Large Language Models are Learnable Planners for Long-Term Recommendation

Wentao Shi
University of Science and Technology of China
Hefei, China
shiwentao123@mail.ustc.edu.cn

Xiangnan He*
University of Science and Technology of China
Hefei, China
xiangnanhe@gmail.com

Yang Zhang
University of Science and Technology of China
Hefei, China
zy2015@mail.ustc.edu.cn

Chongming Gao
University of Science and Technology of China
Hefei, China
chongming.gao@gmail.com

Xinyue Li
University of Science and Technology of China
Hefei, China
lrel7@mail.ustc.edu.cn

Jizhi Zhang
University of Science and Technology of China
Hefei, China
cdzhangjizhi@mail.ustc.edu.cn

Qifan Wang
Meta AI
Menlo Park, USA
wqfcr@fb.com

Fuli Feng*
University of Science and Technology of China
Hefei, China
fulifeng93@gmail.com

ABSTRACT

Planning for both immediate and long-term benefits becomes increasingly important in recommendation. Existing methods apply Reinforcement Learning (RL) to learn planning capacity by maximizing cumulative reward for long-term recommendation. However, the scarcity of recommendation data presents challenges such as instability and susceptibility to overfitting when training RL models from scratch, resulting in sub-optimal performance. In this light, we propose to leverage the remarkable planning capabilities over sparse data of Large Language Models (LLMs) for long-term recommendation. The key to achieving the target lies in formulating a guidance plan following principles of enhancing long-term engagement and grounding the plan to effective and executable actions in a personalized manner. To this end, we propose a Bi-level Learnable LLM Planner framework, which consists of a set of LLM instances and breaks down the learning process into macro-learning and micro-learning to learn macro-level guidance and micro-level personalized recommendation policies, respectively. Extensive experiments validate that the framework facilitates the planning ability of LLMs for long-term recommendation. Our code and data can be found at https://github.com/jizhi-zhang/BiLLP.

CCS CONCEPTS

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

KEYWORDS

Large Language Model, LLM Planner, Long-term Engagement

*Corresponding author.

SIGIR '24, July 14–18, 2024, Washington, DC, USA

ACM ISBN 979-8-4007-0431-4/24/07
https://doi.org/10.1145/3626772.3657683

ACM Reference Format:
Wentao Shi, Xiangnan He, Yang Zhang, Chongming Gao, Xinyue Li, Jizhi Zhang, Qifan Wang, and Fuli Feng. 2024. Large Language Models are Learnable Planners for Long-Term Recommendation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '24), July 14–18, 2024, Washington, DC, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3626772.3657683

1 INTRODUCTION

Recommendation systems have gained widespread adoption in contemporary society to alleviate the overwhelming burden of information overload [5]. Traditionally, researchers primarily focused on optimizing users' immediate responses (*e.g.* clicks) to maximize instant benefits [51]. However, such a greedy recommendation strategy tends to cater to users' immediate interests excessively, neglecting long-term engagement [44] and even influencing the ecology negatively. For instance, some users will be confined within an echo chamber of preferred information and filter bubbles [17]. Therefore, it is essential to investigate long-term recommendation.

To tackle this challenge, it is crucial to integrate planning capabilities into the recommendation decision-making process to develop policies that take into account not only immediate benefits but also long-term consequences. Existing work primarily employs Reinforcement Learning [7, 31, 48, 63] to acquire planning capabilities implicitly through training models from scratch with the objective of maximizing cumulative rewards. However, these approaches are entirely data-driven, and their efficacy is significantly constrained by the quality and quantity of available data [15, 17, 38]. Unfortunately, recommendation data is typically sparse and naturally long-tail distributed [6]. This poses a significant challenge for RL to acquire planning ability, particularly for sparse or long-tail items and users, resulting in sub-optimal performance.

LLMs have emerged with powerful planning capabilities through pre-training on massive and diverse textual data [1, 36, 42]. Previous studies have demonstrated that LLMs can break down complex textual and agent tasks into subtasks and then execute them sequentially [20, 21, 37, 45]. By conceptualizing multi-round recommendations as analogous to such complex tasks, there is potential to harness the planning prowess of LLMs to devise a multi-round recommendation policy aimed at maximizing long-term engagement.

Upon realization, benefitting from the inherent extensive world knowledge and robust reasoning capabilities in LLMs, it is anticipated to obtain superior planning capabilities even in scenarios with sparse recommendation data, especially for long-tail items.

To achieve the target, the key is to recall the task-solving principles to formulate a plan and make it effective and executable for individual users. However, direct acquisition of such planning capabilities is non-trivial, due to the substantial scenario divergence between LLM pre-training and recommendation. In the realm of recommendation tasks, the LLM itself may not naturally exhibit an inherent understanding (or commonsense) of the principles that enhance long-term engagement. Additionally, when tailoring recommendations for individual users, a personalized and item-specific strategy becomes essential, far beyond the mere awareness of such guiding principles. It is necessary to inspire or teach the LLM to acquire the desired principles and make them personalized.

We propose a novel **Bi**-level **L**earnable **L**LM **P**lanning (BiLLP) framework for long-term recommendation. BiLLP breaks down the learning process into macro-learning and micro-learning through a hierarchical mechanism. Macro-learning, aiming at acquiring high-level guiding principles, includes a Planner and Reflector, both implemented as LLM instances. The Planner leverages memorized high-level experiences that imply guiding principles to formulate high-level plans for long-term goals, while the Reflector reflects on the finished trajectory to gather new experiences for updating the Planner. Micro-learning includes the LLM-based Actor-Critic component to acquire planning personalization. The Actor personalizes high-level plans into executable actions for users. The Critic functions similarly to the Reflector but operates on a more fine-grained level. It can promptly evaluate the long-term *advantage* of an action given a state, facilitating the swift update of the Actor policy and mitigating high-variance issues in Q-values [9].

The main contributions of this work are summarized as follows:

- We introduce the idea of exploring the planning ability of LLMs with a bi-level planning scheme to enhance long-term engagement in recommendation.
- We propose a new BiLLP framework with four modules, which learns the planning ability at both macro and micro levels with low variance estimations of Q-values.
- We conduct extensive experiments, validating the capability of LLMs to plan for long-term recommendation and the superiority of the BiLLP framework.

2 RELATED WORK

• **Interactive Recommendation.** Interactive recommendation is a typical setting to study long-term recommendation, where a model engages in online interactions with a user [16, 48]. In contrast to the static recommendation setting, where the focus is on identifying "correct" answers within a test set, interactive recommendation assesses the efficacy of results by accumulating rewards obtained throughout the interaction trajectories. To improve the performance of interactive recommendation, extensive effort [10, 23, 62] has been made to model the recommendation environment as a Markov decision process (MDP) and then utilize advanced RL algorithms to deliver the optimal policy [7, 31, 48, 63]. CIRS [17] learns a causal user model on historical data to capture the overexposure effect of items on user satisfaction, facilitating the planning of the RL policy. DORL [15] alleviates Matthew Effect of Offline RL to improve long-term engagement. However, these RL-based methods exhibit suboptimal learning efficiency and poor planning performance when confronted with sparse recommendation data.

• **LLM for Recommendation.** LLM-based Recommendation paradigm has achieved remarkable advancements [11, 29, 50] owing to the extraordinary abilities of LLMs such as GPT4 [1] and Llama2 [42]. Distinguishing from existing LLM-based recommendation methods that are limited to directly using in-context learning [8, 19, 32, 47, 52, 58] or tuning [3, 26, 30, 49, 59, 60] for immediate response in the recommendation, our proposed BiLLP delves deeply into how the powerful planning ability of LLMs can empower the long-term engagement of recommendation systems. Some other approaches attempt to explore LLMs' planning capability in managing API tools [12, 22, 43] for recommendation, but they are also restricted to immediate responses and lack focus on users' long-term engagement. In contrast to the previous approach of LLM-based recommendation, our proposed BiLLP deeply harnesses the planning capabilities of LLM and utilizes it to enhance long-term engagement for users which is particularly challenging to optimize in traditional recommendations.

• **LLM Planner.** After undergoing pre-training and instruction tuning, the LLMs have attained extensive world knowledge and proficient planning capabilities. Recent work [20, 21, 37, 45] exploit these powerful capabilities to generate better control plans for robots and agents. ReAct [55] effectively integrates the action decision with planning and results in promising performance. SwiftSage [28] integrates fast and slow thinking to solve complex tasks. However, these methods lack the ability to learn from past experiences, which allows for better task planning. To enable self-improvement without fine-tuning, Reflexion [40] verbally reflects on task feedback signals. ExpeL [61] utilize cross-task persistent memory to store insights and trajectories. AdaPlanner [41] can learn from past failure, past success, or both. In addition to these macro-level refinements, some work [4, 39, 57] integrates LLMs with RL algorithm to learn from the micro-level interaction experiences. However, they suffer from the issues of high variance estimations of Q-values, which could be alleviated by our proposed *Critic module*.

3 PRELIMINARY

• **Problem Definition.** Following recent work on long-term recommendation [17], we adopt the interactive recommendation setting. The target is to learn a recommendation model that recommends items $i \in \mathcal{I}$ (*i.e.*, makes an action[1] a_n) to a user $u \in \mathcal{U}$ at each step n based on the current state[2] s_n. As to applying LLMs for interactive recommendation, the recommendation process at each step n involves two main operations: generating a problem-solving plan, referred to as thought t_n, and subsequently providing an item recommendation, denoted as action a_n. Based on this, the entire interaction episode can be denoted as

$$\mathcal{H}^{1\cdots N} = \{s_1, t_1, a_1, r_1, \cdots, s_N, t_N, a_N, r_N\}, \tag{1}$$

[1] As an initial attempt, we constrain each action a_n to recommend only one item.
[2] s_1 is the initial state before interacting with the model.

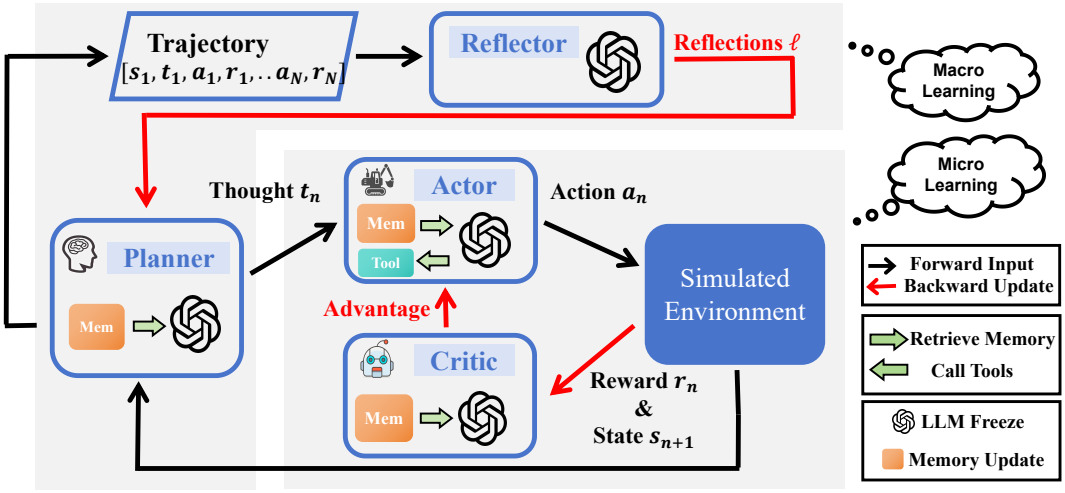

Figure 1: The overview of the proposed BiLLP framework. The black line indicates that the data serves as a prompt input for the subsequent module. The red line denotes that the data is utilized to update the memory of the subsequent module.

and the trajectory $\mathcal{H}^{1\cdots n}, (1 \leq n \leq N)$ can be thought of as a subsequence of an episode.

• **Simulated Environment.** The interactive recommendation setting requires immediate user feedback for recommendation actions. Collecting online feedback from users can be financially burdensome, we thus follow [15, 17] and construct "Simulated Environment" with offline data for both model learning and testing. This environment can mimic users' behaviors in online scenarios, accept recommendations (*i.e.* action a) from the model, and provide feedback (*i.e.*, reward r) accordingly.

4 METHOD

In this section, we present the proposed BiLLP framework for improving long-term engagement in the interactive recommendation. As shown in Figure 1, the recommendation process of our framework involves two main steps:

- The Planner generates problem-solving plans (*i.e.*, thoughts t), where the recommendation task is broken into sequential step-by-step sub-plans, striking a harmonious balance between exploration and exploitation.
- The Actor recommends items (*i.e.*, takes actions a) to the user by incorporating both macro-level sub-plans (thoughts) and micro-learning experiences.

To generate appropriate plans and personalized item recommendations, the key lies in teaching LLMs to learn from past interaction episodes. To enhance the learning process, the BiLLP framework employs a hierarchical mechanism (See Figure 1):

- **Macro-learning** involves the Planner and Reflector to generate more appropriate plans, where the Reflector extracts high-level guiding principles from historical episodes and incorporates them into the input of Planner to enhance the quality of plans.
- **Micro-learning** involves the Actor and Critic to generate more personalized recommendations, where Critic assesses the user's current satisfaction level (action advantage value) and updates the policy of Actor to enhance personalized recommendations.

4.1 Macro-Learning

The macro-learning refers to a process in which the Reflector generates reflections based on historical episodes and subsequently updates them into the memory of the Planner. The Planner then retrieves the most relevant reflections from the memory and utilizes them as prompts to enhance the quality of plan generation. Next, we present the details of the Reflector and Planner, and the procedure of the micro-learning process.

4.1.1 Reflector. The Reflector is designed to extract guiding principles from historical episode data. When a user ends his interaction with the model, we utilize this complete interaction episode $\mathcal{H}_c^{1\cdots N}$ as input, and then generate reflections ℓ as follows:

$$\ell_c = \textbf{Reflector}(\mathcal{H}_c^{1\cdots N}). \tag{2}$$

We implement the Reflector as an LLM instance. Based on the predefined instruction prompt and few-shot examples $\mathcal{P}_R$, the reflection ℓ generation process can be formulated as:

$$\ell_c = \text{LLM}(\mathcal{P}_R, \mathcal{H}_c^{1\cdots N}). \tag{3}$$

The obtained reflections are then used to update the memory $\mathcal{M}_P$ in the Planner, denoted as $\ell_c \rightarrow \mathcal{M}_P$. To facilitate understanding, we provide an example of reflection in Table 1, which primarily covers two high-level aspects: *analysis of withdrawal reasons* and *prospective guidance*. Specifically, in the example, the users' disengagement is identified as stemming from the repetitive recommendation of identical items, and the guiding principle for future recommendations emphasizes prioritizing diversity. Both aspects do not involve specific items.

4.1.2 Planner. The Planner module is designed to generate forward-looking plans and decompose the high-level plan into sub-plans, indicated in outputted thoughts, where thoughts facilitate the Actor to execute actions. The Planner is implemented as a frozen LLM instance equipped with a memory library $\mathcal{M}_P$ storing past reflections for reference. At each step n of a new episode, the Planner utilizes the historical trajectory $\mathcal{H}^{1\cdots n-1}$ and the current state s_n from the environment as input to generate the thought t_n with reflections

Table 1: Example of reflections.

Reflection Case 1
The user became dissatisfied with the final recommendation, which was a repeat of a previously recommended game. This suggests that the user may have been looking for more variety in their recommendations. In the future, it would be beneficial to avoid repeating recommendations and instead focus on providing a diverse range of games across different genres.

obtained from the memory:

$$t_n = \textbf{Planner}(\mathcal{H}^{1\cdots n-1}, s_n; \mathcal{M}_P), \tag{4}$$

where the $\mathcal{M}_P$ is a set of past episode reflections. Formally, we have $\mathcal{M}_P = \{\ell_m | m = 1, 2, ...\}$. When starting a new interaction process, meaning a new episode, multiple relevant reflections $\ell^K_{\mathcal{M}_P}$ are retrieved from the memory library $\mathcal{M}_P$ as guidance for generating new thoughts. For the following steps in the episode, we utilize the same reflections and other inputs to prompt LLM to generate thoughts. We next introduce these parts.

•**Reflection retrieval**. To ensure that the retrieved reflections are helpful for planning. We select K reflections with a minimal distance to this planning process. Taking the initial state s_1 to represent the process for distance computation, we have

$$\ell^K_{\mathcal{M}_P} = \{\ell | rank(d(\ell, s_1)) < K, \ell \in \mathcal{M}_P\}, \tag{5}$$

where $d(\cdot, \cdot)$ is defined as the Euclidean distance between the two encoded textsimplemented by *Facebook AI Similarity Search (FAISS)* [24], a library that allows us to quickly search for similar documents. $rank(\cdot)$ gets the rank of a value in ascending order.

•**Thought generation**. We leverage the macro-level guidance from the memory to generate a thought. For each input $(\mathcal{H}^{1\cdots n-1}, s_n)$, we can sample a thought from LLM policy as follows:

$$t_n \sim \text{LLM}(\mathcal{P}_P, \ell^K_{\mathcal{M}_P}, \mathcal{H}^{1\cdots n-1}, s_n). \tag{6}$$

Here, t_n is a sample from the Planner policy, the arguments in the function LLM$(\cdot)$ represent the prompt input to the LLM including task instruction of the Planner $\mathcal{P}_P$ (including few-shot examples), retrieved reflections $\ell^K_{\mathcal{M}_P}$, state s_n, and historical trajectory $\mathcal{H}^{1\cdots n-1}$ in the current episode.

Table 2 presents examples of our input prompt template and two representative thoughts, encapsulating common outputs of generated thoughts. In the input prompt, we integrate historical interaction sequences and reflections, prompting the LLM to generate appropriate thoughts for guiding subsequent actions. In these examples, we could find some interesting properties of the generated thoughts. In the case of thought example 1, we observe that LLM can analyze users' interests and decompose multiple rounds of recommendation tasks into distinct sub-plans. By leveraging its planning capabilities, the LLM can generate suggestions extending beyond immediate choices, considering their potential long-term impact on user satisfaction. This enables the method to take into account various factors to optimize user long-term engagement and satisfaction. In thought example 2, we note that LLMs can adhere to the previous plan, maintaining the consistency and continuity of the recommendation strategy.

4.1.3 Update. The macro-learning involves updating the Planner during the training. After an episode is completed, we update the Planner module by injecting the new reflections for this episode into its memory. The Planner memory update can be formulated as

$$\mathcal{M}_P \leftarrow \ell_c, \tag{7}$$

where ℓ_c denotes the reflections of the complete episode.

4.2 Micro-Learning

The micro-learning refers to a process in which the Actor grounds the thoughts into executable actions to environments and the Critic provides evaluations of these actions. By updating the policy of Actor based on the feedback of Critic and updating the policy of Critic based on the feedback of the environment, Actor and Critic learn to provide personalized recommendations in specific situations. The learning mechanism is similar to the Planner-Reflector but operates in a more granular dimension, *i.e.*, directly considering the recommendation of items. In essence, the micro-learning process bears analogies to the Advantage Actor-Critic (A2C) algorithm [33]. In the following, we first introduce the details of the Actor and Critic modules and present the procedure of the micro-learning process.

4.2.1 Actor. The Actor module aims to customize high-level plans into executable actions for each users. As illustrated in Figure 1, similar to the Planner module, we implement it as an LLM instance equipped with a memory $\mathcal{M}_A$ storing micro-level experiences. Additionally, considering that some knowledge is valuable for personalization but challenging for LLMs to handle [2], we add a tool library denoted as Tl to access such knowledge. At each step n of an episode, the actor utilizes the historical trajectory $\mathcal{H}^{1\cdots n-1}$, the current state s_n, and the corresponding thought t_n from the Planner module as inputs to generate an executable action a_n with knowledge obtained from the memory and the tool library as follows:

$$a_n = \textbf{Actor}(\mathcal{H}^{1\cdots n-1}, s_n, t_n; \mathcal{M}_A, Tl).$$

Here, the memory $\mathcal{M}_A$ can be formulated as a set of micro-level experiences, where the m-th experience is a previous interaction record, including three factors: state s_m, action a_m, and corresponding value v_m. Formally, we have $\mathcal{M}_A = \{(s_m, a_m, v_m) | m = 1, 2, \dots\}$.

Upon receiving these inputs, each generation process comprises three operations: 1) retrieving valuable experiences from the memory $\mathcal{M}_A$, 2) utilizing the tools to gather valuable statistical information of the current state, and 3) integrating the results of the first two steps and other inputs to prompt LLM to generate an action. We next elaborate on these operations.

• **Retrieval.** Similar to the retrieval operation in the Planner module, we rely on the similarity between the experience and input to select valuable experiences from the memory. Specifically, we leverage the distance between the state of an experience and the input state to measure the similarity, and we select all experiences with distances smaller than a threshold τ_A. The process can be formulated as follows:

$$\Psi^n_A = \{(s_m, a_m, v_m) | d(s_m, s_n) < \tau_A, s_m \in \mathcal{M}_A\}, \tag{8}$$

where Ψ^n_A denotes the retrieved results, and $d(\cdot, \cdot)$ is the same to that in Equation (5).

• **Tool analysis.** We utilize the tools in the tool library Tl to analyze users' interaction history, extracting valuable information that is

Table 2: Example of the input and output for the Planner module.

Instruction Input	
Instruction:	Solve a recommendation task with interleaving Thought, Action, and Observation steps. Thought can reason about the current situation and current user interest. Your goal is to meet the user's interest as much as possible and make recommendations to users as many times as possible. Note that if the user is not satisfied with your recommendations, he will quit and not accept new recommendations. You may take as many steps as necessary. Here are some examples: <Few-shot Examples> (END OF EXAMPLES) Reflection: {**Reflections** $\ell^K_{\mathcal{M}_P}$} {**Historical interaction sequence** $\mathcal{H}^{1\cdots n-1}$}
Output: Thought	
Case 1:	"The user seems to enjoy a mix of Action and Independent video games. They also seem to appreciate Adventure games. I would first recommend the user their favorite action games, and then recommend some other niche genre games that they like."
Case 2:	The user seems to be satisfied with the recommendations so far. Following the previous plan, I should recommend some other niche genre games that they like, such as RPG games..

challenging for the LLM to handle. In this study, we primarily focus on leveraging the *Category Analysis Tool*. At the n-th step, given the state s_n, the tool can identify a list of categories associated with each legal action and conduct statistical analysis on the user's viewing history. Formally,

$$O_n = Tl(s_n), \tag{9}$$

where O_n denotes the tool output in text format. Notably, the methodology described here can be adapted and applied to various other tools.

• **Action generation.** We leverage both the guidance from the Planner module, micro-level knowledge obtained from the memory and tool to prompt the LLM of the Actor module to generate an action. For the input $(\mathcal{H}^{1\cdots n-1}, s_n, t_n)$ with t_n representing the thought, once we obtain the corresponding retrieval results Ψ^n_A and tool analysis result O_n, we can sample an action a'_n from the LLM policy as follows:

$$a'_n \sim \text{LLM}(\mathcal{P}_A, \Psi^n_{\mathcal{M}_A}, O_n, \mathcal{H}^{1\cdots n-1}, s_n, t_n), \tag{10}$$

where $\mathcal{P}_A$ represents the task instruction for the Actor. Note that the temperature coefficient of the LLM should be set to a non-zero value, ensuring the generation of non-deterministic results.

Item grounding. The final action should be specific to an item within the candidate pool. Note that the LLM may generate items a'_n that are not necessarily included in the pool. To address this issue, we adopt the grounding strategy from [2] to map a'_n to an actual item with the highest similarity. Formally, the final action is obtained as follows:

$$a_n = \arg\min_{a \in \mathcal{I}} \text{sim}(\mathbf{e}_a, \mathbf{e}_{a'_n}), \quad \text{sim}(\mathbf{e}_a, \mathbf{e}_{a'_n}) := \|\mathbf{e}_a - \mathbf{e}_{a'_n}\|, \tag{11}$$

where $\mathbf{e}_a$ represents the embedding of the action (item) α encoded by *Llama2-7b* [42], $\text{sim}(\cdot,\cdot)$ denotes the embedding similarity measured by the *L2* distance, and $\|\cdot\|$ signifies the L_2 norm.

4.2.2 Critic. The Critic module is an LLM-based evaluator, providing evaluative feedback on the long-term goals for the actions generated by the Actor module to help update the policy of Actor. The Critic module also contains a memory $\mathcal{M}_C$ to store previous experiences. Inspired by the A2C algorithm, we utilize the advantage value v_n of action a_n in the given state s_n as the measurement. In particular, Critic takes the state s_n, action a_n, and the history trajectory $\mathcal{H}^{1\ldots n-1}$ as inputs and then outputs the advantage v_n with the experiences in $\mathcal{M}_C$ as references, which can be abstracted as follows:

$$v_n = \textbf{Critic}(s_n, a_n; \mathcal{M}_C). \tag{12}$$

To compute advantage values, similar to A2C, we first estimate the state-value function $V(s_n)$, and then, based on it, we use the advantage function [33] to determine the advantage value:

• **Estimating state-value.** The function $V(s_n)$ provides an estimation of the value of being in state s_n when following the Actor policy. We directly model the function with the LLM of the Critic module. In particular, we leverage in-context learning with few-shot examples and previous estimations in the memory $\mathcal{M}_C = \{(s_m, V(s_m)) | m = 1, 2, \dots\}$ to predict the values of a given state s_n. Formally, we have:

$$V(s_n) = \text{LLM}(\mathcal{P}_C, \Phi^n_{\mathcal{M}_C}, \mathcal{H}^{1\cdots n-1}, s_n), \tag{13}$$

where $\mathcal{P}_C$ represents the used task prompt (including few-shot examples), and $\Phi^n_{\mathcal{M}_C}$ denotes the selected experiences from $\mathcal{M}_C$, which is obtained as follows:

$$\Phi^n_{\mathcal{M}_C} = \{(s_m, V(s_m)) | d(s_m, s_n) < \tau_C, s_m \in \mathcal{M}_C\}, \tag{14}$$

where τ_C is a threshold, and $d(\cdot,\cdot)$ denotes the same distance function in Equation 5.

• **Computing advantage value.** We next utilize the advantage function to determine the advantage value v_n of action a_n given the state s_n at the n-th step. The advantage value is:

$$v_n = \sigma(A(s_n, a_n)), \quad A(s_n, a_n) = r_n + \gamma * V(s_{n+1}) - V(s_n), \tag{15}$$

where $A(\cdot,\cdot)$ is the commonly used advantage function, r_n denotes the environmental reward at step n, and s_{n+1} denotes the next-step state if taking action a_n at state s_n. Regarding the function σ, we have $\sigma(x) = 1$ if $x \geq 0$ else 0. Note that this approach mitigates the issue of high variance estimation of the Q-value in previous work [33] (*cf.* Section 5.4).

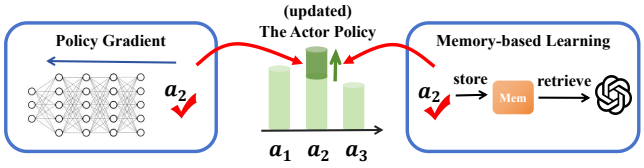

Figure 2: The memory-based learning methods and policy gradient based methods have a comparable impact on the Actor policy.

4.2.3 Update. The micro-learning involves updating both the Actor and the Critic during their iteration. At each step n, after obtaining the advantage value v_n for the action a_n, we update the two modules by injecting the new experience into their memory.

- The Critic memory update can be formulated as

$$\mathcal{M}_C \leftarrow (s_n, r_n + \gamma * V(s_{n+1})), \tag{16}$$

  where $r_n + \gamma * V(s_{n+1})$ can be considered as a more accurate estimation of $V(s_n)$ [33].
- The Actor memory update can be formulated as

$$\mathcal{M}_A \leftarrow (s_n, v_n). \tag{17}$$

The updated memory incorporates new experiences, helping enhance the next step of processing.

4.3 Discussion

Next, we compare the policy update of our BiLLP framework with traditional gradient-based policy updates to illustrate why our approach, based on in-context learning, can learn planning without the need for gradient updates. As shown in Figure 2, for traditional methods, when a favorable action is identified for a state (possibly determined based on Q-values in the REINFORCE algorithm [34] or the Advantage Function in A2C [33]), the purpose of the gradient update is to adjust the policy to increase the probability of sampling that specific action for the given state. In contrast, for our method, although no gradient updates are performed, the specific state and action are recorded in external memory. When encountering a similar state again, the retrieving probability of that specific state and action from the memory will increase, which would further enhance the probability of executing that specific action in that state. This achieves a similar effect to gradient updates. This is the underlying learning principle for our BiLLP framework.

5 EXPERIMENTS

In this section, we evaluate the proposed BiLLP framework in the interactive recommendation settings. Our experiments aim to address the following questions:

- **RQ1:** How does BiLLP perform compared to state-of-the-art RL-based methods and other LLM frameworks in the interactive recommendation setting?
- **RQ2:** To what extent can macro-learning and micro-learning mechanisms improve the LLMs' planning ability?
- **RQ3:** Can the proposed Critic module effectively estimate the state-value function to facilitate the update of the Actor module?
- **RQ4:** Whether the proposed BiLLP framework is robust to different recommendation environments and base LLM models?

5.1 Experiments Setup

We introduce the experimental settings with regard to simulated experiments and baselines, which are implemented based on the EasyRL4Rec library[3] [56].

5.1.1 Recommendation Experiments. In the interactive recommendation setting, we are interested in examining the potential of models to mitigate the issue of filter bubbles and maximize users' long-term engagement. Conducting direct online experiments for model learning and testing can be prohibitively costly. As a result, following [15], we resort to creating simulated interactive environments using high-quality logs.

- **Steam** [25] contains reviews and game information. The dataset compiles titles and genres of games. we consider users who engage in gameplay for a duration exceeding 3 hours to have a rating of 5, while others are assigned a rating of 2. We filter out users and items that interact less than 5 times in the log.
- **Amazon-Book** [35] refers to a book recommendation dataset, the "book" subset of the famous Amazon Product Review dataset[4]. This dataset compiles titles and genres of books from Amazon, collected between 1996 and 2018, with review scores ranging from 1 to 5. We filter out users and items that interact less than 90 times in the log.

To better reflect the issue of filter bubbles and simulate real-world recommendation scenarios, we follow [15, 17, 54] to introduce a quit mechanism. The interaction will terminate if any of the following conditions are met:

- The similarity between a recommended item and the items in the recent recommendation list (with a window size of W) is below a predefined threshold β.
- The online reward r of a recommended item is less than 2.

In this sense, a model that effectively captures users' interests and mitigates the risk of continuously recommending similar items that reinforce the filter bubble phenomenon is crucial for achieving a longer interaction trajectory and maximizing cumulative rewards. To estimate the online reward, we first split the dataset evenly into training and test sets in chronological order. For each set $\mathcal{D} \in \{\mathcal{D}_{train}, \mathcal{D}_{test}\}$, we utilize the DeepFM model [18] to fit the data and obtain vector representations for users $\mathbf{e}_u^{\mathcal{D}}$ and items $\mathbf{e}_i^{\mathcal{D}}$. Then we can calculate the online rewards:

$$r_{u,i}^{\mathcal{D}} = \text{DeepFM}(\mathbf{e}_u^{\mathcal{D}}, \mathbf{e}_i^{\mathcal{D}}), \quad u \in \mathcal{U}, i \in \mathcal{I}, \tag{18}$$

and the similarity between the two items:

$$\text{sim}(\mathbf{e}_i^{\mathcal{D}}, \mathbf{e}_j^{\mathcal{D}}) = |\mathbf{e}_i^{\mathcal{D}} - \mathbf{e}_j^{\mathcal{D}}|_2, \quad i, j \in \mathcal{I}, \tag{19}$$

It is noteworthy that we have established separate training and test environments for each dataset in order to simulate real-world scenarios where the user interests may have evolved during online training and model deployment. For now, the simulated environments can play the same role as the online users. Therefore, we can train the model on the training simulated experiments and evaluate the model on the test simulated experiments as the process shown in Figure 1. The statistics of the datasets are illustrated in Table 4.

[3] https://github.com/chongminggao/easyrl4rec
[4] https://cseweb.ucsd.edu/~jmcauley/datasets/amazon_v2/

Table 3: Average results of all methods in two environments (Bold: Best, Underline: Runner-up).

Methods	Steam			Amazon		
	Len	R_{each}	R_{traj}	Len	R_{each}	R_{traj}
SQN	2.183 ± 0.177	3.130 ± 0.050	6.837 ± 0.517	4.773 ± 0.059	4.303 ± 0.017	20.570 ± 0.245
CRR	4.407 ± 0.088	3.263 ± 0.427	14.377 ± 1.658	3.923 ± 0.162	4.537 ± 0.103	17.833 ± 1.129
BCQ	4.720 ± 0.343	3.997 ± 0.068	18.873 ± 1.092	4.847 ± 0.721	4.367 ± 0.053	21.150 ± 2.893
CQL	5.853 ± 0.232	3.743 ± 0.147	21.907 ± 0.299	2.280 ± 0.185	4.497 ± 0.039	10.263 ± 0.882
DQN	4.543 ± 0.693	4.500 ± 0.069	20.523 ± 3.618	4.647 ± 0.498	4.290 ± 0.083	19.923 ± 1.909
A2C	9.647 ± 0.848	4.367 ± 0.069	42.180 ± 3.937	7.873 ± 0.310	4.497 ± 0.026	35.437 ± 1.453
DORL	9.467 ± 0.862	4.033 ± 0.098	38.300 ± 4.173	7.507 ± 0.174	4.510 ± 0.014	33.887 ± 0.655
ActOnly	5.567 ± 0.160	4.537 ± 0.021	25.250 ± 0.637	6.383 ± 0.176	4.490 ± 0.008	28.660 ± 0.761
ReAct	11.630 ± 0.741	**4.559 ± 0.047**	52.990 ± 2.925	7.733 ± 0.450	4.603 ± 0.033	35.603 ± 1.806
Reflexion	12.690 ± 1.976	4.523 ± 0.026	57.423 ± 8.734	8.700 ± 0.535	**4.670 ± 0.073**	40.670 ± 2.954
BiLLP	**15.367 ± 0.119**	4.503 ± 0.069	**69.193 ± 1.590**	**9.413 ± 0.190**	4.507 ± 0.012	**42.443± 0.817**

Table 4: Statistics of experiment datasets.

Datasets	#Users	#Items	#Train	#Test
Steam	6,012	190,365	1,654,303	958,452
Amazon	3,109	13,864	339,701	137,948

5.1.2 Evaluation Metrics. In this paper, we utilize three metrics: the **trajectory length** (Len), the average **single-round reward** (R_{each}), and the **cumulative reward** of the whole trajectory (R_{traj}) to evaluate the model performance in the interactive recommendation setting. Longer trajectory lengths and higher cumulative rewards demonstrate the model's ability to maximize long-term engagement. However, it is important to note that a higher average reward is not necessarily better. Excessively high average rewards may indicate a model's overemphasis on immediate responses.

5.1.3 Baselines. To comprehensively and fairly evaluate the superiority of our proposed BiLLP, we choose some representative RL-based methods and LLM-based methods as baselines. For the RL-based methods, we choose seven representative baselines including the State-Of-The-Art (SOTA) method for long-term engagement optimization to mitigate filter bubble problems:

- **DQN**, or Deep Q-Networks [34], is a deep reinforcement learning algorithm that combines deep neural networks with the Q-learning algorithm.
- **SQN**, or Self-Supervised Q-learning [53], consists of two output layers, namely the cross-entropy loss head and the RL head. The RL head is utilized to generate the final recommendations.
- **BCQ**, or Batch-Constrained deep Q-learning [14], a modified version of conventional deep Q-learning designed for batch reinforcement learning. It utilizes the discrete-action variant [13], which focuses on discarding uncertain data and updating the policy solely based on high-confidence data.
- **CQL**, or Conservative Q-Learning [27], is a model-free RL method that adds a Q-value regularizer on top of an actor-critic policy.
- **CRR**, or Critic Regularized Regression [46], is a model-free RL method that learns the policy by avoiding OOD actions.
- **A2C**, or Advantage Actor-Critic [33], improves the Actor-Critic algorithm and stabilizes learning by using the Advantage function as Critic instead of the Action value function.
- **DORL**, or Debiased model-based Offline RL [15], add a penalty term to relax the pessimism on states with high entropy to alleviate the Matthew effect in offline RL-based recommendation. This is the SOTA method of maximizing users' long-term engagement to alleviate filter bubble issues.

Ensuring fair comparison, we also implement three LLM-based baselines utilizing the same LLM backbone as BiLLP:

- **ActOnly**, a baseline that recommends items to users according to instruction prompts without thought and planning.
- **ReAct**, [55] utilizes LLMs to generate both reasoning traces and task-specific actions in an interleaved manner, allowing for greater synergy between the reasoning and acting.
- **Reflexion**, [40] verbally reflects on task feedback signals, then maintains their own reflective text in an episodic memory buffer to induce better decision-making in subsequent trials.

5.1.4 Implementation Details. For a fair comparison, all RL-based methods are trained with 100,000 episode data, and all LLM-based methods are trained with 100 episode data. For model-based RL methods DORL, we use the same DeepFM model as [15]. For the Reflection and BiLLP methods, we set the number of most similar reflections $K = 2$. For the BiLLP method, we set the similarity threshold $\tau_A = 0.01$ and $\tau_C = 0.1$. The discount factor γ is set to 0.5. All methods in two environments are evaluated with the quit parameters: $W = 4$, $\beta_{Steam} = 50$, and $\beta_{Amazon} = 15$. The maximum round is set to 100. For all the RL-based methods, we leverage DeepFM [18] as the backbone following [15], and for all the LLM-based methods, we utilize the "gpt-3.5-turbo-16k" provided by OpenAI as the LLM backbone for its strong long context modeling ability. And the temperature is set to 0.5 for all experiments.

5.2 Main Results Comparison (RQ1)

After training, we evaluate all methods with 100 episodes (*i.e.*, interaction trajectories) in two interactive environments. The results are shown in Table 3, where each result in the table is averaged over three random experiments with distinct seeds for robustness and reliability. From the results, we observe that:

- BiLLP consistently achieves the best long-term performance (Len and R_{traj}) over RL-based methods and LLM-based baselines across

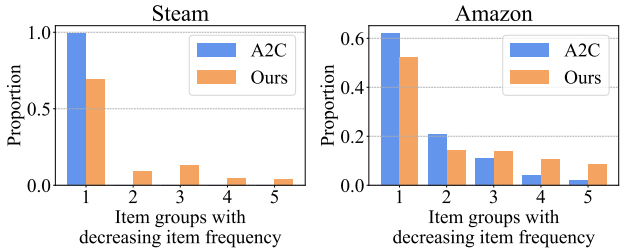

Figure 3: The frequency distribution of items recommended by our method and A2C in the two environments.

two datasets. This demonstrates the effectiveness of our proposed framework and its ability to stimulate and adapt the long-term planning capacity of LLMs. For single-round reward R_{each}, BiLLP obtains a relatively higher score, which indicates it successfully captures user interests while avoiding excessive emphasis on immediate responses.

- The ActOnly method, which only utilizes LLMs to generate actions (recommendations), exhibits inferior performance compared to certain RL-based methods and LLM-based methods that incorporate planning. Drawing upon this, we can infer that an explicit thinking and planning process is crucial for enhancing the planning capabilities of LLMs.
- The ReAct method, which integrates the thinking process and action process, still performs worse than Reflexion and BiLLP. This underscores the significance of self-improvement in LLMs, in order to improve their planning abilities for long-term recommendation tasks.

In addition to the overall performance comparison, we conduct an in-depth analysis of the recommended items for RL-based method A2C and LLM-based method BiLLP. We first calculate the items' popularity (occurrence frequencies) both in the training set and test set. Subsequently, we evenly divide the items into five groups with decreasing popularity: 1, 2, 3, 4, 5. We analyze the proportion of items belonging to each group among the recommended items generated by A2C and BiLLP, where the results are shown in Figure 3. From the figure, we observe that:

- RL-based method A2C tends to overfit on popularity items and lack planning capabilities on long-tail items.
- In contrast, BiLLP exhibits better planning capabilities on long-tail items, which could effectively alleviate the issue of filter bubbles and maximize long-term engagement.

5.3 Ablation Study (RQ2)

In this subsection, we conduct ablation studies to evaluate the effect of the two learning mechanisms. Concretely, **w/o Macro** refers to a variant of the BiLLP framework that does not use the reflective text to enhance its Planner module, and **w/o Micro** refers to a variant that does not use the micro-learning experience to enhance its Actor module. From Table 5, we can observe that:

- The absence of either of the two learning mechanisms would result in a decline in performance, thereby indicating that both learning mechanisms have contributed to the enhancement of long-term engagement.

Table 5: Average results of all methods in the two environments (Bold: Best).

Methods	Steam		
	Len	R_{each}	R_{traj}
w/o Macro	14.363 ± 0.467	4.523 ± 0.012	64.960 ± 2.011
w/o Micro	14.270 ± 0.190	**4.535 ± 0.005**	64.720 ± 0.920
BiLLP	**15.367 ± 0.119**	4.503 ± 0.069	**69.193 ± 1.590**
Methods	**Amazon**		
	Len	R_{each}	R_{traj}
w/o Macro	8.947 ± 0.480	4.530 ± 0.057	40.547 ± 2.622
w/o Micro	8.800 ± 0.432	**4.707 ± 0.026**	41.420 ± 2.003
BiLLP	**9.413 ± 0.190**	4.507 ± 0.012	**42.443 ± 0.817**

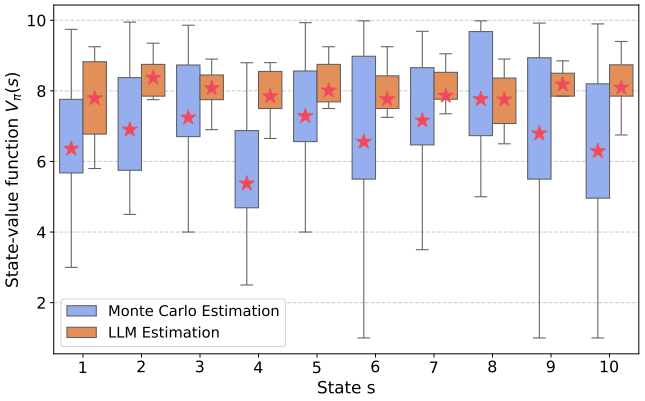

Figure 4: The memory-based in-context learning methods and policy gradient-based methods have a comparable impact on the Actor policy.

- Based on the experimental details presented in Section 5.1.4, the aforementioned improvements are achieved using only 100 episodes of data for both learning mechanisms. This suggests the high efficiency of in-context learning compared to fine-tuning and training from scratch.

5.4 Effects of Critic Module (RQ3)

In this subsection, our objective is to demonstrate the effectiveness of the Critic module in estimating the state-value function, denoted as $V_\pi(s)$, which is crucial for facilitating the update process of the Actor module. The state-value function $V_\pi(s)$ gives the expected cumulative discounted reward if we start from state s and act according to the policy.

To get an accurate and unbiased estimation of $V_\pi(s)$, for a specific state s, we sample 1000 complete trajectories according to the Actor policy and calculate the cumulative discounted reward for each trajectory. Figure 4 illustrates the distribution of these reward samples, along with their mean value. The mean value serves as an accurate and unbiased estimation of the state-value function $V_\pi(s)$. It is worth noting that prior studies [4, 57], have utilized a single trajectory's cumulative discounted reward to estimate either the state-value function $V_\pi(s)$ or the state-action value function $Q_\pi(s, a)$, which suffers from the issue of high variance estimation.

Table 6: Average Results of all methods in the two environments (Bold: Best).

Methods	Steam		
	Len	R_{each}	R_{traj}
GPT-4-32k	**25.400 ± 2.800**	**4.635 ± 0.115**	**118.235 ± 15.915**
GPT-3.5-16k	15.367 ± 0.119	4.503 ± 0.069	69.193 ± 1.590
Llama-2-7B	13.800 ± 1.105	4.610 ± 0.065	63.767 ± 6.015
Methods	Amazon		
	Len	R_{each}	R_{traj}
GPT-4-32k	**12.450 ± 1.250**	4.580 ± 0.070	**57.180 ± 6.570**
GPT-3.5-16k	9.413 ± 0.190	4.507 ± 0.012	42.443 ± 0.817
Llama-2-7B	8.100 ± 1.512	**4.603 ± 0.054**	37.300 ± 6.895

In contrast to these approaches, we leverage the Critic module to estimate the state-value function $V_\pi(s)$. To evaluate our estimation, we repeat the estimation 100 times. The resulting estimations, as well as their distribution and mean value, are also depicted in Figure 4. Based on the analysis of ten different states, it can be inferred that the utilization of the Critic module effectively mitigates estimation variance, despite the presence of a small bias in the estimation.

5.5 Robustness of the Framework (RQ4)

5.5.1 Results with Different Environments. To validate that BiLLP can work robustly in different environment settings, we vary the window size W in the exit mechanism and fix the similarity threshold β to simulate different effects of filter bubbles on user disengagement. The evaluation results are shown in Figure 5, where all results are averaged over three random experiments with distinct seeds. We visualize all three metrics and observe that:

- As the window size W increases, the performance of the trajectory length and the cumulative reward metrics decrease for all methods. This implies that when users are more susceptible to the influence of filter bubbles, the model faces greater challenges in learning to improve users' long-term engagement.
- BiLLP outperforms all baselines in terms of both the trajectory length and the cumulative reward metrics, which demonstrates the robustness of BiLLP in different environments.
- BiLLP obtains a relatively higher score in terms of the single-round in different environments, which indicates it successfully captures user interests while avoiding excessive emphasis on immediate responses.

5.5.2 Results with Different Base Models. To validate the robustness of the BiLLP framework across various base models, we conduct additional experiments with other different LLM backbones: "gpt-4-32k" and "Llama-2-7b". The results are presented in Table 6. From the table, several noteworthy observations can be made:

- BiLLP showcases superior performance compared to traditional RL-based methods with different base models, as demonstrated in Table 6. This indicates that our framework is robust across different LLMs.
- The performance of BiLLP based on "GPT-3.5-16k" is superior to that based on "Llama-2-7B", while inferior to that based on "GPT-4-32k". This observation suggests a positive correlation between the strength of the LLM backbone and the performance enhancement of BiLLP.

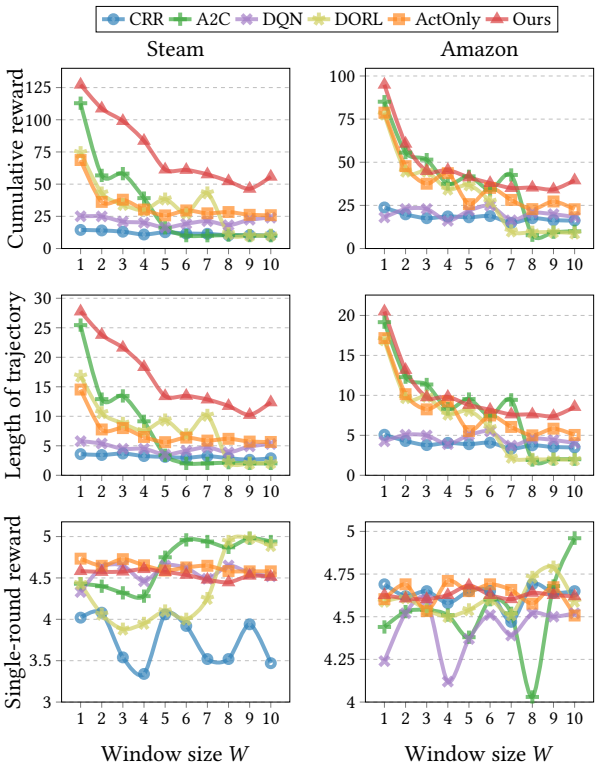

Figure 5: Results under different simulated environments.

6 CONCLUSION

In this work, we explore the integration of planning capabilities from Large Language Models (LLMs) into the recommendation to optimize long-term engagement. To bridge the gap between the pre-training scenarios and recommendation scenarios, we propose a bi-level learnable LLM planning framework called BiLLP, where the learning process can be divided into macro-learning and micro-learning using a hierarchical mechanism. This hierarchical approach improves learning efficiency and adaptability. Extensive experiments validate the capability of LLMs to plan for long-term recommendation and the superiority of the BiLLP framework.

A potential avenue for future research involves exploring techniques to enhance the planning capabilities of small-scale models in the context of recommendation tasks. Additionally, exploring the integration of reinforcement learning algorithms within the planning framework could provide further insights into optimizing long-term engagement in recommendation systems.

ACKNOWLEDGMENTS

This work is supported by the National Key Research and Development Program of China (2022YFB3104701), the National Natural Science Foundation of China (62272437, 62121002), and the CCCD Key Lab of Ministry of Culture and Tourism.

REFERENCES

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774* (2023).

[2] Keqin Bao, Jizhi Zhang, Wenjie Wang, Yang Zhang, Zhengyi Yang, Yancheng Luo, Fuli Feng, Xiangnan He, and Qi Tian. 2023. A Bi-Step Grounding Paradigm for Large Language Models in Recommendation Systems. *CoRR* abs/2308.08434 (2023).

[3] 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. *arXiv preprint arXiv:2305.00447* (2023).

[4] Ethan Brooks, Logan Walls, Richard L. Lewis, and Satinder Singh. 2023. Large Language Models can Implement Policy Iteration. arXiv:2210.03821 [cs.LG]

[5] Jiawei Chen, Hande Dong, Xiang Wang, Fuli Feng, Meng Wang, and Xiangnan He. 2020. Bias and Debias in Recommender System: A Survey and Future Directions. *CoRR* abs/2010.03240 (2020).

[6] Jiawei Chen, Junkang Wu, Jiancan Wu, Xuezhi Cao, Sheng Zhou, and Xiangnan He. 2023. Adap-τ : Adaptively Modulating Embedding Magnitude for Recommendation. In *WWW*. ACM, 1085–1096.

[7] Minmin Chen, Alex Beutel, Paul Covington, Sagar Jain, Francois Belletti, and Ed H. Chi. 2019. Top-K Off-Policy Correction for a REINFORCE Recommender System. In *WSDM*. ACM, 456–464.

[8] 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. *arXiv preprint arXiv:2305.02182* (2023).

[9] Thomas Degris, Martha White, and Richard S. Sutton. 2012. Off-Policy Actor-Critic. *CoRR* abs/1205.4839 (2012).

[10] Gabriel Dulac-Arnold, Richard Evans, Peter Sunehag, and Ben Coppin. 2015. Reinforcement Learning in Large Discrete Action Spaces. *CoRR* abs/1512.07679 (2015).

[11] Wenqi Fan, Zihuai Zhao, Jiatong Li, Yunqing Liu, Xiaowei Mei, Yiqi Wang, Jiliang Tang, and Qing Li. 2023. Recommender systems in the era of large language models (llms). *arXiv preprint arXiv:2307.02046* (2023).

[12] Yue Feng, Shuchang Liu, Zhenghai Xue, Qingpeng Cai, Lantao Hu, Peng Jiang, Kun Gai, and Fei Sun. 2023. A Large Language Model Enhanced Conversational Recommender System. *arXiv preprint arXiv:2308.06212* (2023).

[13] Scott Fujimoto, Edoardo Conti, Mohammad Ghavamzadeh, and Joelle Pineau. 2019. Benchmarking Batch Deep Reinforcement Learning Algorithms. *CoRR* abs/1910.01708 (2019).

[14] Scott Fujimoto, David Meger, and Doina Precup. 2019. Off-Policy Deep Reinforcement Learning without Exploration. In *ICML (Proceedings of Machine Learning Research, Vol. 97)*. PMLR, 2052–2062.

[15] Chongming Gao, Kexin Huang, Jiawei Chen, Yuan Zhang, Biao Li, Peng Jiang, Shiqi Wang, Zhong Zhang, and Xiangnan He. 2023. Alleviating Matthew Effect of Offline Reinforcement Learning in Interactive Recommendation. In *SIGIR*. ACM, 238–248.

[16] Chongming Gao, Wenqiang Lei, Xiangnan He, Maarten de Rijke, and Tat-Seng Chua. 2021. Advances and Challenges in Conversational Recommender Systems: A Survey. *AI Open* 2 (2021), 100–126.

[17] Chongming Gao, Shiqi Wang, Shijun Li, Jiawei Chen, Xiangnan He, Wenqiang Lei, Biao Li, Yuan Zhang, and Peng Jiang. 2024. CIRS: Bursting Filter Bubbles by Counterfactual Interactive Recommender System. *ACM Trans. Inf. Syst.* 42, 1 (2024), 14:1–14:27.

[18] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: A Factorization-Machine based Neural Network for CTR Prediction. In *IJCAI*. ijcai.org, 1725–1731.

[19] 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 preprint arXiv:2305.08845* (2023).

[20] Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. 2022. Language Models as Zero-Shot Planners: Extracting Actionable Knowledge for Embodied Agents. In *ICML (Proceedings of Machine Learning Research, Vol. 162)*. PMLR, 9118–9147.

[21] Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Tomas Jackson, Noah Brown, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. 2022. Inner Monologue: Embodied Reasoning through Planning with Language Models. In *CoRL (Proceedings of Machine Learning Research, Vol. 205)*. PMLR, 1769–1782.

[22] Xu Huang, Jianxun Lian, Yuxuan Lei, Jing Yao, Defu Lian, and Xing Xie. 2023. Recommender ai agent: Integrating large language models for interactive recommendations. *arXiv preprint arXiv:2308.16505* (2023).

[23] Eugene Ie, Vihan Jain, Jing Wang, Sanmit Narvekar, Ritesh Agarwal, Rui Wu, Heng-Tze Cheng, Tushar Chandra, and Craig Boutilier. 2019. SlateQ: A Tractable Decomposition for Reinforcement Learning with Recommendation Sets. In *IJCAI*. ijcai.org, 2592–2599.

[24] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data* 7, 3 (2019), 535–547.

[25] Wang-Cheng Kang and Julian J. McAuley. 2018. Self-Attentive Sequential Recommendation. In *ICDM*. IEEE Computer Society, 197–206.

[26] 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 preprint arXiv:2305.06474* (2023).

[27] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. 2020. Conservative Q-Learning for Offline Reinforcement Learning. In *NeurIPS*.

[28] Bill Yuchen Lin, Yicheng Fu, Karina Yang, Prithviraj Ammanabrolu, Faeze Brahman, Shiyu Huang, Chandra Bhagavatula, Yejin Choi, and Xiang Ren. 2023. SwiftSage: A Generative Agent with Fast and Slow Thinking for Complex Interactive Tasks. *CoRR* abs/2305.17390 (2023).

[29] Jianghao Lin, Xinyi Dai, Yunjia Xi, Weiwen Liu, Bo Chen, Xiangyang Li, Chenxu Zhu, Huifeng Guo, Yong Yu, Ruiming Tang, et al. 2023. How Can Recommender Systems Benefit from Large Language Models: A Survey. *arXiv preprint arXiv:2306.05817* (2023).

[30] Jianghao Lin, Rong Shan, Chenxu Zhu, Kounianhua Du, Bo Chen, Shigang Quan, Ruiming Tang, Yong Yu, and Weinan Zhang. 2023. ReLLa: Retrieval-enhanced Large Language Models for Lifelong Sequential Behavior Comprehension in Recommendation. *arXiv preprint arXiv:2308.11131* (2023).

[31] Feng Liu, Ruiming Tang, Xutao Li, Yunming Ye, Haokun Chen, Huifeng Guo, and Yuzhou Zhang. 2018. Deep Reinforcement Learning based Recommendation with Explicit User-Item Interactions Modeling. *CoRR* abs/1810.12027 (2018).

[32] Junling Liu, Chao Liu, Renjie Lv, Kang Zhou, and Yan Zhang. 2023. Is chatgpt a good recommender? a preliminary study. *arXiv preprint arXiv:2304.10149* (2023).

[33] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous Methods for Deep Reinforcement Learning. In *ICML (JMLR Workshop and Conference Proceedings, Vol. 48)*. JMLR.org, 1928–1937.

[34] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. 2013. Playing Atari with Deep Reinforcement Learning. *CoRR* abs/1312.5602 (2013).

[35] Jianmo Ni, Jiacheng Li, and Julian J. McAuley. 2019. Justifying Recommendations using Distantly-Labeled Reviews and Fine-Grained Aspects. In *EMNLP/IJCNLP (1)*. Association for Computational Linguistics, 188–197.

[36] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems* 35 (2022), 27730–27744.

[37] Shreyas Sundara Raman, Vanya Cohen, Eric Rosen, Ifrah Idrees, David Paulius, and Stefanie Tellex. 2022. Planning with Large Language Models via Corrective Re-prompting. *CoRR* abs/2211.09935 (2022).

[38] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. *CoRR* abs/1707.06347 (2017).

[39] Junjie Sheng, Zixiao Huang, Chuyun Shen, Wenhao Li, Yun Hua, Bo Jin, Hongyuan Zha, and Xiangfeng Wang. 2023. Can language agents be alternatives to PPO? A Preliminary Empirical Study On OpenAI Gym. *CoRR* abs/2312.03290 (2023).

[40] 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]

[41] Haotian Sun, Yuchen Zhuang, Lingkai Kong, Bo Dai, and Chao Zhang. 2023. AdaPlanner: Adaptive Planning from Feedback with Language Models. *CoRR* abs/2305.16653 (2023).

[42] Hugo Touvron et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288* (2023).

[43] Yancheng Wang, Ziyan Jiang, Zheng Chen, Fan Yang, Yingxue Zhou, Eunah Cho, Xing Fan, Xiaojiang Huang, Yanbin Lu, and Yingzhen Yang. 2023. Recmind: Large language model powered agent for recommendation. *arXiv preprint arXiv:2308.14296* (2023).

[44] Yuyan Wang, Mohit Sharma, Can Xu, Sriraj Badam, Qian Sun, Lee Richardson, Lisa Chung, Ed H. Chi, and Minmin Chen. 2022. Surrogate for Long-Term User Experience in Recommender Systems. In *KDD*. ACM, 4100–4109.

[45] Zihao Wang, Shaofei Cai, Anji Liu, Xiaojian Ma, and Yitao Liang. 2023. Describe, Explain, Plan and Select: Interactive Planning with Large Language Models Enables Open-World Multi-Task Agents. *CoRR* abs/2302.01560 (2023).

[46] Ziyu Wang, Alexander Novikov, Konrad Zolna, Josh Merel, Jost Tobias Springenberg, Scott E. Reed, Bobak Shahriari, Noah Y. Siegel, Çaglar Gülçehre, Nicolas Heess, and Nando de Freitas. 2020. Critic Regularized Regression. In *NeurIPS*.

[47] Wei Wei, Xubin Ren, Jiabin Tang, Qinyong Wang, Lixin Su, Suqi Cheng, Junfeng Wang, Dawei Yin, and Chao Huang. 2023. Llmrec: Large language models with graph augmentation for recommendation. *arXiv preprint arXiv:2311.00423* (2023).

[48] Junda Wu, Zhihui Xie, Tong Yu, Handong Zhao, Ruiyi Zhang, and Shuai Li. 2022. Dynamics-Aware Adaptation for Reinforcement Learning Based Cross-Domain Interactive Recommendation. In *SIGIR*. ACM, 290–300.

[49] Likang Wu, Zhaopeng Qiu, Zhi Zheng, Hengshu Zhu, and Enhong Chen. 2023. Exploring large language model for graph data understanding in online job recommendations. *arXiv preprint arXiv:2307.05722* (2023).
[50] Likang Wu, Zhi Zheng, Zhaopeng Qiu, Hao Wang, Hongchao Gu, Tingjia Shen, Chuan Qin, Chen Zhu, Hengshu Zhu, Qi Liu, et al. 2023. A Survey on Large Language Models for Recommendation. *arXiv preprint arXiv:2305.19860* (2023).
[51] Qingyun Wu, Hongning Wang, Liangjie Hong, and Yue Shi. 2017. Returning is believing: Optimizing long-term user engagement in recommender systems. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 1927–1936.
[52] Yunjia Xi, Weiwen Liu, Jianghao Lin, Jieming Zhu, Bo Chen, Ruiming Tang, Weinan Zhang, Rui Zhang, and Yong Yu. 2023. Towards Open-World Recommendation with Knowledge Augmentation from Large Language Models. *arXiv preprint arXiv:2306.10933* (2023).
[53] Xin Xin, Alexandros Karatzoglou, Ioannis Arapakis, and Joemon M. Jose. 2020. Self-Supervised Reinforcement Learning for Recommender Systems. In *SIGIR*. ACM, 931–940.
[54] Shuyuan Xu, Juntao Tan, Zuohui Fu, Jianchao Ji, Shelby Heinecke, and Yongfeng Zhang. 2022. Dynamic Causal Collaborative Filtering. In *CIKM*. ACM, 2301–2310.
[55] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. In *ICLR*. OpenReview.net.
[56] Yuanqing Yu, Chongming Gao, Jiawei Chen, Heng Tang, Yuefeng Sun, Qian Chen, Weizhi Ma, and Min Zhang. 2024. EasyRL4Rec: A User-Friendly Code Library for Reinforcement Learning Based Recommender Systems. *arXiv preprint arXiv:2402.15164* (2024).
[57] Danyang Zhang, Lu Chen, Situo Zhang, Hongshen Xu, Zihan Zhao, and Kai Yu. 2023. Large Language Model Is Semi-Parametric Reinforcement Learning Agent. *CoRR* abs/2306.07929 (2023).
[58] 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. *arXiv preprint arXiv:2305.07609* (2023).
[59] Junjie Zhang, Ruobing Xie, Yupeng Hou, Wayne Xin Zhao, Leyu Lin, and Ji-Rong Wen. 2023. Recommendation as instruction following: A large language model empowered recommendation approach. *arXiv preprint arXiv:2305.07001* (2023).
[60] Yang Zhang, Fuli Feng, Jizhi Zhang, Keqin Bao, Qifan Wang, and Xiangnan He. 2023. Collm: Integrating collaborative embeddings into large language models for recommendation. *arXiv preprint arXiv:2310.19488* (2023).
[61] Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. 2023. ExpeL: LLM Agents Are Experiential Learners. *CoRR* abs/2308.10144 (2023).
[62] Xiangyu Zhao, Long Xia, Liang Zhang, Zhuoye Ding, Dawei Yin, and Jiliang Tang. 2018. Deep reinforcement learning for page-wise recommendations. In *RecSys*. ACM, 95–103.
[63] Guanjie Zheng, Fuzheng Zhang, Zihan Zheng, Yang Xiang, Nicholas Jing Yuan, Xing Xie, and Zhenhui Li. 2018. DRN: A Deep Reinforcement Learning Framework for News Recommendation. In *WWW*. ACM, 167–176.