A Multi-Agent Framework for Recommendation with Heterogeneous Sources

Yabin Zhang[1], Weiqi Shao[1], Xu Chen[1*], Yali Du[2], Xiaoxiao Xu[3], Dong Zheng[3],
Changhua Pei[4], Shuai Zhang[3], Peng Jiang[3], Kun Gai
[1]Gaoling School of Artificial Intelligence, Renmin University of China, Beijing, China
[2]King College London, UK
[3]Kuaishou Technology, Beijing, China
[4]Chinese Academy of Sciences, Beijing, China
{yabin.zhang,shaoweiqi,xu.chen}@ruc.edu.cn,yali.du@kcl.ac.uk,chpei@cnic.cn,
{xuxiaoxiao05,zhengdong,zhangshuai,jiangpeng}@kuaishou.com, gai.kun@qq.com

2023 International Joint Conference on Neural Networks (IJCNN) | 978-1-6654-8867-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/IJCNN54540.2023.10191154

***Abstract*—With the ever prospering of the web technologies, there is a common need to make recommendations from heterogeneous sources, such as recommending products and advertisements together on the e-commerce websites. People usually solve such recommendation problem by a two-stage paradigm, where the first stage is generating candidates from each source, and the second one is aggregating and ranking the generated heterogeneous candidates to produce the final results. While existing models have achieved many successes, they mostly optimize the above two stages separately, where the user preferences can only be used to supervise the second stage, while for the first one, there is no signal to tell whether the generated candidates are accurate enough to cover the user preference. To solve the above problem, in this paper, we design a multi-agent framework to jointly optimize the above two stages. In specific, suppose there are N sources in our problem, then we deploy N+1 agents, where the first N agents correspond one-to-one with the sources, aiming to select the sources-specific candidates, and the last agent is designed to aggregate the candidates from different sources for the final recommendation. All the agents play a cooperative game, aiming to maximize the rewards revealing user preferences. We implement our idea based on the Deep Q-network, where we design a decomposable reward to enhance the training efficiency. We adapt our model to a real-world recommendation problem abstracted from a famous short video platform–*Kuaishou.com*. We conduct extensive experiments to demonstrate the effectiveness of our model.**

***Index Terms*—recommender system, multi-agent reinforcement learning, heterogeneous sources**

I. Introduction

Recommender system (Recsys) has been deployed in a wide-range of real-world applications, bringing a large amount of business values [1], [2], [3]. Traditional models usually recommend items from the same source (*e.g.*, products on the e-commerce website [4], [5], movies on the video sharing platform [6]). However, with the development of the web technology, recommender system becomes more and more complex, there is a common need to generate recommendations from heterogeneous sources. For example, recommending videos and advertisements together to jointly achieve high user utilities and business purposes. The videos and advertisements are from different sources or even various departments in a company, and their feature spaces are complete different, which makes them hard to be directly compared.

To solve this problem, people usually adopt a two-stage strategy [7]: in the first stage, the items from different sources are compared separately to generate source-specific candidates. In the second stage, the generated candidates are aggregated and ranked to produce the final results. Usually, the above two stages are realized separately, which may fail to leverage the user preference to jointly optimize the them together. More specifically, there can be two straightforward strategies: on one hand, one can implement the first stage by heuristic rules like selecting the most popular items and so on, and the second stage is used to select the items that the user is interested in. In this method, the first stage is irrelevant with the user preference, thus it is hard to guarantee that the candidates from the first stage can cover comprehensive user preferences. On the other hand, one can also implement the first and second stages by using the same user preference signals. However, such method cannot distinguish the contributions of the first and second stages for the final user behaviors. For example, with the same supervision signals, it is hard to tell whether the user satisfaction on an item is because that the first stage has correctly selected this item or the second stage has higher-ranked it. It is even harder to identify how different stages contribute the supervision signals.

To better accomplish the task of recommendation with heterogeneous sources, we propose a **M**ulti-**A**gent framework to jointly optimize the above **T**wo-**S**tages, denoted as MATS. While this seems to be a promising direction, there are many challenges: to begin with, how to formulate the algorithms in the first and second stages by the agents, and how to define the multi-agent cooperative game, so as to jointly optimize the user rewards require our careful designs. In addition, the joint action space in the first and second stages can be extremely large, how to design methods to reduce the computation cost may also challenge our implementation. In order to solve these challenges, we assign each source with an agent, which is used to select source-specific candidates from the item pools, named

*Corresponding author

as *selector*. Based on the selected candidates, another aggregation agent is deployed in the second stage to rank the items generated from the first stage for making recommendations, named as *aggregator*. To improve the computation efficiency, we introduce a decomposable Q-network, where the Q value for each agent is independent, and the joint Q function is the sum of the Q values for different agents. We conduct extensive experiments to demonstrate the effectiveness of the MATS.

In a summary, the main contributions of this paper can be summarized as follows:

• We formally define the task of recommendation with heterogeneous sources, which to the best of our knowledge is the first time in the recommendation domain.

• To accomplish the above task, we propose a multi-agent framework to jointly optimize different stages, where we also introduce a decomposable reward for the first and the seond stage to enhance the training efficiency.

• We conduct extensive experiments to prove the effectiveness of the proposed model based on a real-world application.

II. Preliminary

A. Recsys with Heterogeneous Sources

For homogeneous source, the item pool only has one source (*e.g.*, advertisements). Let $\mathcal{U}$ and $\mathcal{V}$ denote a set of users and items, respectively, and $u \in \mathcal{U}$ or $v \in \mathcal{V}$ represent a user or an item. We denote $\mathcal{U}$ and $\mathcal{V}$ interaction history information as X (*e.g.*, interaction time, interaction context), and r_a is the user feedback on action (or items) (*e.g.*, click, rating, watch-time). Recsys with homogeneous sources aims to learn a policy π which maps interaction history information $x \in X$ to distributions π_x over action a, such that the expected reward $E_x E_{a\sim\pi_x}[r_a|x]$ is maximized [7].

While the item pool is heterogeneous sources, we denote heterogeneous sources item pool as $D_n \in \mathcal{D}(D_n \neq \emptyset, \cup_n D_n = \mathcal{D})$ where $n(n > 1)$ is number of the source. $A_i \in \mathcal{A}$ represents a finite set of action of i-th source based on item pool $D_i(i \in n)$. Heterogeneous sources differ from the homogeneous source ones by the two-step strategy of selecting a. First, each source selector picks a single candidate a_n from its assigned action pool A_n. Second, aggregator take an action a from the union pool of different candidate $C_n := a_1, a_2 \dots a_n$, and observes the reward $r = r_a$ associated with action a. The goal of Recsys with heterogeneous sources in two-stage strategy is expected reward maximization, *i.e.*, obtain optimal action set a_n^* and a^{**} from two-stage, formulate as follow:

$$a_n^* = \arg\max_{a_n \in A_n} f^*(x, a_n; \theta^*), a^{**} = \arg\max_{a \in C_n^*} f^{**}(x, a; \theta^{**}) \tag{1}$$

where $C_n^* = \cup_n a_n^*$, $f^*(.)$ and $f^{**}(.)$ represent selector and aggregator respectively. θ^* and θ^{**} mean model parameter.

B. Multi-agent Reinforcement Learning

However, due to the inherent problem of separate implementation in two-stage strategy, typical training process might fail to leverage the user preference to jointly optimize the them together, resulting in suboptimal performance. We provide a promising direction to improve two-stage strategy by leveraging multi-agent reinforcement learning (MARL).

MARL is derived from reinforcement learning (RL). RL [8] is a framework that enables an agent to learn via interactions with the environment. For traditional RL, at each step t, agent takes an action a_t according to its observation o_t from environment. And then the environment changes its state s_t, and given a reward r_t to the agent. After many interactions between agent and environment, the agent is able to learn a optimal policy π that can maximize the accumulated discount reward $R(s_t, a_t) = r(s_t, r_t) + \gamma r(s_{t+1}, r_{t+1}) + \gamma^2 r(s_{t+2}, r_{t+2}) + \cdots$, where γ is a discount factor.

Both Deep Q-learning (DQN) [9] and Double DQN [10] are value-based method, which is approximated by a "twins" deep neural networks (evaluate network and target network). The target network with parameters θ^-, are copied every τ step from evaluate network θ, and θ^- kept fixed on all other step. So "twins" network with same θ^-, the target network of DQN is defined as:

$$Y_t^{DQN} = R_{t+1} + \gamma Q(S_{t+1}, \arg\max_a Q(S_{t+1}, a; \theta_t^-); \theta_t^-) \tag{2}$$

and Double DQN [10] (DDQN) decouple the selection from evaluation, rewrite Eq. 2 as:

$$Y_t^{DDQN} = R_{t+1} + \gamma Q(S_{t+1}, \arg\max_a Q(S_{t+1}, a; \theta_t); \theta_t^-) \tag{3}$$

where the main different is that the selection of action a, in the argmax, is still due to the evaluate weights θ_t [10]. The design of selector and aggregator is inspired by DDQN.

In multi-agent reinforcement learning. In MARL [11], [12], [13], [14], there has only one common environment, but the number of agent is not only one. All agents interact with this common environment, concretely, each agent has their individual observation and is responsible for choosing actions from its own action set based on its individual policy function. In order to better utilize MARL for recommendation with heterogeneous source modeling, we present four main statements. More specifically: **Multi-Agent:** we model four agents contains three source agents (selector) and an aggregation agent (aggregator), each agent is given an independent network architecture. **Session-Decision:** three heterogeneous sources data are collected in session-based interaction, and four agents interacts with environment by sequentially choosing with timestamp. At each time step, first the state is represented by a user's previously interacted history, and then the agent takes an action to respond this state. **Semi-Cooperative:** four agents are semi-cooperative in two-stage strategy. In the first stage, three source agents not pass messages to each other for communication. In the second stage, each source agents with pass messages to aggregation agent for communication. However, there is still no communication between each source agent. **Partially-Observable:** in our multi-agent reinforcement learning model, the environment is partially observable, and each agent only receives a local information

instead of the global information of the environment. The overall performance of these four agents are evaluated by a centralized total Q-value. The total Q-value as a supervisory signal to directly affect aggregation agent recommendation and indirectly influence candidate selection of source agent.

The modeling difficulties of MARL for heterogeneous sources recommendation mainly include partially-observable and instability. Partially-observable is because each agent can only observe and make decisions from its own perspective, and cannot observe and make decisions from the global perspective, so it is unable to learn the global optimal strategy. Instability is because each agent is learning and the strategy is constantly changing, resulting in the update of the Q-value of different agents will be very unstable. Inspired by VDN [15], we introduce a global $Q(S, \mathcal{A})$ through value decomposition, and the formula as follows:

$$Q_{total}(S, \mathcal{A}; \theta) = \sum_{i=1}^{N} Q_i(S_i, A_i; \theta_i) \tag{4}$$

where N is the number of agents in environment, in our paper, N represents 4 that includes SR, SA, SF and AA, as shown in Fig. 1. After getting approximate $Q_{total}(S, \mathcal{A}; \theta)$, we updates $Q_{total}(S, \mathcal{A}; \theta)$ through global reward r with the idea of Double DQN, and the total loss function as follow:

$$L(\theta) = \frac{1}{M} \sum_{j=1}^{M} (y_j - Q_{total}(S, \mathcal{A}; \theta))^2 \tag{5}$$

where M is batch size, $y_j = r_j + \gamma \arg\max_a Q^-_{total}(\hat{S}, \mathcal{A}; \theta)$, $Q^-_{total} = \sum_{i=1}^{N} Q^-_i(S_i, A_i; \theta_i)$, Q^-_{total} is target network in Double DQN. Q_i is obtained via neural network, which is a tensor. Then we can concatenate all of Q_i in order to get Q_{total}. Next, update Q_{total} through TD-error [16], and the gradient will be back propagation to each Q_i through Q_{total} to update them. In this way, we can update Q_i from a global perspective with a global reward.

To sum up, the specific task of this paper is, given three heterogeneous sources data $\mathcal{D}$ [1] that contains recommending video (rec), advertisements video (ad) and fanstop video (fans), to infer users' true next preferences.

III. The MATS Model

A. Overview

The overview of MATS as shown in Fig. 1, which is proposed to solve the two-stage recommendation problem. Specifically, we set three heterogeneous videos selector agent (SR, SA, SF) for the first stage and a recommendation aggregator agent (AA) for the second stage. Selector decides which items in current source are selected as candidate item for the second stage aggregation. Aggregator then decides the recommendation order of the selected candidate items from the first stage. In general, the total model process of selector and aggregator as follow: 1) SR selects a ranked

[1] $\mathcal{D}$ is collected from one datastream in a famous short video platform–*Kuaishou.com*, and the **fans** is a special short video type in *Kuaishou.com*.

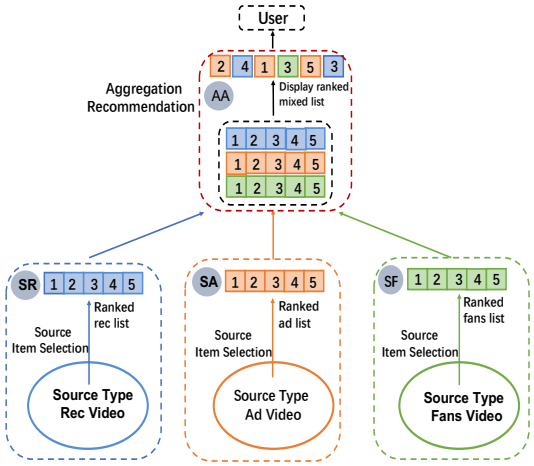


Fig. 1. The total overview of MATS model. Three source video (Rec, Ad, Fans) selector (SR, SA, SF) and a recommendation aggregator (AA).

list containing five rec videos $[1, 2, 3, 4, 5]$ from rec video pool, and the result $[1, 2, 3, 4, 5]$ is used as rec candidate items for the second stage; Similarly, SA and SF select ad and fans candidate videos list respectively; 2) three candidate list of rec, ad and fans are are fused to a new candidate item ($[[1, 2, 3, 4, 5], [1, 2, 3, 4, 5], [1, 2, 3, 4, 5]]$); 3) AA decides a ranked recommendation list $[2, 4, 1, 3, 5, 3]$ from the fuse candidate item for user. Each agent is assigned a novel Double DQN model architecture, which is described in subsection III-C and Fig. 2, and none of them share network parameters.

A general RL-based recommendation model has a sequence of experiences $s_1, a_1, r_1, s_2, a_2, r_2, \cdots, s_t, a_t, r_t$. We model four agents SR, SA, SF and AA, each agent corresponding to a particular optimization policy ($\pi_a, \pi_b, \pi_c, \pi_m$) and an action ($a_t, b_t, c_t, m_t$). As we statement in section II, the environment in our task is partially observable and not all informations are global observable. Specifically, the state s_t and reward r_t of the environment are global and shared by all four agents, while policies ($\pi_a, \pi_b, \pi_c, \pi_m$) and actions ($a_t, b_t, c_t, m_t$) are private, only possessed by each agent itself. Based on these particularities, we formulate our multi-agent reinforcement learning for two-stage strategy as follows:

1) Selector in the first stage: At the time step t, the state s_t is defined as the concatenation of triplet from time step t-1: state $s_{t\text{-}1}$, action $a_{t\text{-}1}$ and reward $r_{t\text{-}1}$, *i.e.*, $s_t = concat[s_{t\text{-}1}, a_{t\text{-}1}, r_{t\text{-}1}]$. For three heterogeneous videos, we have three selectors which means we have three private policies π_a, π_b, π_c, perceives state s_t and actions a_t, b_t, c_t. The action a_t, b_t and c_t will trigger the aggregator to decide how to present the items from these selected candidate items. The selector tries to capture the sequential patterns of user behaviors onto aggregated results in different recommendation item list. Note that the items from each source also have been ranked according to their private policy. In other words, in the first stages, each selector not only to select the suitable items from its source pool, but also tries to rank and get optimal order result. Formulate Q-value of SR, SA and SF at time t

Algorithm 1 Build Online Environment Simulator.

input: Initialize simulator network O with random weight. User's historical sessions in train data.
output: Simulator network O

for session = $1, M$ **do**
  observe initial state s_0
  for length = $1, T$ **do**
    observe current state s
    Initialize current list predict reward $r' = 0$
    for for item order $l = 1 : L$ **do**
      observe current action item a
      predict current reward r via simulator $O(s, a)$
      $r' += 0$
    end for
    update state s
    observe current list ground truth reward r
    minimize$(r - r')$
  end for
end for

as: $Q_{\pi_a}(s_t, a_t; \theta_a)$, $Q_{\pi_b}(s_t, b_t; \theta_b)$ and $Q_{\pi_c}(s_t, c_t; \theta_c)$, where θ_a, θ_b and θ_c mean model parameters.

2) Aggregator in the second stage: The aggregator in the second stage aims to rank the items generated from the first stage. Aggregator shared the same state s_t with three selectors at the time t, but the policy π_m is private. Aggregator selects an action m_t according to s_t and π_m. The action m_t means ranking an aggregation recommendation list for user. Then, the total model will receive the feedback reward r_t from user. The r_t is obtained by an online environment simulator which is shown in subsection III-B. Formulate Q-value of aggregator at time t as: $Q_{\pi_m}(s_t, m_t; \theta_m)$ and θ_m is model parameters.

B. Build Online Environment Simulator

Since the real feedback (dwell time) of user-item pairs is sparse in real datasets of *Kuaishou* platform, in this section, we need to build an online environment simulator. According to this simulator, we can predict a reward based on current unknown pair of state and action *i.e.*, $f : (s_t, a_t) \rightarrow r_t$. We follow previous work [17], [18] to build simulator for approximating the user reward generation process, which is shown in the Algorithm 1. Concretely, our simulator is designed as a two-layer fully connected neural network with ReLU as the activation function. The hidden layer size is (32, 16) and the learning rate is 1e-5, the optimizer is Adam. The input is a state-action pair, and the outputs are the estimated dwell time for recommending different item. The user simulator is learned based on the training sets, and the average reward (dwell time) prediction performance is based on terms of RMSE between prediction value and ground truth value, which is about 0.0052. The simulated online environment is trained on users' log data, which is different from the data used to train the recommendation model.

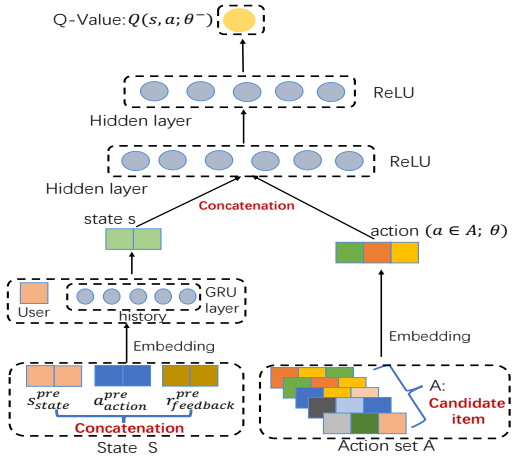


Fig. 2. A novel DDQN framework is proposed for selectors and aggregator.

C. A Novel Double DQN Framework

As we mentioned in section II, in our method, both three source agents and an aggregation agent are based on Double DQN (DDQN). We propose a novel DDQN framework based on our heterogeneous sources recommendation task. As shown in Fig. 2, it uses a multi-layer neural networks with parameters θ to evaluate the Q function and use a "twins" θ^- based on θ to select. The state s_t from previous sessions, which consists of three type elements, *i.e.*, previous state s^{pre}_{state}, previous action a^{pre}_{action}, previous feedback (reward) $r^{pre}_{feedback}$. In order to clearly describe state, we divide total state s_t into two parts, including user (user id) and history (browsed items, action, reward), *i.e.*, $s_t = [user, history]$. The user id is mapped to its user embedding to u_t first, then those history items id and rewards $i_t = [i_1, i_2, \cdots, i_N, r_t]$ are embedded through item embedding to $I_t = [I_1, I_2, \cdots, I_N, R_t]$, One GRU history layer is used in E.q 6,

$$w_t = \text{ReLU}(W_\phi \cdot \phi(I_i) + b_\phi), \;\; h_t = \text{GRU}(w_t, h_{t\text{-}1}) \quad (6)$$

where $\phi(I_i)$ is the vector representation of the history items and reward, h_t is last hidden layer of GRU. Combining user and history item embedding, we set state $s_t = [u_t, h_t]$.

The novel double DQN could help users to select or aggregation their favorite next item according to their previous preferences no matter in the first or second stage. Then, we input a state-action pair by concatenating $[s_t, a_t]$ into multi-layer neural networks. Finally, our DDQN outputs the Q-value $Q(s, a; \theta^-)$ corresponding to a specific action a.

D. Off Policy Training and Online Test

1) Off Policy Training: In this section, we will introduce the training process and online test of the model MATS. We train the proposed model based on generated trajectory data which consists of six elements: $(s_t, a_t, b_t, c_t, m_t, r_t, s_{t+1})$ at time step t. Standard double DQN is used to minimize the squared error between target Q-value: $Y_t^T = r_{t+1} + \gamma Q(s_{t+1}, \arg\max_a Q(s_{t+1}, a; \theta_t); \theta_t^-)$ and estimate Q-value:

$Y_t^E = Q(s_t, \arg\max_a Q(s_t, a; \theta_t); \theta_t)$ based on Bellman Optimally Condition [19]. In our model, we train MATS based on *VDN* policy, the total Q-value function with E.q 4 as follow:

$$Q_{total}(S, \mathcal{A}) = \sum_{n=1}^{N} Q_i(S_i, A_i) = Q_a(s_t, a_t; \theta_a) + Q_b(s_t, b_t; \theta_b) + Q_c(s_t, c_t; \theta_c) + Q_m(s_t, m_t; \theta_m), \ N = \{a, b, c, m\} \tag{7}$$

The total model tries to minimize loss functions Eq. 8:

$$L_{\theta_N} = \mathbb{E}_{s_t, a_t, b_t, c_t, m_t, r_t, s_{t\text{-}1}}[y_t^{target} - Q_{total}(S, A)]^2 \tag{8}$$

where y_t^{target} in E.q 8 is formulated as follow Eq. 9. And Eq. 9 is the total target Q-value for the current interaction t.

$$\begin{aligned} y_t^{target} = \mathbb{E}_{s_{t+1}}[r_t + \gamma(&Q_a(s_{t+1}, \arg\max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \theta_a); \theta_a^-) \\ &+ Q_b(s_{t+1}, \arg\max_{b_{t+1}} Q(s_{t+1}, b_{t+1}; \theta_b); \theta_b^-) \\ &+ Q_c(s_{t+1}, \arg\max_{c_{t+1}} Q(s_{t+1}, c_{t+1}; \theta_c); \theta_c^-) \\ &+ Q_m(s_{t+1}, \arg\max_{m_{t+1}} Q(s_{t+1}, m_{t+1}; \theta_m); \theta_m^-)] \end{aligned} \tag{9}$$

We use a pair of "*twins*" parameter θ (evaluation networks) and θ^- (target networks) to increase smooth the learning and avoid the divergence of parameters [20]. The parameters of the target network θ^- are fixed when optimizing the loss function L_{θ_N}. The derivatives of loss function L_{θ_n} with respective to parameters θ_a, θ_b, θ_c and θ_m:

$$\begin{aligned} \theta_a &= \theta_a + \alpha L_{\theta_N} \cdot \nabla_{\theta_a} Q_{\theta_a}(s_t, a_t; \theta_a) \\ \theta_b &= \theta_b + \alpha L_{\theta_N} \cdot \nabla_{\theta_b} Q_{\theta_b}(s_t, b_t; \theta_b) \\ \theta_c &= \theta_c + \alpha L_{\theta_N} \cdot \nabla_{\theta_c} Q_{\theta_c}(s_t, c_t; \theta_c) \\ \theta_m &= \theta_m + \alpha L_{\theta_N} \cdot \nabla_{\theta_m} Q_{\theta_m}(s_t, m_t; \theta_m) \end{aligned} \tag{10}$$

We detail the entire training process of the model as shown in Algorithm 2. In each interaction on a training recommendation session, there also have two stages. Generating transitions stage (*i.e.*, selector stage and aggregator stage) and model training stage. Specifically, for generating transitions stage, first in selector stage, given the state s_t, the source agent SR selects an item list a_t based on the off-policy π_a (with random probability less ϵ_a select action a_t, otherwise, $a_t = argmax_{a_t \in A_t} \ Q_a(s_t, a_t)$) (line 10); Similarly, we use the same way to select ad and fans source items (line 11); Get fusion candidate items set C3 = fusion $[a_t, b_t, c_t]$ for the second stage of aggregation (line 12). In aggregator stage: given the state s_t, the aggregation agent recommends an item list m_t from the off-policy π_m, which follows the DQN action selected way (with random probability less ϵ_m select action m_t, otherwise, $m_t = argmax_{m_t \in M_t} \ Q_m(s_t, m_t)$) (line 14). Then recommend list m_t for user and observes the reward feedback r_t (lines 16), and next update the state to s_{t+1} (line 16-17); and finally store transitions $(s_t, a_t, b_t, c_t, m_t, r_t, s_{t+1})$ into replay memory $\mathcal{D}$ (line 18). For model training stage: we samples mini-batch of transitions (s, a, b, c, m, r, s') from replay memory $\mathcal{D}$ (line 20), and the updates the parameters

Algorithm 2 Off-policy Training of MATS Framework.

1: Initialize action-value function of selector Q_a, Q_b, Q_c and aggregator Q_m with random weights θ_a, θ_b, θ_c θ_m .
2: $\theta_a^- = \theta_a$, $\theta_b^- = \theta_b$, $\theta_c^- = \theta_c$, $\theta_m^- = \theta_m$
3: Initialize the capacity of replay buffer D
4: **for** session = $1, M$ **do**
5: initialize state s_0 from previous sessions
6: **for** t = $0, T$-1 **do**
7: // Generating transitions stage
8: Observe current state s_t
9: // selector stage
10: Set rec action policy π_a = $\begin{cases} \text{random sample action } \mathrm{a_t}, \ random() < \epsilon_a \\ \mathrm{argmax_{a_t \in A_t} Q_a(s_t, a_t; \theta_a)}, \ otherwise \end{cases}$
11: Set ad and fans action policy π_b and π_c which is the same settings as π_a.
12: Execute action policy π_a, π_b, π_c, obtain fusion candidate items sets $C3_t = [a_t, b_t, c_t]$
13: // aggregator stage
14: Set aggregation policy π_m = $\begin{cases} \text{random sample action } \mathrm{m_t}, \ random() < \epsilon_m \\ \mathrm{argmax_{m_t \in C3_t} Q_b(s_t, m_t; \theta_m)}, \ otherwise \end{cases}$
15: Execute action policy π_m, obtain action m_t, obtain reward r_t from user (online simulator)
16: Obtain next state s_{t+1} = concatenate(s_t, m_t, r_t)
17: Update state s_t = s_{t+1}
18: Store transition $(s_t, a_t, b_t, c_t, m_t, r_t, s_{t+1})$ into the replay buffer $\mathcal{D}$
19: // Model training stage
20: Sample mini-batch of transition (s, a, b, c, m, r, s') from replay buffer $\mathcal{D}$
21: Set $y = r + \gamma(Q_a(s', \arg\max_{a'} Q(s', a'; \theta_a); \theta_a^-) + Q_b(s', \arg\max_{b'} Q(s', b'; \theta_b); \theta_b^-) + Q_c(s', \arg\max_{c'} Q(s', c'; \theta_c); \theta_c^-) + Q_m(s', \arg\max_{m'} Q(s', m'; \theta_m); \theta_m^-))$
22: minimize$(y - (Q_a(s, a; \theta_a) + Q_b(s, b; \theta_b) + Q_c(s, c; \theta_c) + Q_m(s, m; \theta_m)))$.
23: Assign $\theta_a^- = \theta_a$, $\theta_b^- = \theta_b$, $\theta_c^- = \theta_c$, $\theta_m^- = \theta_m$, every C step
24: **end for**
25: **end for**

according to Eq. 8, Eq. 9 and Eq. 10 (lines 21-22). Finally, we update target network parameter after C step (line 23).

Note 1: In order to avoid selecting the same action while ignoring exploration. The strategy for selecting probability value $\epsilon_n (n \in [a, b, c, m])$ as follows:

$$\epsilon_n = 0.01 + 0.99 \times 1/e^{(session_{id}/session_{half})} \tag{11}$$

where $session_{id}$ is training data session id, $session_{half}$ is half of total session number *i.e.*, M/2.

Note 2: The final objective of our model is to maximize the expected return R in the whole aggregation recommend process, the formulated as follows:

$$R = \sum_{m=1}^{M} \sum_{t=1}^{T} \sum_{k=1}^{K} \gamma^{K(t-1)+k} r_{t,k} \tag{12}$$

where M is number of sessions, T is step length of a session, K is the length of the recommended list, and $\gamma \in [0, 1]$.

The intuition of Eq. 12 is that the reward of the top step in a session and top item recommended list in the current step has a higher contribution to the overall rewards, which forces: 1) SA, SR and SF selected items that user most interesting item from source items pool; 2) AA rank items that user may order in the top of the recommended list.

2) Online test: We also do online test in test data. The overall online recommendation service is based on weight θ_a^-, θ_b^-, θ_c^- and θ_m^- trained by Algorithm 2.

IV. Experiments

A. Experimental settings

Adaptation to the real-world problem. As we mentioned above, our model is proposed to solve the real-world shortcoming of two-stage strategy widely adopted in the industry. Since there are no public heterogeneous source datasets, we conduct extensive experiments on a real-world data, which is collected from a famous short video platform **Kuaishou.com**. We collect 66698 sessions in temporal order, following [21], sessions are divided in the principle of whenever there exists a time gap of more than 3 days. The statistics of the datasets are shown in Table I. The meaning of number please reference Fig. 1 (e.g., the 3 in "SR SA and SF selector" is number of three source agents selected candidate items)

Baselines. To evaluate the performance of our proposed model, we use four representative recommender system methods as baselines. For all baselines, the experimental data format needs to be tuned. Specifically, we put three heterogeneous sources data into together as a candidate items pool. We compare our model MATS with the following methods: **BPR** [22]: This baseline is a well-known general recommender method, which is to model user implicit feedback. We use the number of negative sample is 3 in our experiments. **NCF** [23]: This baseline is also a well-known general recommender method, which model the user-item interactions with a multi-layer feedforward neural network, and the number of negative sample is 3 in our experiments. **NARM** [24]: This baseline is an attention mechanism-based RNN approach that aims to learn a user's sequential behavior in the current session. **GRU4rec** [25]: This baseline is a sequential recommender model, which is to predict the user will click next based on the clicking histories with a Gated Recurrent Units (GRU).

Implementation details. The proposed MATS model has four agents, all agents are designed as a multi-layer neural network and more detailed model information of agents. We use Adam for learning parameters with a learning rate fine-tune range in $[0.0001, 0.001, 0.01, 0.1]$. More model parameter settings are shown in Table II. **Metrics**. To measure model MATS online performance, we leverage the *Average Dwell Time* (ADT: the mean value of a user's dwell time in minutes on a recommendation list) in all sessions: $R = \frac{1}{M'} \sum_1^{M'} \sum_1^{T} r_t$ as the accumulate reward metric. We adopted the leave-one-out evaluation, which has been widely used in literature [22], [26]. For each user, we held-out her/his latest session as the testing set and utilize the remaining data for training and validating.

TABLE I
STATISTICS OF THE *kuaishou.com* VIDEO DATASETS.

sessions	**user**	**heterogeneous sources**
66698	8801	3
rec video	**dwell time range**	**rec, ad, fans video pool**
367744	[0, 1]	6
ad video	**session length range**	**SR, SA, SF selector**
65035	[1, 100]	3
fans video	AA **aggregator**	**ground-truth rewards**
12504	3	3

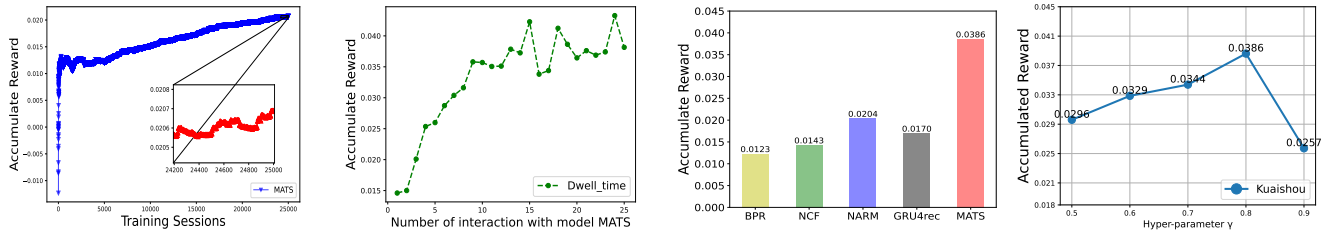
TABLE II
THE NETWORK ARCHITECTURE AND PARAMETER SETTINGS OF MATS.

Rec Source Agent			Ad Source Agent		
Input	Output	Act.	Input	Output	Act.
$[state, action]$	32	ReLU.	$[state, action]$	32	ReLU.
32	16	ReLU.	32	16	ReLU.
16	1		16	1	
Fans Source Agent			**Aggregation Agent**		
Input	Output	Act.	Input	Output	Act.
$[state, action]$	32	ReLU.	$[state, action]$	32	ReLU.
32	16	ReLU	32	16	ReLU.
16	1		16	1	

Hyper-parameter : Setting	
Learning rate α: 1e-3	Optimizer: Adam
Memory D size: 3e4	Mini-batch size: 32
Target θ^- update period C: 1e2	Hidden layer size: (32,16)
User&item embedding size: 32	GRU hidden layer size: 96
Discount factor γ: 0.8	Weight decay:1e-6
Memory_capacity to update agent: 4e4	

B. Overall Comparison

Fig. 3 (a) illustrates the training process of MATS. It can be observed that model converges with around 25000 training sessions as shown in red line. Fig. 3 (b) shows that our model test performance metrics ADT first gradually increases and then tends to slight change within a range $[0.35, 0.45]$. We calculate the average of the last five test results as the final predict result in our experiment. For baselines, we also use the trained simulator to calculate the cumulative reward according to the recommended results of baselines, and both in a unified environment. The overall performances between our model and baselines are shown in Fig. 3 (c), from which we can see: overall performance comparison, MATS $>$ NARM $>$ GRU4rec $>$ NCF $>$ BPR, we can conclude that our proposed model outperforms all other baselines. Session-based methods (NARM, GRU4rec) achieves better performance than general-based methods (NCF, BPR), which further confirm the validity of the model RNN or GRU to capture the user's sequential behavior. The NARM achieves better performance than



(a) (b) (c) (d)

Fig. 3. (a) : Model performance changes, and red line shows model converges after training about 25000 sessions. (b) : Accumulative rewards among 909 users interaction 25 times with the model MATS. (c) : Overall performance comparison in online test.(d) :Accumulated reward changes with different parameters γ.

GRU4rec, which demonstrates that attention-based models may be better at dealing with heterogeneous sources session information. NCF perform better than BPR, which utilizes a neural network (NN) to model user-item interactions, but BPR does not. It shows that NN-based methods are more effective than the tranditional-based method for heterogeneous sources.

C. Hyper-parameter Analysis

In this section, we analyze the influence of the key hyper-parameters. Our method has a key parameters γ that is used to balance the short-term and long-term rewards. Generally speaking, select the smaller γ, MATS focus on the users' immediate preference, while select larger γ, MATS focus on the future engagement. In this section, we investigate how the proposed framework MATS performs with the changes of γ by tuning it in the range of $[0.5, 0.6, 0.7, 0.8, 0.9]$. The results are presented in Fig. 3(d). We find that the optimal γ is 0.8.

D. Ablation Studies

In this section, we conduct ablation studies to verify the effectiveness of each agents. We set the following 7 variants: MATS-α: This variant is to evaluate the effectiveness of aggregator, which we only train the aggregator and the heterogeneous source consists of rec, ad and fans videos. MATS-β_1: This variant is to illustrate the useful for SR agent, we only train the SR and heterogeneous source is rec video. Similarly, variant MATS-β_2 and MATS-β_3 are to illustrate the useful for the SA agent and SF agent. MATS-β_4: This variant is to illustrate the effectiveness of SR and SA. When we train SR, SA and AA, the candidate heterogeneous sources are rec and ad videos. Similarly, MATS-β_5 and MATS-β_6 are to illustrate the useful for the pairwise combination of other agents. To describe the results, we following [20], which can demonstrate the improvement of MATS over each variants. The result is shown in Table III, from which we can see: 1) MATS-α validates the effectiveness of introducing SR, SA and SF to select items that meets the user's interests from their source type videos pool in the first stage. 2) MATS-β_1,$\cdots$,β_6 validates the effectiveness of aggregator to decide the recommendation order in the second stage. 3)All component experimental results show the effectiveness of our proposed model MATS to solve two-stage strategy.

TABLE III
ABLATION STUDY RESULTS FOR MATS MODEL.

method	reward	improvement	p-value
MATS-α	0.03003	28.61%	0.01065
MATS-β_1	0.02296	68.19%	0.00063
MATS-β_2	0.02663	45.02%	0.00414
MATS-β_3	0.02514	53.63%	0.00008
MATS-β_4	0.02905	32.96%	0.00381
MATS-β_5	0.02836	36.19%	0.00722
MATS-β_6	0.02741	40.90%	0.00072
MATS	0.03862	*	*

V. Related Work

A. Recommender systems based on RL

In recent years, Recsys based on RL has attracted more and more attention in academia and industry [17], [27], [28], [18], [29], [30], [31]. To the best of our knowledge, the existing RL Recsys methods mainly focus on homogeneous source with single agent. For example, [17] utilizes actor-critic framework to build a list-wise recommendation framework with a single DQN agent. [28] propose a novel method to incorporate positive and negative feedback into DQN with a single agent. [18] employs generative adversarial network to build a model-based RL framework for Recsys. [20] proposes a novel Deep Q-network architecture for online advertising that can continuously update its advertising strategies. Existing RL for Recsys pay attention to how to design effective agents for homogeneous source, while ignore heterogeneous sources. In this work, we design a MARL framework, which model user long-term feedback on three heterogeneous source.

B. Heterogeneous Sources Aggregation

Result aggregation has a wide range of tasks, such as web search [32], [33] and E-commerce search [34]. Specially, [33] address the issue of integrating search results from a news vertical into web search results. [32] focus on vertical selection, predicting relevant verticals for queries issued to the search engine's main web search page. [34] exploits hierarchical reinforcement learning for aggregating search results from heterogeneous sources in an E-commerce environment. In this paper, we study the two-stage in heterogeneous task.

VI. Conclusion and Future Work

In this paper, we propose a novel framework MATS, which leverages multi-agent reinforcement learning to solve two-stage recommendation with heterogeneous source videos. First, we build three source agents to automatically learn the optimal heterogeneous videos select policy. And then we build an aggregation agent to automatically learn the optimal aggregation recommendation policy. Last, source agent and aggregation agent cooperate based on decomposable reward to achieve overall optimization. To demonstrate our model's effectiveness, extensive experiments are conducted on a real-world *Kuaishou.com* datasets. In future studies, we plan to include some other micro information, in particular, real textual and visual information.

Acknowledgment

This work is supported in part by National Natural Science Foundation of China (No. 62102420 and No. 61832017), Beijing Outstanding Young Scientist Program NO. BJJWZYJH012019100020098, Intelligent Social Governance Platform, Major Innovation & Planning Interdisciplinary Platform for the "Double-First Class" Initiative, Renmin University of China, and Public Computing Cloud, Renmin University of China.

References

[1] G. Linden, B. Smith, and J. York, "Amazon. com recommendations: Item-to-item collaborative filtering," *IEEE Internet computing*, vol. 7, no. 1, pp. 76–80, 2003.

[2] J. Wang, P. Huang, H. Zhao, Z. Zhang, B. Zhao, and D. L. Lee, "Billion-scale commodity embedding for e-commerce recommendation in alibaba," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 839–848.

[3] M. O'Dair and A. Fry, "Beyond the black box in music streaming: the impact of recommendation systems upon artists," *Popular Communication*, vol. 18, no. 1, pp. 65–77, 2020.

[4] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," *arXiv preprint arXiv:1301.7363*, 2013.

[5] R. J. Mooney and L. Roy, "Content-based book recommending using learning for text categorization," in *Proceedings of the fifth ACM conference on Digital libraries*, 2000, pp. 195–204.

[6] W. Carrer-Neto, M. L. Hernández-Alcaraz, R. Valencia-García, and F. García-Sánchez, "Social knowledge-based recommender system. application to the movies domain," *Expert Systems with applications*, vol. 39, no. 12, pp. 10 990–11 000, 2012.

[7] J. Hron, K. Krauth, M. Jordan, and N. Kilbertus, "On component interactions in two-stage recommender systems," *Advances in Neural Information Processing Systems*, vol. 34, pp. 2744–2757, 2021.

[8] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[9] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[10] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, no. 1, 2016.

[11] Y. Shoham, R. Powers, and T. Grenager, "Multi-agent reinforcement learning: a critical survey," Technical report, Stanford University, Tech. Rep., 2003.

[12] L. Panait and S. Luke, "Cooperative multi-agent learning: The state of the art," *Autonomous agents and multi-agent systems*, vol. 11, no. 3, pp. 387–434, 2005.

[13] L. Busoniu, R. Babuska, and B. De Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 2, pp. 156–172, 2008.

[14] L. Buşoniu, R. Babuška, and B. D. Schutter, "Multi-agent reinforcement learning: An overview," *Innovations in multi-agent systems and applications-1*, pp. 183–221, 2010.

[15] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls *et al.*, "Value-decomposition networks for cooperative multi-agent learning," *arXiv preprint arXiv:1706.05296*, 2017.

[16] S. Thrun and M. L. Littman, "Reinforcement learning: an introduction," *AI Magazine*, vol. 21, no. 1, pp. 103–103, 2000.

[17] X. Zhao, L. Zhang, L. Xia, Z. Ding, D. Yin, and J. Tang, "Deep reinforcement learning for list-wise recommendations," *arXiv preprint arXiv:1801.00209*, 2017.

[18] X. Chen, S. Li, H. Li, S. Jiang, Y. Qi, and L. Song, "Generative adversarial user model for reinforcement learning based recommendation system," in *International Conference on Machine Learning*. PMLR, 2019, pp. 1052–1061.

[19] R. Bellman, "Dynamic programming," *Science*, vol. 153, no. 3731, pp. 34–37, 1966.

[20] X. Zhao, C. Gu, H. Zhang, X. Yang, X. Liu, H. Liu, and J. Tang, "Dear: Deep reinforcement learning for online advertising impression in recommender systems," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 1, 2021, pp. 750–758.

[21] Y. Feng, F. Lv, W. Shen, M. Wang, F. Sun, Y. Zhu, and K. Yang, "Deep session interest network for click-through rate prediction," *arXiv preprint arXiv:1905.06482*, 2019.

[22] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," *arXiv preprint arXiv:1205.2618*, 2012.

[23] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proceedings of the 26th international conference on world wide web*, 2017, pp. 173–182.

[24] J. Li, P. Ren, Z. Chen, Z. Ren, T. Lian, and J. Ma, "Neural attentive session-based recommendation," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 2017, pp. 1419–1428.

[25] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, "Session-based recommendations with recurrent neural networks," *arXiv preprint arXiv:1511.06939*, 2015.

[26] H.-J. Xue, X. Dai, J. Zhang, S. Huang, and J. Chen, "Deep matrix factorization models for recommender systems." in *IJCAI*, vol. 17. Melbourne, Australia, 2017, pp. 3203–3209.

[27] G. Zheng, F. Zhang, Z. Zheng, Y. Xiang, N. J. Yuan, X. Xie, and Z. Li, "Drn: A deep reinforcement learning framework for news recommendation," in *Proceedings of the 2018 World Wide Web Conference*, 2018, pp. 167–176.

[28] X. Zhao, L. Zhang, Z. Ding, L. Xia, J. Tang, and D. Yin, "Recommendations with negative feedback via pairwise deep reinforcement learning," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 1040–1048.

[29] X. Zhao, L. Xia, L. Zhang, Z. Ding, D. Yin, and J. Tang, "Deep reinforcement learning for page-wise recommendations," in *Proceedings of the 12th ACM Conference on Recommender Systems*, 2018, pp. 95–103.

[30] X. Bai, J. Guan, and H. Wang, "A model-based reinforcement learning with adversarial training for online recommendation," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[31] X. Chen, Y. Du, L. Xia, and J. Wang, "Reinforcement recommendation with user multi-aspect preference," in *Proceedings of the Web Conference 2021*, 2021, pp. 425–435.

[32] J. Arguello, F. Diaz, J. Callan, and J.-F. Crespo, "Sources of evidence for vertical selection," in *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, 2009, pp. 315–322.

[33] F. Diaz, "Integration of news content into web results," in *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, 2009, pp. 182–191.

[34] R. Takanobu, T. Zhuang, M. Huang, J. Feng, H. Tang, and B. Zheng, "Aggregating e-commerce search results from heterogeneous sources via hierarchical reinforcement learning," in *The World Wide Web Conference*, 2019, pp. 1771–1781.