Re2LLM: Reflective Reinforcement Large Language Model for Session-based Recommendation

Ziyan Wang
wang1753@e.ntu.edu.sg
Nanyang Technological University
Singapore

Yingpeng Du
dyp1993@pku.edu.cn
Nanyang Technological University
Singapore

Zhu Sun
sunzhuntu@gmail.com
Agency for Science, Technology and Research
Singapore

Haoyan Chua
haoyan001@e.ntu.edu.sg
Nanyang Technological University
Singapore

Kaidong Feng
kaidong3762@gmail.com
Yanshan University
China

Wenya Wang
wangwy@ntu.edu.sg
Nanyang Technological University
Singapore

Jie Zhang
zhangj@ntu.edu.sg
Nanyang Technological University
Singapore

ABSTRACT

Large Language Models (LLMs) are emerging as promising approaches to enhance session-based recommendation (SBR), where both prompt-based and fine-tuning-based methods have been widely investigated to align LLMs with SBR. However, the former methods struggle with optimal prompts to elicit the correct reasoning of LLMs due to the lack of task-specific feedback, leading to unsatisfactory recommendations. Although the latter methods attempt to fine-tune LLMs with domain-specific knowledge, they face limitations such as high computational costs and reliance on open-source backbones. To address such issues, we propose a Reflective Reinforcement Large Language Model (Re2LLM) for SBR, guiding LLMs to focus on specialized knowledge essential for more accurate recommendations effectively and efficiently. In particular, we first design the Reflective Exploration Module to effectively extract knowledge that is readily understandable and digestible by LLMs. To be specific, we direct LLMs to examine recommendation errors through self-reflection and construct a knowledge base (KB) comprising hints capable of rectifying these errors. To efficiently elicit the correct reasoning of LLMs, we further devise the Reinforcement Utilization Module to train a lightweight retrieval agent. It learns to select hints from the constructed KB based on the task-specific feedback, where the hints can serve as guidance to help correct LLMs reasoning for better recommendations. Extensive experiments on multiple real-world datasets demonstrate that our method consistently outperforms state-of-the-art methods.

CCS CONCEPTS

• **Information systems** → **Recommender systems**.

KEYWORDS

Session-based Recommendation, Large Language Model, Self Reflection

ACM Reference Format:
Ziyan Wang, Yingpeng Du, Zhu Sun, Haoyan Chua, Kaidong Feng, Wenya Wang, and Jie Zhang. 2024. Re2LLM: Reflective Reinforcement Large Language Model for Session-based Recommendation. .

1 INTRODUCTION

Session-based recommendation (SBR) [2, 10, 14, 21, 51, 57] plays a crucial role in real-world applications. It aims to capture users' dynamic preferences and predicts the next item that users may prefer based on previous interactions within a session. However, the user-item interactions are often scarce, and users' profiles are inaccessible in anonymous sessions, hindering the accurate recommendation result due to data sparsity and cold-start issues [28, 36].

Large Language Models (LLMs) have emerged to show potential in addressing these issues with their extensive knowledge and sophisticated reasoning capabilities. Recently, numerous methods have integrated LLMs into recommender systems (RSs), primarily through two ways: prompt-based methods and fine-tuning-based methods. The former methods exploit in-context learning and prompt optimization [8, 16, 37, 42] to engage LLMs (e.g., ChatGPT[1]) as recommenders without training as shown in Fig.1 (a). However, the crafted prompts not only require extensive expert knowledge and human labor but also may not align well with LLMs' understanding of SBR tasks, making LLMs vulnerable to hallucinations without effective supervision. The latter methods focus on fine-tuning LLMs [1, 9, 50, 60] with domain-specific knowledge, such as user-item interactions, in a supervised manner as shown in Fig.1 (b). However, such methods often suffer from high computational costs, catastrophic forgetting, and reliance on open-source backbones. These drawbacks restrict the practical application of existing LLM-based methods for SBR.

In this paper, we propose to direct LLMs to effectively and efficiently leverage specialized knowledge for recommendation without costly and inconvenient fine-tuning. However, there remain two primary challenges to this goal: (1) How can we effectively extract

[1] https://chat.openai.com/

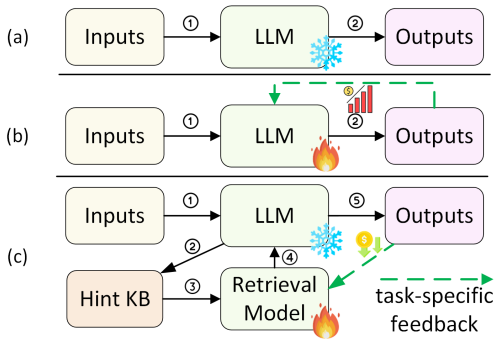

Figure 1: Our method (c) obtains effective task-specific feedback compared with prompt-based (a) and fine-tuning-based (b) methods.

and craft specialized knowledge embedded within extensive user-item interactions to better align with LLM comprehension? (2) How can we enable LLMs to utilize the specialized knowledge efficiently for better recommendations, without relying on resource-intensive supervised fine-tuning?

To overcome these challenges, we propose a Reflective Reinforcement Large Language Model (Re2LLM) for SBR, aiming to capture and utilize specialized knowledge embedded in the extensive user-item interaction data by LLMs for more accurate recommendation. The proposed method consists of two main components: the Reflective Exploration Module and the Reinforcement Utilization Module.

- **Reflective Exploration Module** leverages the self-reflection of LLMs [27, 29] to extract specialized knowledge that is comprehensible and digestible by LLMs for SBR. Specifically, we employ LLMs to identify common errors in their responses, and then generate corresponding specialized knowledge (i.e., hints) to rectify these errors through LLMs' self-reflections. Hence, the specialized knowledge can align seamlessly with LLM comprehension as it is summarized by the LLM itself. Besides, we construct a hint knowledge base that serves as a pool to maintain specialized knowledge (hints), using an automated filtering strategy to ensure the effectiveness and diversity of retained hints.
- **Reinforcement Utilization Module** employs a lightweight retrieval agent to select relevant specialized knowledge, thus eliciting the correct reasoning of LLMs without costly fine-tuning. Specifically, the retrieval agent is trained to select accurate hints from the knowledge base, which can prevent LLMs from potential errors in their inference process. To overcome the absence of explicit labels about the effects of retrievals, we employ deep reinforcement learning (DRL) to simulate the real-world RSs for the agent. In detail, we first design an agent that selects hints (i.e., action) from the hint knowledge base by considering the session's contextual information (i.e., observation state). Then, we measure the improvement (i.e., reward) of recommendation results owing to the selected hint, and use them (rewards, observation states, and actions) as the task-specific feedback to update the agent via Proximal Policy Optimization (PPO) [33] strategy.

In summary, as illustrated in Fig.1 (c), we develop a new learning paradigm to direct LLMs to effectively explore specialized knowledge and efficiently utilize it for more accurate SBR. Although the LLM backbone remains frozen, it can be guided by a lightweight retrieval that consumes task-specific feedback without costly fine-tuning. Therefore, our method combines the strengths of the large-scale LLM's effective reasoning capabilities and the efficiency of the lightweight retrieval model training. The key contributions of this paper are three-fold:

- We propose a new learning paradigm beyond in-context learning and fine-tuning of LLMs, which seamlessly bridges between general LLMs and specific recommendation tasks. It alleviates the issues of unaligned knowledge and costly supervised fine-tuning for LLMs, contributing to better SBR predictions.
- Our Re2LLM benefits from LLMs' self-reflection capabilities (i.e., Reflective Exploration module) as well as the flexibility of the lightweight retrieval agent (Reinforcement Utilization Module). This enables us to effectively extract specialized knowledge understandable by LLMs, which can be then efficiently utilized for better LLM inference by learning from task-specific feedback.
- Our extensive experiments demonstrate that Re2LLM outperforms state-of-the-art methods, including deep learning-based models and LLM-based models, in both few-shot and full-data settings across two real-world datasets.

2 LITERATURE REVIEW

2.1 Session-based Recommendation

SBR methods learn to model users' preferences and make recommendations based on short and dynamic sessions. The early classic work FPMC [32] combines Matrix Factorization [20] and Markov Chain to capture sequential patterns and long-term user preference. With the development of deep learning technologies, various advanced techniques have been widely applied in SBR. Hidasi et al. [13] first propose to use the Recurrent Neural Network (RNN) for SBR due to its strength in modeling sequential session interactions. Several methods propose variants by data augmentation [39], novel ranking loss function [12], and mixture-channel purpose routing networks (MCPRN) [43]. Furthermore, the attention mechanism [41] is introduced to capture more informative item representations for users' dynamic preferences for SBR, such as NARM [22] and STAMP [26]. Recently, graph-based methods [24, 30, 47, 49, 54] leverage the Graph Neural Network (GNN) to better learn high-order transitions within the sessions from the graph structure for SBR. For example, GCEGNN [45] builds a global-level graph along with the item-level graph to exploit item transitions over all sessions for enhancement. Besides adopting different networks in SBR, many studies explore auxiliary information (e.g., attribute, description text) [3, 15, 48, 59] for better session profiling. For example, MMSBR [58] leverages both descriptive and numeric item attributes to characterize user intents. Nevertheless, these methods may still suffer from inadequate user-item interactions and limited auxiliary information, hindering the accuracy of recommendation results.

2.2 Large Language Model for Recommendation

LLMs have emerged to show potential in addressing the aforementioned issues with their extensive knowledge and sophisticated reasoning capability, gaining their popularity for recommendation tasks. Among LLM-based recommendation methods, most of the existing works attempt to utilize the LLM in two key strategies: prompt-based methods and fine-tuning (FT)-based methods.

Prompt-based methods [4, 6, 8, 11, 25, 56] retrieve information from recommendation data (e.g., historical user-item interactions) for direct outputs through prompt enhancement. LLMRank [16] and NIR [42] are two representatives that utilize prompt templates to extract dynamic preferences in anonymous sessions for SBR. To enrich LLMs with task-specific supervision, FT-based methods adopt either fully [9, 19, 23, 50] or parameter-efficient (PEFT) approaches [1, 7, 18, 55], such as LoRA [17], to fine-tune pre-trained LLMs for recommendation tasks. For example, with PEFT, TALLRec [1] bridges the gap between LLMs and the recommendation tasks. PALR [50] fully fine-tunes LLaMA-7b [40] based on historical interactions to improve sequential recommendation performance. However, their effectiveness is constrained by the substantial computational demands, reliance on the availability of open-source LLM backbones, and inferior capabilities compared with larger-scale language models such as ChatGPT. To address these challenges, we employ a lightweight retrieval agent to efficiently select relevant specialized knowledge summarized by LLMs, to elicit the correct reasoning of LLMs without costly fine-tuning.

2.3 Self-reflection in Large Language Models

Recent advances in prompting strategies have effectively enhanced LLMs' ability to handle complex tasks. Chain-of-Thought [46] and Tree-of-Thought [52] strategies allow LLMs to reason through a specified path. However, LLMs are still prone to hallucination and incorrect reasoning [29]. Consequently, much effort is devoted to exploring LLMs' self-reflection [27, 31, 34, 53] capabilities, where they iteratively refine outputs based on self-generated reviews or feedback without additional training. For example, Madaan et al. [27] apply this feedback-and-refine framework to improve LLMs' performance across NLP tasks. Pryzant et al. [31] improve prompts by having an LLM critique on generations and then refine them based on its feedback. Yao et al. [53] train a retrospective language model as a prompt generator, using the Proximal Policy Optimization (PPO) algorithm [33] based on environment-specific rewards for optimization. However, the potential of LLMs' self-reflection in recommendation remains underexplored. To the best of our knowledge, only DRDT [44] is related to reflection for recommendation, which triggers LLMs to conduct the iterative reflection on a specific session until hitting the ground truth item. However, it is essentially a case-by-case reflection strategy, which hinders summarising the specialized knowledge from global sessions. To this end, we propose to explore different sessions to maintain a hint knowledge base, whose knowledge (i.e., hints) can be utilized to enhance LLMs reasoning for all sessions.

3 METHODOLOGY

In this section, we present our proposed Re2LLM approach, whose overall architecture is illustrated in Fig. 2. Re2LLM comprises two key components: the Reflective Exploration Module and the Reinforcement Utilization Module. The Reflective Exploration Module facilitates the self-reflection of LLMs in identifying and summarizing their inference errors for SBR, thereby generating hints (known as specialized knowledge) to avoid these potential errors. These hints, further maintained by the automated filtering strategy to ensure their effectiveness and diversity, are stored in a knowledge base. The Reinforcement Utilization Module employs DRL to train a lightweight retrieval agent based on task-specific feedback without costly fine-tuning. The agent learns to select relevant hints to guide LLMs to mitigate potential errors in future inference. Finally, the inference is conducted by the LLM to deliver recommendations enhanced by these hints. For ease of presentation, we first introduce the notations used in this paper and provide problem formulation.

Notations and Problem Formulation. Let $\mathcal{V} = \{v_1, v_2, ..., v_N\}$ denote the item set with N items. Each session s with l interacted items is denoted as $s = \{v_1^s, v_2^s, ..., v_l^s\}$, where $v_i^s \in \mathcal{V}$. In addition, we are supposed to have side information on items, such as titles, genres, and actors of movies. Compared to traditional recommendation tasks, users' historical behaviors (i.e., interactions with items) on other sessions and their profiles are inaccessible in SBR due to the anonymous nature of sessions. The task of SBR is to predict the next item v_{l+1}^s that the user is likely to interact with based on session history s. For clarity, we denote the prompts designed in our methodology as P_j where j is the index. The notation $LLM(P_j)$ indicates the output of the LLM backbone given prompt P_j.

3.1 Reflective Exploration Module

Intuitively, accurate inference of LLMs requires domain-specific knowledge for recommendation tasks. However, the main challenge lies in that the domain-specific knowledge is usually embedded in the massive user-item interaction records, which may not well align with LLMs' comprehension. Alternatively stated, it is essential to extract the specialized knowledge that is intelligible for LLMs regarding recommendation tasks. Therefore, we propose to leverage LLMs' strong self-reflection capability for the extraction of specialized knowledge (i.e., hints) as LLMs can understand what they have summarized by themselves more easily (Section 3.1.1). Additionally, we introduce an automated filtering strategy (Section 3.1.2) that maintains effective and diverse hints to construct a knowledge base for further utilization.

3.1.1 Multi-Round Self-Reflection Generation. To activate the self-reflection ability of LLMs, we propose to identify errors in LLMs' inference by comparing their predictions against the ground truth. Specifically, we first instruct the LLM to generate a ranking list $O(s)$ from a candidate set C containing $|C|$ items based on current session s:

$$O(s) = LLM(P_1(s, C)), O \subseteq C, \tag{1}$$

where P_1 is the *basic prompt* consisting of titles of items that the user interacted with in the session s, candidate set C, and task instruction. The details are shown in Prompt 1.

Prompt 1 (basic prompt)

I watched/purchased the following movies/items in order: {s}. Based on these interactions, recommend a movie/item for me to watch/purchase next from a candidate set: {C}. Please recommend from the candidate set. List the top 10 recommendations in numbered bullet points.

Then, we identify the incorrectly predicted sessions where the ranking list fails to hit the ground truth item at time step $l + 1$ (i.e., $v_{l+1}^s \notin O(s)$). Afterward, we prompt the LLM to conduct self-reflection, that is, analyze and summarize the prediction errors, to generate potential hints in natural language to rectify such errors.

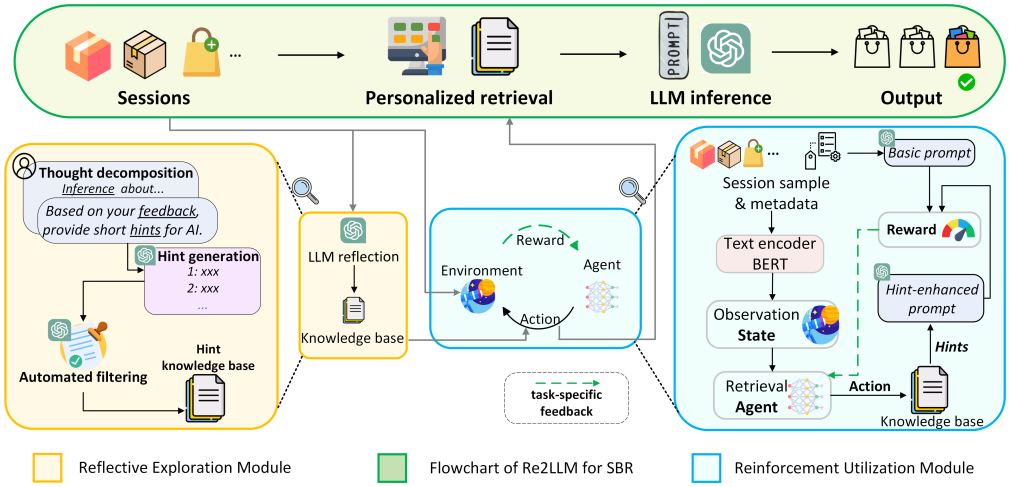

Figure 2: The overall architecture of our proposed Re2LLM approach. The overall flowchart (green) contains a Reflective Exploration Module (yellow) for the generation of session-aware hints as specialized knowledge and a Reinforcement Utilization Module (blue) for the learning to retrieve obtained specialized knowledge.

These hints thus can be considered as specialized knowledge for better recommendations. Inspired by Chain-of-Thought prompting [46], our approach focuses on analyzing and alleviating these errors in a step-by-step and progressive reasoning chain, producing hints that are tailored to be compatible with LLM comprehension capabilities, as introduced below.

Prompt 2

Question: {P_1} (basic prompt).
ChatGPT: { $O(s)$} (top-10 results: 1. Casino Royale 2. Batman 3...).
Now, know that none of these answers is the target. Infer about possible mistakes in the ChatGPT's predictions.
***Example Output**:*
Here are potential mismatches or mistakes in the incorrect recommendations:
Casino Royale: While this is an action film, it leans more towards the spy genre, which isn't strongly represented in the watched list.
Batman: This recommendation assumes ...

Prompt 3

The correct answer is { v^s_{l+1} } (target).
Analyze why ChatGPT missed the target item from a wide variety of aspects. Explain the causes for the oversights based on earlier analysis.
***Example Output**:*
Given v^s_{l+1} is the correct item, we can infer several potential causes for the mismatch in the original recommendations:
*1. **Missing Subtle Preferences**: Your list contains a mixture of different genres, but there is a subtle preference for comedies.*
*2. **Popularity Bias**: ...*

Prompt 4

Provide short hints for AI to try again according to each point of the previous step, without information leakage of the target item.

Particularly, LLMs can yield incorrect results in SBR due to various causes, which underscores the importance of having diverse and accurate hints to rectify the errors. To broaden the exploration of the diverse causes for the errors, we first only present the LLM with incorrectly predicted sessions and make it conduct self-reflection to identify possible causes for the errors without the ground truth item, as shown in Prompt 2. This allows the LLM to explore more diverse causes that may lead to errors. Then, to pursue more accurate causes for the errors, we further stimulate the LLM's analytical skills by revealing the ground truth item. Specifically, given the ground truth item, we ask the LLM to review previous diverse causes to select more relative ones, as shown in Prompt 3, thus identifying accurate causes of mistakes. Finally, for effective utilization in the next stage, we ask the LLM to summarize obtained causes as hints (known as specialized knowledge) in Prompt 4, which are more understandable for LLMs and easily improve the recommendation outcome. By following the provided reasoning chain (i.e., Prompts 2-4), the LLM can identify frequent and prominent errors in SBR inference, and then produce qualified hints to address potential errors effectively.

3.1.2 Automated Filtering. Recognizing that no single hint can rectify all the errors made by LLMs, we aim to build a hint knowledge base to store the most effective hints for further utilization, denoted as $\mathcal{H}$. To enhance the quality of the hints in $\mathcal{H}$, we develop an automated filtering strategy to maintain the hint knowledge base with two key properties: effectiveness and non-redundancy.

Prompt 5 (hint-enhanced prompt)

*I watched/purchased the following movies/items in order: {s}. Based on these interactions, recommend a movie/item for me to watch/purchase next from a candidate set: {C}. Please recommend from the candidate set. List the top 10 recommendations in numbered bullet points. **Hint:** {h} (retrieved hint).*

To ensure the effectiveness, we only add a new hint to $\mathcal{H}$ if it can lead to performance improvement. Specifically, we first construct a *hint-enhanced prompt* P^* (Prompt 5) to trigger the LLM with the hint-enhanced prompt for recommendation inference:

$$O^*(s) = LLM(P^*(s, C, h)), \tag{2}$$

where $O^*(s)$ is the recommendation list for session s based on the hint-enhanced prompt P^*. Then, we compare $O^*(s)$ with $O(s)$ obtained by Eq. 1 using the basic prompt without hint enhancement. The hint h can be recognized as effective if it leads to a better prediction by the LLM, i.e., $v^s_{l+1} \in O^*(s) \ \ \& \ \ v^s_{l+1} \notin O(s)$.

Prompt 6

'Does hint [h'] convey a similar idea for these hints: [h]? Return 1 if true, else return 0.'

Meanwhile, some hints may be redundant generated by LLMs, as multiple sessions may suffer from similar causes. For example, '*Consider the release years of movies in the user's history for era preference*' and '*Think about the production year of watched movies, recommend movies in that era from the candidate set*' share a similar semantic meaning. The redundancy of hints may result in a high cost of the maintenance and utilization of the hint knowledge base $\mathcal{H}$, e.g., larger size of the hint knowledge base and increased complexity in retrieving the targeted hint. To reduce the redundancy in the hint knowledge base $\mathcal{H}$, we only incorporate new hints that are distinct from existing ones. Specifically, we employ LLMs to detect the semantic similarity between the candidate hint h' and existing ones $\{h \in \mathcal{H}\}$, given by,

$$\sum\nolimits_{h \in \mathcal{H}} LLM(P_6(h', h)) == 0, \tag{3}$$

where the detail of P_6 is shown in Prompt 6, instructing the LLM to return 0 if h' shares a different idea with h and else return 1. By doing this, only distinct candidate hint h' will be incorporated into the knowledge base $\mathcal{H}$ if it satisfies Eq. 3.

Through the automated filtering process, we iteratively update the hint knowledge base until reaching capacity. It contains qualified hints to correct the errors across different session patterns and thus improve recommendation results. For a practical illustration, we randomly select five hints from the movie and the video game domains in the diagrams below.

Hint examples - Movie

- *Consider the release years of movies in the record for era preference.*
- *Pay attention to actors who appear in multiple movies from the watched list and consider other films featuring these actors.*
- *Aim for a personalized recommendation list that offers variety and reflects a range of moods and themes.*
- *Consider films that originated from other media, such as books, radio plays, or TV shows.*
- *Do not rely solely on mainstream popularity; consider critically acclaimed films that may not be box office hits.*

Hint examples - Video Game

- *Pay attention to the platforms prominently featured in the user's purchase history and include items that are relevant to those platforms.*
- *Give preference to games released in the era aligning with purchased games.*
- *Pay attention to the previous purchase history, which indicates an interest in gaming accessories and hardware enhancements.*
- *Ensure the games are suitable for a wide range of ages and have a universal appeal.*
- *Focus on items that specifically enhance or are used in conjunction with mobile devices.*

3.2 Reinforcement Utilization Module

To guide LLMs to infer more accurate SBR predictions, we propose to utilize the constructed hint knowledge base to prevent errors in the inference process of LLMs.

In practical scenarios, the absence of explicit labels on the hints' efficacy leads to a challenge in conducting supervision on hint selection, as computing the efficacy for all hints on each sample is costly and redundant. Fortunately, DRL with a replay buffer enables us to collect reward signals (i.e., hint efficacy) and update the network spontaneously to speed up. To this end, we employ the Proximal Policy Optimization (PPO) [33] algorithm to ensure stable and efficient training of our retrieval agent. Specifically, our agent is trained to select the most relevant hints based on session-related context information, thereby preventing LLMs from similar reasoning mistakes for recommendation. In addition, the proposed DRL framework allows for the balance between exploitation and exploration. It is also compatible with more complex extensions such as retrieving multiple hints and multi-round agent-based interactions.

3.2.1 Markov Decision Process (MDP). To outline the basics of our DRL environment, we define MDP as the tuple $MDP = (\mathcal{Z}, \mathcal{A}, R, T)$, where $\mathcal{Z}$, $\mathcal{A}$, R and T denote the state space, action space, reward function, and transition, respectively.

State. To model the session-related context information for our retrieval agent, we concatenate item titles and attributes into a single text string and convert it into embedding for semantic extraction. We use pre-trained BERT [5] as the text encoder due to its robust contextual language understanding capabilities. The state can be denoted as $\mathbf{z} = BERT(s)$, where $\mathbf{z} \in \mathcal{Z}$ is the d-dimensional output of the text encoder.

Action. To select relevant hints from the constructed knowledge base for LLMs' inference, we define a discrete action space for the agent, represented as $\mathbf{a} \in \mathcal{A}$, which is a $(|\mathcal{H}| + 1)$-dimensional vector. Here, $|\mathcal{H}|$ is the size of the knowledge base, and an action $\mathbf{a}$ corresponds to either choosing a hint or opting not to select any.

Reward. To direct the agent in making accurate hint selections, we employ a comparative function R to provide the reward signals for the agent's actions. This function evaluates the improvement in the LLM prediction accuracy by the hint-enhanced prompt (Prompt 5) versus the basic prompt (Prompt 1). For each episode, the reward value r is denoted as $r = m(O^*(s)) - m(O(s))$, where m is a recommendation evaluation metric (e.g., NDCG@10).

Transition. In each step, the agent can receive task-specific feedback by simulating the real-world RSs. The agent observes the state of the current session, takes action to select a hint for the LLM, and receives a reward from the environment. Subsequently, the environment transits to the next session's state, and then the transition is denoted as $T(\mathbf{z}) = \mathbf{z}'$. This setup can flexibly accommodate the scenario where multiple steps per session are involved for more sophisticated learning processes in our future studies.

Replay buffer. To facilitate efficiency in policy optimization, we maintain a replay buffer $D = (\mathbf{z}, \mathbf{a}, r, \mathbf{z}')$ to store the tuples of observation state, agent action, reward, and next observation state. With the records in the replay buffer, the retrieval agent can refine successful strategies and learn from erroneous trials. This technique

Algorithm 1: Re2LLM

Input: Training data $\mathcal{S}_{tr}$ for SBR, hint knowledge base size n_h.
// Reflective Exploration Module
while $|\mathcal{H}| < n_h$ **do**
  Sample s from S_{tr};
  if $m(O(s))$=0 // If incorrectly predicted **then**
    Obtain multiple hints by *Prompt 2 - 4*;
    for *each obtained hint* h **do**
      Assess two aspects of quality by automated filtering;
      if *True* **then**
        Add h to $\mathcal{H}$;
return $\mathcal{H}$
Input: Training data $\mathcal{S}_{tr}$ for SBR, constructed $\mathcal{H}$, max episode N.
Initialize policy network π_{θ};
// Reinforcement Utilization Module
for $i = 1, 2, ..., N$ **do**
  Shuffle $\mathcal{S}_{tr}$;
  for s *in* $\mathcal{S}$ **do**
    Encode $\mathbf{z} \in \mathcal{Z}$ based on context information of s;
    Sample action a by Eq. 4 and ϵ-greedy exploration;
    Compute reward by $r = m(O^*(s)) - m(O(s))$;
    Transit to the next state $\mathbf{z}'$;
    Store the tuple $(\mathbf{z}, \mathbf{a}, r, \mathbf{z}')$ into the replay buffer;
  Sample from the replay buffer, update policy by Eq. 5;
return π_{θ}

significantly accelerates the DRL training as the LLM backbone is relatively slow for inference and producing reward signals.

3.2.2 PPO Training. To model the actions of the retrieval agent, we implement a policy network parameterized by MLPs to define our agent's policy π_{θ}, which maps the environmental space $\mathcal{Z}$ to the action space $\mathcal{A}$:

$$\mathbf{a} = softmax(\mathbf{W} \cdot \mathbf{z}), \tag{4}$$

where $\mathbf{W} \in \mathbb{R}^{(|H|+1)\times d}$ is the learnable weight matrix, and $\mathbf{z} \in \mathbb{R}^d$ denotes the state of the current session. The action of the agent corresponds to the largest value among the softmax logits of $\mathbf{a}$.

During the training process, the retrieval agent adopts the ϵ-greedy ($\epsilon = 0.05$) strategy to explore the environment, with probability ϵ to take a random action while with probability $(1 - \epsilon)$ to exploit the learned policy π_{θ}. Our objective function for PPO training can be formulated to maximize the total reward, given by,

$$\mathcal{L}_{PPO} = \mathbb{E}\left[R(\mathbf{z}, a) - \beta KL(\pi_{\theta_{old}}, \pi_{\theta})\right], \tag{5}$$

where the first term is the reward measured by recommendation tasks, and the second term is the KL-divergence of policies with a coefficient β for the regulation in policy updates. In summary, we train a retrieval agent with task-specific feedback via DRL, which can select the appropriate action to improve LLM's performance through hint-enhanced prompts for better recommendations.

3.2.3 Retrieval-Enhanced LLMs for Recommendation. With the constructed knowledge base and the trained retrieval agent, we achieve more accurate recommendations by automatically selecting appropriate hints for prompt enhancement during inference. The final recommendation output $\hat{O}$ can be denoted as follows:

$$\hat{O} = LLM(P^*(s, C, \pi_{\theta}(\mathbf{z}_s))), \tag{6}$$

where $\pi_{\theta(\mathbf{z}_s)}$ corresponds to the selected hint h by the trained retrieval agent for session s with its state $\mathbf{z}_s$. The overall algorithm of the proposed method Re2LLM is shown in Algorithm 1.

Table 1: Statistics of datasets.

	Movie	Game
# Sessions	468,389	387,906
# Items	8,233	22,576
Avg. length	10.17	6.28
Utilized Item Side Information	title, genre, actor, year, country, director	title, category, tag, brand, description

4 EXPERIMENTAL RESULTS

In this section, we evaluate the effectiveness of the proposed method Re2LLM through comprehensive experiments and analysis[2]. We aim to answer the following research questions:

- **RQ1**: Whether the proposed method outperforms baseline methods, including the deep learning models and other LLM-based models, for SBR?
- **RQ2**: How can key components affect our proposed method? Specifically, how is the efficacy of the proposed Reflective Exploration Module and Reinforcement Utilization Module?
- **RQ3**: How do key hyper-parameters impact the performance of our proposed method?

4.1 Experimental Setup

4.1.1 Dataset. In this paper, we evaluate the proposed Re2LLM and baseline methods on two real-world datasets, namely Hetrec2011-Movielens and Amazon Game:

- Hetrec2011-Movielens[3] dataset contains user ratings of movies. It also contains the side information of movies, e.g., title, production year, and movie genre.
- Amazon Game[4] dataset is the 'Video Game' category of the Amazon Review Dataset, which is collected from the Amazon platform with users' reviews on various types of games and peripherals. It also contains metadata of games, e.g., title, brand, and tag.

For each user, we take all the interactions within one day as a session by the recorded timestamps. We filter out sessions and items that have less than 3 records and treat records with ratings as positive implicit feedback. Following the prior study [39], we adopt the data augmentation strategy to extend a session s with length $|s|$ to $(|s| - 1)$ sessions as training samples. The statistics of the datasets are summarized in Table 1.

4.1.2 Baselines. We compare our proposed method with eight state-of-the-art baseline methods[5], and the details are listed as follows. **FPMC** [32] combines matrix factorization with the first-order Markov chain. **GRU4Rec** [13] uses Gated Recurrent Unit (GRU), a typical RNN layer, to model whole sessions. **NARM** [22]

[2]Our source code is available at https://anonymous.4open.science/r/Re2LLM-34DC/.
[3]https://grouplens.org/datasets/hetrec-2011/
[4]https://jmcauley.ucsd.edu/data/amazon/
[5]Our method does not require fine-tuning, and due to the high computational cost, fine-tuning-based methods are not in our scope.

Table 2: Performance of all methods under two settings. The performance of our Re2LLM is in bold; the best performance achieved by baselines in full and few-shot settings are underlined with '_' and '‗', respectively. The improvements (Imp.) of our model against the best-performed baselines in both settings are reported, where * and † indicate statistically significant improvement by t-test (p < 0.05) for full* and few-shot† settings, respectively.

Setting	Model	Movie				Game			
		HR@5	HR@10	NDCG@5	NDCG@10	HR@5	HR@10	NDCG@5	NDCG@10
full dataset	FPMC	0.1492	0.2885	0.0922	0.1749	0.2463	0.3617	0.1788	0.2110
	GRU4Rec	0.2516	0.3861	0.1564	0.2103	0.2915	0.4107	0.2004	0.2392
	NARM	0.3867	0.5674	0.2605	0.3319	0.4743	0.5635	0.3732	0.4021
	SRGNN	0.3879	0.5720	0.2676	0.3342	0.4864	0.5335	0.3786	0.4128
	GCEGNN	0.3962	0.5895	0.2659	0.3281	0.4826	0.5523	0.4002	0.4332
	AttenMixer	0.3880	0.5727	0.2693	0.3472	0.4885	0.5838	0.3911	0.4226
	Imp.	4.14%*	0.14%	5.42%*	-3.39%	15.95%*	22.87%*	4.82%*	8.24%*
few-shot	NARM	0.2701	0.4412	0.1707	0.2298	0.1891	0.2765	0.1310	0.1586
	NARM-attr	0.3108	0.4835	0.1920	0.2481	0.1977	0.2915	0.1372	0.1672
	GCEGNN	0.3227	0.4813	0.2080	0.2584	0.1905	0.2798	0.1362	0.1659
	GCEGNN-attr	0.3254	0.4893	0.2136	0.2678	0.2019	0.3992	0.1408	0.1710
	AttenMixer	0.2774	0.4668	0.1774	0.2377	0.1899	0.2925	0.1357	0.1581
	AttenMixer-attr	0.3397	0.5056	0.2223	0.2795	0.2084	0.3121	0.1371	0.1727
	LLMRank	0.3649	0.5110	0.2436	0.2784	0.4947	0.6636	0.3898	0.4531
	NIR	0.3587	0.4901	0.2214	0.2648	0.5291	0.6434	0.4016	0.4346
	Re2LLM	**0.4126**	**0.5735**	**0.2839**	**0.3358**	**0.5664**	**0.7173**	**0.4195**	**0.4689**
	Imp.	13.07%†	12.23%†	16.54%†	20.14%†	7.05%†	8.09%†	4.46%†	3.49%†

introduces the attention mechanism into RNNs to better model the sequential behavior. **SRGNN** [47] constructs session graphs and employs GNN layers to obtain embeddings. **GCEGNN** [45] further includes global-level information as an additional graph to enhance the model performance. **AttenMixer** [57] achieves multi-level reasoning over item transitions by the attention-based readout method. **LLMRank** [16] is the LLM-based method, which demonstrates the promising zero-shot inference ability of LLMs through recency-focused prompting and in-context learning. **NIR** [42] is also LLM-based method that employs a multi-step prompt to capture user preferences first, then select representative movies, and finally perform the next-item prediction.

For a comprehensive comparison, we introduce two settings for the implementation of these baseline methods. In the *full dataset setting*, we investigate how Re2LLM performs in comparison with baseline methods trained on the full augmented dataset using item IDs. In the *few-shot setting*, we examine whether our Re2LLM shows superior performance compared to the baselines trained with limited data[6]. The '-attr' suffix denotes the incorporation of item attributes for deep learning-based baselines. We first concatenate all attributes (title included) of the item and then encode them into a text-aware embedding by BERT. Finally, we further aggregate the ID embedding and text-aware embedding of the item as its underlying representation.

4.1.3 Evaluation Strategy and Metrics. For each dataset, we apply the split-by-ratio strategy [38] for sessions to obtain training, validation, and test sets with a ratio of 7:1:2. For the full dataset setting, we use the entire training set. For the few-shot setting, we sample 500 training samples from the entire training set. In terms of evaluation, we consider the last item in each session as the target. Considering the cost and efficiency of LLMs, we randomly sample a subset of 2,000 sessions from the test set. For the efficiency of evaluation, we sampled 49 negative items for each positive item (i.e., target). In addition, we adopt two commonly used metrics for model evaluation, including normalized discounted cumulative gain ($NDCG@n$) and hit rate ($HR@n$), with $n = \{5, 10\}$. The results show the average of five runs with different random seeds.

4.1.4 Implementation Details. For a fair comparison, all methods are optimized by the Adam optimizer with the same embedding dimension 768 that aligns with the BERT encoder dimension[7]. Following [35], we employ the Optuna[8] framework to efficiently optimize the hyper-parameter of all methods. We conduct 20 trials on the following searching spaces, i.e., *learning rate*: $\{1e^{-2}, 1e^{-3}, 1e^{-4}\}$, *weight decay*: $\{1e^{-4}, 1e^{-6}, 1e^{-8}\}$, and *batch size*: {16, 64, 256}. For our method Re2LLM, we set candidate set size $|C|$ to 50, knowledge base size n_h to 20, and few-shot training size n_e to 500. The impacts of essential hyper-parameters on Re2LLM can be found in Section 4.4. For LLM-based methods, we use 'gpt-4' API as the backbone model for these methods. For the rest hyper-parameters in baseline methods, we follow the suggested settings in the original papers. The experiments are run on a single Tesla V100-PCIE-32GB GPU. On average, around 50 data samples are required to generate 20 hints. The average cost to analyze and conduct inference on each sample is around USD 0.015 during the hint generation, DRL training, and final testing.

4.2 Overall Comparison (RQ1)

Table 2 shows the performance of our method Re2LLM and various baseline methods under two evaluation settings, i.e., full dataset setting and few-shot setting. The experimental results lead to several key conclusions. **First**, our proposed method significantly outperforms baselines in few-shot settings, with average improvements

[6]We select more representative and stronger baselines for the few-shot setting and incorporate item attribute information in the few-shot setting only for a fair comparison.

[7]Empirically, using MLP to reduce dimension leads to a consistent performance drop.
[8]https://optuna.org/

Table 3: The results of ablation study across all datasets.

Models	Movie				Game			
	HR@5	HR@10	NDCG@5	NDCG@10	HR@5	HR@10	NDCG@5	NDCG@10
w/o-HE	0.3923	0.5458	0.2704	0.3198	0.5462	0.6771	0.4182	0.4550
w/o-AF	0.4013	0.5592	0.2774	0.3289	0.5652	0.6923	0.4146	0.4602
w/o-DRL(RAN)	0.3542	0.4960	0.2446	0.2908	0.5072	0.6345	0.3722	0.4107
w/o-DRL(ALL)	0.3910	0.5434	0.2746	0.3235	0.5638	0.6897	0.4164	0.4592
Re2LLM	**0.4126**	**0.5735**	**0.2839**	**0.3358**	**0.5664**	**0.7173**	**0.4195**	**0.4689**

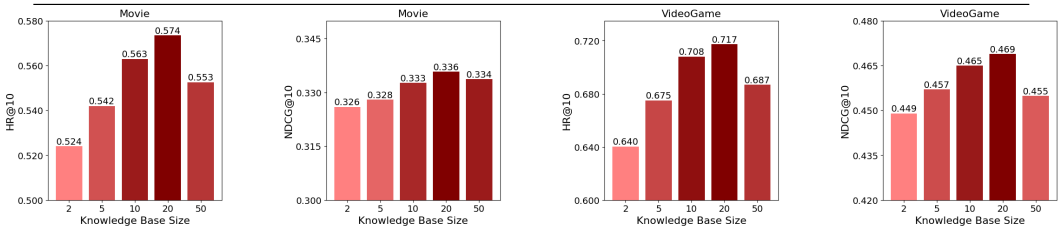

Figure 3: Performance of the proposed method with varying knowledge base size. Darker color indicates higher value.

of 15.49% and 5.77% across all metrics when compared to the top-performing baseline models on Movie and Game, respectively. This is mainly attributed to the effectiveness of Re2LLM, which not only extracts specialized and comprehensible knowledge by self-reflection but also effectively utilizes the knowledge based on a well-trained retrieval agent for better recommendations. **Second**, Re2LLM achieves comparable and even better performance than baseline methods trained on full datasets, which also verifies the effectiveness of our module design. Furthermore, the LLM-based methods (e.g., LLM-Ranker and NIR) also achieve comparable performance to baseline methods trained on full datasets, which indicates the promising way of employing LLMs as recommenders. **Third**, deep learning-based models show a significant performance drop in the few-shot setting compared with the full dataset setting, pointing to the data sparsity issue as a primary challenge. **Last**, the incorporation of item attributes slightly boosts the performance of baseline methods, showing the potential of using side information of items in addressing the sparsity issue in SBR.

4.3 Ablation Studies (RQ2)

To verify the impact of each component, we compare our method Re2LLM with its variants:

- w/o-HE: We only adopt the *basic prompt* (Prompt 1) to trigger the LLM to generate recommendations instead of Hint-Enhanced prompt as in Eq. 6.
- w/o-AF: We remove the Automated Filtering mechanism from the hint knowledge base construction, that is, incorporating all hints generated by LLMs into the hint knowledge base regardless of their quality.
- w/o-DRL: We remove the Deep Reinforcement Learning with task-specific feedback for the retrieval agent. Instead, we use two straightforward strategies for hint retrieval, namely RANdom and ALL retrieval strategies. Specifically, the variant w/o-DRL(RAN) randomly samples hints from the hint knowledge base for prompt enhancement, whereas the variant w/o-DRL(ALL) concatenates all hints in the knowledge base instead of the selective retrieval.

Table 3 shows the performance of all ablation studies. **First**, Re2LLM outperforms its variant w/o-HE, which indicates the necessity of activating LLM's reflection mechanism with hint enhancement in our Reflective Exploration Module. **Second**, Re2LLM outperforms variants w/o-DRL(RAN) and w/o-DRL(ALL), indicating the importance of our Reinforcement Utilization Module for the training of the retrieval agent with task-specific feedback signals. The variant w/o-DRL(RAN) has the worst performance by random sampling hints because inaccurate hints can mislead the LLM inference for recommendation, resulting in negative impacts. The variant w/o-DRL(ALL) shows limited improvement compared to the variant w/o-HE, revealing that simply feeding all hints into LLMs without session-aware selection is not optimal. **Third**, the variant w/o-AF achieves relatively better performance among all variants. Nevertheless, it still underperforms our Re2LLM due to the positive impact of the automated filtering mechanism on the hint knowledge base construction. In summary, the above results exhibit the efficacy of different modules in the proposed Re2LLM for more accurate SBR.

4.4 Hyper-parameter Analysis (RQ3)

We now examine the impact of several key hyper-parameters and designs on our proposed Re2LLM, including knowledge base size n_h, few-shot training size n_e, and reward design.

4.4.1 Knowledge Base Size. We investigate the correlation between model performance and the size of the knowledge base constructed by the Reflective Exploration Module. As shown in Fig. 3, there is an improvement in both evaluation metrics as the size of the knowledge base increases to 20. The improvement is attributed to the diversity of the generated reflective hints by covering a broader range of common errors of LLM inference on SBR. However, when the knowledge base size becomes too large, there is a slight performance drop due to the increased complexity and difficulty in the retrieval agent optimization. Thus, we suggest adopting a moderate hint knowledge base size.

4.4.2 Few-shot Training Size. We also study how model performance is affected by the number of few-shot training samples used for the retrieval model training. Fig. 4 shows the comparison of Re2LLM, its variant w/o-DRL(RAN), and representative baselines across various few-shot training sizes[9]. **First**, using too few samples

[9]We use previously mentioned data augmentation [39] for baselines. The results are from the Movie domain. Similar trends occur for the VideoGame domain.

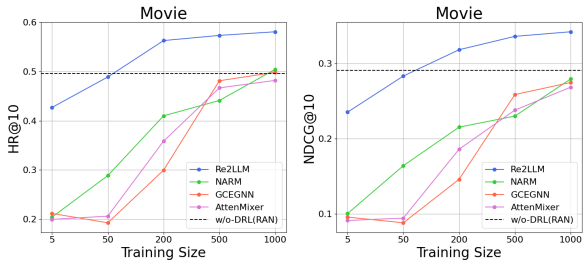

Figure 4: Performance of the representative baselines and our method with varying few-shot training size for retrieval agent. The dotted lines indicate w/o-DRL(RAN) performance.

(e.g., less than 50) for retrieval agent training results in inferior performance, which is even lower than w/o-DRL(RAN) with random hint selection. This is because the retrieval agent is under-trained with only very few samples. **Second**, as the number of training samples increases, the performance of our method increases consistently, showing the growth of the generalization ability of the retrieval agent. Baseline methods also improve, but remain inferior to our proposed method across various few-shot training sizes. **Third**, we also observe the limited improvement of Re2LLM when the number of samples exceeds a threshold. For the balance between effectiveness and efficiency, 500 is an appropriate selection for the few-shot training size of Re2LLM.

4.4.3 Reward Design. In Table 4, we report the experimental results using two distinct reward designs for the Reinforcement Retrieval Module: comparative reward and absolute reward. The comparative reward $R = m(O^*(s)) - m(O(s))$ measures the improvement achieved by the hint-enhanced prompt over the basic prompt. The absolute reward $R' = m(O^*(s))$ evaluates the performance of the hint-enhanced prompt alone. We find that the comparative reward yields superior performance compared to the absolute reward, as the comparative strategy focuses on improvement owing to the selected hint rather than its overall performance. As a result, we adopt the comparative reward for our Re2LLM.

Table 4: Performance of Re2LLM with two reward designs.

Reward	Movie		Game	
	HR@10	NDCG@10	HR@10	NDCG@10
absolute	0.5143	0.3147	0.7026	0.4542
comparative	**0.5735**	**0.3358**	**0.7173**	**0.4689**

4.5 Case Study (RQ2)

To further study the effectiveness of the retrieval agent, we compare Re2LLM, variant w/o-HE, and variant w/o-DRL(RAN) in two cases. Case 1 illustrates the scenario where the LLM backbone fails to hit the target item without a hint (i.e., the variant w/o-HE). Our trained retrieval agent in Re2LLM successfully retrieves the tailored hint and hits the target item at the third place of the top-10 recommendation list, demonstrating its ability to learn from task-specific feedback. However, the variant w/o-DRL(RAN) fails to replicate the correct prediction as a random hint may not be relevant to the session. Case 2 illustrates the scenario where the LLM backbone succeeds in hitting the target item without a hint (i.e., the variant w/o-HE). The trained agent in Re2LLM can further improve recommendation results by hitting the target item in a higher-ranking position (i.e., from the third place to the first place). On the contrary, a randomly selected hint by the variant w/o-DRL(RAN) shows a negative impact on the inference result, making the LLM miss the target item. Specifically, the session shows a preference for comedy, but the inappropriate hint instructs LLM's attention on the movie production years, thus misleading the LLM to generate incorrect predictions. These two cases reveal the necessity of effectively exploring and efficiently utilizing specialized knowledge (i.e., hints) for LLMs in SBR, validating our motivation and the efficacy of key modules in our Re2LLM.

Case Study 1: *Trained retriever*

***Prompt**: I watched the following movies in order: {Titanic; Leaving Las Vegas...}. Based on these interactions, recommend a movie for me to watch next from a candidate set: {C}. Please recommend from the candidate set. List the top 10 recommendations in numbered bullet points. Hint: {h}.*
***Answer by variant w/o-HE**, where h = '':*
1. The Day After Tomorrow; 2. The Bourne Identity; 3. Last Man Standing; 4. The Last Castle; 5. Resident Evil: Extinction; 6. Deep Blue Sea; 7. True Grit; 8. Courage Under Fire; 9. Kiss of the Dragon; 10. The Wrestler
***Answer by Re2LLM**, where **retrieved hint** h = **'Consider films that originated from other media, such as books, radio, or TV shows'**:*
*1. The Bourne Identity; 2. True Grit; **3. Aladdin**; 4. The Parent Trap; 5. The Picture of Dorian Gray; 6. Breakfast at Tiffany's; 7. The Jazz Singer; 8. The Last Castle; 9. The Stepfather; 10. Cujo*
***Answer by w/o-DRL(RAN)**, where **random hint** h = **'Pay attention to actors who appear in multiple movies from the watched list and consider other films featuring these actors'**:*
1. The Day After Tomorrow; 2. Taken; 3. The Bourne Identity; 4. Last Man Standing; 5. The Last Castle; 6. The Wrestler; 7. Redbelt; 8. The Stepfather; 9. Kiss of the Dragon; 10. Courage Under Fire

Case Study 2: *Random retriever*

***Prompt**: I watched the following movies in order: {Xizao; Hero at Large; ...} Based on these interactions, recommend a movie for me to watch next from a candidate set: {C}. Please recommend from the candidate set. List the top 10 recommendations in numbered bullet points. Hint: {h}.*
***Answer by variant w/o-HE**, where h = '':*
*1. Iceman; 2. Banlieue 13; **3. Siu lam juk kau** ; 4. Hot Fuzz; 5. King of New York; 6. Cidade dos Homens; 7. Blood Diamond; 8. Fantastic Four; 9. The Perfect Storm; 10. Chaplin*
***Answer by Re2LLM**, where **retrieved hint** h = **'Focus on films with a strong comedic element, particularly those that blend humor with other genres'**:*
***1. Siu lam juk kau**; 2. Iceman; 3. Banlieue 13; 4. Hot Fuzz; 5. Broken Flowers; 6. The Perfect Storm; 7. Fantastic Four; 8. Blood Diamond; 9. King of New York; 10. Chaplin*
***Answer by w/o-DRL(RAN)**, where **random hint** h = **'Consider the release years of movies in the user's history for era preference'**:*
1. Blood Diamond; 2. Hot Fuzz; 3. Banlieue 13; 4. Freedom Writers; 5. Fantastic Four; 6. 2 Fast 2 Furious; 7. The Perfect Storm; 8. Iceman; 9. Ice Station Zebra; 10. Exit Wounds 2001

5 CONCLUSION

In this paper, we propose Re2LLM, a Reflective Reinforcement LLM for SBR, aiming to improve performance by identifying and rectifying common errors in LLM inference. We present a novel learning paradigm that merges the capabilities of LLMs with adaptable model training procedures. Specifically, Re2LLM harnesses the self-reflection ability of LLMs to capture specialized knowledge and create a hint knowledge base with an automated filtering strategy.

Then, trained via DRL, a lightweight retrieval model learns to select proper hints guided by task-specific feedback to facilitate session-aware inference for better recommendation results. We demonstrate the effectiveness of our method through extensive experiments in two real-world datasets across two evaluation settings. In addition, ablation studies and hyper-parameter analysis further validate our underlying motivations and designs. In future work, the retrieval model could be extended with more flexible functionalities, such as integrating hints and multi-modal contextual information.

REFERENCES

[1] Keqin Bao, Jizhi Zhang, Yang Zhang, Wenjie Wang, Fuli Feng, and Xiangnan He. 2023. TALLRec: An effective and efficient tuning framework to align large language model with recommendation. In *Proceedings of the 17th ACM Conference on Recommender Systems (RecSys)*. 1007–1014.

[2] Jingfan Chen, Guanghui Zhu, Haojun Hou, Chunfeng Yuan, and Yihua Huang. 2022. AutoGSR: Neural Architecture Search for Graph-based Session Recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '22)*. 1694–1704.

[3] Qian Chen, Zhiqiang Guo, Jianjun Li, and Guohui Li. [n. d.]. Knowledge-enhanced Multi-View Graph Neural Networks for Session-based Recommendation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '23)*. 352–361.

[4] Sunhao Dai, Ninglu Shao, Haiyuan Zhao, Weijie Yu, Zihua Si, Chen Xu, Zhongxiang Sun, Xiao Zhang, and Jun Xu. 2023. Uncovering ChatGPT's Capabilities in Recommender Systems. In *Proceedings of the 17th ACM Conference on Recommender Systems (RecSys)*. 1126–1132.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, 4171–4186.

[6] Yingpeng Du, Di Luo, Rui Yan, Hongzhi Liu, Yang Song, Hengshu Zhu, and Jie Zhang. 2023. Enhancing job recommendation through llm-based generative adversarial networks. *arXiv preprint arXiv:2307.10747* (2023).

[7] Yingpeng Du, Ziyan Wang, Zhu Sun, Haoyan Chua, Hongzhi Liu, Zhonghai Wu, Yining Ma, Jie Zhang, and Youchen Sun. 2024. Large language model with graph convolution for recommendation. *arXiv preprint arXiv:2402.08859* (2024).

[8] Yunfan Gao, Tao Sheng, Youlin Xiang, Yun Xiong, Haofen Wang, and Jiawei Zhang. 2023. Chat-REC: Towards Interactive and Explainable LLMs-Augmented Recommender System. arXiv:2303.14524 [cs.IR]

[9] Shijie Geng, Shuchang Liu, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. 2022. Recommendation as language processing (RLP): a unified pretrain, personalized prompt & predict paradigm (P5). In *Proceedings of the 16th ACM Conference on Recommender Systems (RecSys)*. 299–315.

[10] Qilong Han, Chi Zhang, Rui Chen, Riwei Lai, Hongtao Song, and Li Li. 2022. Multi-Faceted Global Item Relation Learning for Session-Based Recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '22)*. 1705–1715.

[11] Zhankui He, Zhouhang Xie, Rahul Jha, Harald Steck, Dawen Liang, Yesu Feng, Bodhisattwa Prasad Majumder, Nathan Kallus, and Julian Mcauley. [n. d.]. Large Language Models as Zero-Shot Conversational Recommenders. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management (CIKM '23)*. 720–730.

[12] Balázs Hidasi and Alexandros Karatzoglou. 2018. Recurrent Neural Networks with Top-k Gains for Session-based Recommendations. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018, Torino, Italy, October 22-26, 2018*. ACM, 843–852.

[13] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. [n. d.]. Session-based recommendations with recurrent neural networks. In *4th International Conference on Learning Representations, ICLR 2016*.

[14] Yupeng Hou, Binbin Hu, Zhiqiang Zhang, and Wayne Xin Zhao. 2022. CORE: Simple and Effective Session-Based Recommendation within Consistent Representation Space *(SIGIR '22)*. 1796–1801.

[15] Yupeng Hou, Shanlei Mu, Wayne Xin Zhao, Yaliang Li, Bolin Ding, and Ji-Rong Wen. [n. d.]. Towards Universal Sequence Representation Learning for Recommender Systems. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22)*. 585–593.

[16] Yupeng Hou, Junjie Zhang, Zihan Lin, Hongyu Lu, Ruobing Xie, Julian McAuley, and Wayne Xin Zhao. 2023. Large language models are zero-shot rankers for recommender systems. arXiv:2305.08845

[17] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*.

[18] Jianchao Ji, Zelong Li, Shuyuan Xu, Wenyue Hua, Yingqiang Ge, Juntao Tan, and Yongfeng Zhang. 2023. GenRec: Large Language Model for Generative Recommendation.

[19] Wang-Cheng Kang, Jianmo Ni, Nikhil Mehta, Maheswaran Sathiamoorthy, Lichan Hong, Ed Chi, and Derek Zhiyuan Cheng. 2023. Do LLMs Understand User Preferences? Evaluating LLMs On User Rating Prediction. arXiv:2305.06474

[20] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42 (2009), 30–37.

[21] Siqi Lai, Erli Meng, Fan Zhang, Chenliang Li, Bin Wang, and Aixin Sun. 2022. An Attribute-Driven Mirror Graph Network for Session-Based Recommendation *(SIGIR '22)*. 1674–1683.

[22] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural attentive session-based recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management (CIKM)*. 1419–1428.

[23] Youhua Li, Hanwen Du, Yongxin Ni, Pengpeng Zhao, Qi Guo, Fajie Yuan, and Xiaofang Zhou. 2023. Multi-modality is all you need for transferable recommender systems. *ArXiv* abs/2312.09602 (2023).

[24] Yinfeng Li, Chen Gao, Hengliang Luo, Depeng Jin, and Yong Li. 2022. Enhancing hypergraph neural networks with intent disentanglement for session-based recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1997–2002.

[25] Qijiong Liu, Nuo Chen, Tetsuya Sakai, and Xiao-Ming Wu. 2024. ONCE: Boosting Content-based Recommendation with Both Open- and Closed-source Large Language Models. In *Proceedings of the Seventeen ACM International Conference on Web Search and Data Mining*.

[26] Qiao Liu, Yifu Zeng, Refuoe Mokhosi, and Haibin Zhang. 2018. STAMP: Short-term attention/memory priority model for session-based recommendation.. In *KDD*. ACM, 1831–1839.

[27] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Sean Welleck, Bodhisattwa Prasad Majumder, Shashank Gupta, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-Refine: Iterative Refinement with Self-Feedback. arXiv:2303.17651 [cs.CL]

[28] Wenjing Meng, Deqing Yang, and Yanghua Xiao. 2020. Incorporating User Micro-Behaviors and Item Knowledge into Multi-Task Learning for Session-Based Recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20)*. 1091–1100.

[29] Liangming Pan, Michael Saxon, Wenda Xu, Deepak Nathani, Xinyi Wang, and William Yang Wang. 2023. Automatically Correcting Large Language Models: Surveying the landscape of diverse self-correction strategies. arXiv:2308.03188 [cs.CL]

[30] Zhiqiang Pan, Fei Cai, Wanyu Chen, Honghui Chen, and Maarten de Rijke. 2020. Star graph neural networks for session-based recommendation. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management (CIKM)*. 1195–1204.

[31] Reid Pryzant, Dan Iter, Jerry Li, Yin Lee, Chenguang Zhu, and Michael Zeng. 2023. Automatic Prompt Optimization with "Gradient Descent" and Beam Search. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Singapore, 7957–7968.

[32] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized Markov chains for next-basket recommendation. In *Proceedings of the 19th International Conference on World Wide Web (WWW '10)*. 811–820.

[33] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. arXiv:1707.06347 [cs.LG]

[34] Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language Agents with Verbal Reinforcement Learning. arXiv:2303.11366 [cs.AI]

[35] Zhu Sun, Hui Fang, Jie Yang, Xinghua Qu, Hongyang Liu, Di Yu, Yew-Soon Ong, and Jie Zhang. 2022. DaisyRec 2.0: Benchmarking Recommendation for Rigorous Evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2022).

[36] Zhu Sun, Qing Guo, Jie Yang, Hui Fang, Guibing Guo, Jie Zhang, and Robin Burke. 2019. Research commentary on recommendations with side information: A survey and research directions. *Electronic Commerce Research and Applications* 37 (2019), 100879.

[37] Zhu Sun, Hongyang Liu, Xinghua Qu, Kaidong Feng, Yan Wang, and Yew-Soon Ong. 2023. Large Language Models for Intent-Driven Session Recommendations. arXiv:2312.07552 [cs.CL]

[38] Zhu Sun, Di Yu, Hui Fang, Jie Yang, Xinghua Qu, Jie Zhang, and Cong Geng. 2020. Are We Evaluating Rigorously? Benchmarking Recommendation for Reproducible Evaluation and Fair Comparison. In *Proceedings of the 14th ACM Conference on Recommender Systems*.

[39] Yong Kiam Tan, Xinxing Xu, and Yong Liu. 2016. Improved recurrent neural networks for session-based recommendations. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*. Association for Computing Machinery, 17–22.

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

Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMA: Open and Efficient Foundation Language Models. arXiv:2302.13971

[41] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc.

[42] Lei Wang and Ee-Peng Lim. 2023. Zero-Shot Next-Item Recommendation using Large Pretrained Language Models. arXiv:2304.03153

[43] Shoujin Wang, Liang Hu, Yan Wang, Quan Z. Sheng, Mehmet Orgun, and Longbing Cao. 2019. Modeling multi-purpose sessions for next-item recommendations via mixture-channel purpose routing networks. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. 3771–3777.

[44] Yu Wang, Zhiwei Liu, Jianguo Zhang, Weiran Yao, Shelby Heinecke, and Philip S. Yu. 2023. DRDT: Dynamic Reflection with Divergent Thinking for LLM-based Sequential Recommendation. arXiv:2312.11336

[45] Ziyang Wang, Wei Wei, Gao Cong, Xiao-Li Li, Xian-Ling Mao, and Minghui Qiu. 2020. Global context enhanced graph neural networks for session-based recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 169–178.

[46] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In *Advances in Neural Information Processing Systems*, Vol. 35. Curran Associates, Inc., 24824–24837.

[47] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-based recommendation with graph neural networks. In *Proceedings of the Thirty-Third AAAI Conference*.

[48] Yueqi Xie, Peilin Zhou, and Sunghun Kim. 2022. Decoupled Side Information Fusion for Sequential Recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '22)*. 1611–1621.

[49] Chengfeng Xu, Pengpeng Zhao, Yanchi Liu, Victor S. Sheng, Jiajie Xu, Fuzhen Zhuang, Junhua Fang, and Xiaofang Zhou. 2019. Graph contextualized self-attention network for session-based recommendation. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI)*. 3940–3946.

[50] Fan Yang, Zheng Chen, Ziyan Jiang, Eunah Cho, Xiaojiang Huang, and Yanbin Lu. 2023. PALR: Personalization aware LLMs for recommendation. arXiv:2305.07622

[51] Heeyoon Yang, YunSeok Choi, Gahyung Kim, and Jee-Hyong Lee. 2023. LOAM: Improving Long-tail Session-based Recommendation via Niche Walk Augmentation and Tail Session Mixup *(SIGIR '23)*. 527–536.

[52] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of Thoughts: Deliberate Problem Solving with Large Language Models. arXiv:2305.10601 [cs.CL]

[53] Weiran Yao, Shelby Heinecke, Juan Carlos Niebles, Zhiwei Liu, Yihao Feng, Le Xue, Rithesh Murthy, Zeyuan Chen, Jianguo Zhang, Devansh Arpit, Ran Xu, Phil Mui, Huan Wang, Caiming Xiong, and Silvio Savarese. 2023. Retroformer: Retrospective Large Language Agents with Policy Gradient Optimization. arXiv:2308.02151 [cs.CL]

[54] Feng Yu, Yanqiao Zhu, Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2020. TAGNN: Target attentive graph neural networks for session-based recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1921–1924.

[55] Zhenrui Yue, Sara Rabhi, Gabriel de Souza Pereira Moreira, Dong Wang, and Even Oldridge. 2023. LlamaRec: Two-Stage Recommendation using Large Language Models for Ranking. arXiv:2311.02089 [cs.IR]

[56] Jizhi Zhang, Keqin Bao, Yang Zhang, Wenjie Wang, Fuli Feng, and Xiangnan He. 2023. Is ChatGPT fair for recommendation? Evaluating fairness in large language model recommendation. In *Proceedings of the 17th ACM Conference on Recommender Systems (RecSys)*. 993–999.

[57] Peiyan Zhang, Jiayan Guo, Chaozhuo Li, Yueqi Xie, Jae Boum Kim, Yan Zhang, Xing Xie, Haohan Wang, and Sunghun Kim. 2023. Efficiently leveraging multi-Level user intent for session-based recommendation via atten-mixer network. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining (WSDM)*. 168–176.

[58] Xiaokun Zhang, Bo Xu, Fenglong Ma, Chenliang Li, Liang Yang, and Hongfei Lin. 2023. Beyond Co-occurrence: Multi-modal Session-based Recommendation. *IEEE Transactions on Knowledge and Data Engineering* (2023), 1–12. https://doi.org/10.1109/TKDE.2023.3309995

[59] Xiaokun Zhang, Bo Xu, Liang Yang, Chenliang Li, Fenglong Ma, Haifeng Liu, and Hongfei Lin. 2022. Price DOES Matter! Modeling Price and Interest Preferences in Session-Based Recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '22)*. 1684–1693.

[60] Yuhui Zhang, Hao Ding, Zeren Shui, Yifei Ma, James Zou, Anoop Deoras, and Hao Wang. 2021. Language models as recommender systems: Evaluations and limitations. In *NeurIPS 2021 Workshop on I (Still) Can't Believe It's Not Better*.