Reinforcement Learning-based Recommender Systems with Large Language Models for State Reward and Action Modeling

Jie Wang
j.wang.9@research.gla.ac.uk
University of Glasgow
Glasgow, UK

Alexandros Karatzoglou*
alexandros.karatzoglou@gmail.com
Amazon
Barcelona, Spain

Ioannis Arapakis
ioannis.arapakis@telefonica.com
Telefonica Research
Barcelona, Spain

Joemon M. Jose
joemon.jose@glasgow.ac.uk
University of Glasgow
Glasgow, UK

ABSTRACT

Reinforcement Learning (RL)-based recommender systems have demonstrated promising performance in session-based and sequential recommendation tasks. Existing offline RL-based sequential recommendation methods face the challenge of obtaining effective user feedback from the environment. Developing a model for the user state and shaping an appropriate reward for recommendation remains a challenge. In this paper, we leverage language understanding capabilities and adapt large language models (LLMs) as an environment (LE) to enhance RL-based recommenders. The LE is learned from a subset of user-item interaction data, thus reducing the need for large training data, and can synthesize user feedback for offline data by: (i) acting as a state model that produces high-quality states that enrich the user representation, and (ii) functioning as a reward model to accurately capture nuanced user preferences on actions. Moreover, the LE allows us to generate positive actions that augment the limited offline training data. We propose a LE Augmentation (LEA) method to further improve recommendation performance by optimising jointly the supervised component and the RL policy, using the augmented actions and historical user signals. We use LEA, the state, and reward models in conjunction with state-of-the-art RL recommenders and report experimental results on two publicly available datasets[1].

CCS CONCEPTS

• **Information systems → Recommender systems**.

KEYWORDS

Sequential Recommendation, Reinforcement Learning, Augmentation, Large Language Models

ACM Reference Format:
Jie Wang, Alexandros Karatzoglou, Ioannis Arapakis, and Joemon M. Jose. 2024. Reinforcement Learning-based Recommender Systems with Large Language Models for State Reward and Action Modeling . 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/10.1145/3626772.3657767

*Work was done while the author was part of Google.
[1]The implementation can be found at https://github.com/jieWANGforwork/LEA

SIGIR '24, July 14–18, 2024, Washington, DC, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0431-4/24/07.
https://doi.org/10.1145/3626772.3657767

1 INTRODUCTION

Recommender Systems (RS) have become an essential tool that navigates through extensive data to deliver relevant and engaging content to users [17, 45] in commercial platforms. Sequential or next-item recommendation [12, 21, 38, 46] have gained prominence, especially in music and video streaming RS, to recommend the next relevant item based on user-item interactions within a recent active session. Typically, sequential recommendation models [12], such as those based on gated recurrent units (GRU), convolutional neural networks (CNN) [45], and Transformer [21], have been trained to predict the next interacted items based on historical user data in a self-supervised manner. To address suboptimal recommendations due to the dependence on supervised learning, self-supervised reinforcement learning (SSRL) has been proposed. This approach trains an RL agent to satisfy user expectations, e.g., the desire for diverse content, by employing sequential models with rewards tailored to accommodate various behaviours [37, 42]. However, recent efforts [35, 42, 43] to develop off-policy/offline RL policies trained on historical user data have been met with the challenge of constructing a high-quality environment that provides meaningful user feedback, e.g., state representation and reward function.

Large Language Models (LLMs) with knowledge-transferring capabilities have recently received significant attention in RS [4, 11, 47]. In the context of sequential recommendation, LLMs have been shown to be acceptable zero-shot [1] or pre-trained [7] RS. These LLM-recommenders have been proven to perform on par with, or even outperform, conventional models. Recent research has heavily focused on fine-tuning/pre-training with user data to adapt LLM for recommendation. This results in large models with orders of magnitude more parameters than traditional recommendation models. Despite the adoption of efficient tuning methods [16, 27], the computational cost of inference with LLM models with billions of parameters remains very high, making serving LLM recommendation models impractical in real-world scenarios.

Motivated by their generative and language-understanding capabilities, we propose adapting LLM as an environment (LE) to model user behavior and return feedback for training the RL-based recommenders. More importantly, LE can help train leaner and more adaptable sequential recommendation models that outperform those trained with standard techniques without incurring additional computational costs at inference time. Specifically, we address the following limitations of the above-mentioned state and reward problems for RL-based RS: Recommendation approaches based on reinforcement learning (RL) typically utilize state representations derived from generative sequential models. These models, such as Transformers or RNNs, process user-logged actions to produce output or hidden states, which are then employed to generate recommendations. LLMs have been shown to be world models [8, 23], which may help generate more accurate representations of user state given the sequence of historical user actions. Additionally, in typical RL-based recommenders, rewards are usually predefined with respect to behavior categories [42], for example, purchase and click. However, a simple uniform reward setting might not accurately reflect user satisfaction across items, resulting in an agent who is unable to capture latent differences between actions. LLMs have been shown to be capable of generating good reward estimates [22] that can be used to train RL algorithms. To this end, we capitalize on these powerful LLM properties to learn an environment (LE) that acts as state and reward model to return high-utility feedback for training RL-based recommendation models.

To construct the LE we fine-tune the LLM by introducing an item tokenization strategy, using autoregressive training. We first learn semantically-rich tokens by using the items' textual descriptions. We subsequently fine-tune the LLM through a small subset of user data and adapters to obtain the reward model (RM) and state model (SM). Specifically, we prompt the LLM with instructions based on user-item token interactions to output the scalar reward by a score head. We then learn effective state representations by contrasting the user-item tokens' interactions with the positive and negative actions. In our modular architecture, RM and SM constitute the LE that enables the acquisition of the state representations and scalar rewards for the RL-based recommendation model.

Moreover, in an offline setting, the agent is trained on fixed historical user-item interactions without probing the environment. Thus, we further propose an LE Augmentation (LEA) method prompting the obtained LE to enrich the offline data for RL-based sequential recommendation. In particular, LE is tasked with selecting potentially positive feedback by prompting it with a combination of user historical behavior and a sampled list of items. This step aims at leveraging the predicted positive items as positive samples to augment the training of the supervised learning component, and as positive actions to reinforce the training of the RL agent.

Finally, we train the supervised loss and the RL loss on the original historical user data and the augmented positive samples. At the inference stage, only the sequential model with the supervised head is used for the evaluation of the top-k recommendation performance, to guarantee efficiency. To validate the effectiveness of our method, we compare LEA with two state-of-the-art Q-value-based RL frameworks, with two sequential models as the backbone. We also directly apply LE to the above frameworks by enhancing the state representation and reward function, and demonstrate significant performance gains on two publicly available datasets. Our contributions can be summarised as follows:

- We propose an LLM-based Environment (LE), acting as the state model and reward function, to improve the performance of the offline RL-based recommender systems.
- We present an efficient fine-tuning method for adapting LLMs for LE using limited user data. Additionally, we propose an item-tokenization strategy to incorporate user data and improve training efficiency.
- We introduce a positive feedback augmentation approach LEA to enhance both supervised learning and Q-learning. The LE is utilized as the behaviour policy to infer positive signals from historical user data.
- We apply the environment LE and augmentation method LEA to two state-of-the-art RL-based sequential recommendation models. Experimental results on real-world datasets show a general improvement across recommendation performance metrics.

2 RELATED WORK

2.1 Supervised Reinforcement Learning for Sequential Recommendation

Deep learning-based sequential recommender systems (DSRS) model users' historical interactions as next-item recommendation tasks to predict their future preferences. One of the first models was proposed by Hidasi et al. [12] utilizing Gated Recurrent Units (GRU) to model user sequences. Subsequently, Convolutional neural networks (CNN) [46] and the Transformer architecture [38] were also adopted for the task. In an offline setting, the sequential recommendation problem can be viewed as a Markov Decision Process (MDP) [29, 41, 49], a framework often employed in e-commerce scenarios with reinforcement learning (RL). Hong et al. [13] integrated wireless sensing to RL Monte Carlo tree search algorithm to improve music recommendation performance. Wang et al. [41] introduced a Knowledge-guided RL framework that enriches the environment by a Knowledge Graph. Moreover, Xin et al. [42] introduced a self-supervised reinforcement learning framework for sequential recommendation (SRLSR). The Sequential Q-Network (SQN) [42] and Soft Actor-Critic (SAC) [43] architectures combine a Double Q-learning head [9] with supervised learning to enhance the baselines accuracy with respect to user clicks and purchases. They further enhanced the frameworks with SNQN and SA2C [43] by using a sampling strategy to integrate negative feedback. More recent research [44] proposed a paradigm of modeling the desirable cumulative rewards rather than the expected returns at each timestamp to guide the training. In other recent work, Ren et al. [35] augmented the original states to improve SQN by contrastive learning.

2.2 LLMs for Sequential Recommendation

LLMs [26, 28, 48] pre-trained on massive natural language datasets with continuously enhanced transfer capabilities have garnered attention in the field of RS [2, 5, 6, 25]. Existing adaptations of LLMs for recommendation tasks involve primarily training the LLM to be a new recommender through pre-training [7], fine-tuning [28], prompt-based tuning [26, 32] etc. For sequential recommendation, Geng et al. [7] pre-train T5 [33] with four tasks, e.g., rating, explanation, review, and direct recommendation, for new domains in RS. Other recent works [14, 40] train recommenders through user-item interactions based on the item features extracted from BERT [3] in the source domain and showed encouraging performance for cross-domain recommendation. Rajput et al. [34] generates the semantic IDs for item representations by Sentence-T5 [30] and then predicts the semantic IDs of the next item autoregressively. Bao et al. [1] efficiently fine-tune LLaMA-7B [39] with LoRA adapters [16] by an instruction prompt including item text descriptions to realize few-shot recommendations. However, these methods of employing LLMs as recommenders come with significant pre-training costs or face difficulties in preserving user signals within lengthy text sequences. Moreover, research efforts that aim to harness LLMs efficiently to improve the performance of existing ID-based sequential models have been sparse. Recent work [22] has shown that using LLM as reward model outperforms learned rewards in the context of RL applications in games.

This work diverges from existing approaches that focus solely on training new LLM-based sequential recommenders with heavy computation at training and particularly at inference time. Instead, we aim at efficiently adapting LLMs to serve as an environment component within an RL framework, with the objective of augmenting the performance of existing recommenders.

3 METHOD

3.1 Task Formulation

Let I denote the set of items in the system, $x_{1:t} = \{x_1, \ldots, x_t\}$ denote the user-item interaction sequence, where $x_i \in I (0 < i \leq t)$ is the index of the interacted item ordered by timestamp. The goal is to recommend the next item x_{t+1} at timestamp $t + 1$ for the user from the whole item set I, such that the user might be interested in given the previous interactions $x_{1:t}$. Sequential recommendation methods [42] have been proposed to tackle the task as a self-supervised learning problem and map the sequence $x_{1:t}$ into the hidden state h_t by a sequential model $G(\cdot)$, i.e., $h_t = G(x_{1:t})$. A fully connected layer then follows to map h_t to classification logits (ranking scores) on all candidate items at the next step. In RL-based recommendation, the task is further modeled as an MDP, where users are treated as the environment. As Figure 1a shows, the state from the environment is represented by the user's historical interactions, and an RL agent is additionally trained to interact with users by taking actions (i.e., recommending items) to maximize the cumulative rewards.

In previous methods, the user state is taken (re-used) from the hidden state h_t of the sequential model $G(\cdot)$, and the observed rewards are usually predefined w.r.t specific behaviours, e.g., purchase=1, click=0.2. However, the uniform reward setting for all positive or negative actions fails to accurately capture the nuances of user preferences, resulting in an agent that cannot discern the latent differences between items for a user. Moreover, user states generated via $G(\cdot)$, based on implicit data comprised of item IDs, cannot reflect user behaviours on specific content. In offline training, the agent is usually trained on fixed historical user-item interactions without probing the environments due to the significant costs associated with error-prone explorations, which introduces the challenge of insufficient positive and negative signals. The proposed solution involves developing a state model to represent nuanced user states based on historical data, along with formulating an accurate reward model that assigns rewards contingent upon specific user states and actions. Additionally, we attempt to distill user's potential interests from the environment to augment the positive feedback for offline training, thereby reinforcing the exploration capabilities of RL-based RS.

3.2 Large Language Model as Environment

LLMs are considered as emergent world representations [8, 23] with transfer learning capabilities. They can adapt to specific downstream tasks with minimal data through fine-tuning or prompt-based methods, e.g., optimization based on prompts and adapters that adjust a small number of parameters [10, 15] to generate question answers [18]. In this work, we leverage LLM as an offline user environment (LE) to return user feedback for RL-based RS attributes for three reasons. First, LLMs are ideal for creating environments that can simulate user queries/behaviours and feedback due to their ability to generate natural language [11, 24]. Second, environments created with LLMs significantly decrease the cost incurred by RL experiments in online settings, as error explorations could impair the user experience. Third, LLMs can generate feedback and shape rewards on various objectives, aiding RL-agents in learning complex signals that reflect user satisfaction. Therefore, we construct the LE based on the Transformer decoder-based LLM and use the latest Mistral 7B model for its simplicity and efficiency.

In summary, our objective is to efficiently tune LLMs with a small subset of offline user data, transforming them into a user environment (LE) capable of providing user feedback, including states, rewards, and predicted positive actions.

•**Item tokenization.** We first propose an initial step of tokenizing items based on their textual content, a preparation step for efficiently fine-tuning the LLM with user-item interaction data. Item tokenization addresses two issues. First, when representing items by textual content, e.g., descriptions of a product in Amazon, excessively long user-item interactions can lead to truncation and inefficiency [19], and the dilution of user signals for training an LE. Second, recent studies [34] represent semantic items via outputs of an encoder fed with textual content to support another recommendation model. However, such item embeddings are not suitable as the input of the LLM since they deviate from the LLM's token embedding distribution. Therefore, we aim to tokenize each item into the LLM's token embedding space $\mathcal{W}$ without losing its semantics and improve efficiency in learning an LE.

We achieve this goal by condensing the textual information of an item into a new item embedding space I^e belonging to $\mathcal{W}$. We refer to $i_i^e \in I^e$ as an item token, since it is not associated with an actual word, but is rather another representation of the items in $\mathcal{W}$. Given an item $i_i \in I$ with textual content, we build a sentence T and adopt an optimization-based approach that performs tokenization by updating the randomly initialized item token i_i^e for each item:

An example of T**:** S_* track is titled Live Forever from album Definitely Maybe, its artist is Oasis.

where S_* signifies a placeholder of i_i^e. T is fed to the decoder-only LLM. Figure 2a shows the process. The objective is to generate the next tokens of the sentence autoregressively, conditioned on the only-optimized item token. The final obtained item token encapsulates signals of descriptive content and serves as a semantic representation in LLM. We finally obtain the environment item

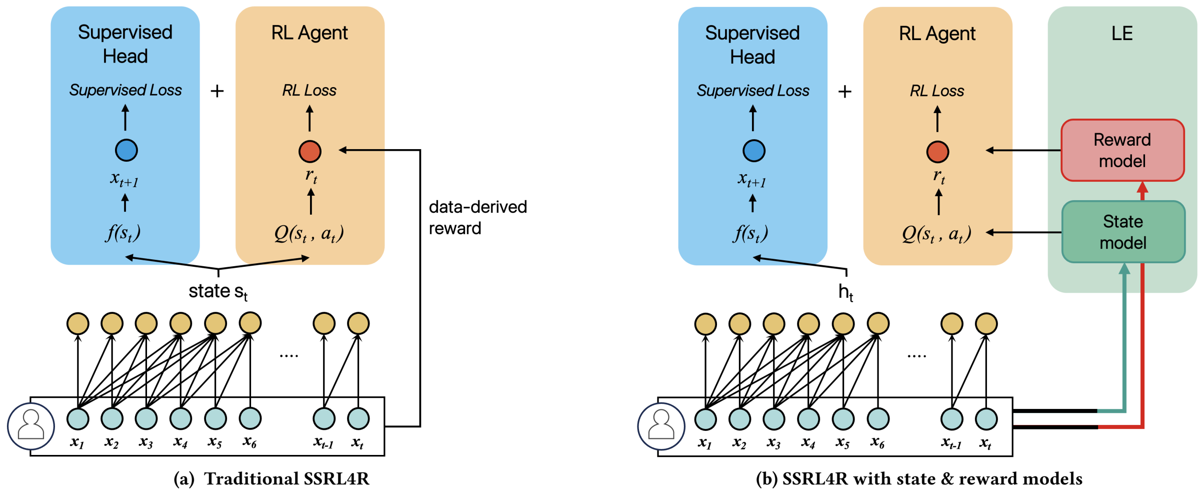


Figure 1: Self-supervised Reinforcement Learning for Recommendation (SSRL4R). (a) shows the previous offline structure, where the state for the RL agent is the hidden state from the sequential model, and the reward value is a predefined scalar. (b) shows our proposed structure, where the state is generated from a separate state model, and the reward is from a reward model.

token set I^e and the corresponding item ID embedding set I in the system, where their indices are aligned. The user-item interactions $x_{1:t}$ in the environment is denoted as $x^e_{1:t}$,where $x^e_i \in I^e (0 < i \le t)$.
•Reward Model (RM). To learn a reward model from a small amount of historical data, we fine-tune the LLM to take input as a reward prompt concatenated by a user prompt p_t and an action prompt p_{a_t} to output a scalar reward:

> **Reward Prompt** is the concatenated text of p_t and p_{a_t}: "$p_t p_{a_t}$", where
> p_t ←The user has listened to these tracks in chronological order: $x^e_{1:t}$
> p_{a_t} ←Compute the likelihood that $item_X$ be the next track to be listened to based on the listening history.

where $item_X \in \{a^+_t, a^-_t\}$. a^+_t is the truly interacted item x^e_{t+1} at next step and a^-_t is the negatively sampled one. As Figure 2b shows, the reward prompt p_t, p_{a_t} guides the LLM to generate the reward score based on the similarity between user-item token interaction and the specific action. A linear network ϕ is added as score head to generate rewards by the last hidden state. We use the Low-Rank Adaptation architecture (LoRA) [16] adapter ϕ for each Transformer block for computational efficiency[2]. The loss function for updating the RM is defined as:

$$\mathcal{L}^e_{rm} = -log[\sigma(r^e_{\theta+\phi}(p_t, p_{a^+_t}) - r^e_{\theta+\phi}(p_t, p_{a^-_t}))], \quad (1)$$

where $r^e \in [r^+_t, r^-_t]$ is the scalar reward conditioned on the reward prompt for action a^+_t and a^-_t at timestamp t, while θ and ϕ represent the trained parameters of the score head and the adapter.
•State Model (SM). We learn an SM to refine state representations from historical interactions. As shown in Figure 2b, the user-item token interactions $x^e_{1:t}$ is fed to the LLM to generate the user state representation s'^e_t at timestamp t, which is represented by the last hidden state of the outputs. Given that the state is modeled from a sequence of actions, where the difference between two consecutive states is the addition of the next interacted item in the input of the LE, there is a possibility that consecutive states could be quite similar. To ensure the model can differentiate states even when their interaction patterns are similar, we employ a contrastive loss:

$$\mathcal{L}^e_{sm} = -\frac{1}{B}\sum_{j=1}^{B} log(\sigma(s'^e_t \cdot a^+_t - s'^e_t \cdot a^j)). \quad (2)$$

where B is the batch size. For each sample, s'^e_t is the state at timestamp t generated by $x^e_{1:t}$, a^+_t is the next interacted item and a_j is the j-th action of the batch. Parameter ϕ is updated.
•Discussion. The SM loss computes the disparity between the user state and token embeddings of true and sampled actions. While the RM loss takes both user interaction and sampled actions as input, the score differences are generated by the reward values. We train RM and SM simultaneously using a shared adapter on a small amount of offline historical user data, e.g., 10% interactions (Figure 2b). After fine-tuning RM and SM, we obtain the LE that can generate user feedback for our RL-based recommendation tasks.

3.3 LE for RL-based Recommenders

We reframe the RL-based sequential recommendation as a novel LE-based Markov Decision Process (LEMDP), discussed in section 3.1. The process by which the agent interacts with the LE to obtain user feedback is shown in Figure 3, which can be represented by a tuple of $(\mathcal{S}^e, \mathcal{A}, \mathcal{P}, r^e, \gamma)$:
• State space $\mathcal{S}^e$ generated by the State Model (SM) $\mathcal{E}^e$ in LE. The set of states with the time series, modeled by the user's historical interacted item tokens $x^e_{1:t}$ via the SM and the hidden state h_t from the sequential model:

$$s'^e_t = \mathcal{E}^e(x^e_{1:t}), \quad (3)$$
$$s^e_t = s'^e_t \parallel h_t, \text{where } h_t = G(x_{1:t}) \quad (4)$$

•Action space $\mathcal{A}$. The discrete action set is comprised of the candidate items. Taking action in LEMDP means recommending items.

[2]We utilize the established prompt engineering [24, 26] and adapter techniques [1, 16].

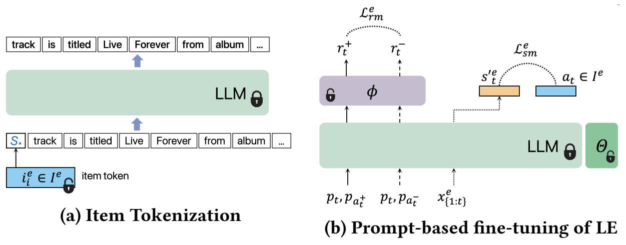


Figure 2: Our approach of adapting decoder-only LLM as Environment (LE). (a) we produce token $i_i^e \in I^e$ for item $i_i \in I$ by optimizing the objective of generating the next tokens of its textual content autoregressively. (b) we learn the LE by parameter-efficient adapters ϕ on a small subset of user data. User-item token interactions $x_{1:t}^e$, where $x_i^e \in I^e$, is the input to generate the state representation s_t^e. We enhance the state representation by comparing the similarity between the state and actions through loss $\mathcal{L}_{sm}$. Reward prompt p_t, p_{a_t} contains $x_{1:t}^e$ and action $a_t \in [a_t^+, a_t^-]$, where a_t^+ is the positive action (next interacted item), and a_t^- is the negative action (sampled uninteracted item). The action-specific reward for a user is produced by a score head θ, and the LLM is trained by comparing user preferences for actions via loss $\mathcal{L}_{rm}$.

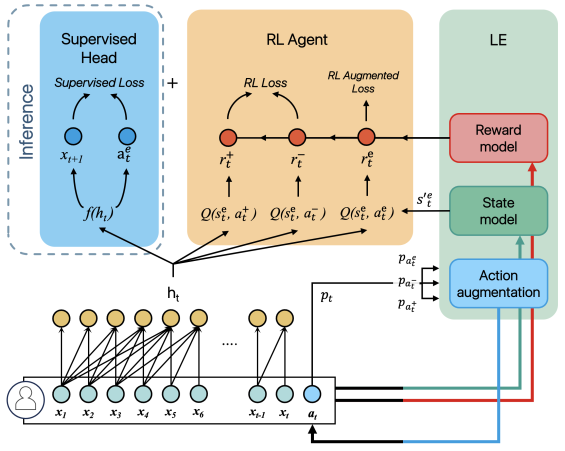


Figure 3: Structure of LEA. *Left*: the LE is applied to offline data. $(x_1^{(e)}, x_2^{(e)}, ..., x_t^{(e)})$ denotes the user-item interaction $x_{1:t}$ for the sequential model, where $x_i \in I$, and $x_{1:t}^e$ denotes the user-item token interaction for the LE, where $x_i^e \in I^e$. a_t^e is the positive action predicted by LE. *Right*: RL policy is trained via the original Q-loss and the augmented one $\mathcal{L}_{aq}$; the base sequential model is jointly trained through RL loss and the supervised loss over the original next item and the augmented one $\mathcal{L}_{ah}$ over a_t^e.

In the offline data, the action $a_t \in A$ at timestamp t is the interacted item in the next step, i.e., $a_t = x_{t+1}^e$.
•**Reward Model (RM) r^e in LE.** The reward function that returns immediate reward r_t as the state s_t^e and the action a_t taken by the agent at step t are observed:

$$r_t = r(s_t^e, a_t) = r^e(p_t, p_{a_t}) \tag{5}$$

where p_t, p_{a_t} is the reward prompt.
•**State Transition Function $\mathcal{P}$.** The transition function describes the next state from the environment given the observed action and the current state. When learning from offline data, only the positive actions affect the state.
• **Discount factor γ.** This defines the discount factor to the future rewards, where $\gamma \in [0, 1]$.

RL aims to learn a target policy $\pi_\psi(a_t|s^e)$ that maps the state $s_e \in \mathcal{S}^e$ to an action distribution $a \in \mathcal{A}$ by maximizing the expected cumulative rewards (returns), where ψ denotes the parameters:

$$\max_{\pi_\psi} \mathbb{E}_{\tau \sim \pi_\psi}[R(\tau)], \text{ where } R(\tau) = \sum_{t=0}^{|\tau|} \gamma^t r\left(s_t^e, a_t\right), \tag{6}$$

where τ denotes the trajectory of (s_t^e, a_t, s_{t+1}^e). We apply the LE to the value-based Q-learning algorithm [43] to train the target policy.

Following the objective in section 4.1, we improve the SNQN and SA2C [43] frameworks to combine supervised learning of the sequential model G and RL via the agent-LE interactions. More specifically, given the input item sequence $x_{1:t}$ and $G(\cdot)$ for self-supervised learning, the hidden representation is formulated as $h_t = G(x_{1:t})$. Then, h_t is used into a fully connected layer and a softmax function to output ranking scores over the candidate items:

$$\hat{Y}_i = softmax(\delta(W_u h_t + b_u)), \tag{7}$$

where $\hat{Y}_i = [\hat{y}_1, \hat{y}_2, ..., \hat{y}_n] \in \mathbb{R}^n$. n is the number of items, δ is the activation function, $W_u \in \mathbb{R}^{d\times n}$ are trainable parameters, and $b_u \in \mathbb{R}^n$ is the bias vector. The self-supervised part is trained via the cross-entropy loss:

$$\mathcal{L}_h = -\sum_{i=1}^{n} y_i \log(\hat{y}_i), \tag{8}$$

$y_i = 1$ if the user interacted with the i-th item in the next timestamp. Otherwise, $y_i = 0$.

Regarding the Q-learning network, we obtain the state s_t^e by the SM in LE and compute the Q-value as follows:

$$Q(s_t^e, a_t) = \delta(W_q s_t^e + b_q) \tag{9}$$

where δ is the activation function, $W_q \in \mathbb{R}^{d\times n}$ are trainable parameters, and $b_q \in \mathbb{R}^n$ is the bias vector. The one-step time difference (TD) Q-learning loss to improve SNQN is defined as:

$$\mathcal{L}_q = \underbrace{(r^e(p_t, p_{a_t^+}) + \gamma \max_{a'} Q(s_{t+1}^e, a') - Q(s_t^e, a_t^+))^2}_{L_p\text{: positive TD error}}$$
$$+ \underbrace{(r^e(p_t, p_{a_t^-}) + \gamma \max_{a'} Q(s_t^e, a') - Q(s_t^e, a_t^-))^2}_{L_n\text{: negative TD error}}, \tag{10}$$

where $a_t^+$and $a_t^-$are the positive action and sampled unobserved (negative) action at timestamp t, respectively. In our method, we only sample one negative action. We follow the assumption that taking negative actions will not affect the state. Therefore, the maximum operation is performed in $Q(s^e{}_t, a')$ other than $Q(s_{t+1}^e, a')$ in the negative TD error L_n. In SA2C, the advantage Q-value [44] is calculated to formulate the supervised loss:

$$A(s_t^e, a_t^+) = (Q(s_t^e, a^+) - Q(s_t^e, a_t^-))/2 \tag{11}$$

Then the supervised loss $\mathcal{L}_h$ is formulated as $\mathcal{L}_h \leftarrow \mathcal{L}_h \cdot A(s_t^e, a_t^+)$ for SA2C. The corresponding loss function used in the LE to directly improve the above RL frameworks is formulated as follows:

$$\mathcal{L}_c = \mathcal{L}_h + \mathcal{L}_q \tag{12}$$

3.4 Augmentation via LE

Since the recommender is trained on historical data without online exploration, both the supervised model and the RL agent can only be trained on user-item interactions that exist in the offline training data. As a result, the models may fail to estimate the value functions for unseen user feedback. We propose an LE augmentation (LEA) method to augment the historical user data for offline training. We prompt the LE to predict items a_t^e the user is likely to interact with in the next timestep, these predicted positive actions are used for both supervised learning and Q-learning. We build an augmentation prompt template constructed by the user prompt p_t and a selection prompt p_{l_t} to generate feedback:

Augmentation Prompt is the text concatenation of p_t and p_{l_t}: "$p_t p_{l_t}$". p_{l_t} ←In the list of following 5 tracks:[$list_t$], based on the history, select the number of the track that he is most likely to continue to listen to.

where $list_t$ is the token sequence of the sampled items. After an early training stage, we sample the top-5 items from the supervised head to construct $list_t$. We feed the augmentation prompt into the LE and obtain the predefined item position classification (e.g., "`first`" for the first item in the list). Then we select the item with the highest label score as the predicted positive action. The predicted item will be used to augment both the supervised learning for the sequential model and the Q-learning for RL agent:

$$\mathcal{L}_{ah} = -\sum_{j=1}^{n} y_j^e \log(\hat{y}_j), \tag{13}$$

$$\mathcal{L}_{aq} = (r^e(p_t, a_t^e) + \gamma \max_{a'} Q(s_{t+1}^e, a') - Q(s_t^e, a_t^e))^2 \tag{14}$$

$y_j^e = 1$ if the LE predicts that the user will interact with the j-th item at the next timestamp. Otherwise, $y_j^e = 0$. The max operation of $\mathcal{L}_{aq}$ is performed in $Q(s_{t+1}^e, a')$, since the positive actions will transit the current state to the next state.

•**Training.** We jointly train the supervised and Q-learning loss on the original offline user data, with the augmented supervised and Q-learning loss on the predicted positive feedback:

$$\mathcal{L} = \mathcal{L}_c + w_{ah}\mathcal{L}_{ah} + w_{aq}\mathcal{L}_{aq} \tag{15}$$

where w_{ah} and w_{aq} are the weights of augmented supervised learning loss and Q-learning loss, respectively.

•**Inference.** Our goal is to distill knowledge from LLMs into the RL-based RS. Previous methods train LLMs to act as recommenders, which are difficult to deploy in real world settings due to low inference speed. During the inference stage, we only use the sequential model with the supervised head to generate the top-k recommendations without compromising efficiency.

Algorithm 1 Training procedure of LEA

```
Input: user-item interaction sequence set $\mathcal{X}$, recommendation model $G$,
    reinforcement head $Q$, supervised head, environment LE comprised of
    state model $\mathcal{E}^e$ and reward model $r^e$, item set $I$, item token set $I^e$
Output: all parameters in the learning space $\Theta$
1: Initialize all trainable parameters
2: Create $G'$ and $Q'$ as copies of $G$ and $Q$, respectively
3: repeat
4:     Draw a mini-batch of $(x_{1:t}, a_t)$ from $\mathcal{X}$, the corresponding user-
       item token interaction $x_{1:t}^e$, sample a negative action $a_t^-$, augmented
       action $a_t^e$
5:     $s^e{}_t = G(x_{1:t})||\mathcal{E}^e(x_{1:t}^e)$, $s^e{}_{t+1} = G(x_{1:t+1})||\mathcal{E}^e(x_{1:t+1}^e)$
6:     $s^{e\prime}_t = G'(x_{1:t})||\mathcal{E}^e(x_{1:t}^e)$ , $s^{e\prime}_{t+1} = G'(x_{1:t+1})||\mathcal{E}^e(x_{1:t+1}^e)$
7:     Generate random variable $z \in (0,1)$ uniformly
8:     if $z \leq 0.5$ then
9:         $a^* = \mathrm{argmax}_a\, Q(s^e{}_{t+1}, a)$
10:        Calculate $\mathcal{L}_q$ and $\mathcal{L}_{aq}$ by $Q(s^e{}_t, a_t^{(e)})$, $Q'(s^{e\prime}_{t+1}, a^*)$
11:        Calculate $\mathcal{L}_h$ and $\mathcal{L}_{ah}$ by $G$
12:    else
13:        $a^* = \mathrm{argmax}_a\, Q'(s^{e\prime}_{t+1}, a)$
14:        Calculate $\mathcal{L}_q$ and $\mathcal{L}_{aq}$ by $Q'(s^{e\prime}_t, a_t^{(e)})$, $Q(s^e{}_{t+1}, a^*)$
15:        Calculate $\mathcal{L}_h$ and $\mathcal{L}_{ah}$ by $G'$
16:    end if
17:    Perform updates by $\nabla_\Theta \mathcal{L}$
18: until converge
19: Return all parameters in $\Theta$
```

Table 1: Dataset statistics. seq. denotes sequence, inter. denotes interaction.

Dataset	#item	#seq.	#inter.	Item content
LFM	18,927	11,073	146,255	title; album; artist
Industry	5,814	10,935	71,872	title; category; brand; description

3.5 Discussion

We compare LEA with two SOTA RL frameworks: SNQN and SA2C [43]. The difference between LEA and the direct application of LE in existing RL frameworks is that it implements an augmentation method through the LE. We illustrate the training procedure of LEA utilizing LE as both state and reward models in Algorithm 1, where double Q-learning [9] is adopted to alternatively train two copies of trainable Q-networks. The function of LE as a state model or a reward model will be discussed in the experimental section by directly replacing the corresponding part of existing RL methods.

4 EXPERIMENTAL SETUP

4.1 Research Questions

We detail the experimental setup for validation of our augmentation method via the LLM-based environment (LEA). We aim to answer the following research questions:

RQ1: How does LEA, integrated with feedback from LE, perform compared with the existing RL-based recommendation frameworks?

RQ2: Does action augmentation affect the performance of LEA?

RQ3: How does the user feedback from LE, i.e., rewards and state, affect the RL-based recommenders?

RQ4: Do various fine-tuning strategies for LE affect its performance, i.e., item tokenization, data scale, state model loss, the use of LLMs?

4.2 Datasets

We perform experiments on two real-world datasets (Table 1): **LFM** [36] and **Industry** [31]. The LFM dataset was collected from the music streaming platform Last.fm[3] and contains more than one billion listening events by 120k users. We sample a subset of 3-day listening events on tracks as experimental data. The textual description for each track (item) includes its title, album and the artist. We filter out sequences with fewer than three interactions and tracks listened to less than three times, obtaining a dataset comprising 18,927 items and 11,073 user sequences, with a total of 146,255 interactions. The second Industry dataset is from the publicly available Amazon review dataset[4] of the Industrial and Scientific category. The textual description for each product (item) includes the title, category, and product description. Similarly, we filter out sequences with fewer than three interactions and products reviewed less than three times, yielding a dataset containing 5,814 items and 10,93 sequences, with 71,872 interactions.

4.3 Baselines

We compare LEA with two state-of-the-art RL frameworks: **SNQN** and **SA2C** [42, 43] which are introduced in section 3.3, under two backbone sequential models: **GRU4Rec** [12], the first sequential recommendation RS based on RNN, and **SASRec** [21], a renowned sequential recommendation model based on self-attention. **Normal** denotes the original sequential models with normal supervised loss. By applying LEA method, we obtain the following strategies: (1) **LEAR** denotes training RL framework with the rewards from LE. (2) **LEAS** denotes that the state for the RL component is derived from LE. (3) **LEASR** denotes that both state and rewards are from the LE. In SA2C, the advantage calculation allows the training of the RL policy to affect the updates of the sequential model when only s'^{e}_{t} is used, therefore, we introduce an additional method (4) **LEAS′R** for SA2C that the state for RL agent is solely from LE, i.e., $s^{e}_{t} = s'^{e}_{t}$ in Eq. 3. In ablation studies, we replace each feedback in the compared RL framework with the corresponding one from LE to examine the LE environment: (1) **LER** and (2) **LES** denote the rewards and states in the baseline RL methods are generated by LE, respectively. Moreover, (3) **LEA** denotes only the augmented training strategy applied to the baseline models.

4.4 Metrics

We apply two commonly used metrics: Hit Ratio (HR@k) [43] and Normalized Discounted Cumulative Gain (NDCG@k) [20] to measure the relevance of recommended items for the evaluated sessions, where $k \in \{5, 10, 20\}$. HR@k evaluates whether the ground-truth item is in the top-k positions of the recommendation list. NDCG@k, $k \in \{5, 10, 20\}$ measures the rank of the ground-truth item in the top-k recommendation list. We report the average results over all interactions of the test sequences.

4.5 Implementation Details

For *training the LE*, we perform 300 iterations with a learning rate of $5e^{-3}$ for optimization-based item tokenization. The sequence length for the fine-tuning of LE for state generation is set to 10. The epochs for all datasets is 10 with a batch size of 20. We employ the Adam optimizer for fine-tuning. The batch size and epochs is 10 for all datasets. For efficient training, we only use a 10% subset of the original dataset to obtain the LE. For all *compared models and our LEA methods*, the length of sequences is set to 10, and a padding token is added to shorter sequences for all datasets. We train all models and variants with the Adam optimizer [37, 42]. The mini-batch size is 100 for LFM and Industry. The learning rates are $1e^{-3}$ for LFM, and $2e^{-3}$ for Industry. We evaluate the validation set every 1,000 and 500 steps of updates on LFM and Industry, respectively. All experiments are performed in one H100 GPU. To ensure a fair comparison, the item embedding size and hidden size are set to 64 for all models. Since there is only one kind of behaviour, we set the reward to 1 for all interactions in original SNQN and SA2C [43].

[3]http://www.last.fm/api

[4]https://nijianmo.github.io/amazon

5 EXPERIMENTAL RESULTS

5.1 Performance Comparison (RQ1)

The performance comparison of the two public datasets is shown in Table 2. We observe the following: (1) In contrast to standard sequential models, our RL-based methods consistently outperform across all datasets. This demonstrates that the integration of augmented RL and supervised learning, with feedback from LE, contributes to enhancing recommendation performance. (2) LEA further improves the performance over the RL frameworks of SNQN and SA2C. This suggests that the augmentation method improves the learning performance on the supervised component, as well as the RL head. (3) On the LFM dataset, LEASR achieves the highest results on both HR and NDCG metrics, across two frameworks, outperforming other strategies. This indicates that the state and reward obtained from LE, when combined, can significantly improve the performance of RL-based recommenders. On the Industry dataset, the overall relative improvement of the RL method by the LEA approach is not as pronounced as on the LFM dataset. This could be attributed to the LLM potentially incorporating more knowledge about music, an artefact due to its exposure to such content during pre-training, compared to product review data from the Amazon dataset. In conclusion, the proposed LEA methods are effective to improve the recommendation performance of RL frameworks.

5.2 Effect of Action Augmentation (RQ2)

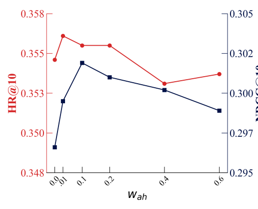
We assess the impact of the LE generated augmented positive actions. We first examine the influence of the feedback on supervised learning (sv) and Q-learning (q) separately. Here, sv+q denotes that both components are augmented. Results are reported on two datasets utilizing the SASRec as the backbone. As Table 3 shows, the first pair is the original sequential models (Normal) and its sv augmented version. We can see that when the augmented samples are taken as additional supervised labels, the recommendation performance increases in all cases and datasets. This confirms the LE's capability to generate high-quality positive samples for offline training. The second comparison is considering the SA2C framework and augmentation on sv, q, and sv+q. We observe that the sv augmentation produces the same trends as in the first pair we compare. Furthermore, the improved performance of q over SA2C indicates the effectiveness of augmenting the Q-loss with the positive feedback. Notably, when combining sv and q augmentations,

Table 2: Performance on LFM and Industry datasets. NG is short for NDCG. Boldface denotes the highest score and the second-best scores are marked with __. $*$ denotes the significance p-value < 0.01 compared with second best baseline.

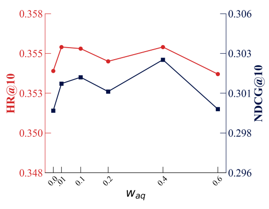
Model		LFM						Industry					
		HR@5	NG@5	HR@10	NG@10	HR@20	NG@20	HR@5	NG@5	HR@10	NG@10	HR@20	NG@20
SASRec	Normal	0.2798	0.2324	0.3115	0.2646	0.3271	0.2685	0.0788	0.0684	0.1171	0.0909	0.1374	0.0961
	SNQN	0.3001	0.2602	0.3337	0.2828	0.3459	0.2855	0.0926	0.0738	0.1202	0.0913	0.1419	0.0968
	LEAR	0.3286	0.2879	0.3504	0.2951	0.3721	0.3005	0.1027	0.0864	0.1232	0.0929	0.1514	0.0983
	LEAS	0.3298	0.2905	0.3529	0.2981	**0.3773***	0.3039	0.1047	0.0856	**0.1281***	0.0931	**0.1581***	**0.1008***
	LEASR	**0.3323***	**0.2914***	**0.3533***	**0.2982***	0.3772	**0.3042***	**0.1059***	**0.0866***	0.1245	**0.0933***	0.1538	0.1001
	SA2C	0.2902	0.2501	0.3372	0.2851	0.3516	0.2885	0.0931	0.0735	0.1205	0.0912	0.1416	0.0966
	LEAR	0.3269	0.2573	0.3429	0.2942	0.3571	0.2977	0.1037	0.0861	0.1231	0.0922	0.1525	0.0996
	LEAS	0.3302	0.2907	0.3533	0.2982	0.3773	0.3043	**0.1055***	0.0868	**0.1278***	0.0938	0.1528	0.1002
	LEAS'R	0.3298	0.2853	0.3511	0.2931	0.3706	0.2981	0.1036	**0.0874***	0.1264	**0.0948***	0.1521	**0.1012***
	LEASR	**0.3325***	**0.2926***	**0.3572***	**0.3011***	**0.3801***	**0.3072***	0.1042	0.0862	0.1261	0.0932	**0.1533***	0.1001
GRU4Rec	Normal	0.2757	0.2536	0.2934	0.2594	0.3113	0.2641	0.0901	0.0751	0.1128	0.0824	0.1399	0.0893
	SNQN	0.2781	0.2526	0.2931	0.2575	0.3103	0.2619	0.0891	0.0727	0.1083	0.0896	0.1392	0.0931
	LEAR	0.3211	0.2878	0.3407	0.2941	0.3574	0.2983	0.0972	0.0887	0.1205	0.0907	0.1466	0.0975
	LEAS	0.2881	0.2596	0.3134	0.2679	0.3359	0.2735	0.0976	0.0826	0.1232	0.0909	0.1525	0.0982
	LEASR	**0.3234***	**0.2909***	**0.3464***	**0.2983***	**0.3667***	**0.3034***	**0.0978***	**0.0892***	**0.1238***	**0.0915***	**0.1557***	**0.0998***
	SA2C	0.2896	0.2743	0.3173	0.2797	0.3337	0.2838	0.0865	0.0705	0.1145	0.0842	0.1398	0.0913
	LEAR	0.3142	0.2782	0.3333	0.2844	0.3518	0.2891	0.0954	0.0816	0.1191	0.0883	0.1421	0.0948
	LEAS	0.3139	0.2846	0.3364	0.2921	0.3588	0.2977	0.0986	0.0814	0.1224	0.0891	0.1531	0.0968
	LEAS'R	0.3048	0.2795	0.3214	0.2849	0.3393	0.2894	0.0959	0.0818	0.1195	0.0889	0.1416	0.0953
	LEASR	**0.3281***	**0.2926***	**0.3502***	**0.2998***	**0.3711***	**0.3051***	**0.0981***	**0.0828***	**0.1234***	**0.0892***	**0.1559***	**0.0969***

Table 3: Effect of action augmentation on supervised learning (sv) and Q-learning (q). NG is short for NDCG. Boldface denotes the highest scores.

Model		LFM						Industry					
		HR@5	NG@5	HR@10	NG@10	HR@20	NG@20	HR@5	NG@5	HR@10	NG@10	HR@20	NG@20
SASRec	Normal	0.2798	0.2324	0.3115	0.2646	0.3271	0.2685	0.0788	0.0684	0.1171	0.0909	0.1374	0.0961
	sv	**0.2899**	**0.2493**	**0.3263**	**0.2757**	**0.3408**	**0.2793**	**0.0928**	**0.0759**	**0.1202**	**0.0931**	**0.1418**	**0.0985**
	SA2C	0.2902	0.2501	0.3372	0.2851	0.3516	0.2885	0.0931	0.0735	0.1205	0.0912	0.1416	0.0966
	sv	0.3295	0.2898	0.3541	0.2978	**0.3755**	0.3032	0.1026	0.0847	0.1241	0.0916	0.1554	0.0995
	q	0.3314	0.2897	0.3535	0.2969	0.3735	0.3019	0.1022	0.0836	0.1234	0.0904	0.1535	0.0981
	sv+q	**0.3322**	**0.2915**	**0.3547**	**0.2987**	0.3754	**0.3041**	**0.1032**	**0.0859**	**0.1252**	**0.0929**	**0.1558**	**0.1006**



(a)



(b)

Figure 4: LEASR with different weights of (a) supervised learning and (b) Q-learning augmentation loss on the LFM dataset.

sv+q on the Industry dataset achieves better performance than the individual methods, demonstrating the potential of joint augmentation. On the LFM dataset, sv+q surpasses the baseline and other augmentation methods in most cases, which suggests that the predicted positive samples may require specific reward configurations, distinct from the truth action, to fully realize the benefit.

Furthermore, we investigate the impact of w_{ah} and w_{aq} - the weights of augmentation $\mathcal{L}_{ah}$ and $\mathcal{L}_{aq}$ - in LEASR under the SA2C framework. The results on the LFM dataset are shown in Figures 4a and 4b, respectively. We can see that the performance initially improves with the increase of w_{ah}. The best performance is achieved when w_{ah} is 0.1. As w_{ah} constantly increases, the performance drops gradually. This suggests that over-weighing augmented actions incrementally diminishes the effect of the true labels, leading to a proportional decrease in performance. We achieve the best performance for w_{aq} = 0.01. As w_{aq} increases, we observe an unstable ascending trend in performance. When beyond a certain threshold, the incremental gains drop. The optimal value of w_{aq} is 0.01. This shows that while the augmentation of Q-learning through predicted action is relatively stable, beyond a certain point further augmentation does not produce proportional performance benefits.

5.3 Effect of LE (RQ3)

5.3.1 Effect of LE as reward model. We conduct experiments via the RL-based RS by SA2C framework using the SASRec backbone to see the effect of LE as a reward model (RM). The results on two

Table 4: Effect of LE as reward function. NG is short for NDCG. Boldface denotes the highest score.

	SAS	HR@5	NG@5	HR@10	NG@10	HR@20	NG@20
LFM	SA2C	0.2902	0.2501	0.3372	0.2851	0.3516	0.2885
	LER	**0.3252**	**0.2714**	**0.3401**	**0.2928**	**0.3539**	**0.2961**
	LEA	0.3296	0.2802	0.3405	0.2877	0.3548	0.2912
	LEAR	**0.3298**	**0.2833**	**0.3429**	**0.2942**	**0.3571**	**0.2977**
Industry	SA2C	0.0935	0.0775	0.1145	0.0842	0.1427	0.0913
	LER	**0.0994**	**0.0841**	**0.1219**	**0.0928**	**0.1432**	**0.0982**
	LEA	0.1032	0.0859	**0.1252**	**0.0929**	**0.1558**	**0.1006**
	LEAR	**0.1037**	**0.0861**	0.1231	0.0922	0.1525	0.0996

Table 5: Effect of LE as state model. NG is short for NDCG. Boldface denotes the highest score.

Model		LFM					
		HR@5	NG@5	HR@10	NG@10	HR@20	NG@20
SAS	SA2C	0.2902	0.2501	0.3372	0.2851	0.3516	0.2885
	LES′	0.3202	0.2804	0.3504	0.2956	0.3711	0.3008
	LES	**0.3311**	**0.2913**	**0.3539**	**0.2988**	**0.3734**	**0.3037**
GRU	SA2C	0.2896	0.2743	0.3173	0.2797	0.3337	0.2838
	LES′	0.2973	0.2597	0.3141	0.2641	0.3345	0.2785
	LES	**0.3128**	**0.2818**	**0.3321**	**0.2891**	**0.3518**	**0.2935**

datasets are reported in Table 4. LER method replaces the predefined reward of SA2C as the reward return from RM. We observe that LER outperforms SA2C across all cases, which substantiates the effectiveness of the RM. We further note that the smaller the k the greater the improvement in the top-k recommendation list, suggesting that a well-designed reward model can effectively prioritize correct actions to higher ranks. LEA incorporates the augmentation technique into the SA2C, whereas LEAR applies both the augmentation and the reward model. It is evident that LEAR generally outperforms LEA. This superior performance can be attributed to the common advantages gained from the augmentation feedback provided by the LE, which potentially obscures the enhancements derived from the RL components. However, the superiority of LEAR still holds the second phenomenon as described above.

5.3.2 Effect of LE as state model. We conduct experiments within the SA2C framework on the LFM dataset between three methods: (1) the baseline SA2C, relying on the hidden state from the sequential model, i.e., h_t in Eq. 4, (2) LES′ that uses the state from SM, i.e., s'^{e}_{t} in Eq. 3, and (3) LES that leverages both the state from LE and hidden state from the sequential model i.e., s^e_t in Eq. 4. Table 5 indicates that LES has the highest performance across all backbones and metrics, while LES′ outperforms SA2C in most cases. This demonstrates that leveraging effectively all state representations can introduce performance gains in RL-based RS.

5.4 Effect of learning strategies for LE (RQ4)

5.4.1 Effect of data scale. Despite optimizing the LE on a subset of the training data, we study the effect of data scaling to provide insights into how to more efficiently obtain LE in the future. We train the LE on a portion of the LFM dataset with varying portions: 5%, 10%, 20%, 50%, and 100%. The results on the LEASR method are shown in Figure 5a.

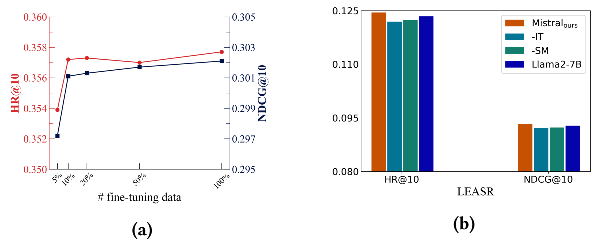


Figure 5: Effect of scaling the training data for LE. The result of LEASR on the LFM dataset.

These indicate that as the data portion increases to 10%, we observe a notable enhancement in performance, demonstrating that a bigger dataset enables the LE to capture complex user-item interactions and return effective user feedback. The scale we set is 10%. As the scale continuously increases, the effect of LE grows slowly and fluctuates within a range. This demonstrates that our method is capable of efficiently optimizing LLMs with a small data portion, producing a state, reward model, and augmented feedback that are scalable to the entire user data.

5.4.2 Effect of item tokenization . We compare the performance of LE utilizing item tokenization against representing items by their textual content (-IT), where user-item interactions are generated from sequences of item titles. We utilize LEASR to examine the impact on LE. Figure 5b shows the results, which suggest that applying item tokenization surpasses the contentenation method. This underscores that the items tokens, represent the semantic information in a concise way, and hence contribute to an effective learning of interaction sequences.

5.4.3 Effect of state loss . To observe the impact of the loss $\mathcal{L}^e_{sm}$ on state generation, we train LE without (-SM) contrastive learning and show the impact by the LES method. Figure 5b shows a decrease in performance when $\mathcal{L}^e_{sm}$ is omitted. This implies that the SM loss plays a critical role in refining the state representation, which is integral to the robustness and accuracy of the LE.

5.4.4 Effect of LLMs. We show the effect of different LLMs on LE by comparing Mistral with Llama2-7B, a model with a 7B parameter count similar to Mistral but launched earlier. Figure 5b shows that the performance of Mistral is slightly higher than Llama2-7B. We argue that as LLMs evolve, there is the potential for continuous improvement in building LE.

6 CONCLUSION AND FUTURE WORK

The use of LLM's as reward, state and action models is shown to be a promising direction in the context of RL driven recommender systems. In contrast to current adaptations of LLMs used as recommenders, the design of LE enhances RL-based sequential models without requiring significant increases in inference computations, hence our model is very easy to deploy in real world recommendation settings. Future directions of research can include the shaping of different rewards and the inclusion of additional user behaviour data in the LLM to create better state representations. We also intent to investigate alternative fine-tuning methods for both reward state and action generation. Moreover the use of more powerful LLM's can lead to even better results.

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. *arXiv preprint arXiv:2305.00447* (2023).

[2] Jin Chen, Zheng Liu, Xu Huang, Chenwang Wu, Qi Liu, Gangwei Jiang, Yuanhao Pu, Yuxuan Lei, Xiaolong Chen, Xingmei Wang, et al. 2023. When large language models meet personalization: Perspectives of challenges and opportunities. *arXiv preprint arXiv:2307.16376* (2023).

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[4] 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).

[5] 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).

[6] Junchen Fu, Fajie Yuan, Yu Song, Zheng Yuan, Mingyue Cheng, Shenghui Cheng, Jiaqi Zhang, Jie Wang, and Yunzhu Pan. 2024. Exploring adapter-based transfer learning for recommender systems: Empirical studies and practical insights. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*. 208–217.

[7] 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*. 299–315.

[8] Wes Gurnee and Max Tegmark. 2024. Language Models Represent Space and Time. In *The Twelfth International Conference on Learning Representations*. https://openreview.net/forum?id=jE8xbmvFin

[9] Hado Hasselt. 2010. Double Q-learning. *Advances in neural information processing systems* 23 (2010).

[10] Shwai He, Liang Ding, Daize Dong, Miao Zhang, and Dacheng Tao. 2022. Sparseadapter: An easy approach for improving the parameter-efficiency of adapters. *arXiv preprint arXiv:2210.04284* (2022).

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

[12] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939* (2015).

[13] Daocheng Hong, Yang Li, and Qiwen Dong. 2020. Nonintrusive-sensing and reinforcement-learning based adaptive personalized music recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1721–1724.

[14] Yupeng Hou, Shanlei Mu, Wayne Xin Zhao, Yaliang Li, Bolin Ding, and Ji-Rong Wen. 2022. Towards universal sequence representation learning for recommender systems. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 585–593.

[15] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *International Conference on Machine Learning*. PMLR, 2790–2799.

[16] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685* (2021).

[17] Yujing Hu, Qing Da, Anxiang Zeng, Yang Yu, and Yinghui Xu. 2018. Reinforcement learning to rank in e-commerce search engine: Formalization, analysis, and application. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 368–377.

[18] Zhiqiang Hu, Yihuai Lan, Lei Wang, Wanyu Xu, Ee-Peng Lim, Roy Ka-Wei Lee, Lidong Bing, and Soujanya Poria. 2023. LLM-Adapters: An Adapter Family for Parameter-Efficient Fine-Tuning of Large Language Models. *arXiv preprint arXiv:2304.01933* (2023).

[19] Wenyue Hua, Shuyuan Xu, Yingqiang Ge, and Yongfeng Zhang. 2023. How to Index Item IDs for Recommendation Foundation Models. *arXiv preprint arXiv:2305.06569* (2023).

[20] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems (TOIS)* 20, 4 (2002), 422–446.

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

[22] Minae Kwon, Sang Michael Xie, Kalesha Bullard, and Dorsa Sadigh. 2023. Reward Design with Language Models. In *The Eleventh International Conference on Learning Representations*. https://openreview.net/forum?id=10uNUgI5Kl

[23] Kenneth Li, Aspen K Hopkins, David Bau, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. 2023. Emergent World Representations: Exploring a Sequence Model Trained on a Synthetic Task. In *The Eleventh International Conference on Learning Representations*. https://openreview.net/forum?id=DeG07_TcZvT

[24] Lei Li, Yongfeng Zhang, and Li Chen. 2023. Prompt distillation for efficient llm-based recommendation. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. 1348–1357.

[25] 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).

[26] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *Comput. Surveys* 55, 9 (2023), 1–35.

[27] Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Lam Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2021. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *arXiv preprint arXiv:2110.07602* (2021).

[28] Bonan Min, Hayley Ross, Elior Sulem, Amir Pouran Ben Veyseh, Thien Huu Nguyen, Oscar Sainz, Eneko Agirre, Ilana Heintz, and Dan Roth. 2023. Recent advances in natural language processing via large pre-trained language models: A survey. *Comput. Surveys* 56, 2 (2023), 1–40.

[29] Omar Moling, Linas Baltrunas, and Francesco Ricci. 2012. Optimal radio channel recommendations with explicit and implicit feedback. In *Proceedings of the sixth ACM conference on Recommender systems*. 75–82.

[30] Jianmo Ni, Gustavo Hernández Ábrego, Noah Constant, Ji Ma, Keith B Hall, Daniel Cer, and Yinfei Yang. 2021. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. *arXiv preprint arXiv:2108.08877* (2021).

[31] Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*. 188–197.

[32] 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.

[33] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research* 21, 1 (2020), 5485–5551.

[34] Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan H Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Q Tran, Jonah Samost, et al. 2023. Recommender Systems with Generative Retrieval. *arXiv preprint arXiv:2305.05065* (2023).

[35] Zhaochun Ren, Na Huang, Yidan Wang, Pengjie Ren, Jun Ma, Jiahuan Lei, Xinlei Shi, Hengliang Luo, Joemon Jose, and Xin Xin. 2023. Contrastive State Augmentations for Reinforcement Learning-Based Recommender Systems. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 922–931.

[36] Markus Schedl. 2016. The lfm-1b dataset for music retrieval and recommendation. In *Proceedings of the 2016 ACM on international conference on multimedia retrieval*. 103–110.

[37] Dusan Stamenkovic, Alexandros Karatzoglou, Ioannis Arapakis, Xin Xin, and Kleomenis Katevas. 2022. Choosing the Best of Both Worlds: Diverse and Novel Recommendations through Multi-Objective Reinforcement Learning. In *Proc. of the Fifteenth ACM International Conference on Web Search and Data Mining*. 957–965.

[38] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the eleventh ACM international conference on web search and data mining*. 565–573.

[39] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971* (2023).

[40] Jie Wang, Fajie Yuan, Mingyue Cheng, Joemon M Jose, Chenyun Yu, Beibei Kong, Xiangnan He, Zhijin Wang, Bo Hu, and Zang Li. 2022. Transrec: Learning transferable recommendation from mixture-of-modality feedback. *arXiv preprint arXiv:2206.06190* (2022).

[41] Pengfei Wang, Yu Fan, Long Xia, Wayne Xin Zhao, ShaoZhang Niu, and Jimmy Huang. 2020. KERL: A knowledge-guided reinforcement learning model for sequential recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 209–218.

[42] Xin Xin, Alexandros Karatzoglou, Ioannis Arapakis, and Joemon M Jose. 2020. Self-supervised reinforcement learning for recommender systems. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 931–940.

[43] Xin Xin, Alexandros Karatzoglou, Ioannis Arapakis, and Joemon M Jose. 2022. Supervised advantage actor-critic for recommender systems. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*. 1186–1196.

[44] Xin Xin, Tiago Pimentel, Alexandros Karatzoglou, Pengjie Ren, Konstantina Christakopoulou, and Zhaochun Ren. 2022. Rethinking Reinforcement Learning for Recommendation: A Prompt Perspective. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1347–1357.

[45] Fajie Yuan, Alexandros Karatzoglou, Ioannis Arapakis, Joemon M Jose, and Xiangnan He. 2019. A simple convolutional generative network for next item recommendation. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. 582–590.

[46] Fajie Yuan, Alexandros Karatzoglou, Ioannis Arapakis, Joemon M Jose, and Xiangnan He. 2019. A simple convolutional generative network for next item recommendation. In *Proceedings of the twelfth ACM international conference on web search and data mining*. 582–590.

[47] Gangyi Zhang. 2023. User-Centric Conversational Recommendation: Adapting the Need of User with Large Language Models. In *Proceedings of the 17th ACM Conference on Recommender Systems*. 1349–1354.

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

[49] Xiangyu Zhao, Liang Zhang, Zhuoye Ding, Long Xia, Jiliang Tang, and Dawei Yin. 2018. Recommendations with negative feedback via pairwise deep reinforcement learning. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 1040–1048.