ECAI 2024
U. Endriss et al. (Eds.)

doi:10.3233/FAIA240742

SUBER: An RL Environment with Simulated Human Behavior for Recommender Systems

Nathan Corecco[a,1], Giorgio Piatti[a,1], Luca A. Lanzendörfer[a,*], Flint Xiaofeng Fan[b,c] and Roger Wattenhofer[a]

[a]ETH Zurich
[b]National University of Singapore
[c]A*STAR Singapore

Abstract. Reinforcement learning (RL) has gained popularity in the realm of recommender systems due to its ability to optimize long-term rewards and guide users in discovering relevant content. However, the successful implementation of RL in recommender systems is challenging because of several factors, including the limited availability of online data for training on-policy methods. This scarcity requires expensive human interaction for online model training. Furthermore, the development of effective evaluation frameworks that accurately reflect the quality of models remains a fundamental challenge in recommender systems. To address these challenges, we propose a comprehensive framework for synthetic environments that simulate human behavior by harnessing the capabilities of large language models (LLMs). We complement our framework with in-depth ablation studies and demonstrate its effectiveness with experiments on movie and book recommendations. Using LLMs as synthetic users, this work introduces a modular and novel framework to train RL-based recommender systems. The software, including the RL environment, is publicly available on https://github.com/SUBER-Team/SUBER.

1 Introduction

In an age defined by the ubiquitous presence of digital platforms in both leisure and commerce, recommender systems have emerged as instrumental tools in guiding user choices. From Netflix tailoring movie suggestions to match the cinematic tastes of users to Amazon presenting personalized products lists to shoppers, recommendation systems are the engines driving enhanced user experiences and the engagement of the platform [41, 3].

Reinforcement Learning (RL), with its principles rooted in learning by interaction, provides a compelling approach to dynamically and adaptively tailor recommendations. Recommender systems should take into account both short- and long-term rewards and direct the interests of users towards appropriate recommendations. An increasing body of research has investigated the use of RL in recommender systems [18, 9, 23, 1, 22]. Although promising, the use of RL for recommendation systems comes with its own set of challenges:

Data Availability: RL algorithms require a significant amount of data from interactions with the environment to learn effective policies. However, in the case of recommender systems, users may quickly abandon the service if they receive random or irrelevant recommendations. This makes it impractical to collect the large amount of data needed to train an RL model without compromising the user experience [50].

Unknown user model: In RL, a reward function is crucial to allow the model to learn effectively. In the context of recommender systems, designing an appropriate synthetic reward function that accurately reflects user satisfaction or preferences can be challenging due to the complexity of modeling human behavior [10, 39].

Model evaluation: A key challenge in recommender systems is the evaluation of models without directly interacting with real users, thus avoiding any potential negative impact on the user experience. On the other hand, evaluating on offline data does not guarantee good recommendation performance in the real world [38, 13].

In this work, we propose a "Simulated User Behavior for Recommender Systems" (SUBER), a novel framework for recommender systems to address the aforementioned challenges. SUBER is a framework for synthetic environments that use Large Language Models (LLM) at its core. SUBER leverages recent advances in LLMs to simulate human behavior [31, 4]. Furthermore, by training on large amounts of data, LLMs have obtained inherent knowledge about movies, books, and various other objects. These strengths, the ability to mimick human behavior coupled with vast knowledge about humanity, uniquely position LLMs as a powerful tool to simulate users in synthetic environments for recommender systems. Therefore, SUBER serves as a versatile playground for researchers, allowing them to experiment with different LLM configurations, fine-tune user specifications, and improve their RL strategies. Our contributions can be summarized as follows:

- We introduce SUBER, a versatile framework for training and evaluating RL-based recommender systems. Our framework includes a gym environment with an LLM designed to simulate human behavior and rate recommended items accordingly.
- We conduct extensive ablation studies to assess the impact of each component in our framework. Moreover, we present findings across multiple LLM families, revealing their influence on the environment's performance and highlighting their effectiveness in replicating human behavior for item recommendations.
- We experimentally validate our environment using both movie and book recommendation settings. Additionally, we have made all code available as open-source.

* Corresponding Author. Email: lanzendoerfer@ethz.ch
[1] Equal contribution.

Table 1. Comparison of simulation environments for recommender systems. We list whether the user and item datasets are real or synthetic. Simulation Engine indicates the different approaches used. For the evaluation strategy, we distinguish between offline evaluation in the original dataset used to train the simulator, online testing on a platform, sanity checks, and case studies.

Simulators	User dataset	Item dataset	Simulation engine	Evaluation strategy
Adversarial [10]	Real	Real	GAN	Offline
VirtualTaobao [39]	Real	Real	GAN	Online
RL4RS [47]	Real	Real	Transformer	Online
KuaiSim [51]	Real	Real	Transformer	Offline
RecoGym [34]	Synthetic	Synthetic	Statistical modelling	Sanity checks
RecoSim [17]	Synthetic	Synthetic	Statistical modelling	Case studies
SUBER (our)	Synthetic	Real	LLM	Sanity checks & case studies

2 Related Work

RL for Recommender Systems. Platforms such as YouTube [18, 9] and BytePlus [23] are two of many recent successful examples of training and evaluating recommender systems with online data. Traditional and neural recommender systems and have been extensively researched in the past three decades [15, 42, 5, 40, 24, 49]. However, since our work focuses on RL in recommender systems (RL4Rec), we limit the related work to this area of research. Although RL4Rec has been the subject of several studies, most of the work has been based primarily on training and evaluation based on offline datasets [1, 22]. As indicated by Afsar et al. [1], online assessment is the preferred approach for evaluation. However, it presents significant challenges with respect to complexity and expense. In contrast, offline evaluation takes place in a static and biased environment. Therefore, Afsar et al. call for the creation of a versatile simulator for RL4Rec similar in nature to OpenAI's Gym for conventional RL tasks [6]. Additional challenges exist in the wider domain of RL, specifically regarding issues related to off-policy learning and offline policy evaluation, which become even more complex when incorporated into recommender systems [33, 14, 21].

Notable efforts have been made to address the limitations of offline learning in recommender systems. To this end, many simulation environments for recommender systems have been developed. Rohde et al. [34] presented RecoGym, a synthetic environment that addresses exploding variance by simulating user responses to different recommendation strategies. RecSim [17] is a customizable synthetic simulation platform that incorporates various assumptions about user preferences, item familiarity, user latent states and dynamics, and choice models. Chen et al. [10] proposed a generator that captures the underlying distribution of historical user interactions and learns to generate realistic interactions. Extending this idea, Shi et al. [39] proposed VirtualTaobao, a virtual shopping environment, and demonstrated the superiority of policies developed in this framework over traditional supervised techniques in real-world settings. Wang et al. [47] introduced the RL4RS dataset to address the lack of validated simulation environments and advanced evaluation methods in RL-based recommender system research. The dataset is collected from a NetEase game and anonymized through a three-step process. Zhao et al. [51] propose KuaiSim, a versatile environment that provides user feedback with multi-behavior and cross-session responses, supporting three tasks: request-level list-wise recommendation task, whole-session-level sequential recommendation task, and cross-session-level retention optimization task. Unlike previous approaches, our work leverages natural language by using LLMs to simulate user behavior. In addition, our framework is not dataset dependent, and therefore, the set of users and items are not restricted to specific domains.

Large Language Models. There have been significant recent advances in the field of LLMs. These models are primarily based on the transformer architectures introduced by Vaswani et al. [46] and have continued to grow in size, capability, and performance. The Generative Pre-trained Transformer (GPT) series by OpenAI [7, 30] is one of the most notable developments in this area, demonstrating the immense potential and scalability of transformer-based models. The recent release of foundation language models such as Llama-1 and Llama-2 [43, 44], has democratized the access to these large LLMs. This has paved the way for the creation of instruction-following models such as Vicuna [52] and Mistral [19]. Meanwhile, numerous efforts have focused on optimizing the memory consumption and inference speed of LLMs. For example, GPTQ Frantar et al. [12] compressed the model parameters to 4 bits, allowing larger models to run on hardware with less memory and without significant loss of performance.

LLMs can generate textual content that rivals the quality of human-generated text [7]. However, their applications go beyond text generation. Park et al. [31] demonstrated how LLMs can be used to simulate human behavior. These simulated agents wake up, cook, go to work, make decisions, and reflect on past experiences in a believable manner. Furthermore, Argyle et al. [4] suggests using language models as surrogates for certain demographic groups within social science research. Their study demonstrates how conditioning GPT-3 on the socio-demographic backgrounds of real human subjects can accurately replicate response distributions among diverse human subgroups.

Contemporary work has also integrated LLMs into recommender systems. Kang et al. [20] demonstrated that fine-tuned LLMs outperform traditional supervised methods in predicting user ratings with less training data, while Wang et al. [48] employed LLMs as a recommendation agent, showcasing their potential to improve recommender systems. Both works show how LLMs can act as a good predictor of the ratings that a user would assign to an item. The authors further investigated whether LLMs can also be used as a recommender directly; they restricted their experiment to choosing an item from a list of 100 items. However, this task is still challenging for LLMs, as they must have knowledge of the entire set of possible items to be recommended. The limited context length does not allow one to provide a list of all possible items in the prompt to an LLM. Therefore, to date, the application of large language models (LLMs) as recommender systems has yet to exceed the performance of traditional recommender systems, which encompass both classical supervised algorithms and those based on reinforcement learning techniques. Our work diverges from these approaches by leveraging LLMs as simulation environments for item recommendation, in contrast to prior efforts that focused on training LLMs to function as the recommender system itself.

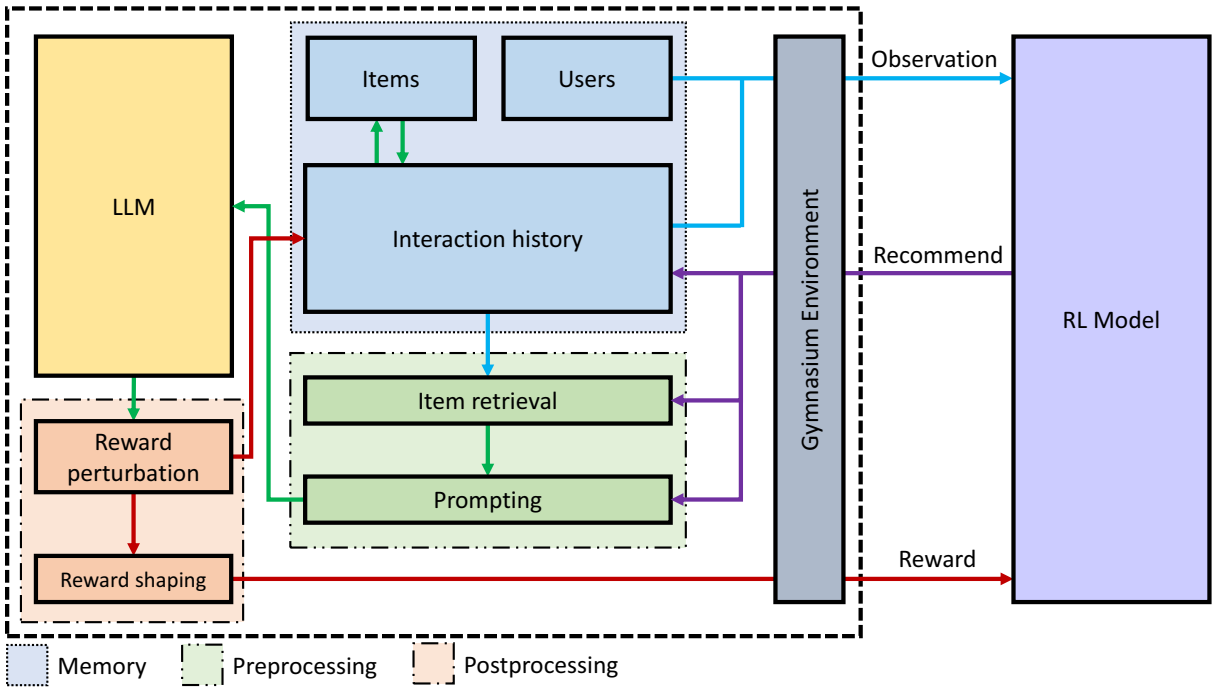

Figure 1. Overview of SUBER. The environment is built as a modular framework where each component can be modified as required. The basic control flow is as follows: The environment provides an observation using the memory module; the RL model returns an item recommendation in the form of an action, which is processed into a prompt by the memory and preprocessing component before being passed to the LLM. The score returned by the LLM is postprocessed, stored in memory and returned as a reward to the RL model.

3 Framework

To address the aforementioned challenges of data availability, unknown user model, and model evaluation, we propose SUBER, an environment designed to simulate human behavior through the integration of LLMs. SUBER serves a dual purpose by generating synthetic data and harnessing the capabilities of LLMs to replicate the behavior of individuals with unknown patterns. Additionally, this dynamic environment can serve as a model evaluation tool for recommender systems.

SUBER consists of an LLM component and three separate modules that contain multiple individual components. An overview of the overall structure is presented in Figure 1. The internal memory module of the environment contains two separate datasets, one for users and one for items. The environment also includes a preprocessing module that retrieves raw data from the memory module and transforms it to ensure compatibility with the LLM. Finally, a postprocessing component transforms the output produced by the LLM before returning it to the RL model.

The interaction with an RL model involves the following information flow: initially, the environment selects a user from memory, along with their interaction history (i.e., items and associated ratings) as the observation for the RL model. The RL model then recommends an item to the user as its action, with an action space equal to the number of items in the environment. The action and observation are subsequently processed through the preprocessing module, the LLM component, and the postprocessing module. Finally, the environment returns a reward corresponding to the post-processed rating predicted by the LLM. We describe each module in more detail in the following sections.

Our environment is designed with easy accessibility and extensibility in mind. Therefore, we chose a modular approach and based the environment interface on the Gymnasium standardized API [45]. Different components can be modified at will, providing additional flexibility in future design choices.

3.1 *Memory*

We introduce the following notation. We define U as the set of users and I as the set of items. For every pair of user-items $(u, i) \in U \times I$, we have a set $R_{u,i}$ that records all interactions between user u and item i. Similarly, for every user u we define with R_u the set of all interactions with all items, defined as follows:

$$R_u = \{(i, h) | i \in I, h \in R_{u,i}\}. \quad (1)$$

The memory module consists of three components: an item dataset, a user dataset, and a record of all interactions between users and items. This interaction history stores the set of interactions $R_{u,i}$ for each pair of user-items (u, i). Every interaction between the RL model and the environment produces a new interaction record between a user and an item, which is added to the interaction history.

3.2 *Pre-processing*

Item Retrieval. As the RL model interacts with the environment, the history of the interaction increases. It may be challenging to extract relevant information from long histories, and the increasing duration of the history will probably exceed the context length of current LLMs [31]. To address this issue, we propose an item-retrieval component responsible for retrieving the most appropriate items for

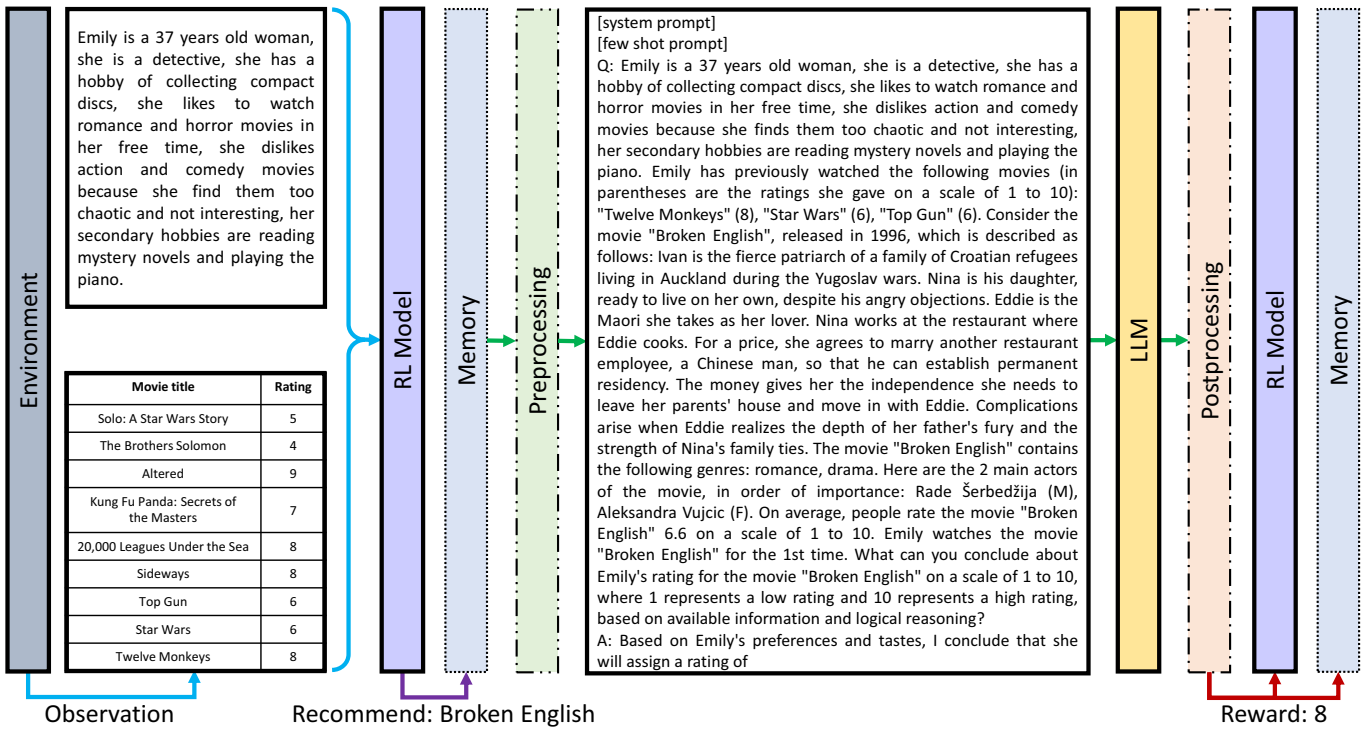

Figure 2. Pipeline of one interaction between the RL model and SUBER. The environment provides an observation in the form of a user description and user-item interaction history to the RL model. The RL model then recommends an item, which is processed into a prompt together with the user description and interaction history. The LLM uses this prompt to generate a reward for the recommended item. The reward is stored as part of the user-item interaction history and returned to the RL model.

the current query from the interaction history of a user. Additionally, as user interests and preferences may evolve over time, relying solely on user features may not accurately capture current interests. Therefore, historical rating data are used to provide a more detailed depiction of their evolving preferences.

Prompting. The prompting component aggregates the information retrieved by the item retrieval component, creating a prompt that contains the necessary details for the LLM, including the user and query item data. The objective of this prompt is to enable the LLM to accurately predict the rating of the current query item. An example of such a prompt as part of an interaction example can be seen in Figure 2.

3.3 Postprocessing

Reward Perturbation. The reward perturbation component introduces noise into the ratings generated by the LLM. This component functions as a simulation of "concept drift" for users [53]. Concept drift refers to the notion that users may change their interests over time and are unlikely to maintain static preferences.

Reward Shaping. Similarly to the reward perturbation component, reward shaping modifies the reward. However, unlike the perturbed reward which is added to the memory, the reward modified by the reward shaping component is returned directly to the RL model and is not stored in memory. The reward shaping module aims to reflect changes in the reward that are not related to a change in the preference of a user, such as spontaneous decisions or fleeting interests.

4 Experiments

To evaluate SUBER, we followed the approach of Rohde et al. [34] and Ie et al. [17]. We perform sanity checks and case studies, which we present in Section 4.2 and Section 4.4. To achieve this, we implemented a movie recommendation and a book recommendation environment in our framework. In the following sections, we discuss our implementation and design choices for these environments, as well as our ablation study and experiments. For the movie setting, we use rewards from 1 to 10, similar to TMDB [2], while for the book setting we use rewards from 1 to 5, as found in the Amazon Reviews Dataset [28].

For both environments, we created a dataset of synthetic users using Vicuna [52] with Guidance [25]. To generate user descriptions, we condition the LLM with information such as the age, liked and disliked genres, hobbies, and profession of the user (cf. Listing 1). We generate the user age by sampling from the age distribution in the United States [8],

We randomly select a hobby and a profession from predefined lists (cf. Appendix G in [11]). These hobby lists are divided into two categories: one tailored to children (aged 4-17) and another for adults (aged 18-75). Users not of working age are assigned the profession "student," while those of retirement age are categorized as "retired." For the movie dataset we use MovieLens (*ml-latest-small*) [16] and collect the respective movie features from TMDB. For the book dataset, we used a subset of the Amazon Book Dataset. For more details, see Appendix A and Appendix B in [11].

[2] https://www.themoviedb.org/

Can you generate details for a person, you need to generate a name, an age, a hobby, a job and a detailed, long and original description that contains the persons interests and secondary hobbies. Please outline the cinematic preferences of the individual, detailing their favorite and least favorite genres. Kindly provide explanations for each genre preference.
Name: Emily Johnson, Age: 37, Gender: F, Hobby: COMPACT DISCS, Job: DETECTIVE
Genres liked: romance, horror, Genres diskliked: action, comedy
Description: she is a detective, she has a hobby of collecting compact discs, she likes to watch romance and horror movies in her free time, she dislikes action and comedy movies because she find them too chaotic and not interesting, her secondary hobbies are reading mystery novels and playing the piano.

Listing 1. User generation and characteristic assignment process example by Vicuna with guidance. Black text shows the template and the instruction, RED TEXT marks the sampled information from external distributions, blue text indicates the content generated by LLM.

4.1 Setup

We implemented three different approaches for the retrieval component: feature retrieval, recency retrieval, and similarity retrieval. The feature-based approach retrieves items based on the Sorensen Coefficient of movie genres, actors, director, and average rating, while for books, we use book category, author, and average rating. The recency algorithm retrieves the most recent interactions. The similarity approach retrieves items from the history based on their similarity to the query item. We generate item-description embeddings using a Sentence-T5 model [29] and compute their similarities based on the cosine distance. To select the item-rating pair to retrieve from memory, we compute the similarity between the query item and all items previously viewed by the current user, selecting the items with the highest similarity. We use the items returned from the retrieval component to construct a prompt to query the LLM. The LLM is tasked with generating a rating of the queried item by the current user, where the queried item corresponds to the item suggested by the recommender system. We construct the prompt such that the user description comes first, allowing us to leverage the key-value cache [32], eliminating the need to recalculate all intermediate embeddings within the layers of the LLM for already encountered prefixes, therefore, increasing execution speed. Furthermore, we experimented with one-shot and two-shot prompting to improve model performance, which has been shown to increase generation quality [7]. In addition to the default system prompt , we created a custom system prompt (see Listing 2 for movie and Appendix B in [11] for books).

You are a highly sophisticated movie rating assistant, equipped with an advanced understanding of human behavior. Your mission is to deliver personalized movie recommendations by carefully considering the unique characteristics, tastes, and past–seen films of each individual. When presented with information about a specific movie, you will diligently analyze its plot, primary genres, actors, and average rating. Using this comprehensive understanding, your role is to provide thoughtful and accurate ratings for movies on a scale of 1 to 10, ensuring they resonate with the person's preferences and cinematic inclinations. Remain impartial and refrain from introducing any biases in your predictions. You are an impartial and reliable source of movie rating predictions for the given individual and film descriptions.

Listing 2. An advanced system prompt guiding the model to provide personalized and unbiased movie ratings.

Tokenization ambiguity can become an issue when generating numbers with LLMs. Since we are dealing with ratings on a scale from one to ten, and because the number "10" can be tokenized in two different ways, this can cause unwanted side effects. To tackle this challenge, we tested two additional strategies for the movie setting: shifting all rewards to the scale of 0-9, and using words for numbers from "one" to "ten."

We experimented with various quantized versions of Llama, Vicuna, Mistral, using LLMs that could run within a 24GB memory limit. A list of the models used in our experiments can be found in Appendix D in [11]. All models were quantized using GPTQ. Since different LLMs influence the simulation of human behavior in different ways, it is important to highlight the inherent trade-off between model size and processing speed. In particular, during training of an RL model, a fast environment is desirable to acquire more samples in a shorter time span. However, smaller LLMs may not adequately emulate the desired human behavior of our synthetic users.

For the reward perturbation experiment, we compared Gaussian noise and greedy noise. Greedy noise alters the LLM rating by 1 with a probability of q, while it remains unchanged with a probability of $1 - q$.

Our implementation of reward shaping operates on the following premise: as a user engages with an item more frequently, their interest in revisiting it diminishes. In contrast, as time passes, the likelihood that the user interacts with the item increases again [35]. Given this insight, let us consider a user u from the set U and an item i with which the user has interacted n_{ui} times. When a time span of Δt has passed since the last interaction with the item, the reward r undergoes a reshaping process, characterized by the following equation:

$$r \leftarrow \max(1, \lfloor r \cdot q^{n_{ui}/\Delta t} \rfloor), \tag{2}$$

where $q \in [0, 1]$. This adjustment takes into account both the frequency of user interaction with the item and the time elapsed since their last interaction, resulting in the modified reward r.

4.2 Ablations

To determine the effect of each component in our environment, we performed ablations across four different test cases. In this section, we present the high-level idea; for more details, see Appendix C in the supplementary material [11].

Genres/Categories. We assess the environment's ability to recognize movie and book genres and its ability to correlate those genres with user preferences to accurately predict ratings. User profiles were manually created for each movie genre, ensuring that they expressed a preference for the selected genre while disliking all others. Afterwards, we queried the environment with users and movies from both their favored and disliked genres. The accuracy of rating predictions is used to measure performance. A similar process is used for the book environment, replacing movie genres with book categories.

High/Low Rating. We assess whether the environment can accurately infer high ratings for users who provide positive-leaning descriptions, while inferring low ratings for users whose descriptions are negative-leaning. We give each user a set of items and test whether the environment is able to generate high or low ratings, depending on the description of the user.

Collection of Items. We evaluate the ability of the environment to leverage the historical item ratings of a user to predict their future ratings. We conduct this test by manually selecting a set of item

Table 2. Ablation results for the movie setting using *Mistral 7B* as our environment. We test the LLM on coherency and realistic ratings for user-movie interactions. We achieve best performance with 0-9 digit rating scale, 2-shot prompting, and our custom system prompt.

Prompt component							
Rating scale	N-shot	System prompt	Genres ↑	High/Low ↑	Collection of movies ↑	Similarity to ML ↑	Agg. score ↑
0-9	0-shot	default	0.80±0.00	**1.00±0.00**	0.67±0.02	0.54±0.00	0.75±0.01
0-9	0-shot	custom	**0.87±0.00**	**1.00±0.00**	0.68±0.02	0.70±0.00	0.81±0.01
0-9	1-shot	default	0.72±0.00	0.96±0.00	**0.71±0.03**	0.73±0.01	0.78±0.01
0-9	1-shot	custom	0.81±0.00	**1.00±0.00**	**0.71±0.02**	0.78±0.00	**0.82±0.01**
0-9	2-shot	default	0.78±0.00	0.99±0.00	0.69±0.01	**0.80±0.00**	**0.82±0.00**
0-9	2-shot	custom	0.79±0.00	**1.00±0.00**	0.67±0.03	0.78±0.00	0.81±0.01
1-10	2-shot	custom	0.50±0.0	0.50±0.00	0.50±0.00	0.51±0.00	0.50±0.00
one-ten	2-shot	custom	0.79±0.00	**1.00±0.00**	0.66±0.01	0.72±0.00	0.79±0.00

Table 3. Ablation results for the book environment using *Mistral 7B* as our environment. We test the performance of the LLM to give coherent and realistic ratings for user-book interactions. We achieve best overall performance when using 2-shot prompting and our custom system prompt.

Prompt component						
Rating scale	N-shot	System prompt	Category ↑	High/low ↑	Collection of books ↑	Agg. score ↑
1-5	0-shot	default	0.68±0.00	**1.00±0.00**	0.65±0.01	0.77±0.00
1-5	0-shot	custom	0.83±0.00	**1.00±0.00**	0.68±0.02	0.83±0.01
1-5	1-shot	default	0.87±0.00	**1.00±0.00**	0.81±0.04	0.89±0.01
1-5	1-shot	custom	**0.89±0.00**	**1.00±0.00**	**0.82±0.02**	**0.90±0.01**
1-5	2-shot	default	0.83±0.00	0.98±0.00	0.73±0.02	0.85±0.01
1-5	2-shot	custom	0.85±0.00	**1.00±0.00**	0.76±0.02	0.87±0.01

collections belonging to a series (e.g., James Bond, Toy Story, etc.). Subsequently, we randomly select a sample of users from our synthetic dataset and fill their history with items from our collection as well as random items. We assign a high rating to all items in the collection history, and the corresponding average rating to the remaining random items. Success is measured by a high rating for the queried item that is part of the collection. The experiment is repeated by assigning low ratings to the collection items to test the ability of the environment to predict low ratings.

Similarity to Real Rating Distribution. We evaluate whether the rating distribution obtained from our movie environment accurately reflects human behavior by comparing it to the rating distribution from MovieLens, which are representative samples of human ratings. We sample with replacement from both our environment and the MovieLens dataset. We then compute the empirical distribution across the dataset and use the total variation distance as a metric to measure similarity. For the book environment, see Appendix C in [11]. The aggregated score is the mean of all test cases. All ablations, except where defined otherwise, were performed using the following configurations. We used the 2-shot prompting, a custom system prompt, three item retrievial via T5-similarity, and no reward perturbation. For movies, we used *Mistral 7B* with rating scale 0-9, and for books we use *Mistral 7B* with scale 1-5.

Results. In the movie environment, we observe that different prompt strategies generally do not differ significantly from each other in the case of Mistral, with the only two exceptions being the 0-shot prompt with the default system prompt, which performs slightly worse, and the weak performance of the 1-10 rating scale due to tokenization ambiguity. Vicuna, on the other hand, is more affected by different prompt strategies, as shown in Appendix D in [11]. Table 2 shows the general trend on how the environment can capture human concepts such as genres and movie franchises. For the book environment (cf. Table 3) it can be observed that the use of few-shot prompting, as well as the custom system prompt, has a positive impact on the different test cases. Additionally, similar to the movie environment, the model is also able to understand human concepts in the book domain. In general, we observe that larger models perform better across model families (cf. Figure 3). In addition, we can see how Mistral performs best among open-source models.

Our ablation of the retrieval component demonstrates that this component plays a crucial role in understanding user interests (cf. Tables 8 and 12 in the supplementary material [11]) . Furthermore, the recency approach proves inadequate, while the best-performing retrieval approach is predicated on the similarity of item features.

4.3 *Human Evaluation*

We conducted a case study to better evaluate the quality of different LLMs in the rating simulation task. For the study, we sampled ten

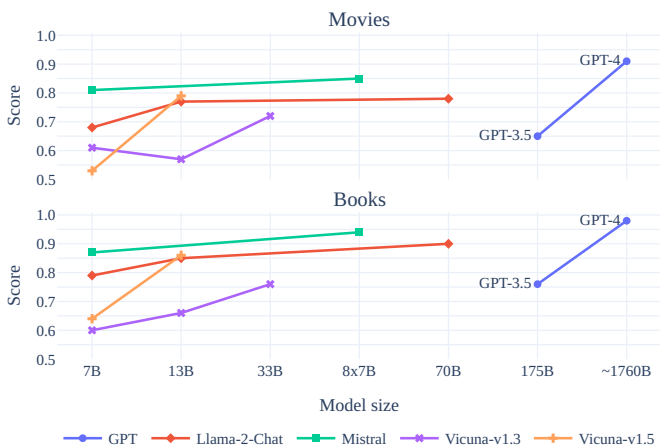

Figure 3. Aggregated score across LLM families for the movie environment (top), and for the book environment (bottom) by varying only the LLM component. For details see Appendix D in [11].

Table 4. Performance Metrics of RL Models Trained on SUBER: Mean Average Precision (MAP@10), Mean Reciprocal Rank (MRR@10), Personalization of the top ten recommendations (Pers.@10). "Liked genres" indicates the proportion of movies in the top ten recommendations aligned with user-preferred genres (see Appendix F in the supplementary material [11] for details).

Algorithm	Average reward	MAP@10 ↑	MRR@10 ↑	Pers.@10 ↑	% Liked genres ↑	% Disliked genres ↓
DQN	6.79±0.06	0.53±0.06	0.85±0.06	0.00±0.00	0.42±0.01	0.15±0.01
PPO	6.91±0.03	0.59±0.01	0.84±0.01	**0.99±0.00**	0.44±0.01	0.15±0.00
TRPO	7.25±0.08	0.67±0.06	0.91±0.02	0.35±0.06	0.45±0.02	0.14±0.01
A2C	**7.93±0.07**	**0.88±0.01**	**0.96±0.01**	0.91±0.03	**0.49±0.02**	**0.11±0.01**

Table 5. Human evaluation scores for various LLMs.

LLM	Score ↑
Random rating	2.87±1.51
Vicuna 13B	3.22±1.32
Llama-2-Chat 13B	3.42±1.22
Mistral 7B	3.80±1.27
GPT-4	**4.47±0.77**

user-movie interactions, for each interaction we queried four different LLMs. The random rating in Table 5 serves as a baseline comparison, allowing us to compare the quality of our proposed approach with a random signal. The answer is constructed by sampling a rating uniformly at random between 1 and 10, and having the LLM (*Vicuna 13B*) generate the explanation for the rating. We then asked participants to rate the quality of the LLM's response on a scale of 1 to 5.

Participants in this study were recruited from among our colleagues and provided informed consent to participate. The study was designed with strict adherence to randomized double-blind procedures to ensure impartiality and reliability of the results. As this user study did not involve ongoing follow-up or monitoring of the participants, our institutional review board (IRB) determined that formal approval was not required. From the survey (cf. Table 5), we find that users agree more with *GPT-4*, outperforming all other models. Furthermore, we find that *Mistral 7B* is the best LLM among open-source models despite only having 7B parameters. More information on the study setting is provided in Appendix E in [11].

4.4 Benchmarks

We demonstrate the viability of our environment to train an RL recommender system. The architecture of the RL model is inspired by the principles of Low-Rank Approximations in collaborative filtering [2]. We implemented four different agents based on A2C [27], PPO [37], TRPO [36], and DQN [26]. We train all models for 1.6M steps on SUBER. Due to space constraints, a more detailed discussion on the training of reinforcement learning models is deferred to Appendix F in the supplementary material [11]. In addition to using classical RecSys metrics, like MAP@10, MRR@10, and personalization, we introduce two additional metrics to assess the alignment of the agent's recommendations with user preferences (See Appendix F in [11] for detail metric definitions). Each user in the training dataset has both preferred and disliked movie genres. Based on these data, the trained RL model generates a list of top-5 movie recommendations for each user: *percentage liked genres* and *percentage disliked genres* . The recommendations are classified into three categories: *liked* (movies matching preferred genres and excluding disliked ones), *disliked* (movies with disliked genres and without preferred ones), and *neutral* (remaining recommendations).

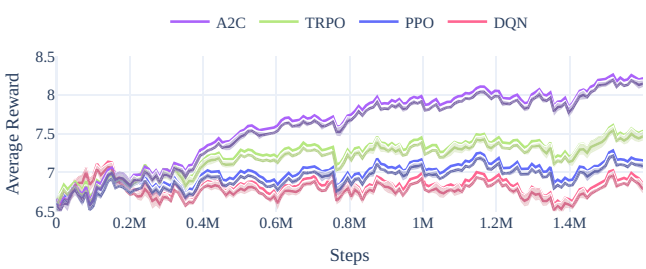

Figure 4. Training plot of various RL models. The y-axis displays the average reward from evaluation samples.

Our evaluation indicates that the A2C algorithm demonstrates the best overall performance in our case study (cf. Table 4). Although the PPO algorithm registers a higher personalization score, indicative of its ability to tailor recommendations, it is less effective than A2C in aligning recommendations with user interests, as reflected in the percentage of liked genres metric. This suggests that A2C is more adept at discerning and catering to user preferences.

5 Future Work

One promising direction is to fine-tune the LLM with human feedback to improve the simulated user behavior. This can be achieved using datasets like MovieLens, which provide a natural reward function for RL methods. For instance, the negative squared difference between the LLM rating and the actual rating can be used as a reward. Currently, the setup considers only static users. Future work could model user evolution over time to reflect changing interests, making synthetic users more realistic and dynamic. Additionally, exploring ways to enrich the feature space of the LLM could be valuable. By incorporating complex features such as item seasonality and user context, RL models could better capture user behavior, leading to more accurate simulations.

6 Conclusion

Our research offers a possible avenue to address the persistent challenge of training recommender systems in the absence of real user interactions. Conventional approaches that depend on user-item interaction histories or synthetic data have often failed to replicate real-world usage scenarios accurately. By introducing SUBER, a novel RL environment designed specifically for recommender system training, and incorporating recent advances in LLMs to emulate human behavior in the training environment, we have proposed a potential solution to this long-standing issue. Our results, as demonstrated through a series of ablation studies, experiments, and human evaluation, underscore the efficacy of our approach. We believe that this work marks a step toward achieving more realistic and practical training environments for recommender systems, even when direct user interactions are unavailable.

References

[1] M. M. Afsar, T. Crump, and B. Far. Reinforcement learning based recommender systems: A survey. *ACM Computing Surveys*, 55(7):1–38, 2022.

[2] C. C. Aggarwal et al. *Recommender systems*, volume 1. Springer, 2016.

[3] S. Agrawal, S. Merugu, and V. Sembium. Enhancing e-commerce product search through reinforcement learning-powered query reformulation. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 4488–4494, 2023.

[4] L. P. Argyle, E. C. Busby, N. Fulda, et al. Out of one, many: Using language models to simulate human samples. *Political Analysis*, 31(3): 337–351, 2023.

[5] J. Bobadilla, F. Ortega, et al. Recommender systems survey. *Knowledge-based systems*, 46:109–132, 2013.

[6] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, et al. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[7] T. Brown, B. Mann, N. Ryder, M. Subbiah, et al. Language models are few-shot learners. *NeurIPS*, 33:1877–1901, 2020.

[8] U. S. C. Bureau. National population by characteristics: 2020-2022, 2022. URL https://www.census.gov/data/tables/time-series/demo/popest/2020s-national-detail.html. 09.21.2023.

[9] M. Chen, A. Beutel, P. Covington, S. Jain, et al. Top-k off-policy correction for a reinforce recommender system. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 456–464, 2019.

[10] X. Chen, S. Li, H. Li, S. Jiang, Y. Qi, and L. Song. Generative adversarial user model for reinforcement learning based recommendation system. In *ICML*, pages 1052–1061. PMLR, 2019.

[11] N. Corecco, G. Piatti, L. A. Lanzendörfer, F. X. Fan, and R. Wattenhofer. SUBER: An RL Environment with Simulated Human Behavior for Recommender Systems. *arXiv preprint arXiv:2406.01631*, 2024. Full version of this paper.

[12] E. Frantar, S. Ashkboos, T. Hoefler, and D. Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

[13] F. Garcin, B. Faltings, O. Donatsch, A. Alazzawi, et al. Offline and online evaluation of news recommender systems at swissinfo. ch. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 169–176, 2014.

[14] C. Gelada and M. G. Bellemare. Off-policy deep reinforcement learning by bootstrapping the covariate shift. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3647–3655, 2019.

[15] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992.

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

[17] E. Ie, C.-w. Hsu, M. Mladenov, V. Jain, et al. Recsim: A configurable simulation platform for recommender systems. *arXiv preprint arXiv:1909.04847*, 2019.

[18] E. Ie, V. Jain, J. Wang, S. Narvekar, et al. Reinforcement learning for slate-based recommender systems: A tractable decomposition and practical methodology. *arXiv preprint arXiv:1905.12767*, 2019.

[19] A. Q. Jiang et al. Mistral 7b, 2023.

[20] W.-C. Kang, J. Ni, N. Mehta, M. Sathiamoorthy, et al. Do llms understand user preferences? evaluating llms on user rating prediction. *arXiv preprint arXiv:2305.06474*, 2023.

[21] A. Kumar, J. Fu, M. Soh, G. Tucker, and S. Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. *NeurIPS*, 32, 2019.

[22] Y. Lin, Y. Liu, F. Lin, L. Zou, et al. A survey on reinforcement learning for recommender systems. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[23] Z. Liu, L. Zou, X. Zou, C. Wang, et al. Monolith: real time recommendation system with collisionless embedding table. *ORSUM@ACM RecSys 2022*, 2022.

[24] J. Lu, D. Wu, M. Mao, W. Wang, and G. Zhang. Recommender system application developments: a survey. *Decision support systems*, 74:12–32, 2015.

[25] S. Lundberg. Guidance. https://github.com/guidance-ai/guidance, 2023.

[26] V. Mnih, K. Kavukcuoglu, D. Silver, et al. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[27] V. Mnih, A. P. Badia, M. Mirza, A. Graves, et al. Asynchronous methods for deep reinforcement learning. In *ICML*, pages 1928–1937. PMLR, 2016.

[28] J. Ni, J. Li, and J. McAuley. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *EMNLP-IJCNLP*, pages 188–197, Hong Kong, China, Nov. 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1018.

[29] J. Ni, G. H. Abrego, N. Constant, J. Ma, K. B. Hall, D. Cer, and Y. Yang. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. *arXiv preprint arXiv:2108.08877*, 2021.

[30] OpenAI. Gpt-4 technical report, 2023.

[31] J. S. Park, J. C. O'Brien, C. J. Cai, M. R. Morris, et al. Generative agents: Interactive simulacra of human behavior. *arXiv preprint arXiv:2304.03442*, 2023.

[32] R. Pope, S. Douglas, A. Chowdhery, J. Devlin, et al. Efficiently scaling transformer inference. *Proceedings of Machine Learning and Systems*, 5, 2023.

[33] D. Precup, R. S. Sutton, and S. Dasgupta. Off-policy temporal-difference learning with function approximation. In *ICML*, pages 417–424, 2001.

[34] D. Rohde, S. Bonner, T. Dunlop, F. Vasile, and A. Karatzoglou. Recogym: A reinforcement learning environment for the problem of product recommendation in online advertising. *arXiv preprint arXiv:1808.00720*, 2018.

[35] C. A. Russell and S. J. Levy. The Temporal and Focal Dynamics of Volitional Reconsumption: A Phenomenological Investigation of Repeated Hedonic Experiences. *Journal of Consumer Research*, 39(2): 341–359, 10 2011. ISSN 0093-5301. doi: 10.1086/662996. URL https://doi.org/10.1086/662996.

[36] J. Schulman, S. Levine, P. Abbeel, et al. Trust region policy optimization. In *ICML*, pages 1889–1897. PMLR, 2015.

[37] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[38] G. Shani and A. Gunawardana. Evaluating recommendation systems. *Recommender systems handbook*, pages 257–297, 2011.

[39] J.-C. Shi, Y. Yu, et al. Virtual-taobao: Virtualizing real-world online retail environment for reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4902–4909, 2019.

[40] Y. Shi, M. Larson, and A. Hanjalic. Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges. *ACM Computing Surveys (CSUR)*, 47(1):1–45, 2014.

[41] H. Steck, L. Baltrunas, E. Elahi, et al. Deep learning for recommender systems: A netflix case study. *AI Magazine*, 42(3):7–18, 2021.

[42] X. Su and T. M. Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in artificial intelligence*, 2009, 2009.

[43] H. Touvron et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[44] H. Touvron et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[45] M. Towers, J. K. Terry, A. Kwiatkowski, J. U. Balis, et al. Gymnasium, Mar. 2023. URL https://zenodo.org/record/8127025.

[46] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, et al. Attention is all you need. *NeurIPS*, 30, 2017.

[47] K. Wang, Z. Zou, M. Zhao, et al. Rl4rs: A real-world dataset for reinforcement learning based recommender system. In *SIGIR'23*, pages 2935–2944, 2023.

[48] Y. Wang, Z. Jiang, Z. Chen, F. Yang, et al. Recmind: Large language model powered agent for recommendation. *arXiv preprint arXiv:2308.14296*, 2023.

[49] S. Zhang, L. Yao, A. Sun, and Y. Tay. Deep learning based recommender system: A survey and new perspectives. *ACM computing surveys (CSUR)*, 52(1):1–38, 2019.

[50] W. Zhang, U. Paquet, and K. Hofmann. Collective noise contrastive estimation for policy transfer learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

[51] K. Zhao et al. Kuaisim: A comprehensive simulator for recommender systems. *Advances in Neural Information Processing Systems*, 36: 44880–44897, 2023.

[52] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *arXiv preprint arXiv:2306.05685*, 2023.

[53] I. Žliobaitė, M. Pechenizkiy, and J. Gama. An overview of concept drift applications. *Big data analysis: new algorithms for a new society*, pages 91–114, 2016.