Aggregation Recommendation from Heterogeneous Videos via Multi-Agent Reinforcement Learning

Yabin Zhang[2], Weiqi Shao[2], Xu Chen[1,2,*], Changhua Pei[3], Zhengpeng Liu[3], Shuai Zhang[3], Dong Zheng[3], Peng Jiang[3], Kun Gai[3]
[1]Beijing Key Laboratory of Big Data Management and Analysis Methods
[2]Gaoling School of Artificial Intelligence, Renmin University of China, Beijing 100872, China
[3]Kuaishou.Inc, Beijing, China
{yabin.zhang,shaoweiqi}@ruc.edu.cn, successcx@gmail.com
{zhongxuan,liuzhengpeng,zhangshuai,zhengdong,jiangpeng,gaikun}@Kuaishou.com

***Abstract*—Recommender systems via reinforcement learning (RL) to display a mixed list from heterogeneous sources for users has become a hot topic both in academic and industrial circles. While many promising results have been achieved, most existing models focus on optimising the ranking of mixed lists, ignoring the interaction between the ranking of mixed lists and the selection of heterogeneous sources items. A major challenge is that the selection of items from heterogeneous sources and the presentation of mixed lists to users are independent of each other, and there is no measure to ensure that both the selected items and the recommended mixed list are globally optimal. To address this challenge, we consider how to model heterogeneous videos sequential aggregation recommendation in the context of RL based recommender system. Specifically, we first decompose the total task into two subtasks: a sources videos item list select task where we model multi sources agents to select item list from their own source items, and an aggregation recommend task where we model an aggregation agent to decide the presentation order of selected fusion items in a recommendation list. And then, we assign each agent with a novel double DQN network framework to select or aggregation items list. Last, we bulid our model on a Multi-Agent RL framework (Value-Decomposition-Networks) achieve the optimal performance through cooperative optimization. We conduct experiments in a real-world *Kuaishou* APP datasets to demonstrate our model's effectiveness.**

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

I. Introduction

Recommender system (RS) has achieved great sucess in intelligent E-commerce application [1], [2]. The main purpose of RS is suggesting item that best fit users' needs and preferences. In recent years, RS has been utilized in a wide range of task domains, such as social networks [3], [4], music [5], book [6], new [7], etc. However, there are still great challenges in dynamic interactive real-world recommender scene. In regard to traditional RS models, it is commonly assumed that the recommendation task is in a supervised learning framworks, we need to model based on a fixed greedy strategy. Sequential recommendation task is more vivid expression of human real interactions behavior. Figure 1 illustrates a typical example of the interactions between recommender system and user in a sequential recomendation scene. The recommender system continues to interact with the user until the user leaves. In each interaction step, system recommends an item to the user, and user response the action with a reward (whether I need?). To effectively utilize the large amount of interaction data generated, reinforment learning based on Markov Decision Process (MDP) [8] has become an effictive and popular method. Typically, the user plays the role of environment in RL and the recommender system plays the role of agent in RL, the interaction mechanism between user and agent [9]–[12] as shown in Figure 2 , recommender agent takes an action (e.g., a video) to users, and the user gives a feedback (e.g., dwell time 30s) for this action. Then, recommender agent takes the next action according to the user's feedback. The final objective of this way is to maximize the expected return in the whole interaction process.

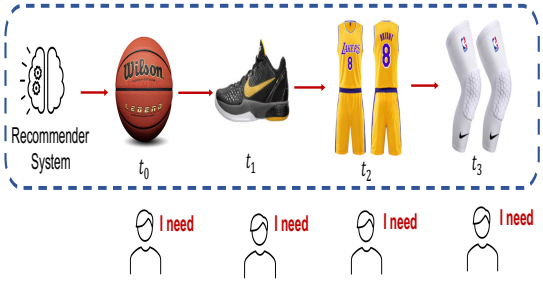

Fig. 1. An example to illustrate the sequential interactions between recommender systems and users.

Reinforcement learning has many successful applications in sequential recommendation in homogeneous source [7], [13]–[15]. For the homogeneous sources, there is only a type item (e.g., ad video), while the heterogenonous sources come form mutli-types item which may include ad, live and rec video, etc. To the best of our knowledge, there is little work to consider heterogenonous sources item for modeling recommendations. The reason we consider recommendations from heterogeneous sources is to find the best information to match the user from different heterogeneous sources. We also named recommendation from heterogenonous sources as

*Corresponding author

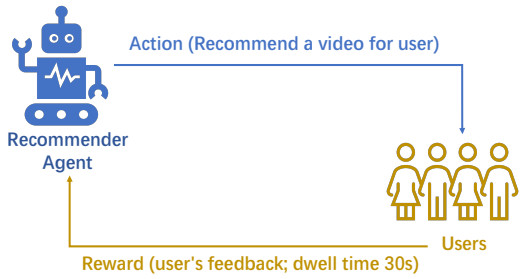

Fig. 2. The interaction mechanism between users and recommender agent based RL.

result aggregation recommendation. We named it as **reslut aggregation recommendation** is because of *reslut aggregation*. For *reslut aggregation*, existing research on reslut aggregation has focused on web search [16]–[18] and E-commerce search [19], while in this paper, we study the reslut aggregation with short video recommendation. We give an example, as shown in Figure 4, to illustrate the video result aggregation recommendation. We assume that our user is a car enthusiast, and videos about car has four heterogeneous sources type, i.e., race , movie , game and ad. The value, in the aggregation list 1, 2 and 3, represents the user's satisfaction level with the video at the current moment $T_i, i \in [0, 1, 2, 3]$. Aggregation recommendation list 3 is most optimized recommendation result compared to recommendation list 1 and 2. For recommendation list 1, there has three Race videos and one Game video, the total feedback is 2.7. For recommendation list 2, both in time T_1 and T_2 the item is Ad which is increase user's negative experience, so the user is leave in time T_3. For recommendation list 3, which is big different with list 1 and 2, has a recommendation movie video named *Fast&Furious 6* in time T_1. This is probably because movie video increases user's satisfaction for next time: ad video reward (0.9) in time T_3 which is 0.8 in time T_1 of list 2 and game video reward (0.7) in time T_4 which is 0.6 in time T_3 of list 1.

So, how to aggregate the optimal list from heterogeneous sources items for users. There are mainly two big challenges to be solved: **1) Recall:** how to select the videos of interest to users from the heterogeneous source data (e.g., Heterogeneous sources Race, Movie, Game and Ad in Fig. 3); **2)Ranking:** how to rank the selected videos and aggregate a list that the most satisfactory for users. (e.g., best aggregation recommendation list 3 in Fig. 3)

In this paper, we design a novel model **A**ggregation **R**ecommendation from Heterogeneous **V**ideo via **M**ulti-Agent Reinforcement Learning (**ARVM**), which adpots the multi-agent reinforcement learning cooperative optimization strategy to handle the earlier described two challenges. In general, we decompose result aggregation in short video recommendation task into two subtasks: source selection and aggregation recommendation. Source selection can also be regarded as **Recall** in the industry that decides which items should be recalled/selected in the heterogeneous sources candidate items. For example, if we want to recomenmend a basketball video for basketball lovers, and we have three sources video about basketball i.e., normal basketbal video, NBA Live video and Nike advertising video. Source selection decides the selected normal items from normal basketbal video, Live items from NBA Live video and advertising item from Nike advertising video respectively. Aggregation recommendation can also be regarded as **ranking** in the industry, which decides the presentation order of the items from heterogeneous sources. Also take the basketball video recommendation above as an example, we obtain three sources candidate items (normal video, NBA video and Nike advertising video) after source selection stage. Aggregation recommendation need to decide the display position of each item with a list length of 3 from three sources candidate items. Meanwhile, recall and ranking stages are optimized by the two groups without considering joint modeling in the industry. It not only wastes model resources and maintenance costs, but also does not consider downstream models and data feedback. Therefore, the establishment of model **ARVM** will have some enlightenment for academic research and solving practical business problems.

Both source selection and aggregation recommendation can be viewed as a sequential decision problem and handled by reinforcement learning. So given the above two subtasks, we utilize multi-agent reinforcement learning to model. For source selection, we set N agents corresponding to N heterogeneous sources respectively. For aggregation recommendation, we only set an agent to aggregation recommendation list. Let's use baskeball aboved as an example again, we have four agents in total consist of three source agents (normal, Live, ad) and a aggregation agent. In our sequential multi-agent RL model, reward is sole indicator to evaluate model whether work well, obtaining a positive reward means that both source agents and aggregation agent are doing well, while getting a negative reward means that at least one of them is not doing well. Reward can better guide source agents to select and aggregation agent to rank. Finally, source agents and aggregation agent achieve the best reward by cooperating with each other.

The main contributions of this paper can be summarized as follows:

- We propose to model the aggregated recommendation from heterogeneous sources via Multi-Agent RL based on value-decomposition networks .
- We extand traditional doublel DQN framework with GRU, which captures more information about user state changes while as a agent is able to dynamically select or ranking the list of optimization items.
- Extensive experiments are conducted based on a real-world short video dataset of *Kuaishou* APP to demostrate our model's effectiveness.

The rest of this paper is organized as follows. Section II briefly discusses background on Recommender system in industry communities, Recommendation as a RL Model and Multi-Agent RL. Section III introduces the technical details of the

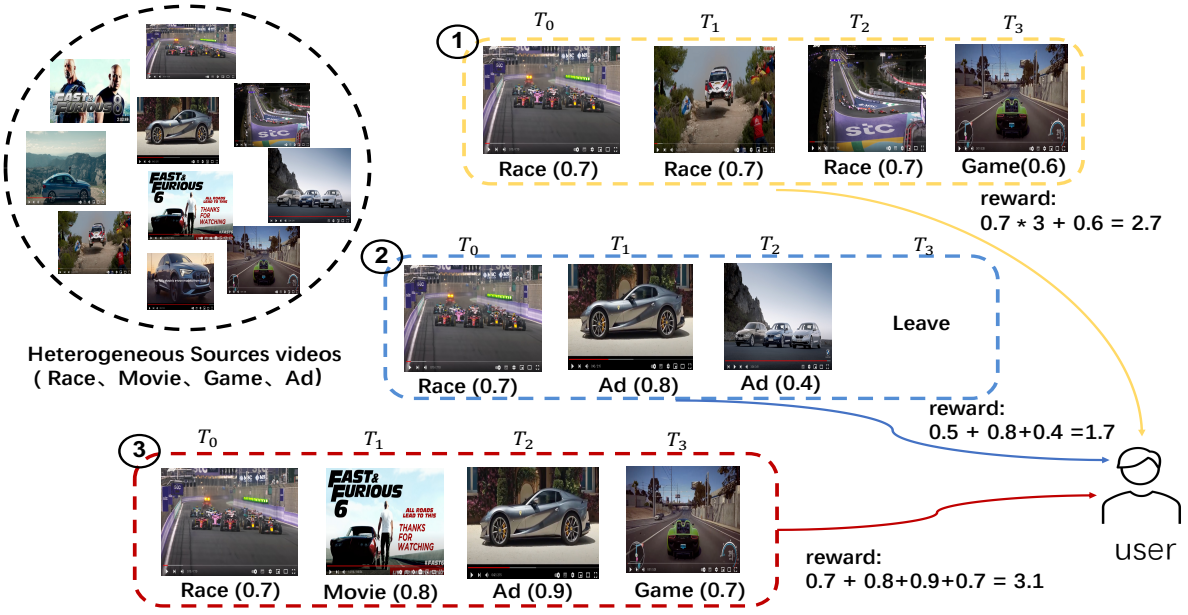

Fig. 3. An result aggregated example of the video recommendation, where the heterogeneous video consist of four type, i.e., Race, Movie, Game and Ad (advertisement). Recommendation list 3 is the most satisfying aggregation result for user.

proposed ARVM method. In Section IV the related work is discussed. Section V provides a detailed report on the experimental comparative results. Finally, Section VI offers a conclusion.

II. BACKGROUND

In this section, we briefly introduce the necessary backgrounds of this work, which is a big support for our proposed model.

A. Recommender System in Industry

Recommendation system in the industry is generally divided into three parts: online, nearline and offline. In this paper, the motivation of our model is mainly based on the online part. The online part generally goes through five stages, 1) recall stage: reduce the items recommended to users to less than 1000; 2) pre-rank stage: optional stage, if there are still too many items returned in the recall stage, the pre-rank stage can be added, which further reduces the items delivered to the subsequent links through some simple ranking models; 3) Rank stage: complex models can be used here to accurately rank a small number of items; 4) Business strategy stage: for a user, even if the result of fine-tuning recommendation comes out, it will not be directly displayed to the user. Some business strategies may be needed, such as going to read, diversifying recommendation, adding advertising, etc; 5) Display to users: form the final recommendation results and display the results to users. As shown in Fig. 4, a simplified version of the recall and ranking flow chart in recommender system.

In general, recall and ranking may be optimized by the two groups without considering joint modeling. They simply use the user behavior feedback data for their own model training. There will be the following two problems, one things is that gap exists in the data magnitude distribution between offline user behavior feedback data and online actual estimated data, which affects the effect of the model. Another thing is that there is not considered to joint modeling, which not only wastes model resources and maintenance costs, but also does not consider downstream models and data feedback. In our paper, we address this gap with two modules, source select and aggregation recommendation.

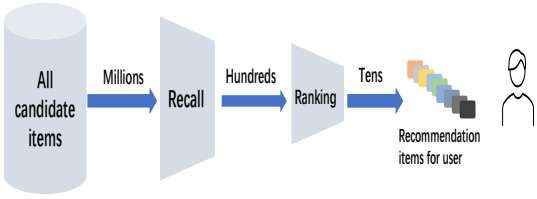

Fig. 4. Recall and ranking process of recommender system in industry.

B. Recommendation as a RL Model

Reinforcement learning [20] is different from supervised learning and unsupervised learning, which 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 based on a policy π 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. RL-based recommender models has similar operating mechanism, typically, recommended item as a action a_t, user as a environment and user's previously browing history as state s_t, user's feedback (e.g., rating, click, dwell time) as a reward r_t. For more operate detail, the recommender agent takes an action based on the user's current state, and the user response a reward for this

action. The state is transformed into next state s_{t+1} based on transition $\mathcal{P}$. After many interactions between users and agent, the recommender agent is able to learn a policy π that can maximize the acumulated 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 fator. With the notations and definitions above, the recommendation as a RL model can be formally defined as Markov Decision Process (MDP) [8], which consists of five elements, i.e., $< \mathcal{S}$,$\mathcal{A}$,$\mathcal{P}$,$\mathcal{R}$,$\gamma >$.

Generally speaking, traditional RL methods can be classified into three branches, including policy-based [21],valued-based [22] and actor-critic [23]. All three types methods have been used in the recommender system. Next, we will give a brief introduction to DQN [22] and DDQN [24] models, which are both valued-based method, and would be used to our proposed model.

1) DQN and DDQN: Deep Q-learning (DQN) is a value-based method, which can be applied to solve the discrete action space problem. DQN only maintains an action-value function $Q(s_t, a_t)$, 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, so that then $\theta_t^- = \theta_t$, and parameters θ^- kept fixed on all other step [24]. The target network used by DQN is $Y_t^{DQN} = R_{t+1} + \gamma \max_a Q(S_{t+1}, a; \theta_t^-)$. Expand this formula, we can get $Y_t^{DQN} = R_{t+1} + \gamma Q(S_{t+1}, \arg\max_a Q(S_{t+1,a}; \theta_t^-); \theta_t^-)$, The max problem of the latter formula is adpots the same parameter to select and evaluate action. This makes it more likely to select overestimated values, resulting in over optimistic value estimates [24].

To change this, Double DQN (DDQN) decouple the selection from the evaluation. Specifically, for each update, evaluate network weights is used to determine the greedy policy select action and the target network to determine its value [24]. For a more clear comparison, we can rewrite the latter foumula as $Y_t^{DDQN} = R_{t+1} + \gamma Q(S_{t+1}, \arg\max_a Q(S_{t+1,a}; \theta_t); \theta_t^-)$. According to this foumula, the main different is that the selection of the action, in the argmax, is still due to the evaluate weights θ_t [24]. In this paper, we employ four double DQN as three source agents and an aggregation agent. We select the action according to the evaluation network outputs, and calculate the Q-value according to the target network outputs.

C. Multi-Agent RL

In multi-agent reinforcement learning (MARL) [25]–[28], 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.

MARL can be generally divided into three group, including fully cooperative, fully competitive, or with mixed strategies. Fully cooperative agents have a common goal and each agent try their best to maximize the same expected return [29]. Fully competitive agents have private goals that each agent will opposite to each other for thier won interest. Mixed strategies fall between the two extremes. Next, we will give a brief introduction fully cooperative mutli agents method VDN [30] , which are used to our proposed model.

1) VDN: The difficulties of MARL 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. These two problems are also reflected in our multi-agent recommendation system. VDN [30] trains a global $Q(s, u)$ through value decomposition, the formula is clearly explained as follows:

$$Q_{total}(s, u) = \sum_{i=1}^{N} Q_i(o_i, u_i) \tag{1}$$

Where N is the number of agents in environment. After getting approximate $Q_{total}(s, u)$, VDN updates $Q_{total}(s, u)$ through global reward r with DQN, and its loss function is expressed as

$$L(\theta) = \frac{1}{M} \sum_{j=1}^{M} (y_j - Q_{total}(s, u))^2 \tag{2}$$

Where M is batch size, $y_j = r_j + \gamma \arg\max_u Q_{total}^-(s^{'}, u)$, $Q_{total}^- = \sum_{i=1}^{N} Q_i^-(o_i, u_i)$, Q_{total}^- is target network in 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 [31], and the gradient will be back propagation to each Q_i through Q_{total} to updata them. In this way, we can update Q_i from a global perspective. In our model, we use the idea of VDV to train our source agents and aggregation agent as a whole model.

III. The Proposed Framework

We develop a deep multi-agent reinforcement learning algorithm for user aggregate recommenda item in this paper. In the following subsections, we first given the problem statement of recommender system via mutli-agent reinforcement learning. Then we bulid an online user-agent interaction environment simulator. Next, we propose ARVM model which has two different stage (named source selection and aggregation recommendation). Finally, we discuss how to online train our model by obtaining data from the simulator and user's offline log, and then utilize the proposed model to online test.

A. Problem Statement

In this paper, we formulate the result aggregation recommendation from heterogeneous videos task as a semi-cooperative, partially observable, multi-agent sequential decision problem. More specifically:

Multi-agent:: in our task, we model four agents for three different heterogeneous sources and an aggregation agent for aggregation recommendation item list in a system. Each agent

is a select or ranking strategy and to learns ite own value function.

Sequential-Decision:: we need multiple agents interacts with environment (users) by sequentially choosing recommendation item over a sequence of time step. At each time step, the state is represented by a user's previously interacted products, then the agent takes an action to respond to this state. The current each source agent action will affect aggregation agent actions in the future.

Semi-Cooperative:: in our model, four agents are semi-cooperative to maximize accumulate reward. Moreover, in source selection stage, three source agents not pass messages to each other for communication. In aggregation recommendation stage, each source agents with aggregation agent pass messages to each other for communication, but each source agents is still no communication, and the overall performance of these four agents are evaluated by a centralized total Q-value.

Partially Observable: for mutli-agent model, its environment is partially observable, and each agent only receives a local information instead of observing the full information of the environment.

Notations: the notations used in this paper are summarized in Table. I.

TABLE I
NATATIONS OF AGGREGATED RECOMMENDATION ITEM

	Notation	Description
Dataset	M	User's browsing history session
	T	Each session step length
	L	Aggregation recommendation list length
	srv, sav, sfv	heterogeneous sources rec/ad/fans video
Model	O	Online simulator
	$SR, SA, SF \vert AA$	Source Agent|Aggregation Agent
	s, S	State, State set
	a, A	Action, Action set
	r, R	Reward, Return
	γ	Discount fator
	Q_i	Action-value function
	$\pi_a, \pi_b, \pi_c, \pi_m$	Policy symbol
	$\theta_a, \theta_b, \theta_c, \theta_m$	Network weight

B. Build Online Environment Simulator

Since the real feedback (dwell time) of user-item pairs is sparse in real *Kuaishou* APP datasets, in this section, we need to build an online environment simulator. According to this simulator, we can predict a reward based on current state and a selected action i.e., $f : (s_t, a_t) \rightarrow r_t$. We follow previous work [9], [32] to build simulators 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, optimization algorithm 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' logs, which is not same data for trainning proposed model.

Algorithm 1 Build Online Environment Simulator.

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

for seesion = $1, H$ **do**
  observe initial state s_0
  for length = $1, T$ **do**
    observe current state s
    Intialize 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 $S(s, a)$
      $r' += 0$
    end for
    update state s
    observe current list ground truth reward r
    minimize$(r - r')$
  end for
end for

C. Total model

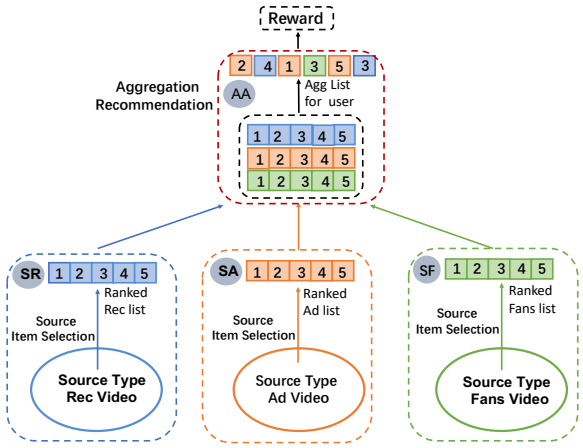

Fig. 5. The Multi-agent RL on aggregation recommendation. This example of recomender agent for user in a session. Each session is composed of three source types video (rec, ad, fans).

The total model consists of three heterogeneous source select agent (selector) and an aggregation recommendation agent (aggregator). The selector decides which items in current source to recall/select for the next stage aggregation. The aggregator then decides the recommendation order of the relevant items from selected three heterogeneous sources item. Both selector and aggregator are trained with the Double DQN (Fig.6). The entire process works as shown in Fig.5. We have three heterogeneous source is Rec (normal video source), Ad (advertisement video source) and Fans ($fans\_top$

is a kind of popularize video source). In general, the total model has three stage: 1) SR selects five (Changeable) rec videos $[1, 2, 3, 4, 5]$ from source type rec video as aggregator rec candidate items; SA selects five(Changeable) ad videos $[1, 2, 3, 4, 5]$ from source type ad video as aggregator ad candidate items. SF selects five(Changeable) fans videos $[1, 2, 3, 4, 5]$ from source type fans video as aggregator fans candidate items; 2) rec, ad and fans candidate item are fused to a new candidate item to be aggregated. 3) AA aggregation a recommendation list $[2, 4, 1, 3, 5, 3]$ for user from the new fuse candidate item $[[1, 2, 3, 4, 5], [1, 2, 3, 4, 5], [1, 2, 3, 4, 5]]$.

A general RL-based recommender models has a sequence of experiences $s_1, a_1, r_1, \cdot, , s_t, a_t, r_t$ where $s/a/r$ correspond to state/action/reward respectively. We model with four agents SR, SA, SF, AA, each agent corrsepoding to a particular optimization source item. As we statement aboved, the environment in our problem is partially observable. In our mutli-agent model setting, not all informations are global observable. Specifically, the state(s_t) of the environment and reward r_t is global, shared by all four agent, while the actions a_t, b_t, c_t, m_t are all private, only possessed by each agent itself. Formulates these particularities on source select and Aggregation recommendation.

1) Source select: At the time t, the state s_t is defined as the concatenation of triplet from time $t-1$: state s_{t-1}, action a_{t-1} and reward r_{t-1}, i.e., $s_t = concat[s_{t-1}, a_{t-1}, r_{t-1}]$. For three heterogeneous sources, we have three source agents which means we have three private policies π_a, π_b, π_c perceives state s and selects an action a_t, b_t, c_t with random weight $\theta_a, \theta_b, \theta_c$. The action a_t, b_t, c_t will trigger the aggregation recommendation to deicde how to present the items from these selected sources candidate items. The source 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 word, in source selection stages, each source agent is trying to select the optimal ranking item list from their source type videos.

Formulates these three source agents Q-value at time t as follow:

$$\begin{aligned} &Q_{\pi_a}(s_t, a_t; \theta_a) : \text{for rec video} \\ &Q_{\pi_b}(s_t, b_t; \theta_b) : \text{for ad video} \\ &Q_{\pi_c}(s_t, c_t; \theta_c) : \text{for fans video} \end{aligned}$$

2) Aggregation recommendation: The aggregation recommendation process aims to decide the recommendation order of the items from the three candidate source items chosen by the source selector. Aggregator shared the same state s_t at the time t with selector, but the policies π_m is private. Aggregator selects an action m_t with radom weight θ_m according to s_t and π_m. The action m_t means ranking a aggregation recommendation list for user. Therefore, the total model will receive the feedback reward r_t from user (trained online simulator in subsection III.B) at the time t. Formulates aggregator's agents Q-value at time t as follow:

$$Q_{\pi_m}(s_t, m_t; \theta_m) : \text{for selected items}$$

D. A novel double DQN Framework

As mentioned above, in our method, both three source agent and an aggregation agent model is based on Double DQN. We propose a novel double DQN framework to select or recommend item. As shown in Fig. 6, It uses a multi-layer neural networks with parameters θ to evaluate the Q function and use a "twins" θ^- based on θ to selecet. As we discussed, the state s_t from previsous 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 S, we divide state 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 item and reward ids $i_t = [i_1, i_2, \cdot, i_N]$ are embedded through item embedding to $I_t = [I_1, I_2, \cdot, I_N]$, One GRU history layer is used, i.e.

$$\begin{aligned} W_t &= \mathrm{ReLU}(W_\phi \cdot \phi(I_i) + b_\phi) \\ h_t &= \mathrm{GRU}(w_t, h_{t-1}) \end{aligned} \tag{3}$$

where $\phi(I_i)$ is the vector representation of the history item, h_t is last hidden layer of GRU. Combining user embedding 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 items according to their previous preferences in source select stage and aggregation recommendation stage respectively.

Next, we input a state-action pair by concatenaing $[s_t, a_t]$ into multi-layer neural networks with ReLU activate function. Last, Double DQN outputs the Q-value ($Q(s, a; \theta^-)$) corresponding to a specific action a.

E. Multi-Agent off Policy Training

We train the proposed model based on generated trajectory data which is consists 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 the target Q-value: $Y_t^{Target} = r_{t+1} + \gamma Q(s_{t+1}, \arg\max_a Q(s_{t+1,a}; \theta_t); \theta_t^-)$ and estimate Q-value: $Y_t^{Estimate} = Q(s_t, \arg\max_a Q(s_t, a; \theta_t); \theta_t)$ based on Bellman Optimality Condition [33] as in Q-learning [34]. In the context of our Multi-Agent Recommender Model, we train our model based on VDN policy, the total Q-value function as follow Eq. (4):

$$\begin{aligned} Q_{total}(s, u) = \sum_{i=1}^{j} Q_i(s_i, u_i) = Q_a(s_t, a_t; \theta_a) + Q_b(s_t, b_t; \theta_b) \\ + Q_c(s_t, c_t; \theta_c) + Q_m(s_t, m_t; \theta_m),\ j \in (a, b, c, m) \end{aligned} \tag{4}$$

The whole model (three source agents and an aggregation agent) tries to minimize a sequence of loss functions Eq. (5) :

$$\begin{aligned} L_{\theta_j} =& \mathbb{E}_{s_t, a_t, b_t, c_t, m_t, r_t, s_{t-1}}[y_t^{target} - Q_{total}(s, u)]^2, \\ & j \in (a, b, c, m) \end{aligned} \tag{5}$$

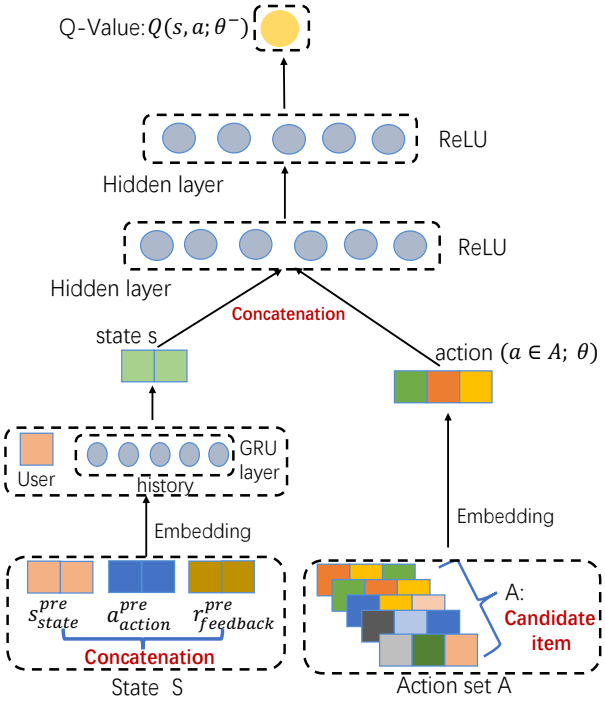

Fig. 6. A novel Double DQN framework is used in both source agent and aggregation agent.

$$
\begin{aligned}
y_t^{target} = \mathbb{E}_{s_{t+1}}[r_t + \gamma(& \\
& Q_a(s_{t+1}, \arg\max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \theta_a); \theta_a^-) \\
& + Q_b(s_{t+1}, \arg\max_{b_{t+1}} Q(s_{t+1}, b_{t+1}; \theta_b); \theta_b^-) \\
& + Q_c(s_{t+1}, \arg\max_{c_{t+1}} Q(s_{t+1}, c_{t+1}; \theta_c); \theta_c^-) \\
& + Q_m(s_{t+1}, \arg\max_{m_{t+1}} Q(s_{t+1}, m_{t+1}; \theta_m); \theta_m^-)]
\end{aligned} \tag{6}
$$

where y_{target} is formulated as aboved Eq. (6), which is the total target q-value for the current interation.

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 [13]. The parameters of the target network θ^- are fixed when optimizing the loss function L_{θ_j}. The derivatives of loss function L_{θ_j} with respective to parameters $\theta_a, \theta_b, \theta_c, \theta_m$:

$$
\begin{aligned}
\theta_a &= \theta_a + \alpha L_{\theta_j} \cdot \nabla_{\theta_a} Q_{\theta_a}(s_t, a_t; \theta_a) \\
\theta_b &= \theta_b + \alpha L_{\theta_j} \cdot \nabla_{\theta_b} Q_{\theta_b}(s_t, b_t; \theta_b) \\
\theta_c &= \theta_c + \alpha L_{\theta_j} \cdot \nabla_{\theta_c} Q_{\theta_c}(s_t, c_t; \theta_c) \\
\theta_m &= \theta_m + \alpha L_{\theta_j} \cdot \nabla_{\theta_m} Q_{\theta_m}(s_t, m_t; \theta_m)
\end{aligned} \tag{7}
$$

We present the whole training procedure of our model in details shown in Algorithm 2. In each interation on a trainning recommendation session, there mainly have two stages. Generating transitions stage (i.e., multi-types source select and aggregation item recommendation) and model trainning stage. Specifically, for generating transitions stage, in mutli-types source selectioin, given the state s_t, the source agent SR selects an item list a_t based on a 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 11); Similarly, we use the same way to select ad and fans source items (line 12-13); Obtain candidate three have selected heterogenous sources items C3 = fusion $[a_t, b_t, c_t]$ (line 14). In aggregation item recommendation: given the state s_t, the aggregation agent recommends an item list m_t from a off-policy π_m, which follows a 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 16). then recommend list m_t for user and observes the reward feedback r_t(lines 17) and next update the state to s_{t+1} (line 18-19); and finally store transitions $(s_t, a_t, b_t, c_t, m_t, r_t, s_{t+1})$ into replay memory D (line 20). For model training stage: we samples mini-batch of transitions (s, a, b, c, m, r, s') from replay memory D (line 22), and the updates the parameters according to Eq. (6) and Eq. (5) (lines 23-24). Finally, we update target network parameter after C step (line 25).

Note 1: the strategy for selecting probility $\epsilon_j (j \in [a, b, c, m])$ is as follows:

$$
\epsilon_j = 0.01 + 0.99 \times 1/e^{(session_{id}/session_{half})} \tag{8}
$$

where $session_{id}$ is training data session id, $session_{half}$ is half of total session number i.e., M/2. In order to avoid selecting the same action while ingoring exploration. **Note 2:** in our model, aggregation recommendation resust is an item list. Following [9], we calculate the accumulate reward r_t of a aggregation recommended list at time t as follows:

$$
r_t = \sum_{k=1}^{K} \Gamma^k r_{t,k} \tag{9}
$$

where k is the order that an item in the aggregation recommended list and K is the length of the recommended list, and $\Gamma \in [0, 1]$. The intuition of Eq.(9) is that the reward of the top item recommended list in the current step has a higher contribution to the overall rewards, which force: 1) source agent selected items that user most interest item from candidate source items; 2) aggregation agent arranging items that user may order in the top of the recommended list.

In the whole algorithm, we introduce widely used techniques to train our framework. For example, we utilize a technique known as experience replay [35] lines(4,20,22), introduce Double DQN (evaluation and target networks) [24], which can avoid overestimated of q-value, and adpots VDN [30] to train multi-agent model.

F. Online test

We also do online test in test data. The overall online test algorithm is presented in Algorithm 3.

IV. Experiments

In this section, to evaluate the performance of our proposed ARVM model, we conduct extensive experiments on a real short video to jointly optimize heterogenous sources item. We mainly focus on three question: 1) how the ARVM performs compared to representative baselines; 2) how the source agent

Algorithm 2 Off-policy Training of ARVM Framework.

Intialize actioin-value function of source types agent (rec, ad and fans videos) Q_a, Q_b, Q_c with random weights θ_a, θ_b, θ_c respectively.
Intialize actioin-value function of aggregation agent Q_m with random weights θ_m.
$\theta_a^- = \theta_a$, $\theta_b^- = \theta_b$, $\theta_c^- = \theta_c$, $\theta_m^- = \theta_m$,
Intialize the capacity of replay buffer D
for seesion = $1, M$ **do**
  initialize state s_0 from previsous sessions
  for t = $0, T-1$ **do**
    // Generating transitions data stage
    Observe current state s_t
    // multi-types source select
    Set rec action policy π_a =

$$\begin{cases} \text{random sample action } \mathrm{a_t}, & random() < \epsilon_a \\ \mathrm{argmax}_{\mathrm{a_t} \in \mathrm{A_t}} \mathrm{Q_a}(\mathrm{s_t}, \mathrm{a_t}; \theta_\mathrm{a}), & otherwise \end{cases}$$

    Set ad action policy π_b =

$$\begin{cases} \text{random sample action } \mathrm{b_t}, & random() < \epsilon_b \\ \mathrm{argmax}_{\mathrm{b_t} \in \mathrm{B_t}} \mathrm{Q_b}(\mathrm{s_t}, \mathrm{b_t}; \theta_\mathrm{b}), & otherwise \end{cases}$$

    Set fans action policy π_c =

$$\begin{cases} \text{random sample action } \mathrm{c_t}, & random() < \epsilon_c \\ \mathrm{argmax}_{\mathrm{c_t} \in \mathrm{C_t}} \mathrm{Q_b}(\mathrm{s_t}, \mathrm{c_t}; \theta_\mathrm{c}), & otherwise \end{cases}$$

    Execute action policy π_a, π_b, π_c, obtain fusion candidate item $C3_t = [a_t, b_t, c_t]$
    // aggregation item recommendation
    Set aggregation policy π_m =

$$\begin{cases} \text{random sample action } \mathrm{m_t}, & random() < \epsilon_m \\ \mathrm{argmax}_{\mathrm{m_t} \in \mathrm{C3_t}} \mathrm{Q_b}(\mathrm{s_t}, \mathrm{m_t}; \theta_\mathrm{m}), & otherwise \end{cases}$$

    Execute action policy π_m, obtain action m_t, obtain reward r_t from user (online simulator)
    Obtain next state s_{t+1} = concatenate(s_t, m_t, r_t)
    Update state s_t = s_{t+1}
    Store transition $(s_t, a_t, b_t, c_t, m_t, r_t, s_{t+1})$ into the replay buffer D
    // Model training stage
    Sample mini-batch of transition (s, a, b, c, m, r, s') from repaly buffer
    Set $y = r + \gamma(Q_a(s', \arg\max_{a'} Q(s', a'; \theta_a); \theta_a^-)$
    $+Q_b(s', \arg\max_{b'} Q(s', b'; \theta_b); \theta_b^-)$
    $+Q_c(s', \arg\max_{c'} Q(s', c'; \theta_c); \theta_c^-)$
    $+Q_m(s', \arg\max_{m'} Q(s', m'; \theta_m); \theta_m^-))$
    minimize$(y - (Q_a(s, a; \theta_a) + Q_b(s, b; \theta_b) +$
    $Q_c(s, c; \theta_c) + Q_m(s, m; \theta_m)))$.
    Assgin $\theta_a^- = \theta_a$, $\theta_b^- = \theta_b$, $\theta_c^- = \theta_c$, $\theta_m^- = \theta_m$, every C step
  end for
end for

Algorithm 3 Online Test of ARVM Framework.

Intialize three source agents action-value function Q_a, Q_b, Q_c with well trained weights θ_a^-, θ_b^-, θ_c^- .
Intialize aggregation agent actioin-value Q_m with well trained weights θ_m^-.
for seesion = $1, M$ **do**
  initialize state s_1 from previsous sessions
  for t = $0, T-1$ **do**
    Observe current state s_t
    // Multi-Types Source selection
    Executalte action a_t following trained policy θ_a^-
    Executalte action b_t following trained policy θ_b^-
    Executalte action c_t following trained policy θ_c^-
    // Aggregation item recommendation
    Obtain candidate aggregation data:
    C3 = $[a_t, b_t, c_t]$
    Executalte action m_t following well trained policy θ_m^- from C3
    Observe reward r_t from users (online simulator)
    Update s_t = s_{t+1} = concat(s_t, m_t, r_t)
  end for
end for

and aggregation agent in this framework contribution to the performance; and 3) how the hyper-parameters impact the model performance.

A. Experiment Setting

1) Datasets: Since there are no public datasets consist of multi-heterogenous sources to model, we train our model in a real-world short video dataset which is from *Kuaishou* APP. There are three types of videos, i.e., normal videos(recommended items), ad videos(advertised items) and fanstops (fans items). We collect 66698 sessions in temporal order, following [36], sessions are divided in the principle of whenever there exists a time gap of more than 3 days. The specific data format is as follows: a session data at time step t: {candidate rec source video: *srv*, seleted rec item, candidate ad source video: *sav*, selected ad video, candidate fans source videos*sfv*, selected fans video, aggregation recommend list, reward}. To evaluate the performance of item recommendation, we adopted the leave-one-out evaluation, which has been widely used in literature [37], [38]. For each user, we held-out her latest session as the test set and utilized the remaining data for training/validating. The statistics of the dataset are in Table II.

2) Parameter Setting: The proposed ARVM model has four recommendation agent network, all of desigend as a multi-layer neural network with Relu activate function. The detailed input and output setup information for source agent and aggregation agent are given in Table III. The reward discount factor fine-tune range is γ = $[0.9, 0.8, 0.7, 0.6, 0.5]$. In our experiments, we used Adam for learning parameters with a learning rate fine-tune range in $[0.0001, 0.001, 0.01, 0.1]$. We

TABLE II
STATISTICS OF THE KUAISHOU VIDEO DATASET

session	user	rec video	ad video	fans video
66698	8801	367744	65035	12504
dwell time	**session length**	**source rec candiate items**	**source ad candiate items**	**source fans candiate items**
[0, 1]	[1, 100]	6	6	6
SR select items	**SA select items**	**SF select items**	**AA aggregate items**	**ground-truth rewards**
3	3	3	3	3

used replay buffer size is 30000 and the minibatch size is 32. More model parameter setting are in Table III.

TABLE III
THE NETWORK ARCHITECTURE OF ARVM.

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-2
Optimization algorithm	Adam
Memory D size	3e4
Mini-batch size	32
Target θ^- update period C	1e2
user&item id embedding size	32
Hidden layer size	(32,16)
GRU hidden layer size	64
Discount factor γ	0.8
Weight decay	1e-6
Memory_capacity to update agent	4e4

3) Metrics: To measure our model ARVM online performance, we leverage the *Average Dwell Time* (ADT: the mean value of a user's dwell time in minutes on a recommendation item list) in all session $R = \frac{1}{M}\sum_1^M \sum_1^T r_t$ as the metric, where M is the number of total sessions in test data; T is the interaction length of a session. To describe the significance of the results, we following [13], which uses the improvement of ARVM over each baseline.

4) Baselines: To answer question 1: "how the ARVM performs compared to representative baselines", we use four representative recommender system methods for baseline. For all baseline methods, the experimental data format needs to be fine tuned, that is, put all the data together. Specifically, we put the normal videos, ad videos and fans videos into together as one candidate item set. We compare our model with the following baselines, including two general-based methods and two session-based methods:

- **BPR [37]:** 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 [39]:** NCF is also a 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 [40]:** This baseline is an attention mechanism-based RNN approach that aims to learn a user's sequential behavior in the current session.
- **GRU4rec [41]:** 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).

5) Overall Performance Comparision: Fig.7 illustrates the training process of the proposed model. It can be observed that model converges with around 25000 trainning sessions as shown in red line. According to line 11-13 & 16 in algorithm 2, at the beginning, the action selected by the strategy will be more random, resulting in model instability and extremely high value point. However, with the progress of training, the strategy will select the action with greater Q-value and the model will gradually converge.

Fig.8 shows that model test performance metrics ADT increases first and then tends to slight change within a range $[0.25, 0.35]$. We calculate the average of the last five test results as the final predict result of in our experiment. The overall performances between our model and baselines are shown in Fig. 9. From Fig. 9, we observe the following facts:

- Overall baselines performance comparison, ARVM $>$ NARM $>$ GRU4rec $>$ NCF $>$ BPR .
- Session-based methods (NARM, GRU4rec) achieves better performance than general-based methods (NCF, BPR), which further confirm the validity of method RNN or GRU to capture both the user's sequential behavior for sequential recommend problem.
- The NARM achieves better performance than GRU4rec, which demonstrates that attention mechanism-based RNN models may be better at dealing with sequence information in our heterogeneous source sessions data.
- NCF perform better than BPR, which utilizes a neural network (NN) to model nonlinear user-item interactions, but BPR does not. Result shows that NN-based methods are more effective than the trainditional-based method in solving heterogeneous sources problem.
- By comparing accumulate reward, the result shows that our proposed model ARVM outperforms all other baselines algorithms.

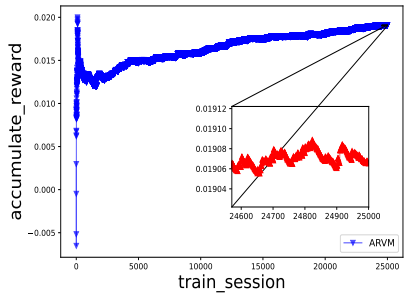

Fig. 7. Model performance with training data, and red line shows model converges after trainning about 25000 sessions.

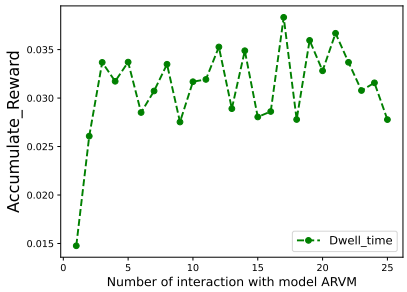

Fig. 8. Cumulative rewards among 909 users interaction 25 times with the multi-agent recommendation model ARVM.

B. Ablation Study

In this section, we conduct ablation studies to answer the second question: "whether different model components are useful for the final result ". We mainly focus on the following variants:

- ARVM-α: This variant is to evaluate the effectiveness for the aggregation agent, while we only train the aggregation agent and the candidate heterogeneous source is consist of rec, ad and fans videos.
- ARVM-β_1: This variant is to illustrate the useful for the rec source agent, while we only train the rec agent and candidate heterogeneous source is only rec video.
- ARVM-β_2: This variant is to illustrate the effectiveness for the ad source agent, while we only train the ad agent and candidate heterogeneous source is only ad video.
- ARVM-β_3: This variant is to illustrate the effectiveness for the fans source agent, while we only train the fans agent and candidate heterogeneous source is only fans video.
- ARVM-β_4: This variant is to illustrate the effectiveness for the rec and ad source agent, while we train the rec&ad&aggregation agents and candidate heterogeneous sources are rec&ad video.
- ARVM-β_5: This variant is to illustrate the effectiveness for the rec and fans source agent, while we train the rec&fans&aggregation agents and candidate heterogeneous sources are rec&fans video.
- ARVM-β_6: This variant is to illustrate the effectiveness for the ad and fans source agent, while we train the ad & fans& aggregation agents and candidate heterogeneous sources are ad&fans video.

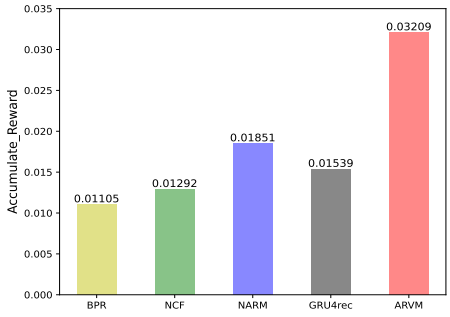

Fig. 9. Overall performance comparison in online test.

The result are shown in Table IV. It can be observed the following facts:

- ARVM-α validates the effectiveness of introducing three heterogeneous source agents to select items that meets the user's interests from their respective source type videos.
- ARVM-$\beta_1, \beta_2, \beta_3, \beta_4, \beta_5, \beta_6$ validates the effectiveness of introducing aggregation agent to decide the recommendation order of the different heterogeneous items.
- All component experimental results show the effectiveness of our proposed multi-agent RL recommender system model.

TABLE IV
ABLATION STUDY RESULTS

method	reward	improvament	p-value
ARVM-α	0.02639	21.63%	0.06422
ARVM-β_1	0.01736	84.92%	0.00060
ARVM-β_2	0.02950	8.79%	0.11483
ARVM-β_3	0.03021	6.25%	0.36509
ARVM-β_4	0.02865	12.01%	0.12032
ARVM-β_5	0.02860	12.21%	0.10540
ARVM-β_6	0.02920	9.90%	0.11219
ARVM	0.03209	*	*

C. Parameter sensitivity analysis.

In order to deal with question 3: "how the hyper-parameters impact the model performance ". Our method has a key parameters, i.e., γ that is used to balance the short-term and long-term rewards. Generally speaking, select the smaller γ, recommender system pays more attention to the users' immediate preference, while select larger γ recommender system pays more focus on the future engagement. In this section, we investigate how the proposed framework ARVM performs with the changs of γ by tunning it in the range of $[0.5, 0.6, 0.7, 0.8, 0.9]$ in Eq. (6). The results are presented in Figure 10. We find that the optimal γ for our proposed model is 0.8.

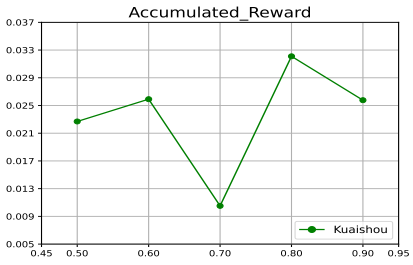

Fig. 10. Model performance with gamma.

V. RELATED WORK

In this paper, we briefly review works related to our study. In general, the related work can be mainly divided into reinforcement learning for recommender system and short video heterogeneous sources aggregation.

A. RS based on RL

In conventional recommendation systems, it is often suggested to list items that users may be interested in. However, the user's long-term interaction is rarely mentioned. In recent years, recommendation system based on reinforcement learning has attracted more and more attention in academia and industry. To the best of our konwledges, the existing RS based on RL methods mainly aim at specific types item in single agent, take user as the environment and recommend agent as the recommendation system. For example, [9] ulitizes actor-critic framework to bulid a list-wise recommendation framework with a sinle DQN agent. [11] propose a novel method to incorporate positive and negative feedback into DQN with GRU with a single agent. [32] emploies generative adversarial network to bulid a model-based reinforcement learning framework for recommendation systems. [15] is also model-based RL for RS, which models the user-agent interaction for offline policy learning. [10] models page-wise recommendation framework and aims to optimizes a page of item. [7] aims to do online new recommendatioin based on RL machanism. [13] proposes a novel Deep Q-network architecture for online advertising that can continuously update its advertising strategies and maximize reward in the user's long-term interaction. In addition to using single agent RL to solve the problem of recommender system, multi-agent is also gradually attracting the attention of researchers. [14] is utilizing multi-agent reinforcement learning method for impression allocation in online display advertising. [29] proposes a model that can optimize ranking strategies collaboratively for multi-scenario ranking problem with a multi-agent RL.

Existing RL for recommender models mainly focus on how to design effective agents for single source types item recommendation, while little effort has been devoted to studying the heterogenous sources on users' long-term engagement in a recommender system. In this work, by revisiting there works, we design a heterogenous sources fusion item recommender system at Kuaishou APP, which model the users' total long-term feedback on three sources types item (rec, ad and fans item).

B. Heterogeneous Sources Aggregation

In recent years, reslut aggregation has a wide range of tasks, such as web search [16], [17] and E-commerce search [19]. To be specific, [17] address the issue of integrating search results from a news vertical into web search results. [16] address the problem of vertical selection, predicting relevant verticals for queries issued to the search engine's main web search page. [19] exploits hierarchical reinforcement learning for aggregating search results from heterogenous sources in an E-commerce environment. To the best of our konwledge, existing research on search reslut aggregation has focused on web search and E-commerce search, while in this paper, we study the task in short video recommendation. Our purpose is to recommend short videos from different heterogeneous sources that users are most interested in, so as to increase users' long-term engagement.

VI. CONCLUSION

In this paper, we propose a novel framework ARVM, which leverages Multi-Agent Reinforcement Learning to solve aggregation recommend problem from heterogeneous sources. First, we build source select agent to automatically learn the optimal heterogeneous sources select policy. And then we also bulid aggregation agent to automatically learn the optimal aggregation recommendation policy. Last, source agent and aggregation agent cooperate to achieve overall optimization. To demonstrate our model's effectiveness, extensive experiments are conducted based on the real-world Kuaishou APP datasets.

In future studies, we plans to include other types of heterogeneous sources into ARVM, in particular, textual and aural information at Kuaishou.

ACKNOWLEDGMENT

This work is supported in part by

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] P. Sun, L. Wu, and M. Wang, "Attentive recurrent social recommendation," in *The 41st international ACM SIGIR conference on research & development in information retrieval*, 2018, pp. 185–194.

[4] H. Gao, J. Tang, X. Hu, and H. Liu, "Content-aware point of interest recommendation on location-based social networks," in *Twenty-ninth AAAI conference on Artificial Intelligence*, 2015.

[5] 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.

[6] F. Ricci, L. Rokach, and B. Shapira, "Introduction to recommender systems handbook," in *Recommender systems handbook*. Springer, 2011, pp. 1–35.

[7] 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.

[8] M. L. Puterman, "Markov decision processes," *Handbooks in operations research and management science*, vol. 2, pp. 331–434, 1990.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] D. Wu, C. Chen, X. Yang, X. Chen, Q. Tan, J. Xu, and K. Gai, "A multi-agent reinforcement learning method for impression allocation in online display advertising," *arXiv preprint arXiv:1809.03152*, 2018.

[15] 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.

[16] 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.

[17] 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.

[18] B. Long and Y. Chang, *Relevance ranking for vertical search engines*. Newnes, 2014.

[19] 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.

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

[21] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," *Advances in neural information processing systems*, vol. 12, 1999.

[22] 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.

[23] V. Konda and J. Tsitsiklis, "Actor-critic algorithms," *Advances in neural information processing systems*, vol. 12, 1999.

[24] 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.

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

[26] 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.

[27] 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.

[28] 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.

[29] J. Feng, H. Li, M. Huang, S. Liu, W. Ou, Z. Wang, and X. Zhu, "Learning to collaborate: Multi-scenario ranking via multi-agent reinforcement learning," in *Proceedings of the 2018 World Wide Web Conference*, 2018, pp. 1939–1948.

[30] 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.

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

[32] 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.

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

[34] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3, pp. 279–292, 1992.

[35] L.-J. Lin, *Reinforcement learning for robots using neural networks*. Carnegie Mellon University, 1992.

[36] 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.

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

[38] 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.

[39] 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.

[40] 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.

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