they primarily rely on the user–item interactions, they lack the ability to effectively integrate external knowledge, making it difficult to recommend newly released items such as books or movies. In contrast, our Critic-LLM-RS generates recommendations using an LLM that stores an extensive amount of factual knowledge obtained from vast collections of documents. That is, Critic-LLM-RS knows large and very recent datasets in almost every domain and can recommend items not present in most training datasets due to the LLM.

Regarding the second question, LLM fine-tuning is a supervised learning process where one uses a dataset of human-curated examples to update the parameters of LLM and improve its ability for specific tasks. It is a powerful technique but has notable drawbacks [1]. First, LLMs have millions or even billions of parameters. Training these massive models requires serious computational power. The costs can quickly become prohibitive for smaller teams or those on a budget. The parameter-efficient fine-tuning technique LoRA [49] drastically reduces the number of parameters that need to be trained by using low-rank matrices to update specific parameters. However, LoRA typically works with open-source LLMs, but when using proprietary LLMs (e.g., GPT models from OpenAI), LoRA is impractical, because such LLMs are machine learning black boxes.

In contrast, we train one Recommendation Critic separately using the user–item interactions in the training data for a particular recommendation task. It provides feedback on the recommendations of either open-source or proprietary LLMs.

3.4. Why Critic-LLM-RS works

By training on the user–item interactions, our Recommendation Critic has the capability of collaborative and content-based filtering. As one of the most successful technologies for building recommender systems, collaborative filtering uses the known preferences of a group of users to make recommendations of the unknown preferences for other users [50]. If two users have a similar interaction history, Recommendation Critic is trained to predict that they will give similar ratings to the same items. Moreover, Recommendation Critic is capable of content-based filtering when the items are not simply represented by item identities but by item features (i.e., textual information). Content-based filtering uses item features to recommend other items similar to what the user likes, based on their previous actions or explicit feedback [51]. By processing the textual information of items, the Recommendation Critic can better understand the item o_j and user u_i preferences from his/her interaction history with items. As a result, Recommendation Critic can more accurately estimate the rating that u_i gives to o_j.

As discussed in Section 2, collaborative and content-based filtering capability has been explored in LLM-as-RS studies. In [40], the capability is captured by taking multiple hops over a user–item graph to help enhance descriptions of users/items before using a traditional recommender system. However, it is constrained by the number of hops in the user–item graph. In [19], the capability is acquired by using in-context examples in the prompt of LLM, like the purchase history of the relevant users, to represent the user preference to support recommendations. However, the number of in-context examples is limited by the upper bound of tokens when using LLM. In contrast, our Critic-LLM-RS acquires collaborative and content-based filtering capability using a machine learning model trained on complete training data. Also, unlike these studies, where the information obtained via collaborative and content-based filtering capability is applied pre-recommendation, our Critic-LLM-RS utilizes such information post-recommendation to provide targeted feedback for each recommended item. This criticize-and-enhance process can be repeated to ensure output enhancement if needed. In short, Critic-LLM-RS can better harness and exploit comprehensive information from collaborative and content-based filtering capabilities for recommendations.

4. Case studies

Tables 1 and 2 demonstrate how to use Critic-LLM-RS to recommend items to two users, one from Movie and the other from Book, respectively. The prompt template for both the initial recommendations and the final recommendations follows prior LLM-as-RS research (e.g., LLaMA4Rec [8], GENREC [10], GPTRec [13]). We represent user interaction histories in natural language and explicitly instruct the LLM to recommend items that match user preferences. Besides that, the prompt template for the final recommendations also has a component considering the feedback from Recommendation Critic. This part of the prompt is inspired by recent studies that refine LLM outputs through critic feedback [42–46]. We transform the rating to each item in the initial recommendations, predicted by Recommendation Critic, into concise textual feedback (e.g., "Item A is highly relevant with a predicted rating of 4.5, while Item B is less preferred with a rating of 2.0"). Then, the prompt template for the final recommendations guides the LLM to promote high-scoring items and suppress low-scoring ones.

In Tables 1 and 2, the recommendations in black font are the movies/books not having real ratings from the user, and the recommendations in blue font are the movies/books having real ratings from the user. For the user in Table 1, no matter whether a movie in the initial recommendations has a real rating or not, the Recommendation Critic can be used to estimate the rating that the user is likely to give. Compared with the initial recommendations, we can observe that the real ratings of movies in the final recommendations increase,

Table 1
A case study – Movie dataset – real (real rating), Critic (rating estimated by the Recommendation Critic).

Prompt template for initial recommendations:

"role": "system", "content""": "You are a movie recommendation system. Given a set of movies (each including the title, directedBy, starring) that a user watched, the user preference is based on the ratings that user give to these movies. Now, please recommend movies based on the user's preferences, and the recommended years for the movie are 1880 to 2021. The format for the recommended movies should be title (year)."""

"role": "user", "content": """Here are the movies that user watched and rated:

< replaced by a list of movies, each with title, directedBy, starring, rating >

Please recommend 10 movies and rank them according to how much the user might like them? Please base the ranking on a rating scale from 1 to 5, where 5 means I like them the most and 1 means I don't like them at all."""

Prompt template for final recommendations:

"role": "system", "content": """You are a movie recommendation system. Given a set of movies (each including the title, directedBy, starring) that a user watched, the user preference is based on the ratings that user give to these movies. Now, please recommend movies based on the user's preferences, and the recommended years for the movie are 1880 to 2021. The format for the recommended movies should be title (year)."""

"role": "user", "content": """Here are the movies that user watched:

< replace by a list of movies, each with title, directedBy, starring, rating >

Please recommend 10 movies and rank them according to how much the user might like them? Please base the ranking on a rating scale from 1 to 5, where 5 means I like them the most and 1 means I don't like them at all."""

"role": "assistant", "content": "< replaced by the movies in the initial recommendations >"

"role": "user", "content": """I have provided ratings for the movies you recommended. Based on the ratings I've provided, please adjust your recommendations to ensure all the movies are ones I tend to enjoy.

The provided rating:

< replaced by the movies in the initial recommendations, each with the rating estimated by the Critic model >"""

Movies watched and rated by the user: [{"**title**": "Airheads (1994)", "**directedBy**": "Michael Lehmann", "**starring**": "Steve Buscemi, Chris Farley, Brendan Fraser, Adam Sandler, Joe Mantegna, David Arquette, Amy Locane, Ernie Hudson, Judd Nelson, Michael McKean, Michael Richards", "**rating**": "4.0"}, {"**title**": "Red Dragon (2002)", "**directedBy**": "Brett Ratner", "**starring**": "Ralph Fiennes, Anthony Hopkins, Harvey Keitel, Edward Norton, Philip Seymour Hoffman, Mary-Louise Parker, Ken Leung", "**rating**": "3.5"}, ...]

Initial recommendations	Real	Critic	Final recommendations	Real	Critic
1 Jurassic Park (1993)	4.0	4.0	1 Pulp Fiction (1994)	4.0	4.0
2 Gladiator (2000)	3.0	2.5	2 Predator (1987)	–	3.0
3 The Dark Knight (2008)	–	3.0	3 Terminator 2: Judgment Day (1991)	4.5	5.0
4 Fast and Furious 7 (2015)	–	4.5	4 Dr. Strangelove or: How I Learned to Stop (1964)	5.0	5.0
5 Die Hard (1988)	5.0	5.0	5 The Night of the Living Dead (1968)	–	3.5
6 The Lord of the Rings: The Return of the King (2003)	–	3.5	6 Apocalypse Now (1979)	5.0	5.0
7 Blade (1998)	2.0	2.0	7 The Terminator (1984): The Winter Soldier (2014)	–	4.0
8 Pulp Fiction (1994)	4.0	4.0	8 Spider-Man: Into the Spider-Verse (2018)	–	3.0
9 The Shawshank Redemption (1994)	–	3.5	9 Gravity (2013)	–	3.5
10 John Wick (2014)	–	3.5	10 Twin Peaks: Fire Walker Stunt Double (2015)	–	3.0

and the movies with the higher ratings are closer to the top of the recommendations. A similar trend can be observed if looking at the ratings estimated using the Recommendation Critic. In Table 2, we have a similar observation that the final recommendations of books are better than the initial recommendation of books to the user according to real ratings or the ratings estimated using the Recommendation Critic.

The Movie dataset and Book database are up to 2020 and 2017, respectively. The traditional recommender system trained on the Movie dataset cannot recommend movies after 2020 and the traditional recommender system trained on the book dataset cannot recommend books after 2017. In contrast, the Critic-LLM-RS can do it by prompting the LLM to recommend movies after 2020 and books after 2017 as illustrated in Table 3. In this situation, we can still use the Recommendation Critic to give LLM feedback on the initial recommendations. Based on the ratings estimated using Recommendation Critic, the final recommendations are better than the initial recommendations.

5. Experiment

We focus on comparing our Critic-LLM-RS with the state-of-the-art of the non-tuning LLM-as-RS. Also, we compare Critic-LLM-RS with the tuning-based LLM-as-RS on open-source LLMs and evaluate the effectiveness of Critic-LLM-RS on a proprietary LLM. Experiments are conducted on a cluster where each node has 8 cores, 64 GB of RAM, and an NVIDIA GeForce RTX 3090. The code and datasets used in experiments will be available upon request.

5.1. Datasets

The evaluations are on two datasets, *Movie* and *Book*. The Movie dataset, extracted from *MovieLens (2021)* [52], comprises 247,383 users and 84,661 movies (up to 2020) with 28,490,116 user–movie interactions. A user–movie interaction includes the movie title, director, main actors, and user rating. Movie ratings range from 0 to 5 with a step of 0.5. The Book dataset, extracted from *Book-Crossing (2022)* [53], comprises 350,332 users and 9374 books (up to 2017) with 5,152,656 user–book interactions. A user–book interaction includes the book title, URL, authors, language, year published, book description, and user rating. Book ratings range from 1 to 5 with a step of 1.

For each dataset, a Recommendation Critic is trained on randomly selected 30k users and the user–item interactions of these users; each of these users has at least 10 user–item interactions. 80% of the training data is used for training, 10% for validation, and 10% for testing. For evaluating Critic-LLM-RS, we randomly sample another 1k users (not in the 30k users) from each dataset; each of these users has at least 30 user–item interactions. When using Critic-LLM-RS to do recommendations for each user, we randomly select 1/3 of user–item interactions as the interaction history of the user and the remaining 2/3 for evaluation. By default, we set $n_{max} = 20$ to standardize the number of items in the interaction history of each user when training Recommendation Critic and using Recommendation Critic in Critic-LLM-RS (see more details in Section 3.2).

5.2. Evaluation metrics

On both the Movie and Book datasets, the performance of Critic-LLM-RS measures to which extent the items in the final recommendations are preferred by users. A user prefers an item if the user rates the item ≥ 4. The evaluation metrics used are Precision, HR (Hit Rate), and NDCG (Normalized Discounted Cumulative Gain).

Table 2
A case study – Books dataset – real (real rating), Critic (rating estimated by the Critic).

Prompt template for initial recommendations:

"role": "system", "content""": "You are a book recommendation system. Given a set of books (each including the title, authors, lang, year, url, description) that a user watched, the user preference is based on the ratings that user give to these books. Now, please recommend books based on the user's preferences. The format for the recommended books should be title."""

"role": "user", "content": """Here are the books that user watched and rated:

< replaced by a list of books, each with title, authors, lang, year, url, description, rating >

Please recommend 10 books and rank them according to how much the user might like them? Please base the ranking on a rating scale from 1 to 5, where 5 means I like them the most and 1 means I don't like them at all."""

Prompt template for final recommendations:

"role": "system", "content""": "You are a book recommendation system. Given a set of books (each including the title, authors, lang, year, url ,and description) that a user watched, the user preference is based on the ratings that user give to these books. Now, please recommend books based on the user's preferences. The format for the recommended books should be title."""

"role": "user", "content": """Here are the books that user watched:

< replaced by a list of books, each with title, authors, lang, year, url, description, rating >

Please recommend 10 books and rank them according to how much the user might like them? Please base the ranking on a rating scale from 1 to 5, where 5 means I like them the most and 1 means I don't like them at all."""

"role": "assistant", "content": "< replaced by the books in the initial recommendations >"

"role": "user", "content": """I have provided ratings for the books you recommended. Based on the ratings I've provided, please adjust your recommendations to ensure all the books are ones I tend to enjoy.

The provided rating:

< replaced by the books in the initial recommendations, each with the rating estimated by the Critic model >"""

Books read and rated by the user: [{"**title**": "Every You, Every Me", "**url**": "https://www.goodreads.com/book/show/9972838-every-you-every-me", "**authors**": "David Levithan", "**lang**": "**eng**", "**year**": 2011, "**rating**": 4, "**description**": "In this high school-set psychological tale, a tormented ..." }, {"**title**": "When the Wind Blows (When the Wind Blows, #1)", "**url**": "https://www.goodreads.com/book/show/13162.When_the_Wind_Blows", "**authors**": "James Patterson", "**lang**": "eng", "**year**": 2000, "**rating**": 3, "**description**": "Frannie O'Neill is a caring young veterinarian living in the Colorado Rockies, ..."}, ...]

Initial recommendations	Real	Critic	Final recommendations	Real	Critic
1 The Picture of Dorian Gray	–	4.5	1 The Da Vinci Code (Robert Langdon, #2)	5.0	5.0
2 The Great Gatsby	3.0	3.0	2 The Martian	4.0	4.5
3 To Kill a Mockingbird	4.0	4.5	3 The Girl with the Dragon Tattoo	–	4.5
4 One Hundred Years of Solitude	–	3.5	4 Ready Player One	–	4.0
5 The Catcher in the Rye	–	5.0	5 The Handmaid's Tale	4.0	4.0
6 The Master and Margarita	–	2.0	6 Pride and Prejudice	–	3.5
7 Pride and Prejudice	–	4.0	7 The Lion, the Witch and the Wardrobe	–	3.0
8 Slaughterhouse-Five	–	3.5	8 The Time Machine	–	3.5
9 The Count of Monte Cristo	–	2.5	9 11/22/63	–	3.5
10 The Stranger	–	3.0	10 The Hunger Games	–	2.5

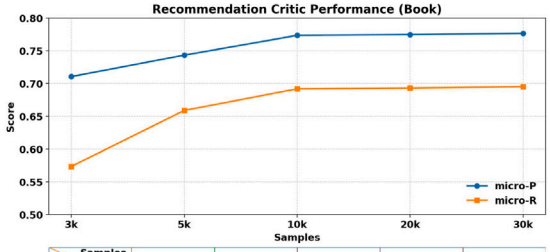
Table 3
A case study — real (real rating), Critic (rating estimated by the Critic).

Initial recommendations — Movies after 2020	Real	Critic	Final recommendations — Movies after 2020	Real	Critic
1 Dune (2021)	–	4.0	1 Last Night in Soho (2021)	–	4.5
2 Spider-Man: No Way Home (2021)	–	3.5	2 The Power of the Dog (2021)	–	5.0
3 No Time to Die (2021)	–	4	3 Lucky Grandma (2021)	–	4.0
4 The Matrix Resurrections (2021)	–	5	4 The Northman (2022)	–	3.5
5 Shang-Chi and the Legend (2021)	–	3.5	5 The Adam Project (2022)	–	4.5
6 Eternals (2021)	–	5			
7 Jungle Cruise (2021)	–	3.5	*—LLM recommends 5 movies only—*		
8 Free Guy (2021)	–	3.0			
9 Don't Look Up (2021)	–	3.0			
10 Promising Young Woman (2020)	–	4.0			
Initial recommendations — Books after 2017	**Real**	**Critic**	**Final recommendations — Books after 2017**	**Real**	**Critic**
1 Girl, Woman, Other (2019)	–	4.5	1 The Poppy War (2018)	–	5.0
2 The Memory Police (2019)	-	5.0	2 The City of Brass (2017)	–	4.0
3 Black Leopard, Red Wolf (2019)	–	3.5	3 Kingdom of the Winds (2020)	–	4.5
4 The Mirror & the Light (2020)	–	3.5	4 The Fifth Season (2015)	–	4.0
5 Where the Crawdads Sing (2018)	–	5.0	5 Project Hail Mary (2021)	–	3.0
6 The Water Dancer (2019)	–	4.0	6 The Gilded Wolves (2019)	–	4.0
7 The Sparrow (1996)	–	4.0	7 Red, White, and Royal Blue (2019)	–	4.5
8 The Invisible Life of Addie LaRue (2020)	–	3.5	8 The Bear and the Nightingale (2017)	–	3.5
9 The Duchess Kingmaker (2021)	–	2.5			
10 Lost in the Lost City (2021)	–	3.0	*—LLM recommends 8 books only—*		

- Precision measures the percentage of items in the top-N recommendations that are preferred by each user.

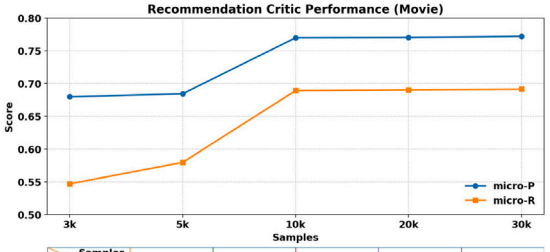
$$\text{Precision} = \frac{\text{Number of items preferred}}{N} \quad (1)$$

The higher Precision value indicates a better performance. Given N, the box plot of Precision values for all users in the test dataset is reported.



Metric \ Samples	3k	5k	10k	20k	30k
micro-R	0.5734	0.6589	0.6919	0.6931	0.6952
micro-P	0.7106	0.7432	0.7735	0.7746	0.7765

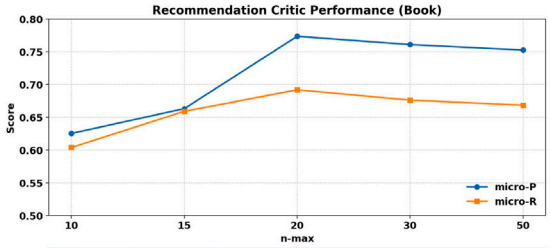
a: On the Book dataset.



Metric \ Samples	3k	5k	10k	20k	30k
micro-R	0.5469	0.5796	0.6892	0.6903	0.6913
micro-P	0.6797	0.6844	0.7698	0.7702	0.7720

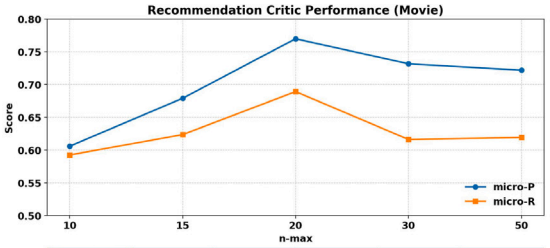
b: On the Movie dataset.

Fig. 2. Predictive accuracy of Recommendation Critic, which is trained on training data of different sizes.



Metric \ n-max	10	15	20	30	50
micro-R	0.6038	0.6590	0.6919	0.6762	0.6685
micro-P	0.6253	0.6631	0.7735	0.7610	0.7525

a: On the Book dataset.



Metric \ n-max	10	15	20	30	50
micro-R	0.5925	0.6235	0.6892	0.6161	0.6192
micro-P	0.6058	0.6790	0.7698	0.7316	0.7218

b: On the Movie dataset.

Fig. 3. Predictive accuracy of Recommendation Critic with different settings of n_{max}.

- HR indicates the number of hits and non-hits in the test dataset. Given a user, it is a *hit* if the top-N recommendations contain one or more items preferred, *non-hit* otherwise. HR = 1 means hits, and HR = 0 means non-hits. More HR = 1 indicates a better performance.
- NDCG evaluates the top-N recommendations considering both rating and ranking:

$$\text{NDCG} = \frac{\text{DCG}}{\text{IDCG}} \tag{2}$$

$$\text{DCG} = \sum_{i=1}^{N} \frac{2^{rel_i} - 1}{\log_2(i+1)} \tag{3}$$

where rel_i is the relevance score of the recommended item in the ith position. $rel_i = 1$ if the ith item is preferred by the user, 0 otherwise. IDCG is DCG when the top-N recommended items are sorted in descending order of relevance score. The higher NDCG indicates a better performance. The box plot of NDCG values for all users in the test dataset is reported.

During evaluation, the items recommended to a user by Critic-LLM-RS may have no real ratings because (i) the items are not in the dataset but recommended, or (ii) the items are in the dataset but the users never rate them. Since HR, NDCG, and Precision require real ratings for the top-N recommendations, we adopt two approaches to address this issue. First, we construct a candidate set of items with real user ratings and prompt the LLM to recommend items from this set only to validate the effectiveness of Critic-LLM-RS. Second, the LLM recommends items directly, without being restricted to a predefined candidate set of items. This approach may result in recommended items without real ratings. In this situation, Recommendation Critic is applied to estimate the ratings by the user. Substituting real ratings with the estimated ones still effectively measures the recommender system's performance. This is attributed to the predictive accuracy of the Recommendation Critic. The predictive accuracy of the Recommendation Critic is evaluated using micro-average precision (micro-P) and micro-average recall (micro-R), defined as:

$$\begin{aligned} \text{micro-P} &= \frac{\sum_{i=1}^{L} TP_i}{\sum_{i=1}^{L}(TP_i + FP_i)} \\ \text{micro-R} &= \frac{\sum_{i=1}^{L} TP_i}{\sum_{i=1}^{L}(TP_i + FN_i)} \end{aligned} \tag{4}$$

where L is the number of classes (i.e., rating levels, 10 for Movie and 5 for Book), TP_i, FP_i, and FN_i are the true positive, false positive, and false negative for class i, respectively.

Figs. 2(a) and 2(b) illustrate the predictive accuracy of Recommendation Critic in relation with size of training data. When the training data increases from $3k$ to $10k$, the predictive accuracy increases significantly. But after $10k$, the performance increase remarkably slows down or even stops. The convergence signals that $10k$ is sufficient for training the Recommendation Critic. The Recommendation Critic is trained using $30k$ users. The results are on the test dataset for evaluating Recommendation Critic, which is different from the test dataset for Critic-LLM-RS.

To ensure the input embedding of Recommendation Critic with the equal dimension, we standardize the number of items in the interaction history of users to n_{max} (see details in Section 3.2). By default, $n_{max} = 20$. We empirically analyze the impact of different n_{max} settings on the predictive accuracy of Recommendation Critic. As shown in Figs. 3(a) and 3(b), the performance peaks when $n_{max} = 20$. The smaller n_{max} (i.e., $n_{max} = 10$) means that the items in the interaction history of users

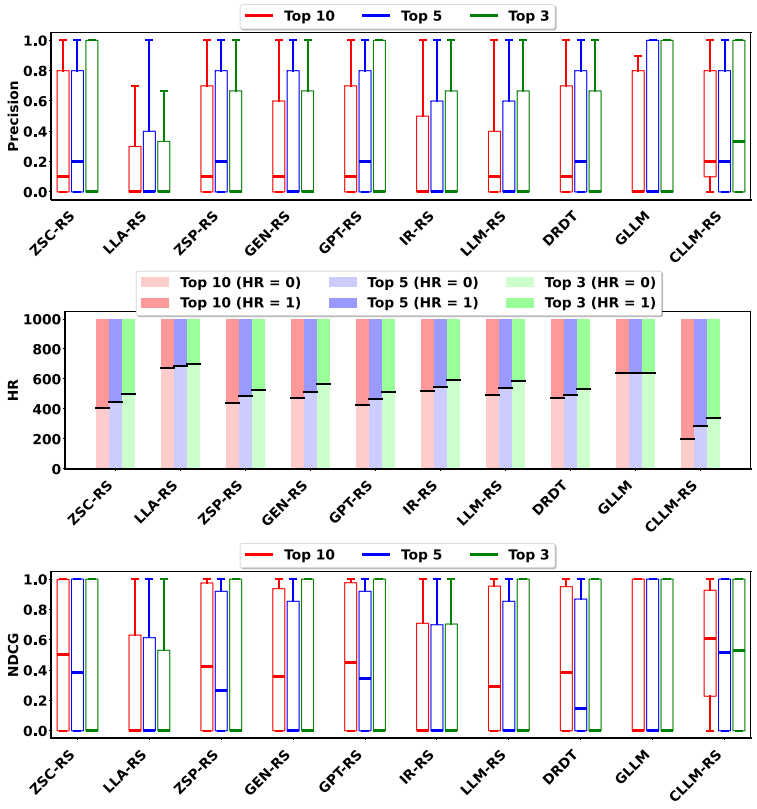


Fig. 4. Evaluation on the Book dataset with estimated ratings. In the final recommendations, the Top-N items are extracted for evaluation (red for Top-10, blue for Top-5, and green for Top-3).

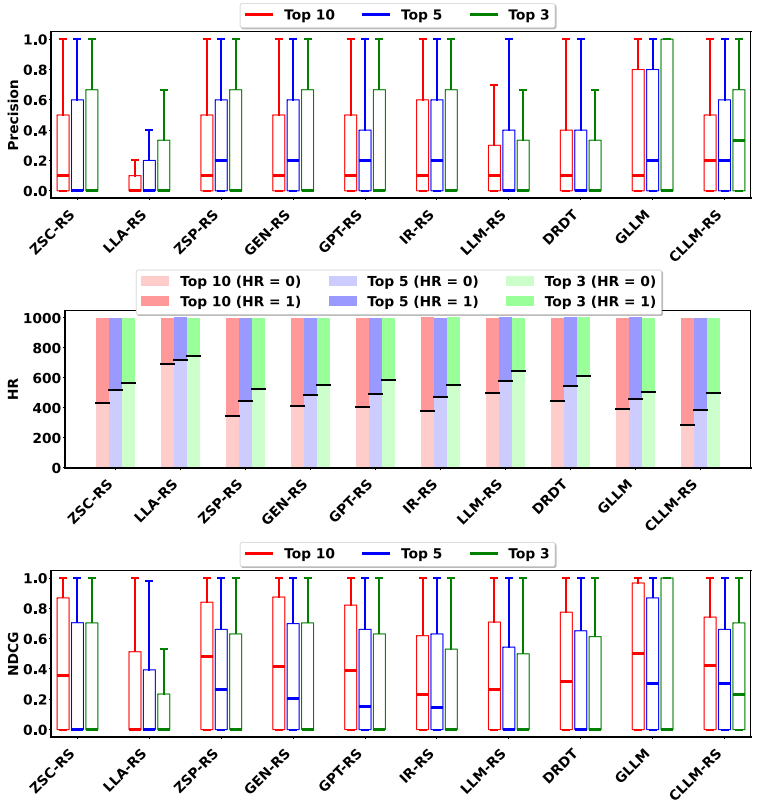


Fig. 5. Evaluation on Movie with the estimated ratings. In the final recommendations, the Top-N items are extracted for evaluation (red for Top-10, blue for Top-5, and green for Top-3).

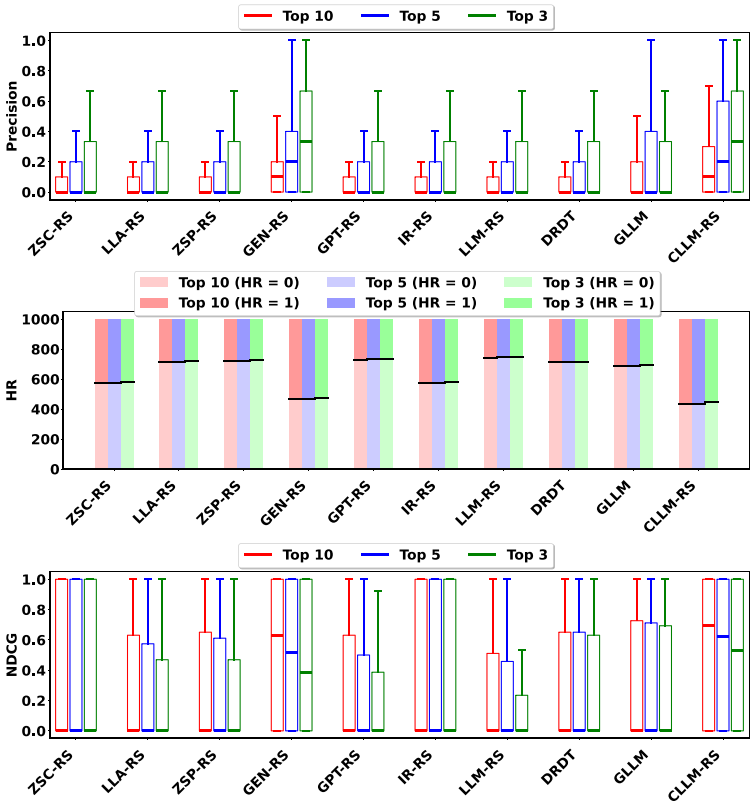


Fig. 6. Evaluation on Book with a candidate set. In the final recommendations, the Top-N items are extracted for evaluation (red for Top-10, blue for Top-5, and green for Top-3).

are limited and therefore the information is insufficient to represent user preferences. The greater n_{max} (i.e., $n_{max} = 30, 50$) provides more information to represent user preferences. However, we noticed that many users do not have 30 or 50 items in their interaction history. In the Movie dataset, 52% of users have interaction history with <30 items, and 65% with <50 items. In the Book dataset, 35% of users have interaction history with <30 items, and 62% with <50 items. Since dummy items are inserted in this situation, this degrades the fidelity of the user preferences and thus negatively impact the predictive accuracy of Recommendation Critic. This empirical study motivates us to set $n_{max} = 20$ by default as a balanced choice between context sufficiency and noise reduction.

5.3. Critic-LLM-RS vs. Non-tuning LLM-as-RS baselines

As Critic-LLM-RS follows the non-tuning LLM-as-RS strategy, eight representative non-tuning LLM-as-RS methods are compared. As discussed in related work, they rely on the inherent prowess of pre-trained LLMs to craft recommendations directly.

- Zero-shot-CoT LLM-RS (ZSC-RS) [54] enhances understanding of complex user preferences and logical reasoning by introducing the *think step-by-step* prompt in the recommendation process.
- Llama4Rec (LLA-RS) [8] is a method that integrates LLMs with traditional recommender systems. It employs data augmentation and prompt augmentation strategies to harness the strengths of both.
- Zero-shot-Prompt LLM-RS (ZSP-RS) [55] utilizes natural language commands to help understand complex user needs without targeted training.
- GENREC (GEN-RS) [10] generates personalized recommendation lists by understanding users' historical interactions and preferences through specialized prompts.
- GPTrec (GPT-RS) [13] aims to tackle the challenges of sequential recommendation.
- InteraRec (IR-RS) [9] leverages Multimodal Large Language Models (MLLMs) to analyze user behavior by integrating movie (or book) posters from user's viewing history with relevant textual information such as movie attributes to deliver precise recommendations.
- DRDT (Dynamic Reflection with Divergent Thinking) [19] utilizes a retriever to extract candidate items from a dataset, like the idea of retrieval-augmented generation (RAG) [41], based on the similarity of user behaviors, and then prompt LLMs to identify the important aspects of user preference and make recommendations.
- GaCLLM (GLLM) [40] combines Graph Convolutional Networks (GCN) and Large Language Models to integrate structured information and textual data of users and items into a unified embedding space, enabling efficient personalized recommendations by calculating matching scores.

We employ a popular open-source large language model called Vicuna [56]. Specifically, we use version 1.5 of the LLM with a size of 7B parameters. For comparability, all baselines are reproduced based on their respective publications: we faithfully follow the described configurations and re-implement the prompting and algorithmic components accordingly.

Figs. 4 and 5 report the performance of Critic-LLM-RS (denoted as CLLM-RS) and baselines for top-3, top-5, and top-10 recommendations. If real ratings are not available, the estimated ratings using the Recommendation Critic are considered. Besides baselines, we also compare with *LLM-RS*, which represents the initial recommendations of Critic-LLM-RS. The colors in Figs. 4 and 5 denote different Top-N recommendations (red for Top-10, blue for Top-5, green for Top-3). The results demonstrate that the performance of our Critic-LLM-RS is better

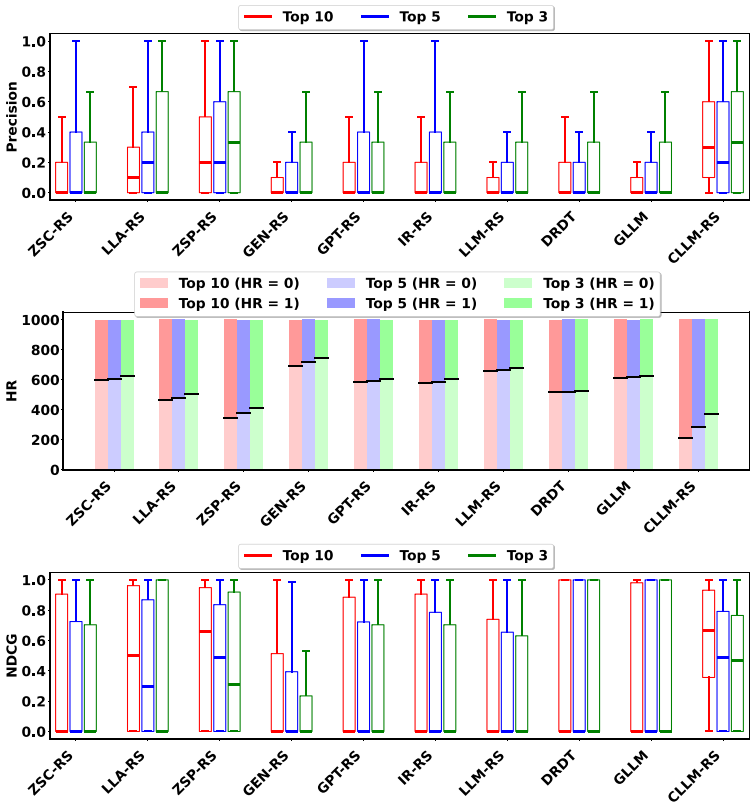


Fig. 7. Evaluation on Movie with a candidate set. In the final recommendations, the Top-N items are extracted for evaluation (red for Top-10, blue for Top-5, and green for Top-3).

than all baselines and LLM-RS in almost all cases on both datasets: the gains are most pronounced at smaller recommendation (Top-3/Top-5) and remain positive at Top-10. Note that the superior performance of Critic-LLM-RS against LLM-RS verifies the effectiveness of Recommendation Critic. Without the feedback from the Recommendation Critic, the initial recommendations (i.e., LLM-RS) do not have an advantage compared with baselines.

Figs. 6 and 7 report the experiment results where the LLM is prompted to recommend items from a candidate set of items with real ratings. The similar performance advantage of our Critic-LLM-RS can be observed as in Figs. 4 and 5. In the candidate-constrained setting, all methods score and rank the same pool of candidates with ground-truth ratings, thereby removing differences arising from open-world candidate recall. Consequently, in Figs. 6 and 7, Critic-LLM-RS consistently outperforms both the baselines and its initial list (*LLM-RS*); the gains are most pronounced at smaller recommendation (Top-3/Top-5) and remain positive at Top-10. Consistent with the observations in Figs. 4 and 5, this indicates that Recommendation Critic provides a stable scoring signal and systematically improves ranking quality.

5.4. Feedback in multiple loops

So far, Recommendation Critic provides feedback on the initial recommendations of LLM, and then LLM takes the feedback to improve its recommendations. What if such a criticize-and-enhance process is repeated for multiple loops? To investigate it, we record the recommendations generated by the LLM after each loop and evaluate them using Precision, HR, and NDCG. The test results are reported in Fig. 8. Interestingly, the performance is almost the same after the first loop. This demonstrates that LLM can effectively learn how to refine the recommendations in a single criticize-and-enhance process.

5.5. Critic-LLM-RS vs. Tuning-based method

We manage to fine-tune an open-source LLM as a recommender system. Given the interaction history of a user, the LLM is fine-tuned to recommend items (i.e., the textual information of movies or books including title, year, director/authors, etc.) that the user prefers (i.e., the items rated ≥4 by the user). The input is the same prompt template for initial recommendations (see case study in Section 4). LoRA [57] is used for the supervised fine-tuning, and the base open-source LLM is Vicuna. We set the learning rate to 1e–4 and the training epochs to 3.

Fig. 9 reports the performance of the fine-tuned LLM-as-RS (LLM-FT-RS), initial recommendations (LLM-RS), and our Critic-LLM-RS (CLLM-RS). It is clear that compared to LLM-RS, both LLM-FT-RS and Critic-LLM-RS exhibit superior performance. Notably, Critic-LLM-RS performs remarkably better than LLM-FT-RS. Look closely, this might be caused by the loss function when performing supervised fine-tuning on the LLM. The loss function is a token-by-token match targeting to output the ground truth text, which deviates from the objective of recommendations.

5.6. Critic-LLM-RS with proprietary LLM

We further evaluate Critic-LLM-RS with four representative proprietary LLMs: three black-box API models (GPT-4o, Gemini-2.5, GLM-4.5-Air) and one open-source/local model (Qwen3-7B). Under the same prompt template (see Section 4), each model first produces initial recommendations, denoted as *no-critic* in Figs. 10, 11, 12, and 13. After injecting feedback from Recommendation Critic, the refined recommendations are returned by the LLM. Here, we repeat the criticize-and-enhance process for three loops, and report the performance at the end of each loop, denoted as *1-critic*, *2-critic*, and *3-critic*, respectively. The results show that incorporating feedback from Recommendation Critic yields consistent improvements with different LLMs, and a single criticize-and-enhance process (1-critic) is sufficient.

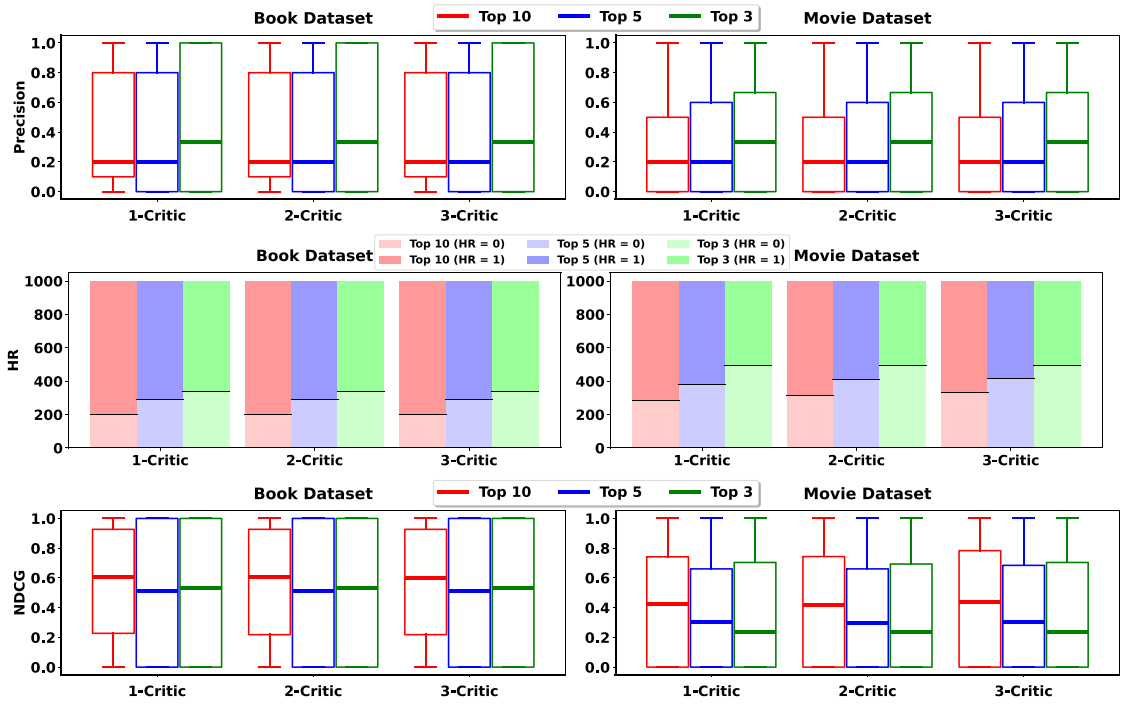


Fig. 8. Provide feedback in multiple loops. In the final recommendations, the Top-N items are extracted for evaluation (red for Top-10, blue for Top-5, and green for Top-3).

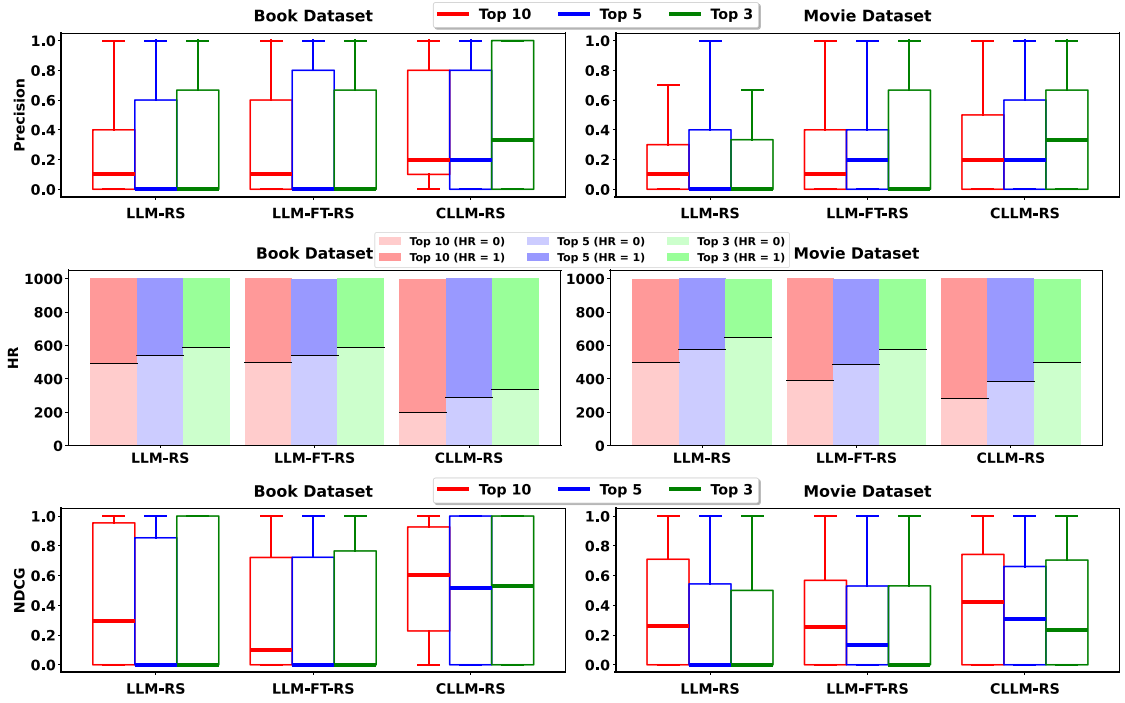


Fig. 9. Critic-LLM-RS vs. Fine-tuned Method. In the final recommendations, the Top-N items are extracted for evaluation (red for Top-10, blue for Top-5, and green for Top-3).

5.7. Time cost of Critic-LLM-RS

Figs. 14, 15, 16, 17, and 18 illustrate the distribution of time consumed using Recommendation Critic with Vicuna, GPT4o, Qwen3-7B, GLM-4.5-Air and Gemini-2.5, respectively. x-axis is the time (seconds) and y-axis is the number of users, in the test dataset, to whom the final recommendation is generated. The results demonstrate that the time consumed by applying the Recommendation Critic is trivial even after multiple loops.

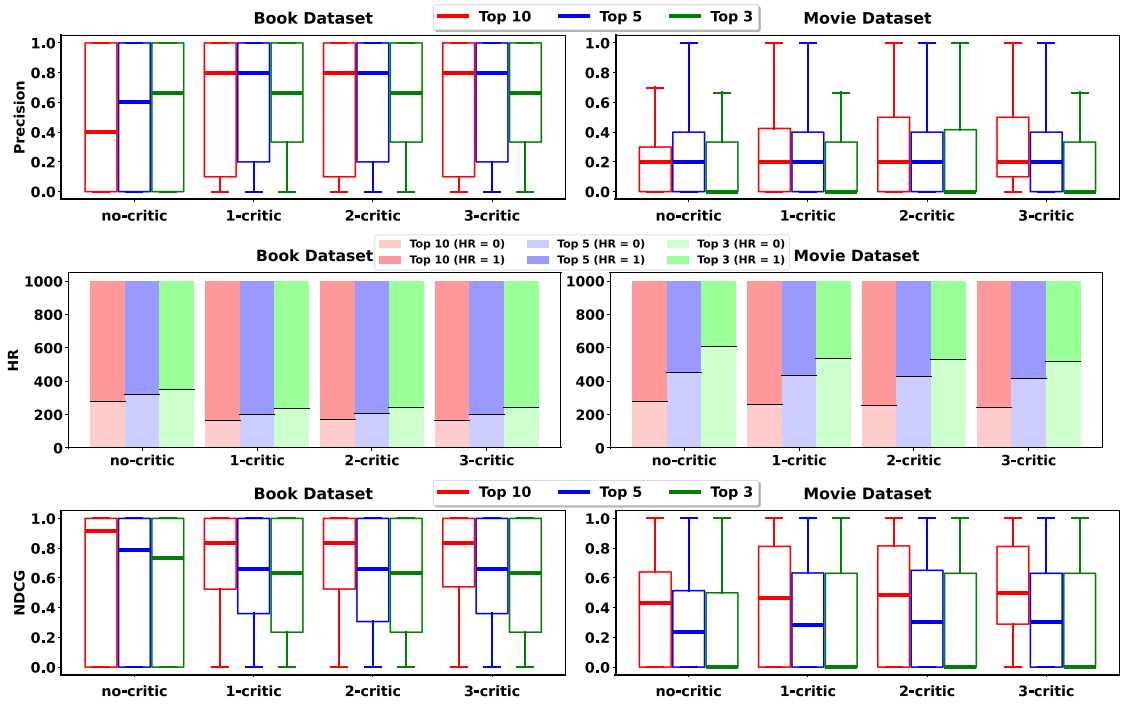


Fig. 10. Critic-LLM-RS with Proprietary GPT4o. In the final recommendations, the Top-N items are extracted for evaluation (red for Top-10, blue for Top-5, and green for Top-3).

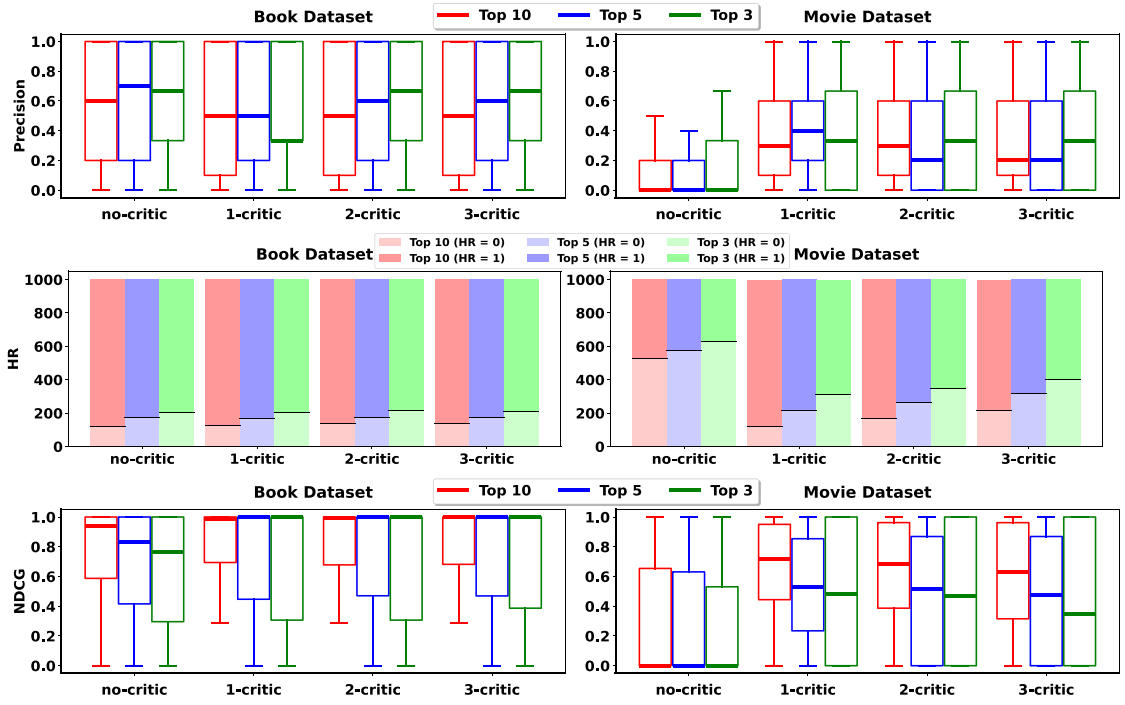


Fig. 11. Critic-LLM-RS with Proprietary Gemini. In the final recommendations, the Top-N items are extracted for evaluation (red for Top-10, blue for Top-5, and green for Top-3).

6. Threats to validity and limitations

6.1. Threats to validity

A classic challenge in recommender system evaluation is how to assess items that lack user ratings (e.g., new items or out-of-dataset entries). This problem also exists in our study. We adopt two methods to evaluate Critic-LLM-RS under such a challenge: (i) An open-world setting in which missing ratings are proxies by the Recommendation Critic's scores; and (ii) A candidate-constrained setting in which all methods re-rank the same pool of candidates with available ground-truth ratings.

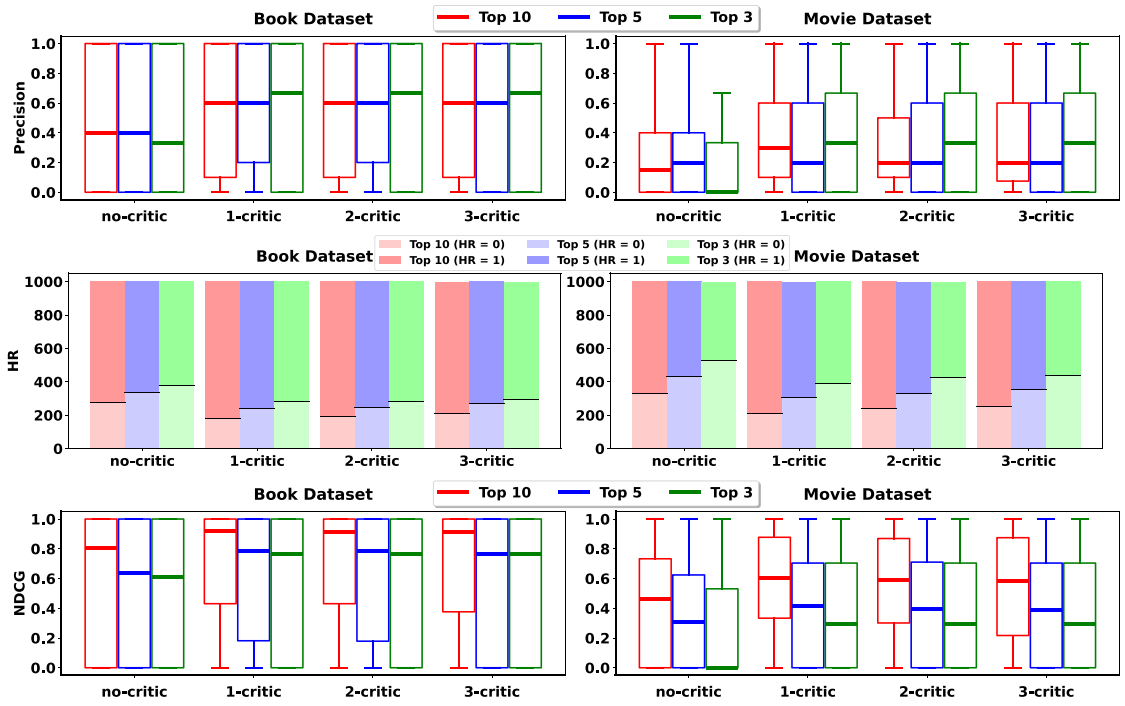


Fig. 12. Critic-LLM-RS with Proprietary GLM-4.5-air. In the final recommendations, the Top-N items are extracted for evaluation (red for Top-10, blue for Top-5, and green for Top-3).

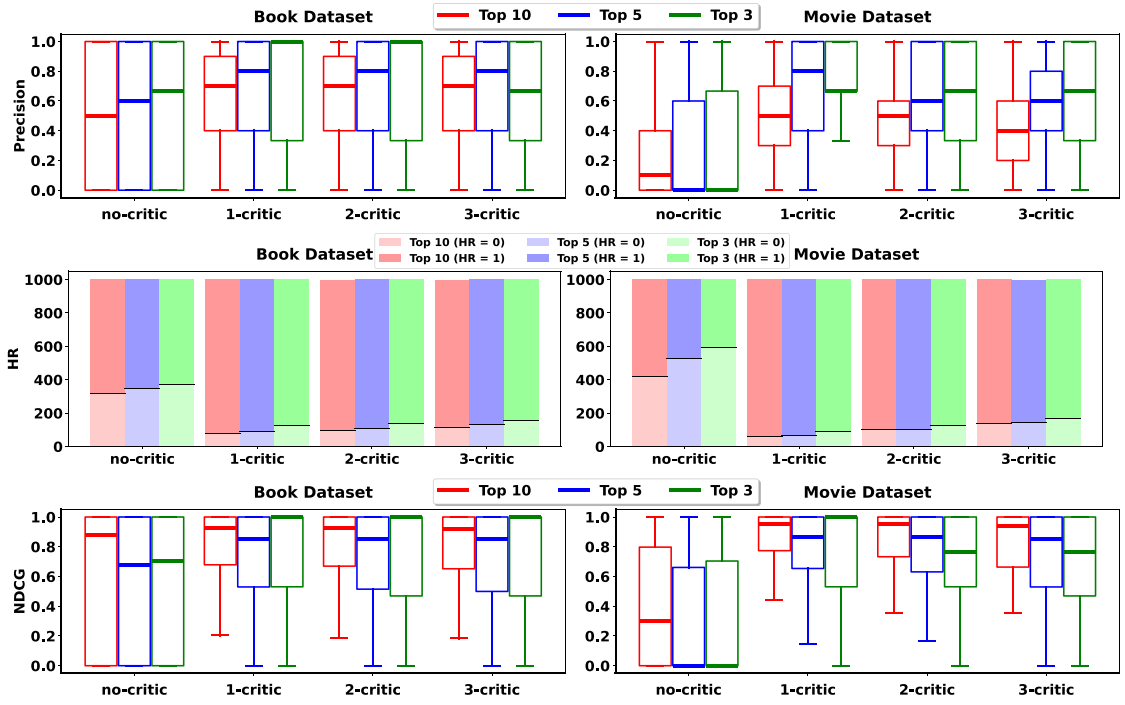


Fig. 13. Critic-LLM-RS with Proprietary Qwen. In the final recommendations, the Top-N items are extracted for evaluation (red for Top-10, blue for Top-5, and green for Top-3).

While the candidate-constrained setting enables a fair comparison on known items, it cannot properly evaluate the performance in an open-world scenario, as it relies solely on ground-truth data. Conversely, the open-world setting may also threaten validity, since evaluating critic-generated proxy ratings might not fully reflect real user preferences. To mitigate these issues, we conduct evaluations under both settings to provide a more comprehensive evaluation of Critic-LLM-RS.

6.2. Limitations

Critic-LLM-RS has two primary limitations: (i) Finite context window of the LLM. Long user histories and verbose item descriptions

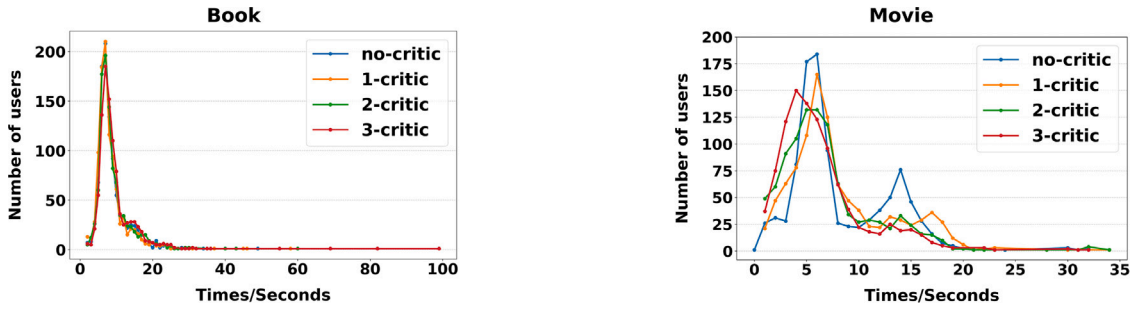


Fig. 14. Distribution of time consumed (Vicuna).

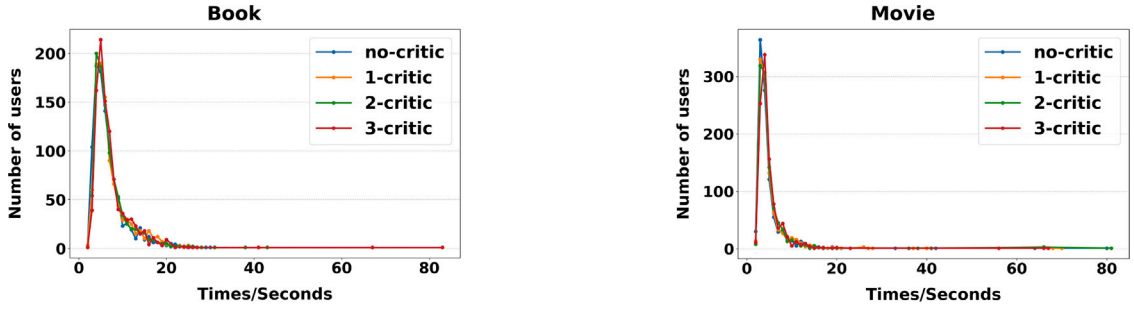


Fig. 15. Distribution of time consumed (GPT4o).

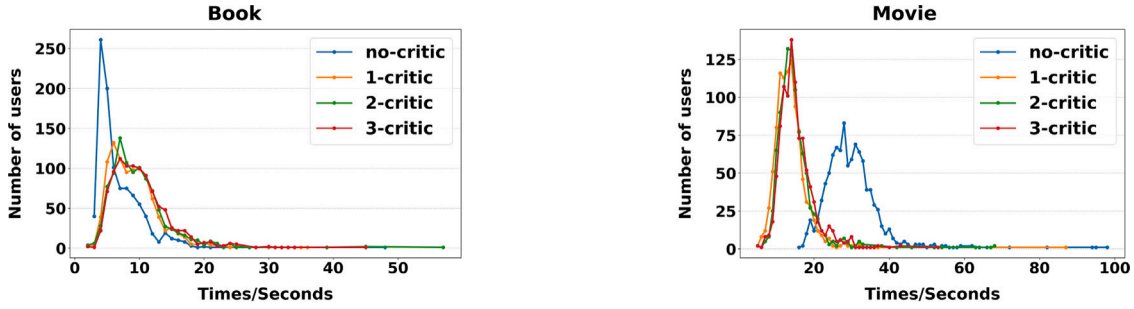


Fig. 16. Distribution of time consumed (Qwen3-7B).

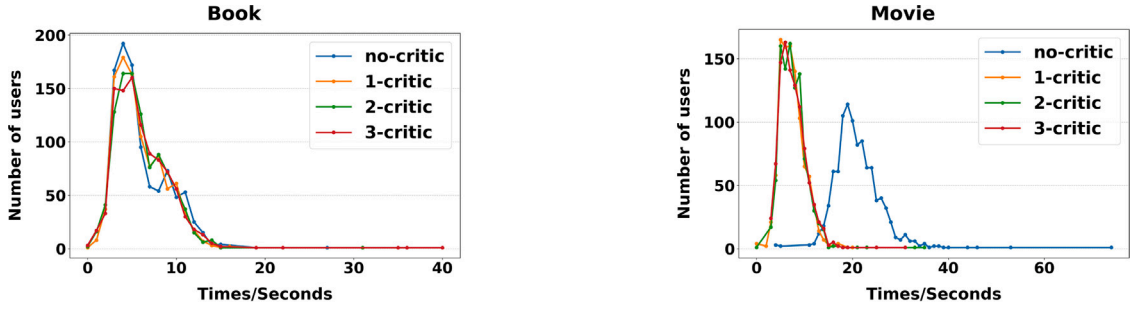


Fig. 17. Distribution of time consumed (GLM-4.5-air).

may be truncated or compressed, which may degrade the quality of the recommendations. Under the settings and data scale of this study, we did not encounter context-length (token) overflows; however, this may become a bottleneck for longer texts. (ii) Dependence on the Recommendation Critic's scoring accuracy and confidence. Since Critic-LLM-RS refines its recommendations based on the Recommendation Critic's feedback, the quality of these refinements directly depends on the reliability of the Recommendation Critic's feedback. If Recommendation Critic is biased or unstable, its feedback can be misleading, causing performance degradation.

In addition, our method can partially mitigate the *item cold-start problem*: Recommendation Critic can rate a new item based on the information of the item, although the item has not been rated by any users before. However, it cannot address the *user cold-start problem*, as

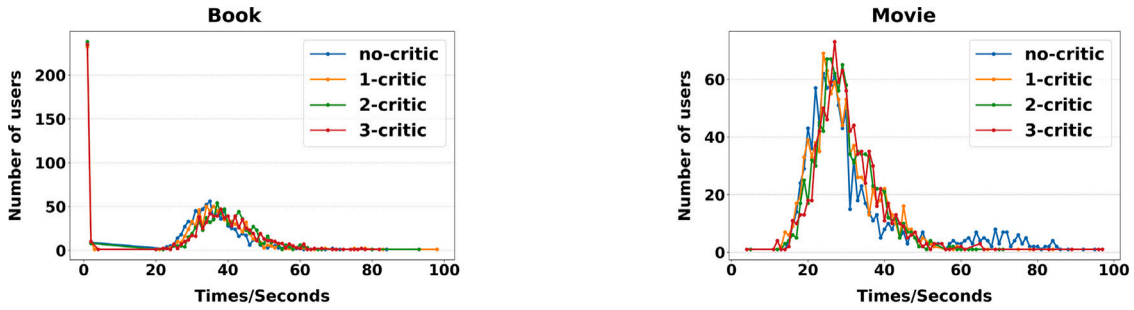


Fig. 18. Distribution of time consumed (Gemini-2.5).

the Recommendation Critic's feedback depends on the user's interaction history; when this history is unavailable, the model cannot infer the user's preferences.

7. Conclusion and future work

This study proposes to train a separate machine-learning model called *Recommendation Critic* for the capability of collaborative and content-based filtering and integrates it with LLM as a recommendation system, named Critic-LLM-RS. By providing LLM feedback on its recommendations, Critic-LLM-RS prompts LLMs to refine recommendations. Critic-LLM-RS enjoys the comprehensive knowledge of pre-trained LLM which can recommend items not included in the recommender system training datasets, but also enjoys accurate recommendations due to comprehensive information from collaborative and content-based filtering. The extensive experiments and a case study have verified the effectiveness of the Critic-LLM-RS for recommendation tasks.

In the future research agenda, we plan to address the token limitation of LLM in Critic-LLM-RS. A promising direction is to identify the most useful items in the user interaction history so that user preferences can be represented with a limited number of user–item interactions. To this end, we will leverage RAG (Retrieval-Augmented Generation), a popular technique that enhances LLMs by allowing them to access and incorporate information from external knowledge bases. The strategy is intended to reduce the LLM input tokens while improving effectiveness through targeted retrieval, achieving a more favorable balance between efficiency and performance.

CRediT authorship contribution statement

Zhisheng Yang: Writing – original draft, Validation, Software, Methodology, Data curation. **Xiaofei Xu:** Writing – review & editing, Supervision, Resources, Methodology. **Ke Deng:** Writing – review & editing, Supervision, Resources, Methodology. **Li Li:** Writing – review & editing, Supervision, Resources, Methodology.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The complete replication materials for this study are publicly available in a GitHub repository (https://github.com/yzsyzs478/Critic-LLM-RS).

References

[1] L. Wu, Z.e.a. Zheng, A survey on large language models for recommendation, 2023, arXiv preprint arXiv:2305.19860.
[2] K. Bao, J. Zhang, et al., Tallrec: An effective and efficient tuning framework to align large language model with recommendation, in: Proceedings of the 17th ACM Conference on Recommender Systems, 2023, pp. 1007–1014.
[3] W.-C. Kang, J. Ni, et al., Do llms understand user preferences? Evaluating llms on user rating prediction, 2023, arXiv preprint arXiv:2305.06474.
[4] W. Xu, Q. Xie, et al., Enhancing content-based recommendation via large language model, 2024, arXiv preprint arXiv:2404.00236.
[5] Z. Chu, H. Hao, et al., Leveraging large language models for pre-trained recommender systems, 2023, arXiv preprint arXiv:2308.10837.
[6] L. Li, Y. Zhang, et al., Prompt distillation for efficient llm-based recommendation, in: Proceedings of the 32nd ACM International Conference on Information and Knowledge Management, 2023, pp. 1348–1357.
[7] P. Liu, W. Yuan, et al., Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing, ACM Comput. Surv. 55 (9) (2023) 1–35.
[8] S. Luo, Y. Yao, et al., Integrating large language models into recommendation via mutual augmentation and adaptive aggregation, 2024, arXiv preprint arXiv:2401.13870.
[9] S.R. Karra, T. Tulabandhula, Interarec: Interactive recommendations using multimodal large language models, in: Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer, 2024, pp. 32–43.
[10] J. Ji, Z. Li, et al., Genrec: Large language model for generative recommendation, in: European Conference on Information Retrieval, Springer, 2024, pp. 494–502.
[11] J. Zhang, K. Bao, et al., Is chatgpt fair for recommendation? evaluating fairness in large language model recommendation, in: Proceedings of the 17th ACM Conference on Recommender Systems, 2023, pp. 993–999.
[12] J. Liu, C. Liu, et al., Is chatgpt a good recommender? A preliminary study, 2023, arXiv preprint arXiv:2304.10149.
[13] A.V. Petrov, C. Macdonald, Generative sequential recommendation with gptrec, 2023, arXiv preprint arXiv:2306.11114.
[14] A. Kumar, S. Sankar, P.P. Das, P.P. Chakrabarti, Using large language models for multi-level commit message generation for large diffs, Inf. Softw. Technol. (2025) 107831.
[15] Y. Hou, J. Zhang, et al., Large language models are zero-shot rankers for recommender systems, in: European Conference on Information Retrieval, Springer, 2024, pp. 364–381.
[16] S. Che, M. Mao, et al., New community cold-start recommendation: A novel large language model-based method, 2024, Available At SSRN 4828316.
[17] S. Sanner, et al., 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, RecSys '23, Association for Computing Machinery, New York, NY, USA, 2023, pp. 890–896, http://dx.doi.org/10.1145/3604915.3608845.
[18] Y. Wang, Z. Jiang, et al., Recmind: Large language model powered agent for recommendation, 2023, arXiv preprint arXiv:2308.14296.
[19] Y. Wang, Z. Liu, et al., Drdt: Dynamic reflection with divergent thinking for llm-based sequential recommendation, 2023, arXiv preprint arXiv:2312.11336.
[20] J. Devlin, M.-W. Chang, et al., Bert: Pre-training of deep bidirectional transformers for language understanding, 2018, arXiv preprint arXiv:1810.04805.
[21] F. Sun, J. Liu, J. Wu, C. Pei, X. Lin, W. Ou, P. Jiang, BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer, in: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, 2019, pp. 1441–1450.
[22] L. Wu, Z. Qiu, et al., Exploring large language model for graph data understanding in online job recommendations, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 38, 2024, pp. 9178–9186.
[23] Z. Qiu, X.e.a. Wu, U-BERT: Pre-training user representations for improved recommendation, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, 2021, pp. 4320–4327.

[24] Y. Yang, Y.e.a. Qiao, Lightweight composite re-ranking for efficient keyword search with BERT, in: Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining, 2022, pp. 1234–1244.
[25] X. Wu, A. Magnani, et al., A multi-task learning framework for product ranking with bert, in: Proceedings of the ACM Web Conference 2022, 2022, pp. 493–501.
[26] Y. Hou, S. Mu, et al., Towards universal sequence representation learning for recommender systems, in: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2022, pp. 585–593.
[27] C. Wu, F. Wu, et al., Empowering news recommendation with pre-trained language models, in: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2021, pp. 1652–1656.
[28] Q. Liu, J. Zhu, et al., Boosting deep CTR prediction with a plug-and-play pre-trainer for news recommendation, in: Proceedings of the 29th International Conference on Computational Linguistics, 2022, pp. 2823–2833.
[29] R. Li, W. Deng, et al., Exploring the upper limits of text-based collaborative filtering using large language models: Discoveries and insights, 2023, arXiv preprint arXiv:2305.11700.
[30] Z. Yuan, F. Yuan, et al., Where to go next for recommender systems? id-vs. modality-based recommender models revisited, in: Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2023, pp. 2639–2649.
[31] J. Zhang, Y. Hou, et al., Agentcf: Collaborative learning with autonomous language agents for recommender systems, in: Proceedings of the ACM on Web Conference 2024, 2024, pp. 3679–3689.
[32] J. Lin, X. Dai, et al., How can recommender systems benefit from large language models: A survey, 2023, arXiv preprint arXiv:2306.05817.
[33] G. Trichopoulos, M. Konstantakis, et al., Large language models as recommendation systems in museums, Electronics 12 (18) (2023) 3829.
[34] W. Wei, X. Ren, et al., Llmrec: Large language models with graph augmentation for recommendation, in: Proceedings of the 17th ACM International Conference on Web Search and Data Mining, 2024, pp. 806–815.
[35] Z. Zheng, Z. Qiu, et al., Generative job recommendations with large language model, 2023, arXiv preprint arXiv:2307.02157.
[36] Y. Xi, W. Liu, et al., Towards open-world recommendation with knowledge augmentation from large language models, 2023, arXiv preprint arXiv:2306.10933.
[37] W. Wang, X. Lin, et al., Generative recommendation: Towards next-generation recommender paradigm, 2023, arXiv preprint arXiv:2304.03516.
[38] X. Wang, K. Zhou, et al., Towards unified conversational recommender systems via knowledge-enhanced prompt learning, in: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2022, pp. 1929–1937.
[39] L. Wang, J. Zhang, et al., Recagent: A novel simulation paradigm for recommender systems, 2023, arXiv preprint arXiv:2306.02552.
[40] Y. Du, Z. Wang, et al., Large language model with graph convolution for recommendation, 2024, arXiv preprint arXiv:2402.08859.
[41] P. Lewis, E. Perez, et al., Retrieval-augmented generation for knowledge-intensive nlp tasks, Adv. Neural Inf. Process. Syst. 33 (2020) 9459–9474.
[42] T. Schick, A.Y. Jane, et al., PEER: A collaborative language model, in: The Eleventh International Conference on Learning Representations, 2022.
[43] A.F. Akyürek, E. Akyürek, et al., RL4F: Generating natural language feedback with reinforcement learning for repairing model outputs, in: Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2023, pp. 7716–7733.
[44] H. Lee, S. Phatale, et al., Rlaif: Scaling reinforcement learning from human feedback with ai feedback, 2023, arXiv preprint arXiv:2309.00267.
[45] W. Yu, Z. Zhang, et al., Improving language models via plug-and-play retrieval feedback, 2023, arXiv preprint arXiv:2305.14002.
[46] G.D. De Siano, A.R. Fasolino, G. Sperlí, A. Vignali, Translating code with large language models and human-in-the-loop feedback, Inf. Softw. Technol. (2025) 107785.
[47] A. Kalyanpur, K. Saravanakumar, et al., LLM-ARC: Enhancing LLMs with an automated reasoning critic, 2024, arXiv preprint arXiv:2406.17663.
[48] Z. Gou, Z. Shao, et al., CRITIC: Large language models can self-correct with tool-interactive critiquing, in: The Twelfth International Conference on Learning Representations, 2024, URL: https://openreview.net/forum?id=Sx038qxjek.
[49] E.J. Hu, yelong shen, et al., LoRA: Low-rank adaptation of large language models, in: International Conference on Learning Representations, 2022, URL: https://openreview.net/forum?id=nZeVKeeFYf9.
[50] X. He, L. Liao, et al., Neural collaborative filtering, 2017, CoRR abs/1708.05031. URL: http://arxiv.org/abs/1708.05031. arXiv:1708.05031.
[51] R. van Meteren, Using content-based filtering for recommendation, 2000, URL: https://api.semanticscholar.org/CorpusID:2088490.
[52] D. Kotkov, A. Maslov, et al., Revisiting the tag relevance prediction problem, in: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2021, pp. 1768–1772.
[53] D. Kotkov, A. Medlar, et al., The tag genome dataset for books, in: Proceedings of the 2022 Conference on Human Information Interaction and Retrieval, 2022, pp. 353–357.
[54] T. Kojima, S.S. Gu, et al., Large language models are zero-shot reasoners, Adv. Neural Inf. Process. Syst. 35 (2022) 22199–22213.
[55] J. Wei, M. Bosma, et al., Finetuned language models are zero-shot learners, 2021, arXiv preprint arXiv:2109.01652.
[56] L. Zheng, W.-L. Chiang, et al., Judging llm-as-a-judge with mt-bench and chatbot arena, Adv. Neural Inf. Process. Syst. 36 (2024).
[57] E.J. Hu, P. Wallis, et al., LoRA: Low-rank adaptation of large language models, in: International Conference on Learning Representations, 2021.