UISA: User Information Separating Architecture for Commodity Recommendation Policy with Deep Reinforcement Learning

AOBO XU, LING JIAN*, YUE YIN, and NA ZHANG, China University of Petroleum East China, Qingdao, China

Commodity recommendation contributes an important part of individuals' daily life. In this context, deep reinforcement learning methods have demonstrated substantial efficacy in enhancing recommender systems' performance. Nevertheless, several recommender systems directly utilize original feature information as a foundational element for decision-making, which seems simplistic and low efficient. Furthermore, the incorporation of sequential decision-making adds complexity to the task of recommendation.

In pursuit of maximizing the long-term sequential returns of recommender systems, our study introduces a novel architecture, named User Information Separating Architecture (UISA). This framework is tailored to align with classic reinforcement learning algorithms and aims to extract the user's interest value through the discrete processing of both static and dynamic user information. Through integration with deep reinforcement learning, the architecture is oriented towards the maximization of long-term profit and is applicable in sequential recommendation scenarios. We conduct experimental assessments by combining the proposed architecture with proximal policy optimization (PPO) and deep deterministic policy gradient (DDPG) algorithms. The outcomes illustrate marked improvements in commodity recommendation, showcasing enhancements ranging from approximately 5% to 40% in both reward and click-through rate metrics across a self-constructed JDEnv environment and the Virtual Taobao environment. Through comparison experiments, the UISA models demonstrate comparable performance.

CCS Concepts: • **Information systems** → **Recommender systems**; • **Theory of computation** → **Reinforcement learning**.

Additional Key Words and Phrases: Recommender Systems, Reinforcement Learning, Sequential Decision-making

1 INTRODUCTION

Personalized recommender systems present a potent strategy for mitigating the information overload and have found application in diverse areas, encompassing film recommendations, job posting [27] and beyond [44]. They serve to filter out irrelevant information and select the appropriate items for individual users. Commodity recommender systems constitute a significant social application, where recommender engines exhibit individual customers the commodities they are most inclined to buy, especially with the developing of e-commence.

Within the context of commodity recommender systems, the items are massive, whereas the user information tends to be characterized by sparsity, typically comprising static and dynamic attributes. Consequently, formulating the decision process and effectively extracting the feature from user data emerge as pivotal factors influencing the ultimate performance of the recommender system, and influencing both merchants' profitability and users' shopping experiences. It's natural to posit a positive correlation between the profitability of merchants and the

*Corresponding author: Ling Jian, bebetter@upc.edu.cn

Authors' address: Aobo Xu, upc20xab@s.upc.edu.cn; Ling Jian, bebetter@upc.edu.cn; Yue Yin, yy4201@nyu.edu; Na Zhang, 2108010424@s.upc.edu.cn, China University of Petroleum East China, Qingdao, Shandong, China.

ACM 2770-6699/2024/4-ART
https://doi.org/10.1145/3654806

quality of user shopping experiences. Furthermore, contemplating sequential recommendations and pursuing for higher long-term profitability brings recommender systems closer to reflecting real-world dynamics, presenting a more challenging problem. Leveraging machine learning algorithms to increase profits and shopping experiences is intuitively beneficial.

Reinforcement learning methods have gained extensive traction across domains such as game agent [24] [28], content generation [19] [38], mathematical reasoning [16] and so on. By constructing a Markov decision process (MDP), reinforcement learning algorithms identify actions that yield the highest cumulative reward. Owing to its emphasis on the long-term returns, the integration with recommender systems holds significant promise.

The demarcation of static and dynamic user information is applicable to most recommendation scenarios. To dig out the latent capacity of user information and harness the formidable decision-making prowess offered by reinforcement learning, we introduce a novel architecture. It processes both static and dynamic user features separately and captures the influence impact of dynamic attributes upon static ones. This approach finds particular suitability in the context of sequential recommender tasks. Notably, the model architecture can be generalized across the conventional reinforcement algorithms for recommendation systems. By the optimization of reinforcement learning algorithms, our structural framework possesses the capability to effectively capture diverse customer features. Collectively, we offer the following contributions:

- Within the context of commodity recommendation, we introduce a dedicated architecture termed the **U**ser **I**nformation **S**eparating **A**rchitecture (**UISA**). It is specifically designed to separately process user information, which aligns well with the framework of reinforcement learning algorithms.
- We conduct empirical experiments on simulated commodity recommendation environments, which indicate the the efficacy and consequential enhancement of future profitability.
- Besides performing experiments on our self-built environment based on JData, named JDEnv, and an existing Virtual Taobao recommendation environment. To facilitate reproducibility, the source code is made publicly available at https://github.com/MyLove-XAB/DRLRec.

In the rest of the sections is organized as follows. We present a comprehensive overview of pertinent literature in Section 2. Subsequently, in Section 3, we elucidate our proposed methodologies, with a particular focus on the User Information Separating Architecture (UISA). Section 4 is dedicated to the empirical validation, wherein we demonstrate the efficacy of UISA through empirical results. Lastly, Section 5 summarizes our contributions and outline the potential avenues for future research.

2 RELATED WORKS

This article revolves around the process of recommendation selection achieved through the extraction of users' features from segregated user information using deep reinforcement learning algorithms. Our work is related to the following three aspects.

2.1 Recommendation Through Feature Emphasis

In the realm of recommendation models and algorithms, leveraging features for decision-making has been proven to be a beneficial practice. Some research commence from the vantage point of user-centric considerations, while others emphasize item-centric viewpoints[10]. The classic collaborative filtering (CF) methods can similarly be categorized into two primary classes: user-based algorithms [47] [8] and item-based algorithms [20]. Certain studies endeavor to incorporate both user-centric and item-centric perspectives [31]. A prevalent approach in recommender systems is the adoption of the Two-Tower structural model [7] [41], which concurrently learns representations for both users and items in parallel.

Various ways for handling user and item features have been investigated, such as matrix decomposition [21] and multi-feature fusion [4] [39], which have demonstrated their utility in enhancing recommender systems. To

achieve more satisfactory recommendations, considering more information is likely to yield positive results. Social networks, for instance, which encapsulate the relationships between users, offer a valuable resource for informed reasoning and decision-making in recommendation systems [30] [29]. Furthermore, the inclusion of context information within recommendations introduces heightened flexibility, considering aspects like transactional context[33] and temporal context[15]. The realm of feature research holds substantial promise, owing to the rich information within user and item profiles. Therefore, we adopt a user-centric approach, involving the extraction of user features and the computation of their similarity with item features to formulate recommendation suggestions. Nevertheless, extant literature appears to have overlooked attempts to segregate user information into static and dynamic components for the extraction of their respective features, a facet that deserves further study.

2.2 Neural Recommender Systems

The deep learning methods are widely sought after in recommender systems research, for efficient computing, attractive features representation, and superior performance. A diverse range of neural network architectures have been applied to recommender systems. Recurrent neural networks and their variants, such as Gated Recurrent Units (GRU) and Long Short-Term Memory (LSTM), excel in processing temporal data and have been effectively employed for tasks involving text information [37] and historical records [6]. In scenarios where the relationship between users and items can be modeled as a graph, Graph Neural Networks (GNNs) have demonstrated superior performance [42] [2]. Recommender systems grounded in knowledge graphs are flourishing, particularly for the multi-source data [27]. Notably, recent advances in Natural Language Processing (NLP) techniques, have seen the utilization of models for feature extraction from text-based data, like BERT and Transformer [18]. Furthermore, the advent of Large Language Models (LLMs) has afforded recommender systems greater personalization and flexibility, achieved through the design of prompts [12] [14]. Session-based recommendation and sequential recommendation have garnered considerable interest from researchers [9] [17]. Due to the inherent nonlinear transformations, deep neural networks often function as black boxes, rendering the analysis of cause-and-effect relationships challenging. Therefore, the interpretability and explainable ability of such models have become subjects of academic discourse [11] [40].

Recent researches on neural recommender systems yield numerous competitive baseline models. In this study, we leverage deep neural networks for the extraction of user and item information, thereby formulating the proposed model. Additionally, we devise a deep recommender system by establishing a reinforcement policy and a critic network rooted in neural network architectures. Furthermore, during the construction of the commodity recommendation environment JDEnv, an auto-encoder is employed for the representation of commodity features.

2.3 Reinforcement Learning Based Recommender Systems

In practical recommendation scenarios, the decision-making process typically unfolds as a sequential and interactive exchange between recommender systems and users. Sequential recommender systems align more closely with users' prolonged preferences, hence garnering escalating interest [34]. Within this framework, reinforcement learning endeavors to explore policies that maximize long-term gains through interactions with the environment. Numerous reinforcement learning algorithms are used to construct recommender systems, such as REINFORCE [1] and DDPG [3]. The incorporation of reinforcement learning provides a theoretical foundation for the contemplation of multi-step recommendation strategies [49] [13]. On the other hand, methods rooted in reinforcement learning offer a promising avenue for investigating the explainability of recommender systems [36] [26]. Leveraging reinforcement learning algorithms to improve the recommender performance is potential [48] [32].

In order to generate a recommendation policy that is both relevant and applicable, it is imperative to operate within an interactive environment that closely mirrors real-world conditions. In terms of simulation methodologies,

numerous research focuses on the utilization of generative adversarial network (GAN) to replicate environment [46] [45]. While some delve into supervised imitation learning [5] or inverse reinforcement learning [23]. In this paper, we craft a JDEnv environment and employ reinforcement learning algorithms to derive an effective recommendation policy. To the best of our knowledge, we are the first to integrate reinforcement learning for the distinct assessment of the varying emphasis on users' interest caused by both static and dynamic information.

3 RECOMMENDATION POLICY LEARNING

In this paper, we consider a scenario involving sequential interactions between commodity recommenders and users. From the recommender engine's perspective, the emphasis lies in precisely handling user information to enhance recommendation outcomes. The overarching objective is to build an intelligent recommender system proficient in item recommendations to users, with the goal of maximizing long-term returns. In this section, we commence by constructing a formalized model with mathematical notation and formulation. Subsequently, we introduce the architecture of separately processing user information and present how its integration with classic reinforcement learning algorithms, using proximal policy optimization (PPO) algorithm [22] as an example.

3.1 Problem Formulation

It is imperative to establish foundational configurations for the sequential commodity recommendation settings. In the context of most commodity recommender systems, the number of items often vastly surpasses the users', making item-centered decision-making require huge computing resources. Thereby, it seems reasonable to make recommendation decisions relying on user information. However, the user information will change with the process of sequential decision-making, which needs to be taken into account. Consequently, it is postulated certain aspects of user information will change while others remain constant, which can be comprised of two distinct components: static and dynamic information. Therefore, we can artificially divide user characteristics into two distinct categories: static information and dynamic information.

Initially, we provide precise definitions for the static and dynamic information. Static user information is characterized by attributes that either remains unaltered or undergoes gradual changes over the course of interactions, which can be regarded as constant during the sequential recommendation, effectively deemed constant throughout sequential recommendation processes, exemplified by attributes such as gender and age. In contrast, the dynamic information is defined by attributes that undergo changes in response to varying recommendations and user actions, illustrated by factors like the characteristics of commodities recently purchased by the user. In the experiments of Section 4, the segregation of static and dynamic information is contingent upon distinct environmental configurations, the specifics of which will be expounded upon later. The static information typically exhibit a persistent nature over extended duration, whereas the dynamic information modifies in response to the evolving interaction history. In each decision stage, the recommender engine ultimately generates a list of items, adhering to a predetermined fixed quantity, aligning with the user's current page view can contains. Then the user will behavior diversely to different recommendations, such as purchasing, clicking, buying and so on, which caused the dynamic information changes. By separating user information, the model is adept at capturing nuanced shifts in user preferences.

Reinforcement learning is a paradigm that learns how to map situations to actions, so as to maximize a numerical reward signal [25]. The agent sequentially makes decisions and engages in dynamic interactions with its environment to iteratively refine a policy. Most reinforcement learning problems can be formulated as a Markov decision process (MDP). Within the previously specific task, the agent is a recommender engine, responsible for selecting commodities to present to customers, while the environment represents the collective of system users. Subsequently, we can formulate the task into a Markov decision process as follows, thus facilitating seamless integration with reinforcement learning algorithms.

- State S: The user information provided to the agent, encompass both static and dynamic information, which serves as the basis for decision-making. As for the item information, it is considered as a part of environment, which will not be conveyed to the recommender agent.
- Action A: For the recommender engine, directly select the suitable commodities from a pool of millions of items is deemed inefficient. As an alternative approach, we advocate to extract continuous feature vectors to construct the space of actions, wherein each action shares equivalent dimensional attributes with the features of the commodities. Subsequently, recommendations are refined through the similarity between user and commodity features. Therefore, a continuous action space composed by the user feature, aligning in dimensions with the item feature space, is employed. The extracted features will be used to calculate the similarity with candidate items, generating a ranked list of commodities. Only a predetermined number of top-ranked commodities are selected as recommendations exposed to the users.
- Reward R: As the environmental feedback signal, the reward signal ties highly with the overarching objective. For the sake of procedural simplicity, the agent is assigned a positive reward when a user engages actively with the recommendations, such as purchasing or clicking. The numerical reward setting will be introduced in the following section varying according to the environments.
- Transition P: Transition function denotes the likelihood of progressing from the current state to the ensuing state, which often hinges upon environmental dynamics. In recommendation context, the transitions predominantly arise from diverse user behaviors. Because the state is composed by the static and dynamic user information, the transition of static state information will unchanged with probability 1, while the dynamic state information will transfer according to a pre-designed rules in response to diverse agent actions.
- Discount factor γ: To consider the time value of the reward, a factor $\gamma \in [0, 1]$ is used to balance the relative significance of between the future rewards and accrued in the present.

Thereby, the Markov decision process can be formally expressed as the quintuple $< S, A, R, P, \gamma >$. Through this formulation, the computation of long-term cumulative rewards at time step t can be expressed as $G_t = \sum_{i=t} \gamma^{i-t} R_i$ for $i \in \{t, t+1, t+2, ...\}$. The objective is to pursue the maximum expectation of the long-term cumulative rewards, donated as $\mathbf{E}[G_t]$. The optimal policy π^* is $\text{argmax}_\pi \mathbf{E}[G_{t=0}|\pi]$.

The interaction process is depicted in Figure 1. From the view of the merchant, once a customer arrives, the recommender engine receives the user information. It then proceeds to extract a feature vector to calculate the similarity scores and select a ranking list of commodities that align with the user's potential interests. Subsequently, the commodities are presented to the user, whose behavior will inflect both the reward and the information, particularly about dynamic updates, provided to the agent.

3.2 UISA: User Information Separating Architecture

In the context of commodity recommendation, it is commonly observed that static information is characterized by sparsity, whereas dynamic information tends to be abundant and heterogeneous. Treating static and dynamic information as a homogeneous entity poses challenges in quantifying their mutual influence and discerning shifts in user implicit preferences induced by alterations in dynamic information. Acknowledging this intrinsic disparity, a rational approach involves separately processing these information, followed by an evaluation of their interplay. Inspired by the potential enhancements in performance attributed to the assessment of user preference drift, particularly those manifested in dynamic information alterations, the discrete treatment of dynamic and static information appears to offer utility. Consequently, we propose a User Information Separating Architecture (UISA), with the objective of optimizing recommendation performance.

The core idea underpinning the proposed architecture centers on the distinct processing of user information. Static information contains the user's basic traits, which are fixed throughout sequential recommendation

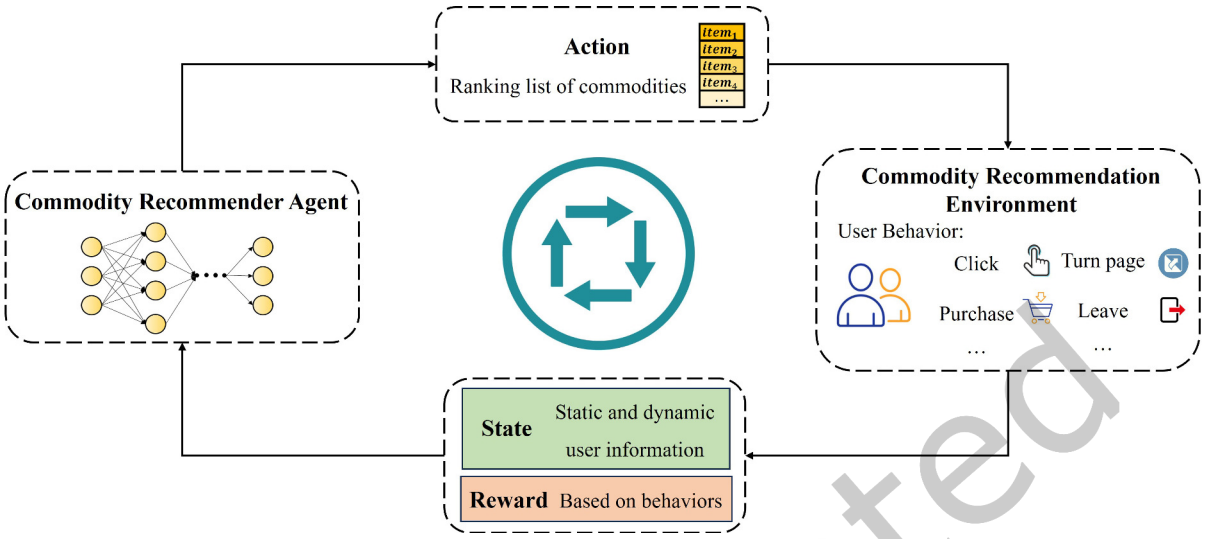

Fig. 1. Interaction between the recommender agent and the user it serves

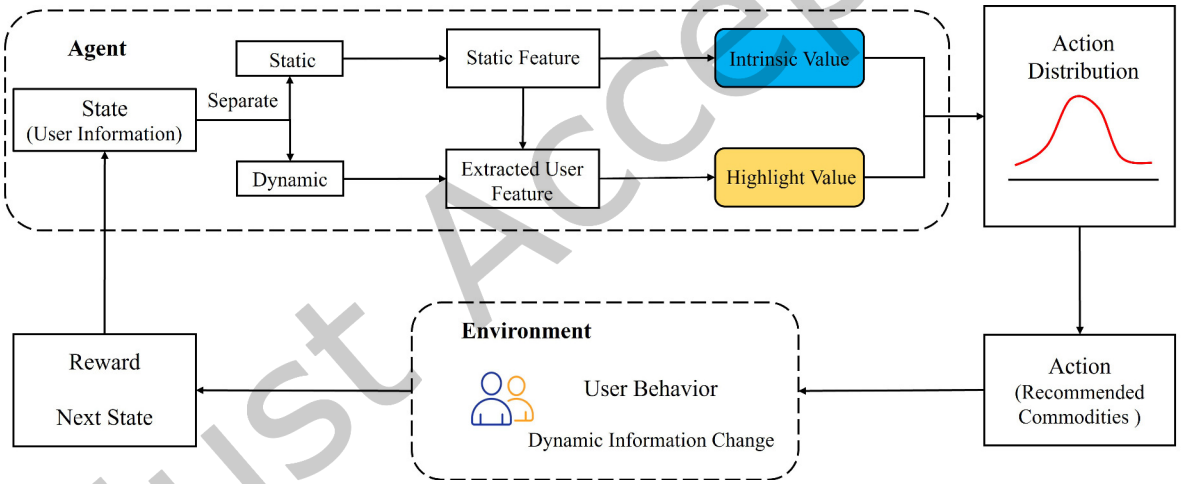

Fig. 2. User Information Separating Architecture (UISA)

processes. Conversely, dynamic information, frequently arising from sequential interactions, represents the user's attention patterns and evolving interests. As depicted in Figure 2, the architecture comprises two distinct streams: one dedicates to processing the static information to extract the latent static feature, while the other focuses on dynamic information and integrating it with the static hidden feature. The static hidden feature is instrumental in computing a value, named intrinsic value, representing the inherent aspects of the user's interest. Subsequently, a neural network is deployed to evaluate the interplay between static and dynamic features, signifying the user's current heightened area of interest, termed highlight value. Ultimately, the combination of intrinsic value and highlight value forms the foundation to generate the action distribution, from which the action list is derived.

Various operations, including addition and multiplication, can be leveraged to combine these two values. We donate the input information as i, static and dynamic information as i_s and i_d respectively. The process of generating the action distribution with the multi-layer perceptions (MLPs) can be expressed by the following briefly:

$$i_s, i_d = \text{sep}(i), \tag{1}$$

$$h_s = \text{MLPs}(i_s), \tag{2}$$

$$h_d = \text{MLPs}(i_d), \tag{3}$$

$$h_u = \text{concat}(h_s, h_d), \tag{4}$$

$$v_{intr} = \text{MLPs}(h_s), \tag{5}$$

$$v_{high} = \text{MLPs}(h_u), \tag{6}$$

$$p = \text{F}(v_{intr}, v_{high}), \tag{7}$$

where the operation of sep separates the static information and dynamic information; h_s and h_u donate the extracted hidden feature of static and dynamic user information; v_{intr} and v_{high} are the evaluated intrinsic and highlight value vector; F is the add or multiply operation on the intrinsic and highlight value, which generates the action distribution p. Specifically, the distribution can be derived with mean and standard variance parameters under the assumption that it follows a normal distribution. The process of input state separation is readily attainable, incurring no significant increase in computational resource requirements.

3.3 Reinforcement Learning Based User Information Separating Recommender Algorithm

The UISA can easily integrate with classic reinforcement learning algorithms, offering advantages in addressing sequential recommendation challenges. To illustrate it succinctly, we combine it with the Proximal Policy Gradient (PPO) algorithm as an example, which integrates UISA into the actor network to generate the action distribution, as illustrated in Figure 3

PPO algorithm employs a clipped objective function within a trust region framework to enhance learning stability and exploration in complex environments. As an actor-critic-based algorithm, PPO consists of an actor network π_θ, responsible for action selection, and a critic network V_ϕ evaluating the corresponding value. In decision-making phase, PPO agent is tasked with assessing the user's interest value across various dimensions. Through integration with our proposed architecture, the actor computes distinct interest values for each feature dimension, which has direct inflection to the final action selection. Therefore, we incorporate the separating architecture into the actor-network. The main actor network divided into two processing channel: one process the static information, while the other channel process the dynamic information and concatenate it with the static feature to compute the highlight value. The two channel merge in the last to generate the mean value and the standard deviation. Under the assumption of a normal distribution, the action of the output dimensions can be sampled according to the distribution. Subsequently, the top-ranked commodity list is acquired through the computation of the similarity scores with the candidate items. We apply MLPs to extract the segregated information, which served as the basis for determining both means and standard deviations of each action dimension, to generate continuous action representations. The architecture of the critic network, which considers the interaction between the user and the recommender system, remains unaltered. In the process of optimizing the network parameters, batch updating is employed. The actor-network aims to optimize a clipped objective L_a based on advantage function A^{π_θ}, as Eq. 8 and Eq. 9,

$$L_a = \min\left(\frac{\pi_\theta(a \mid s)}{\pi_{\theta_k}(a \mid s)} A^{\pi_{\theta_k}}(s,a), \quad \text{clip}\left(\epsilon, A^{\pi_{\theta_k}}(s,a)\right)\right). \tag{8}$$

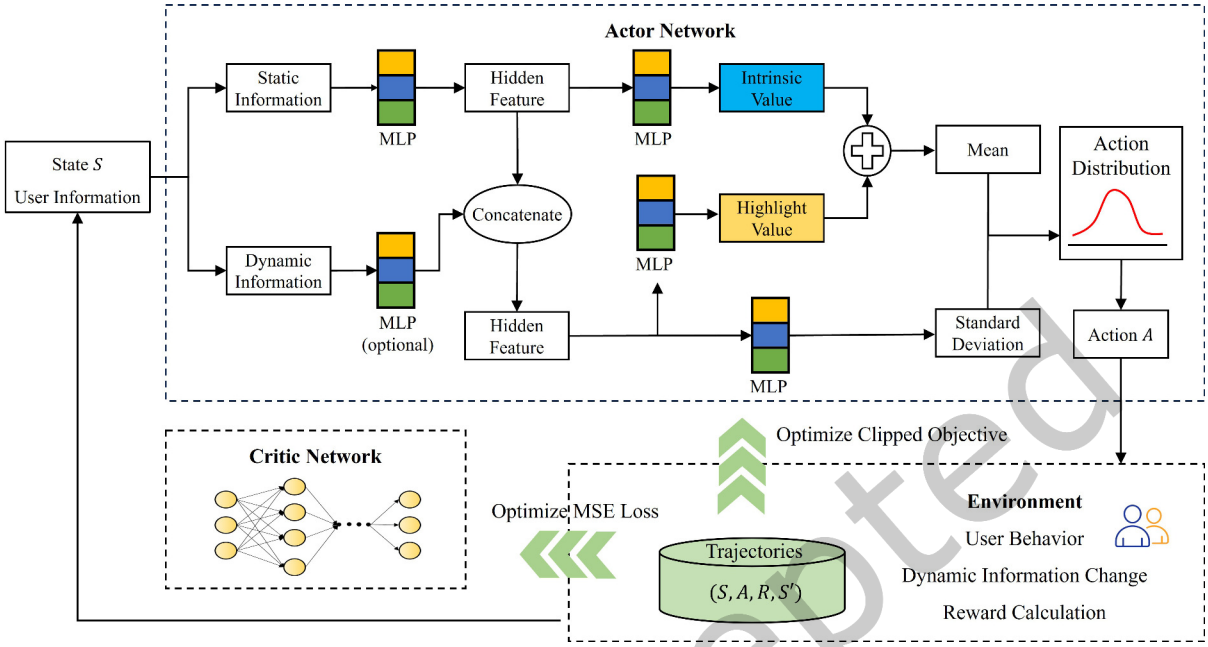

Fig. 3. PPO based User Information Separating Model

$$\text{clip}\,(\epsilon, A) = \begin{cases} (1+\epsilon)A & A \geq 0 \\ (1-\epsilon)A & A < 0 \end{cases}, \tag{9}$$

where the advantage function at time step t can be estimated with Eq. 10,

$$A_t = -V_\phi(s_t) + r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots + \gamma^{T-t} r_T. \tag{10}$$

For the critic, the mean squared error (MSE) loss L_c between the critic value $V_\phi(s)$ and target value y is preferred, as Eq. 11 and Eq. 12. By leveraging policy gradient techniques, the parameters are iteratively updated in the direction that maximizes long-term returns, as Eq. 13 and Eq. 14, where α^θ and α^ϕ are the learning rate parameters. Algorithm 1 shows how to combine the separating architecture with the PPO algorithm.

$$L_c = \min(V_\phi(s) - y)^2, \tag{11}$$

$$y = \begin{cases} r & \text{if episode terminate at the next time step} \\ r + V_\phi(s') & \text{otherwise} \end{cases}. \tag{12}$$

$$\theta_{k+1} = \theta_k + \alpha^\theta \nabla_\theta L_a \tag{13}$$

$$\phi_{k+1} = \phi_k - \alpha^\phi \nabla_\phi L_c \tag{14}$$

Similarly, the integration of UISA with other policy-based or actor-critic reinforcement learning algorithms can be seamlessly accomplished by incorporating the two-channel structural information processing into the agent policy networks. Additionally, experimental results for the deep deterministic policy gradient (DDPG) algorithm with UISA are presented in the next section. While the UISA primarily concentrates on the feature representation, it compiles with reinforcement on the aspect of value and interest.

Algorithm 1 PPO Based User Feature Separating Recommendation Algorithm

Input: Initialize actor network parameters θ_0, critic network parameters ϕ_0, and set training epochs N_{ep}.

1: **for** $k = 0$ to N_{ep} **do**
2: Separate the user information and attain the intrinsic value v_{intr} and highlight value v_{high}, as Eq. 1 - Eq. 6
3: Generate the action distribution p by calculating the means and standard deviations according to v_{intr} and v_{intr}, as Eq. 7
4: Sample actions $a(s)$ according to the distribution p and subsequently calculate the similarity scores to generate a ranked list of items
5: Collect the set of trajectories $D = \{\tau_i = (s_i, a(s_i), r, s_i')\}$ from the interaction between the users and recommendation environment
6: Compute the target value y_t as Eq. 12
7: Compute advantage estimate $\hat{A}_t$ based on the current value function V_{ϕ_k}
8: Update the actor network $\pi(\theta_k)$ by maximizing the clipped objective L_a, as Eq. 8 and Eq. 13
9: Update the critic network V_{ϕ_k} by minimizing the MSE loss L_c, as Eq. 11 and Eq. 14
10: **end for**

4 EXPERIMENTS

In this section, we build a reinforcement recommendation environment named JDEnv and conduct empirical experiments within JDEnv and Virtual Taobao environment [23]. The objective is to substantiate the efficacy of the UISA, with a specific emphasis on verifying the enhancement of recommendation performance and the universality across various reinforcement learning algorithms. Furthermore, we select a neural recommender baseline and conduct comparative experiments on JDEnv environment.

4.1 Commodity Recommendation Environment

Applying the algorithm to the real world directly is costly and risky. Therefore, learning the policy in a virtual interactive environment for a recommender agent and evaluating its performance seems more reasonable. That is, we conduct experiments in an offline manner, based on a self-built JDEnv environment and a Virtual-Taobao environment constructed by [23].

4.1.1 JDEnv Environment. JDEnv environment is constructed based on the JData dataset, which has been extensively employed for recommendation purposes [35] [17]. This dataset originates from a prominent Chinese e-commerce platform [1]. To establish JDEnv environment, we extract data from JData dataset, consisting of 24,187 distinct products, a total of 20,948 unique users, and 3,597,713 interaction records. Among all the distinct products, only 3329 items exhibit interaction records with users, constituting 13.76%. On average, each user maintains interaction records with 13.87 distinct items. The interactions encompass six categories: browsing the product details page, adding to the shopping cart, deleting from the shopping cart, placing an order, following, and clicking. The items within these six categories are stratified into sets comprising four levels, and the statistical information pertaining to the interactions, employed in constructing JDEnv, is presented in Table 1. It is worth noting that for distinct users, the same product may be categorized into different level sets, hence resulting in the sum of items across all four levels exceeding the total number of interaction items. The fourth column provides the average count of interactive products for each user across different levels. Notably, we observe that the act of directly placing an order exhibits a lower count compared to other levels, posing a challenge for recommendation systems.

[1]The e-commerce website is https://www.jd.com/, and the JData dataset can be found at https://www.kaggle.com/datasets/owincontext/jdata2016

Table 1. Division of 4 Item Levels

Level Set	Category of Interactions	Number of Items	Number of Interactive Items per User
Level 1	placing an order	438	0.1241
Level 2	browsing the product details page dding to shopping cart clicking	3329	13.83
Level 3	following	1137	0.36
Level 4	deleting from the shopping cart	1256	0.70

The classification of user actions into four levels is contingent upon their contribution to the final objective. Consequently, the level partition is intricately connected to the reward design and metric calculation processes.

JDEnv serves as a sequential recommendation environment, wherein the recommender agent endeavors to generate the optimal values of mean and standard variance for action sampling. Sequentially, the environment status dynamically changes in response to the user interactions with the recommended commodity lists. The static user information contains 16 dimensions, primarily constituted by fundamental user attributes such as gender and age. Whereas the dynamic information incorporates a broader spectrum, taking into account the user's history behavior, attributes distribution of purchased items, and relevant information regarding recommended items, resulting in a 25-dimensional representation. For the commodity features, an auto-encoder is employed. The resulting commodity features are subsequently encoded from the original 113-dimensional space into a 32-dimensional space, enabling the extraction of meaningful representations. During interactions with the environment, users engage with the recommended commodities, displaying behaviors that are consistent with the four-level interactions previously defined.

The formulation of the reward is rooted in the four-level interaction sets. In accordance with the varying levels of recommendation, distinct rewards are assigned to corresponding actions, as denoted by Eq. 15. This reward design affords increased flexibility, enabling the cumulative reward to serve as a reflective measure of merchants' profitability and users' satisfaction. Moreover, it facilitates straightforward adaptations to align with specific scenario requirements. The cumulative reward is weighted by the discount factor γ. Furthermore, the computation of click-through rate (CTR) is delineated in Eq. 16. Each distinct behavior type induces specific variations in both reward outcomes and click-through rates. While the consideration of CTR is omitted in the construction of reward design, it is evident that CTR exhibits a positive correlation with the reward metric. For each user, the environment conducts a fixed number of sequential recommendation rounds, subsequently providing the reward and CTR for each round. Moreover, the environment can generate a predetermined number of users' trajectories (specifically, we set this value to 256) at the same time step in parallel, which speeds up the process of simulating experience, enhancing computational efficiency.

$$R = \begin{cases} 1 & \text{if the recommended item in level set 1} \\ 0.5 & \text{if the recommended item in level set 2} \\ -1 & \text{if the recommended item in level set 4} \\ 0 & \text{others} \end{cases}, \tag{15}$$

$$CTR = \begin{cases} 1 & \text{if the recommended item in level set 1 or 2} \\ 0 & \text{others} \end{cases}. \tag{16}$$

In the training and testing phases, the dataset is partitioned randomly based on user IDs, ensuring that users within the testing set are entirely distinct from those within the training set. To be specific, the training set encompasses 20,110 unique users, while the testing set comprises 838 distinct users, which do not occur in the training set. In the training and testing stage, sequentially selecting the personalized items from a pool comprising 24,187 distinct products is performed for each user.

4.1.2 Virtual Taobao Environment. Virtual Taobao environment is constructed using a composite of deep learning models trained on authentic e-commerce website data [23]. This environment demonstrates the capability to generate synthetic customer profiles, forecast customer behavior, and compute ensuing interactions. Several fundamental environment settings of Virtual Taobao are as follows: The static user information consists of 11 attributes, encoded into 88 binary dimensions with one-hot technique, and the dynamic information is 3-dimensional, including customer browsing history and current step counts. The commodity information is projected into a 27-dimensional feature space. The customer conveys the feedback signal to the recommendations, through various actions, including buying, turning to the next page, and leaving. Regarding the reward, it is defined such that the engine receives a signal of 1 when a customer makes a purchase, and 0 otherwise. The transitions of the environment states are governed by well-trained models. The users' arrival, actions, and leaving behaviors are all determined by corresponding pre-trained deep learning models.

For Virtual Taobao, the training data are generated through simulation from the environment. To ensure equitable evaluation across various methodologies, instead of generating randomly from the environment, the testing data are selected from the source data released by the project and held constant for testing. Specifically, 964 unique users' information is randomly chosen to initialize the testing environmental state.

4.2 Experiment Results

To evaluate the performance, we conduct experiments involving the integration of the proposed UISA with both DDPG algorithm and PPO algorithm. Moreover, we compare the performance against neural collaborative filtering models, revealing that the UISA models achieve promising results.

4.2.1 Reinforcement Learning Experiments. Due to the distinct handling of user information state, there are slight discrepancies in the neural network structure, yet this adjustment does not significantly impact computational resource requirements. We construct a simplified model, employing one or two fully-connected layers for each MLPs, and maintain consistency in most hyperparameter settings, like batch sizes and the neurons in the hidden layer. For instance, in the PPO-based experiments, we configure the discount factor γ as 0.99, the learning rates for actor and critic as 0.00001 and 0.001, respectively, and set the batch size to 128. Throughout the experiments, particular attention is given to the reward signal and CTR metric. The accumulated discounted reward serves as an intuitive metric for evaluating model performance in terms of profitability, while the CTR metric reflects the alignment between recommended items and user preferences. In order to evaluate different algorithms, we conducted 5 independent runs of experiments for each model and compared the average reward and CTR on the testing dataset in tandem with the training process. Notice that, distinct environments and algorithms entail varying episode lengths, which is caused by the trajectory generation and network update mechanisms. Furthermore, the time spent on training does not exhibit substantial variation for a given environment.

Figure 4 and Figure 5 depict the performance of testing datasets throughout the training process. The average reward for each run is computed over the final 30% of episodes. The results, along with the enhancements over the baseline across various metrics, are presented in Table 2. The notion of reward signifies the merchant's profit, while the CTR serves as an indicator of user satisfaction regarding the recommendations. All the models with UISA demonstrate superior performance on reward and CTR metrics. The outcomes demonstrate that the UISA achieve enhancements ranging from approximately 5% to 40% in both reward and CTR metrics across the

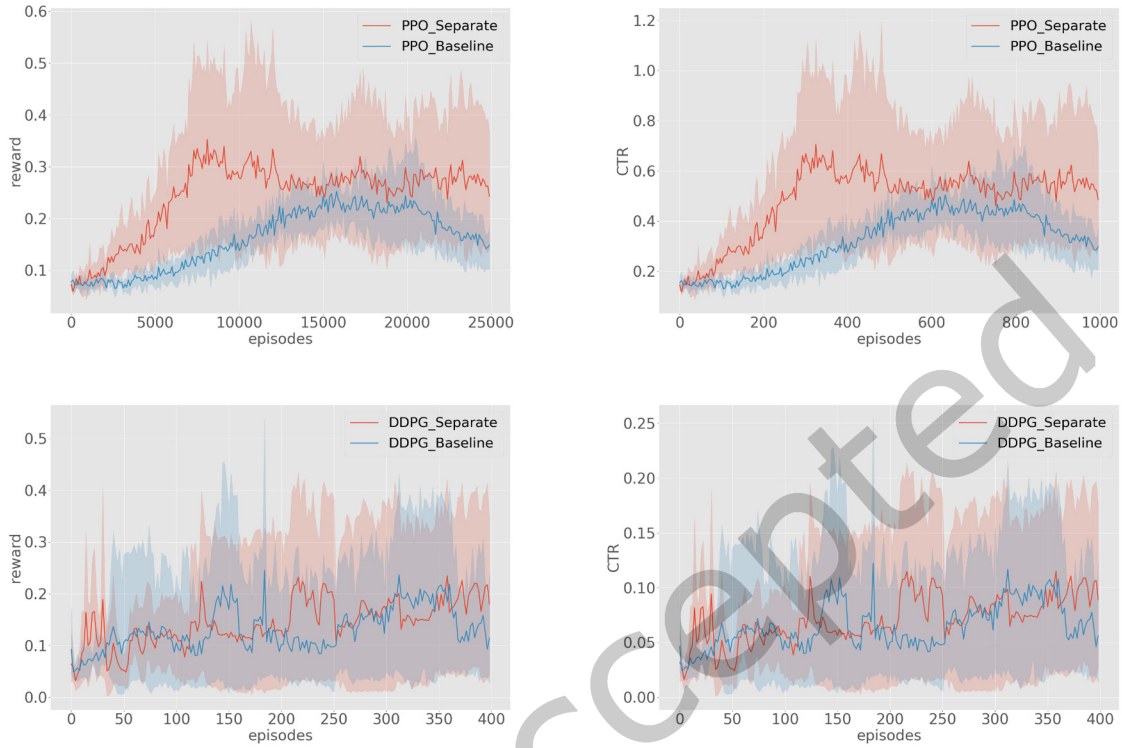

Fig. 4. Commodity Recommendation Experiments on JDEnv Environment

JDEnv and Vritual Taobao environments. Notably, the PPO Separating model exhibits the highest performance in the JDEnv environment with an improvement of approximately 38% over the baselines, possibly attributed to the richness of dynamic features (from 25 dimensions to 3 dimensions) and the adept learning capability inherent in the PPO algorithm. It is important to acknowledge that the comparatively lower CTR observed in the DDPG-based models may stem from the fact that CTR was not explicitly factored into the reward design. However, a lower CTR coupled with a higher reward indicates that the users are able to find their preferred items with fewer clicks. Furthermore, the DDPG-based models may not recommend items that a user clicks on but does not subsequently purchase. In the Virtual Taobao environment, the proposed model can achieve nearly 5% to 10% improvement in both two metrics. The initial slower growth of UISA in the Virtual Taobao Environment of the DDPG algorithm may be attributed to the algorithmic limitations and the relatively small dimensions of dynamic features in relation to the JDEnv. Consequently, the agent may encounter challenges in effectively learning the mutual influence, leading to an increased tendency to engage in exploratory actions during the early stages of training, which contributes to a slower-rising curve. It is essential to recognize that disparities in the magnitude of various performance indicators are primarily attributed to the distinctions between the two evaluation environments.

4.2.2 Comparison Experiments. In addition to reinforcement learning algorithms, we include a deep learning baseline to validate the effectiveness of UISA. Collaborative filtering models are widely employed in practical

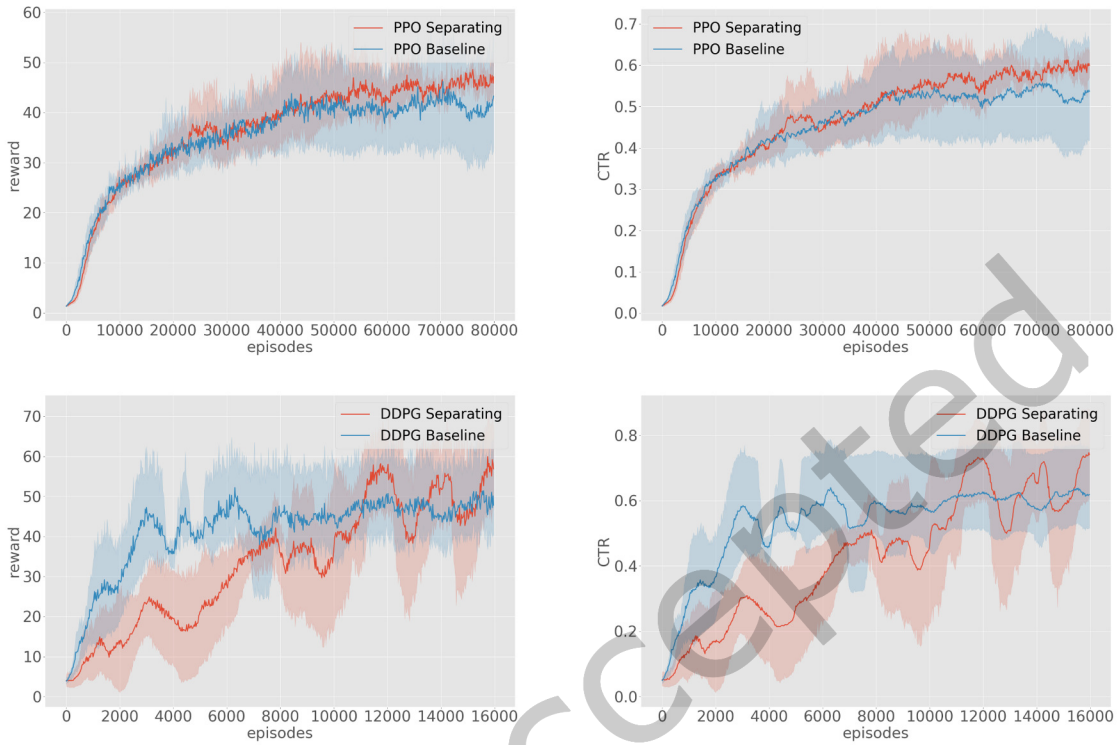

Fig. 5. Commodity Recommendation Experiments on Virtual Taobao Environment

Table 2. Performance and Improvement of Reward and CTR

Environment	Model	Average Reward	Reward Improvement	Average CTR	CTR Improvement
JDEnv	PPO Baseline	0.1969	-	0.3934	-
	PPO Separating	0.2715	+37.89%	0.5422	+37.82%
	DDPG Baseline	0.1666	-	0.0823	-
	DDPG Separating	0.1770	+6.24%	0.0872	+5.95%
Virtual Taobao	PPO Baseline	41.20	-	0.5290	-
	PPO Separating	45.00	+9.22%	0.5787	+9.40%
	DDPG Baseline	47.46	-	0.6106	-
	DDPG Separating	50.21	+5.79%	0.6446	+5.57%

recommender systems. As a deep-learning-based method, the Neural Collaborative Filtering (NCF) method models user-item interactions to perform recommendation [43]. In this context, NCF models, specifically the Multi-Layer Perception (MLP) model and the Generalized Matrix Factorization (GMF) model are chosen as baselines due to their great performance in pratical implementations. These models are designed to predict the probability

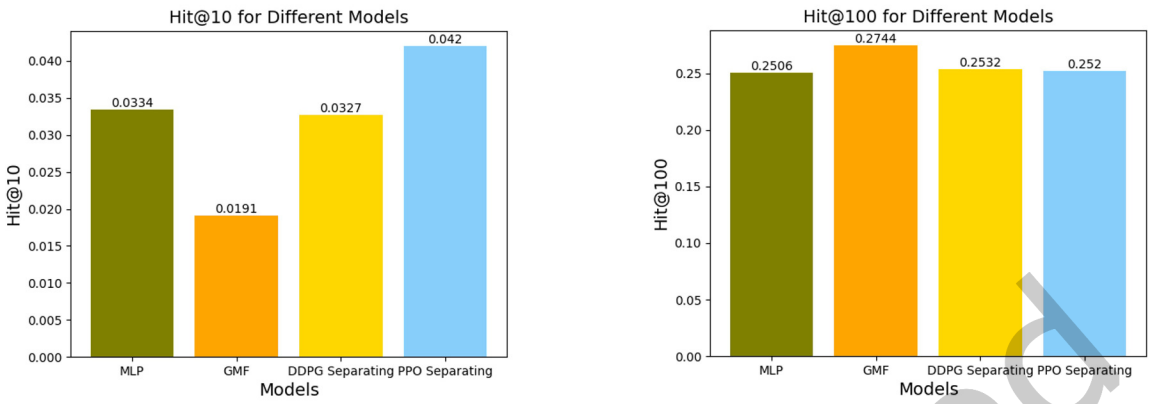

Fig. 6. Comparison Experiments with Neural Baseline Models

of the compatibility between the user and the item. In experiments, NCF models with fully connected layers are constructed. Rather than utilizing embeddings of user and item IDs, the input comprises the same user information as seen in UISA models and item features extracted by auto-encoder. Both MLP and GMF models are configured with a batch size of 128, a learning rate of 0.001, and a factor number of 32. The training and testing data are derived from the JData dataset, with the same testing users as those in the reinforcement learning experiments to ensure the fairness. For the training samples, positive instances are constructed using items from level sets 1 and 2 for each user, while negative instances are randomly selected from level sets 3 and 4, along with other items lacking user interactions. The number of positive and negative samples are 280,753 and 222,352, respectively, maintaining a ratio of approximately 1:1. The performance evaluation employs metrics of hit@10 and hit@100.

During the training process, an intriguing observation emerged: a higher reward does not consistently translate to an elevated hit@10 and hit@100. This phenomenon can be attributed to the fact that a higher accumulated reward may necessitate a higher proportion of positive items in a recommendation list, potentially biasing toward users with a greater inclination to make purchases. However, hit@10 and hit100 metrics only require the presence of at least one positive item in the recommendation list. Therefore, we present the optimal performance achieved by UISA models on the testing set, averaging the results in 5 runs. Figure 6 illustrates the results among MLP, GMF, DDPG Separating, and PPO Separating models. Regarding the hit 10 metrics, PPO Separating exhibits superior performance, while DDPG Separating achieves a comparable standing. In terms of hit 100 metrics, all four models demonstrate similar performance. However, during the testing stage, the NCF models not only required to consider both user and item information but also necessitated computing scores for each user across all 24,187 items through a deep network, which implies that there are 24187 user-item pairs need to be tested for each user, resulting in prolonged testing time. In this content, UISA with reinforcement learning proves to be potent, efficient, and computationally friendly.

Despite the promising outcomes achieved by UISA, certain limitations are evident, particularly concerning the stability of UISA models. Figure 4 and Figure 5 depict the variability of metrics observed on the testing dataset throughout the training phase. The empirical findings indicate that UISA models may not exhibit the same level of stability as neural recommender models, a phenomenon frequently encountered in a range of reinforcement learning contexts. Furthermore, it is acknowledged that reinforcement learning models are sensitive to the hyperparameter settings, which can significantly impact their performance. In our experimental setup, the

testing set remains consistent across both reinforcement learning experiments and comparison experiments. It is crucial to note that NCF models undergo training on the static dataset, whereas UISA models are trained through interactions with dynamic environments, which may contribute to the observed instability. Consequently, UISA models appear to be better suited for dynamic recommendation scenarios and demonstrate comparable performance.

5 CONCLUSION

This paper introduces a novel User Information Separating Architecture (UISA) tailored for acquiring a commodity recommendation policy, aligning well with the reinforcement learning framework. The utilization of both static and dynamic information renders it applicable to a wide array of recommendation scenarios. Empirical experiments conducted in virtual commodity recommendation environments, including Virtual Taobao and the self-built JDEnv, underscore its efficacy in enhancing future income. Based on experimental results, the UISA demonstrates an enhancement in the performance of reinforcement learning algorithms, yielding results comparable to neural baselines. The source code for the implementation is made openly available at https://github.com/MyLove-XAB/DRLRec.

For the future directions, three pivotal avenues merit attention. Firstly, in scenarios characterized by abundant and sequential dynamic features, it is prudent to explore the integration of recurrent neural networks (RNNs) or their variants, given their proficiency in handling temporal dependencies. Secondly, the advancement of a more intricate recommender-user interaction environment through simulation holds promise for enhancing model applicability. A notable gap persists between the environmental simulation and realistic dynamics. Particularly, augmenting JDEnv to mirror the intricacies of real-world e-commerce platforms is a noteworthy endeavor. Existing simulations often rely on historical data records, yet the integration of online data streams into a more realistic environment promises to yield more pragmatic outcomes. Lastly, directing focus towards session-based recommender systems, which accentuate the nuanced dynamics of current user-recommender sessions and interactions, presents a promising avenue for further exploration.

ACKNOWLEDGMENTS

This work was supported by the National Key Research and Development Program of China under Grant Nos. 2021YFA1000100 and 2021YFA1000104, Laboratory Project of Higher Education Institutions in Shandong Province-Energy System Intelligent Management and Policy Simulation Laboratory at China University of Petroleum, and Youth Innovation Team of Higher Education Institutions in Shandong Province-Data Intelligence Innovation Team at China University of Petroleum.

REFERENCES

[1] Minmin Chen, Alex Beutel, Paul Covington, Sagar Jain, Francois Belletti, and Ed H Chi. 2019. Top-k off-policy correction for a REINFORCE recommender system. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. 456–464.

[2] Chen Gao, Xiang Wang, Xiangnan He, and Yong Li. 2022. Graph neural networks for recommender system. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*. 1623–1625.

[3] Alexey Grishanov, Anastasia Ianina, and Konstantin Vorontsov. 2022. Multiobjective Evaluation of Reinforcement Learning Based Recommender Systems. In *Proceedings of the 16th ACM Conference on Recommender Systems*. 622–627.

[4] Xiao Gu, Haiping Zhao, and Ling Jian. 2022. Sequence neural network for recommendation with multi-feature fusion. *Expert Systems with Applications* 210 (2022), 118459.

[5] Jin Huang, Harrie Oosterhuis, Maarten De Rijke, and Herke Van Hoof. 2020. Keeping dataset biases out of the simulation: A debiased simulator for reinforcement learning based recommender systems. In *Proceedings of the 14th ACM Conference on Recommender Systems*. 190–199.

[6] Liwei Huang, Yutao Ma, Shibo Wang, and Yanbo Liu. 2019. An attention-based spatiotemporal lstm network for next poi recommendation. *IEEE Transactions on Services Computing* 14, 6 (2019), 1585–1597.

[7] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. 2333–2338.

[8] Zhiyang Jia, Yuting Yang, Wei Gao, and Xu Chen. 2015. User-based collaborative filtering for tourist attraction recommendations. In *2015 IEEE international conference on computational intelligence & communication technology*. IEEE, 22–25.

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

[10] Ming Li, Mozhdeh Ariannezhad, Andrew Yates, and Maarten de Rijke. 2023. Who will purchase this item next? Reverse next period recommendation in grocery shopping. *Recommender Systems* 1, 2 (2023), 1.

[11] Ninghao Liu, Yong Ge, Li Li, Xia Hu, Rui Chen, and Soo-Hyun Choi. 2020. Explainable recommender systems via resolving learning representations. In *Proceedings of the 29th ACM international conference on information & knowledge management*. 895–904.

[12] Qijiong Liu, Nuo Chen, Tetsuya Sakai, and Xiao-Ming Wu. 2023. A First Look at LLM-Powered Generative News Recommendation. *arXiv preprint arXiv:2305.06566* (2023).

[13] Yuhao Luo, Qianfang Xu, Wenliang Li, Feng Jiang, and Bo Xiao. 2021. A multi-step decision prediction model based on LightGBM. In *2021 IEEE International Conference on Big Data (Big Data)*. IEEE, 5714–5718.

[14] Hanjia Lyu, Song Jiang, Hanqing Zeng, Yinglong Xia, and Jiebo Luo. 2023. LLM-Rec: Personalized Recommendation via Prompting Large Language Models. *arXiv preprint arXiv:2307.15780* (2023).

[15] Yifei Ma, Balakrishnan Narayanaswamy, Haibin Lin, and Hao Ding. 2020. Temporal-contextual recommendation in real-time. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 2291–2299.

[16] Daniel J Mankowitz, Andrea Michi, Anton Zhernov, Marco Gelmi, Marco Selvi, Cosmin Paduraru, Edouard Leurent, Shariq Iqbal, Jean-Baptiste Lespiau, Alex Ahern, et al. 2023. Faster sorting algorithms discovered using deep reinforcement learning. *Nature* 618, 7964 (2023), 257–263.

[17] Wenjing Meng, Deqing Yang, and Yanghua Xiao. 2020. Incorporating user micro-behaviors and item knowledge into multi-task learning for session-based recommendation. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in Information Retrieval*. 1091–1100.

[18] N Nikzad-Khasmakhia, M Reza Feizi-Derakhshia, and Cina Motamedb. 2020. BERTERS: Multimodal Representation Learning for Expert Recommendation System with Transformer. *arXiv preprint arXiv:2007.07229* (2020).

[19] 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.

[20] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*. 285–295.

[21] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2002. Incremental singular value decomposition algorithms for highly scalable recommender systems. In *Fifth international conference on computer and information science*, Vol. 1. Citeseer, 27–8.

[22] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *CoRR, abs/1707.06347* (2017).

[23] Jing-Cheng Shi, Yang Yu, Qing Da, Shi-Yong Chen, and An-Xiang Zeng. 2019. Virtual-taobao: Virtualizing real-world online retail environment for reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 4902–4909.

[24] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of Go with deep neural networks and tree search. *nature* 529, 7587 (2016), 484–489.

[25] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

[26] Shaohua Tao, Runhe Qiu, Yuan Ping, and Hui Ma. 2021. Multi-modal knowledge-aware reinforcement learning network for explainable recommendation. *Knowledge-Based Systems* 227 (2021), 107217.

[27] Chirayu Upadhyay, Hasan Abu-Rasheed, Christian Weber, and Madjid Fathi. 2021. Explainable job-posting recommendations using knowledge graphs and named entity recognition. In *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 3291–3296.

[28] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature* 575, 7782 (2019), 350–354.

[29] Frank Edward Walter, Stefano Battiston, and Frank Schweitzer. 2008. A model of a trust-based recommendation system on a social network. *Autonomous Agents and Multi-Agent Systems* 16 (2008), 57–74.

[30] Fan Wang, Haibin Zhu, Gautam Srivastava, Shancang Li, Mohammad R Khosravi, and Lianyong Qi. 2021. Robust collaborative filtering recommendation with user-item-trust records. *IEEE Transactions on Computational Social Systems* 9, 4 (2021), 986–996.

[31] Jun Wang, Arjen P De Vries, and Marcel JT Reinders. 2006. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*. 501–508.

[32] 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.

[33] Shoujin Wang, Liang Hu, Longbing Cao, Xiaoshui Huang, Defu Lian, and Wei Liu. 2018. Attention-based transactional context embedding for next-item recommendation. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 32.

[34] S Wang, L Hu, Y Wang, L Cao, QZ Sheng, and M Orgun. 2019. Sequential recommender systems: Challenges, progress and prospects. In *Twenty-Eighth International Joint Conference on Artificial Intelligence {IJCAI-19}*. International Joint Conferences on Artificial Intelligence Organization.

[35] Ting-Yun Wang, Chiao-Ting Chen, Ju-Chun Huang, and Szu-Hao Huang. 2023. Modeling cross-session information with multi-interest graph neural networks for the next-item recommendation. *ACM Transactions on Knowledge Discovery from Data* 17, 1 (2023), 1–28.

[36] Xiting Wang, Yiru Chen, Jie Yang, Le Wu, Zhengtao Wu, and Xing Xie. 2018. A reinforcement learning framework for explainable recommendation. In *2018 IEEE international conference on data mining (ICDM)*. IEEE, 587–596.

[37] Xing Wu, Yushun Fan, Jia Zhang, Haozhe Lin, and Junqi Zhang. 2019. QF-RNN: QI-matrix factorization based RNN for time-aware service recommendation. In *2019 IEEE international conference on services computing (SCC)*. IEEE, 202–209.

[38] Aobo Xu and Ling Jian. 2023. A Deep News Headline Generation Model with REINFORCE Filter. In *2023 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–7.

[39] Yang Xu, Lei Zhu, Zhiyong Cheng, Jingjing Li, and Jiande Sun. 2020. Multi-feature discrete collaborative filtering for fast cold-start recommendation. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 34. 270–278.

[40] Zhichao Xu, Hansi Zeng, Juntao Tan, Zuohui Fu, Yongfeng Zhang, and Qingyao Ai. 2023. A Reusable Model-agnostic Framework for Faithfully Explainable Recommendation and System Scrutability. *ACM Transactions on Information Systems* (2023).

[41] Xinyang Yi, Ji Yang, Lichan Hong, Derek Zhiyuan Cheng, Lukasz Heldt, Aditee Kumthekar, Zhe Zhao, Li Wei, and Ed Chi. 2019. Sampling-bias-corrected neural modeling for large corpus item recommendations. In *Proceedings of the 13th ACM Conference on Recommender Systems*. 269–277.

[42] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 974–983.

[43] Eva Zangerle and Christine Bauer. 2022. Evaluating recommender systems: survey and framework. *Comput. Surveys* 55, 8 (2022), 1–38.

[44] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep learning based recommender system: A survey and new perspectives. *ACM computing surveys (CSUR)* 52, 1 (2019), 1–38.

[45] Xiangyu Zhao, Long Xia, Lixin Zou, Hui Liu, Dawei Yin, and Jiliang Tang. 2021. Usersim: User simulation via supervised generativead-versarial network. In *Proceedings of the Web Conference 2021*. 3582–3589.

[46] Xiangyu Zhao, Long Xia, Lixin Zou, Dawei Yin, and Jiliang Tang. 2019. Toward simulating environments in reinforcement learning based recommendations. *arXiv preprint arXiv:1906.11462* (2019).

[47] Zhi-Dan Zhao and Ming-Sheng Shang. 2010. User-based collaborative-filtering recommendation algorithms on hadoop. In *2010 third international conference on knowledge discovery and data mining*. IEEE, 478–481.

[48] Guanjie Zheng, Fuzheng Zhang, Zihan Zheng, Yang Xiang, Nicholas Jing Yuan, Xing Xie, and Zhenhui Li. 2018. DRN: A deep reinforcement learning framework for news recommendation. In *Proceedings of the 2018 world wide web conference*. 167–176.

[49] Fei Zhou, Biao Luo, Tianmeng Hu, Zihan Chen, and Yilin Wen. 2021. A Combinatorial Recommendation System Framework Based on Deep Reinforcement Learning. In *2021 IEEE International Conference on Big Data (Big Data)*. IEEE, 5733–5740.