Multi-agent Reinforcement Learning for Mention Recommendation

Journal:	*ACM Transactions on Recommender Systems*
Manuscript ID	TORS-2022-0043
Manuscript Type:	Original Research Paper
Date Submitted by the Author:	02-Nov-2022
Complete List of Authors:	HUANG, HANCHI; Shanghai Jiao Tong University, Li, Yan; Shenzhen Polytechnic Shen, Li; JD.com Inc Lin, Jianghao; Shanghai Jiao Tong University Xin, Xin; Shandong University
Computing Classification Systems:	Mention Recommendation, Multi-agent Reinforcement Learning, Graph Neural Network

Dear Editor:

We are very interested in submitting a research paper to ACM Transactions on Recommender Systems.

The title of the research paper is: "Multi-agent Reinforcement Learning for Mention Recommendation". No conflict of interest exists in the submission of this manuscript, and the manuscript is approved by all authors for publication. I would like to declare on behalf of my co-authors that the work described was original research that has not been published previously, and not under consideration for publication elsewhere, in whole or in part. All the authors listed have approved the manuscript that is enclosed.

Thank you very much for considering our manuscript for potential publication. Please do not hesitate to contact me with any problems.

Best regards,
HANCHI HUANG, Shanghai Jiao Tong University,
YAN LI, Shenzhen Polytechnic,
LI SHEN, JD Explore Academy,
JIANGHAO LIN, Shanghai Jiao Tong University,
XIN XIN, Shandong University.

Multi-agent Reinforcement Learning for Mention Recommendation

HANCHI HUANG, Shanghai Jiao Tong University
YAN LI, Shenzhen Polytechnic
LI SHEN, JD Explore Academy
JIANGHAO LIN, Shanghai Jiao Tong University
XIN XIN, Shandong University

In social networking platforms such as Twitter, users tend to mention other related users (indicated by the "@" symbol) when posting messages. In these scenarios, automatically suggesting candidate users who are likely to be mentioned can improve communication efficiency and the user experience. Existing work typically uses recent posts or a few randomly selected historical messages to infer user preferences and relationships between them. However, few papers have taken into account two fundamental properties of mention recommendations during this inferring process, namely dynamic cooperation among users and the sparsity of their interactions. To address these issues, we propose an MA-DGNN algorithm, a graph-based multi-agent reinforcement learning (MARL) approach that consists of two key components: a time-delayed aggregation graph policy network and a custom prioritized experience replay (PER). As to the time-delayed aggregation graph policy network, the time-delayed graph support operation is applied to act as a distributed controller for large networks of users with interaction dynamics and sparse communication, where a time-delayed multi-hop information diffusion mechanism helps sparsely interacting users to obtain advice from densely interacting users. As to the prioritized experience replay mechanism, it combines six carefully chosen metrics to handle sample imbalance and non-stationarity of user preferences and to leverage statistical patterns in user interactions. Moreover, regarding the inferring process, MA-DGNN updates the user-user utility matrix by decomposing the neural matrix based on similarity loss to aggregate the output of the policy network and aid in credit assignment in MARL; regarding the loss function, the loss functions of MA-DGNN's policy and critic networks are modeled as minimal maximum objectives to learn robust policies in the face of multiple non-stationarity from both the environment and other agents. Extensive experiments on synthetic and real datasets demonstrate that our MA-GDNN outperforms the existing state-of-the-art approaches.

CCS Concepts: • **Computing methodologies** → **Artificial intelligence**.

Additional Key Words and Phrases: Mention Recommendation, Multi-agent Reinforcement Learning, Graph Neural Network

ACM Reference Format:
Hanchi Huang, Yan Li, Li Shen, Jianghao Lin, and Xin Xin. 2022. Multi-agent Reinforcement Learning for Mention Recommendation. *Proc. ACM Meas. Anal. Comput. Syst.* 37, 4, Article 111 (August 2022), 19 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Mention notification is a very popular tool when people communicate on social networks. By tagging an '@' symbol with a username concatenated in the post message, a 'You have been mentioned' notification would be pushed to the specific user. Automatically suggesting who is the best to be mentioned with an '@' is a key problem

Authors' addresses: Hanchi Huang, qq20001224m@sjtu.edu.cn, Shanghai Jiao Tong University; Yan Li, yb57411@szpt.edu.cn, Shenzhen Polytechnic; Li Shen, mathshenli@gmail.com, JD Explore Academy; Jianghao Lin, chiangel@sjtu.edu.cn, Shanghai Jiao Tong University; Xin Xin, xinxin@sdu.edu.cn, Shandong University.

2476-1249/2022/8-ART111 $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

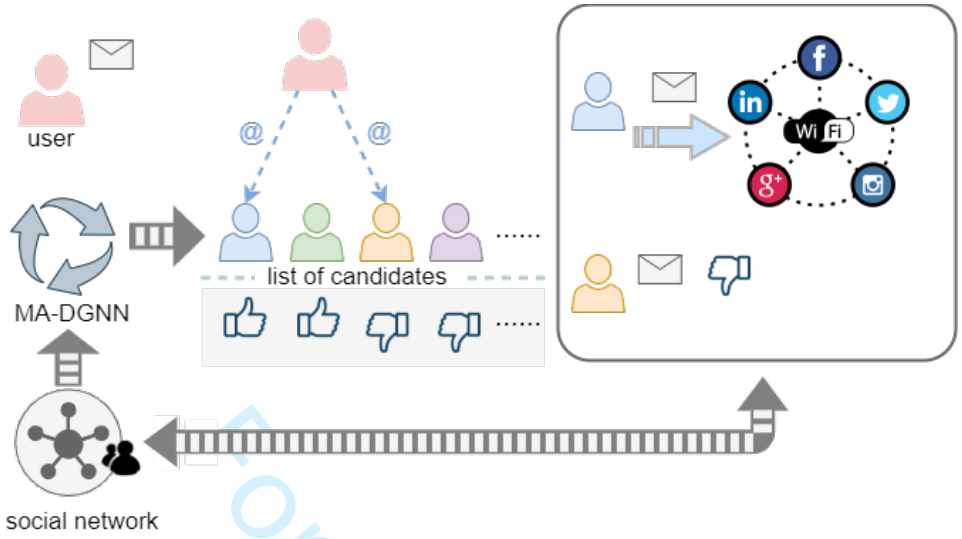

Fig. 1. Mention recommendation processed by our proposed MA-DGNN.

in social networks titled as mention recommendation. It brings in many benefits including but not limited to (i) strengthening the relationship with friends in that the one mentioned feels a special greeting and care from the user who publishes the tweet [27]. The reactions make friends closely connected and spend more time and attention on each other. (ii) bridging over different circles, since the issued post could be forwarded to friends of friends more than we can imagine. It is similar to a view from [14] that the tagging activity promotes the crossover of social graphs on Twitter, which is widely applied in commercial promotion. Breaking the boundaries and touching the potential customers are just what the retailers want most; and (iii) improving reciprocity over a social network. Take LinkedIn for instance, the published post could be kept hot on the LinkedIn platform if the publisher mentions the right persons with great interest as well as great public impact. Similar examples on Facebook could be found in [10, 27].

The whole process of mention recommendation is displayed in Figure 1. For each recommendation round, in the beginning, a user colored pink issues a post and intends to '@' somebody. When the user inputs the '@' symbol, the algorithm (named MA-DGNN in Figure 1) should quickly recommend a list of candidate mentionees (with colors blue, green, yellow, and purple, etc.) for reference. Since the pink user has the biggest preference for the blue user, the pink user mentions the blue user ultimately. After receiving the 'mention' notification from the pink user, the blue user finds the post from the pink user very interesting and continues to share the post with other friends according to his social network. Therefore, by the information diffusion on a social network, the relationship between users is continuously evolving, which exerts influence on not only the algorithm's follow-up recommendation but the subsequent users' '@' decision. Since mention recommendation is a multi-round recommendation process involving the complex and dynamic relationship among the post publisher, the post's content, and other users on social networks, it seems more natural to use reinforcement learning (RL) instead of supervised learning which assumes environments' stationarity, and to use multiple agents instead of a single agent to model evolving interactions between users while making personalized decisions for each user.

Existing studies on mention recommendation could be mainly categorized into three methods: ranking-based methods [17, 32, 34, 45], classification-based methods [5, 16] and Multi-agent Reinforcement Learning (MARL)

based methods [9][1]. For ranking-based methods, the target user would be recommended with a ranked list of potential mentionees achieved by a global scoring function. Similarly, classification-based methods address mention recommendation from a classification perspective, i.e., potential mentionees should be either predicted as positive or negative in each mention behavior towards the target user. However, these two methods fail to consider the following two issues. *Firstly*, the global scoring function or classification model, as a unified strategy for all users, could not well capture the very different styles of different users and their preferences evolving in different directions as time goes by. *Secondly*, the aforementioned methods generally suffer from the sparsity problem, i.e., lack of interaction signals between users. Ideally, different levels of relationships between different users and other related information should be fully leveraged to make better recommendation decisions for users with less historical interaction records.

To address these challenges, we propose MA-DGNN, a graph-based MARL model with deliberately customized prioritized experience replay to solve mention recommendation tasks with a large number of users. Each user on the social network would be treated as an agent and the interactions with other users would be well modeled as cooperations with the environment. The system is expected to automatically generate highly qualified mentionee candidates such that the target user would prefer to hit them with the symbol '@'. First, during state embedding, we employ an LSTM-based graph neural network to model users' complex time-varying dynamics, since the messages along with user interactions diffuse over the social network in a sequential manner. To handle a large number of users and the sparsity and non-stationarity of interactions between users, we adopt the delayed aggregation graph neural network (DGNN) [33] as the policy network of MARL, which combines the time-delayed graph operation of different orders to serve as distributed controllers for large networks of users with interacting dynamics and sparsely available communications. Thanks to different levels of information diffusion along the multi-hop neighbors in DGNN, the relationship between nodes with sparse interactions might be well inferred from the flowing information among other nodes with varying distances. Last but not least, a customized personalized replay buffer (PER) with non-stationarity detection and state clustering techniques is also utilized to make full use of historical statistics and handle imbalanced datasets as well as sparse user interactions.

Below, we summarize our main contributions as four-fold:

- We are the first ones to model each user as an agent in mention recommendation problem and the first one to introduce a graph-based MARL approach with time-delayed graph support which models the sparse and dynamic interactions among users to solve the problem with a large number of users. A customized PER mechanism and a reward reshaping technique are designed elaborately to handle the sample imbalance and the non-stationarity of users' preferences, and take full advantage of the statistical laws inside users' interactions.
- During the training of MARL, we customize minimax objectives to minimize the losses of the policy and critic networks for a worst-case scenario, so as to learn robust policies in the face of multiple non-stationarity from both environments and other agents. The minimax optimization technique, the customized PER mechanism, and the LSTM-based graph neural network for state embedding together empower our algorithm with great adaptivity to the changes of both tweets' styles and users' interactions.
- To make a comprehensive aggregation for the outputs of DGNN as well as performing credit assignment, we construct an agent-agent utility matrix which learns the contribution of each user's suggestion for other users and regard each row of the matrix as the aggregation coefficients. The neural matrix factorization technique, which can well deal with data sparsity by learning hidden factors, is novelly proposed to optimize the matrix. An ablation study is performed to successfully show the great performance improvement when using this decision aggregation mechanism.

[1]Although Gui et al. [9] propose a MARL method to tackle mention recommendation problems, the method only constructs two agents, i.e., two tweet selectors with different functions instead of modeling each user as an individual agent, and thus neglects users' personal preference.

- Extensive experiments on the synthetic and real-world datasets with six classic evaluation metrics demonstrate the absolute performance superiority of our algorithm over existing methods.

2 RELATED WORK

2.1 Mention Recommendation

There are a bunch of studies proposed to address mention recommendation. The most popular employed methods are ranking-based models and classification-based models. For ranking-based methods, a global ranking function could be achieved by (semi-) supervised learning. Specifically, Wang et al. [34] adopt ranking support vector regression with features respecting user interest match, user relationship, and user influence. Tang et al. [32] apply the ranking support vector machine with features regarding content, social, location, and time. Wang et al. [35] introduce a semi-supervised deep model to capture the highly non-linear relationship between nodes.

For classification-based models, whether to recommend each user is deduced by tackling several classification problems. Pramanik et al. [23] introduce a tweet propagation model based on a multiplex network framework to handle the classification problems, which allows analyzing the effects of mentioning on final retweet count. Wang et al. [36] propose a generative model to solve mention recommendation by learning users' semantic patterns and the correlation between users' multi-modal mention documents in a unified way. Yi et al. [39] construct a heterogeneous mention network to model different entities such as authors, messages, and users into a unified low-dimensional embedding vector space, such that the heterogeneous information can be calculated in the same embedding space.

However, the above approaches and the MARL method in [9] take a unified strategy to serve for all users which might neglect users' personalized preferences. Besides, these methods do not fully characterize the two primary properties of mention recommendation, i.e., the dynamic and sparse communications of user behaviors, which make them not quite viable in our problem setting.

2.2 Reinforcement Learning for Recommendation

Reinforcement learning (RL) is the process of agent learning to maximize the long-term rewards when interacting with the environment [31]. It has been widely introduced into recommendation systems due to its consideration of users' long-term feedback. Via interacting with a target user, the recommendation platform would learn her item preference step by step according to the reward, e.g., the user's click or longer dwelling time. However, such a scenario is quite different from our problem setting. In social networks, the node (user) and edge (relationship) adding or disappearing happen all the time. The interaction environment becomes more complex than the general recommendation. Despite this huge difference, we would still introduce the existing RL-based methods over general recommendation as follows. Existing RL methods for recommendation can be roughly divided into two categories, value-based methods [6, 38, 41, 43, 44] and policy-based methods [4, 11, 12, 19, 26, 28, 37, 40, 42].

Value-based methods evaluate the Q value for each candidate item (or item set) and select the item (or item set) with the highest Q-value. Recent literature approximates Q-values mostly via Deep Q-Networks. For example, Zheng et al. [44] propose a Deep Q-Learning based recommendation framework with the user return pattern as a supplement to user feedback. Similarly, Zhao et al. [43] leverage the Deep Q-Network and integrate skipped items (negative feedback) into RL-based recommendation methods. Nevertheless, it can be computation inefficient for value-based methods to evaluate Q-values for all items when the item space is very huge. As a consequence, some literature adopts policy-based methods. They explicitly construct a representation of the policy, which maps environment state to action and store it in memory during the learning process. For instance, Zhao et al. [42] propose a page-wise recommendation framework that leverages Deep Deterministic Policy Gradient (DDPG) to automatically learn the optimal strategies. Meanwhile, Hu et al. [12] develop a policy gradient algorithm to learn

Table 1. Notations.

Notation	Meaning
T	total number of rounds
t	current time
N	number of users
tm_t	message at time t
o_t	linearly weighted value output by DGNN and utility matrix
$thre_t$	threshold which decides whether to recommend the most recent $\mathcal{K}$ users that user i_t has mentioned or to randomly select $\mathcal{K}$ users except user i_t to user i_t
x_j	the feature vector of the j-th latest tweet that user j has issued
y_j	the feature vector of the j-th latest user that user j has mentioned
p_t	the number of historical positive samples till time t
n_t	the number of historical negative samples till time t
$r_{t,i}$	the reward of agent $i \in [N]$ at round t
S_n	a graph shift operator that describes the graph support G_n

an optimal ranking policy, as well as dealing with the high reward variance and unbalanced reward distribution in the search session Markov decision process.

However, seldom do existing RL strategies consider tackling the mention recommendation problem, where the interactions between the target user with the others are affected by the complex time-varying dynamics and the interaction sparsity makes the learning process difficult to address. Although the proposed RL-based strategies share a similar long-term rewards maximization goal, they are actually not feasible for our problem[2].

3 METHODOLOGY

In this part, we first formulate the mention recommendation task in Section 3.1 and then present the overview and core components of our proposed method MA-DGNN in Section 3.2.

3.1 Problem Formulation

Assume there are N users in a social network platform. At round $t \in [T]$, user i_t comes into the system and issues a message tm_t, e.g., a tweet or microblog. Given i_t and tm_t, the system must recommend one or several users from a list of candidate mentioned users $[N]$ [3] according to their long-term historical tweets $\mathcal{H}$. The objective is to have an overall great performance with regard to precision, recall, and F1-score, the three popular evaluation metrics in mention recommendation tasks and traditional recommendation system field.

3.2 Algorithm

Figure 2 shows the architecture of our proposed method MA-DGNN. To model the dynamic relationship between users and make a personalized recommendation for each user, we regard each user as a node in a graph, as well as an agent in the multi-agent reinforcement learning (MARL) system. At round $t \in [T]$, a delayed aggregation graph neural network (DGNN) is utilized to output a real value for each node/user, which applies the delayed graph

[2]To achieve personalized mention recommendation and better utilization on users' interactions, it seems promising to utilize MARL which could model the personalized strategy for each user as each individual agent and infer relationship among users by learning agents' dynamic and cooperative behaviors. Although MARL-based methods could adopt personalized strategies and help alleviate the sparsity problem, they are still in face of specific limitations to be overcome. Firstly, samples from MARL's replay buffer might be out of date due to users' dynamic preference, and the mixture of noisy samples from various users would cause a great interference to policy updating and inferring process. Secondly, when updating a policy for a certain agent, other agents could be regarded as environments and the non-stationarity of other agents (i.e., environments) breaks the Markov assumption of single-agent RL, which might result in performance degradation or even failures of policy learning. To handle these challenges, we equip our MA-DGNN with a series of novel techniques which demonstrate great performances in the experimental session.

[3]In experiments we recommend the candidate mentionees from the whole user set $[N]$ for each target user because the times that MA-DGNN recommends user i_t when user i_t is issuing a message is too rare to influence the ultimate performances.

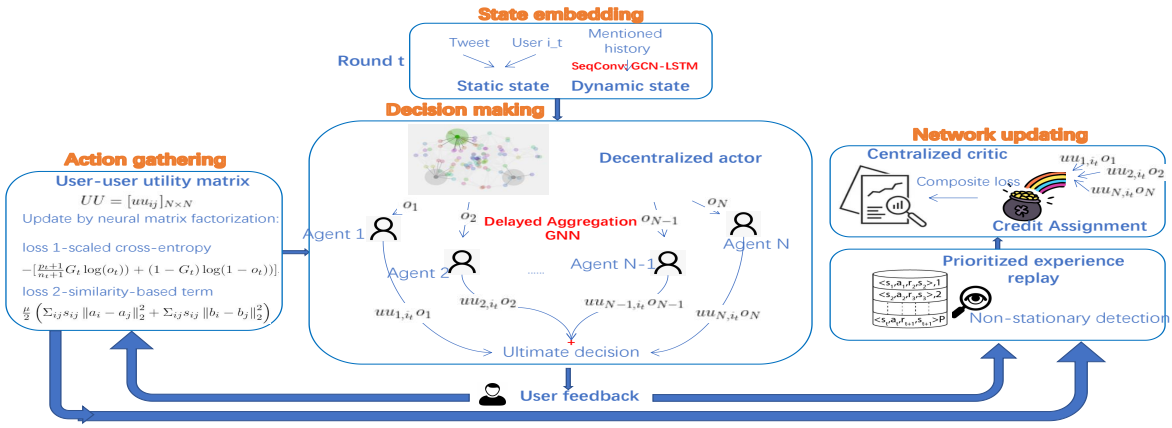

Fig. 2. The overall framework of MA-DGNN.

support operation to serve as distributed controllers for large networks of users with interacting dynamics and sparsely available communications. Then we use the i_t-th row of a user-user utility matrix to aggregate the real values for every user and output a linearly weighted value $o_t \in [0, 1]$ to decide which users should be recommended. The utility matrix models the dynamic relationship between each pair of users and is updated by a neural matrix factorization method with the loss considering the similarity between users. It is optimized independently to DGNN instead of being jointly and thus slowly updated with DGNN via only one feedback at each round. After obtaining o_t, we compare it to a threshold $thre_t$[4]. If $o_t > thre_t$, then we recommend the most recent $\mathcal{K}$ users that user i_t has mentioned; otherwise, $\mathcal{K}$ users except user i_t are randomly selected for the recommendation to user i_t.

We use a unified critic network and decentralized actor networks in MARL with minimax optimization for robust policy learning and adopt a customized experience replay mechanism with state clustering and non-stationarity detection to update both the policy network and the critic network, which consider both the sample diversity and users' varying preferences. As for the critic loss, we perform credit assignment to the N agents with weights being the i_t-th row of the user-user utility matrix, which is proven to obviously improve the performance of our MA-DGNN in the ablation study part. Lastly, to handle the seriously imbalanced data, we scale rewards in different scenarios and the different terms in the loss of neural matrix factorization according to the relative ratio between positive samples and negative samples[5].

Below, we give a detailed description of each module.

3.2.1 The composition of Markov Decision Process. We model the long-term mention recommendation process as a Markov Decision Process (MDP) and introduce the state, action, and reward configuration in the MDP in detail.

State. We regard each user as an agent in MARL. For $i \in [N]$, the state of agent i consists of two parts: the static and dynamic parts.

[4] $thre_t$ is a sliding-window threshold which will be described later.

[5] Positive samples refer to samples in the rounds where the mentionee belongs to the most recent $\mathcal{K}$ users that the currently served user has mentioned. The negative samples are samples in the remaining rounds.

- The *static* part is the pre-trained feature vector of user i, which shows the user's inherent features such as birth, birthplace, education, etc.
- With regard to the *dynamic* part, since the historically issued tweet list and the mentionee list of a user are essential to the user's characterization, we also consider SeqConv [**?**], an LSTM-based graph convolutional operator, which is useful for graph datasets where each node represents a sequence, to deal with the two aforementioned time series and form the dynamic part of a user's state. Specifically, the dynamic part of user (agent) i's state is $SeqConv([x_1, y_1], [x_2, y_2], \cdots, [x_{N_{list}}, y_{N_{list}}])$, where $x_j (j \in [N_{list}])$ refers to the feature vector of the j-th latest tweet that user i has issued, and $y_j (j \in [N_{list}])$ refers to the feature vector of the j-th latest user that user i has mentioned. In the beginning, when the system has not collected enough samples, the uncollected x_j and y_j are set to be the zero vectors.

The state input to the policy network and the critic network as stated below is the concatenation of all agents' states.

Action. The action is the concatenation of the N real values output by the graph-based policy network, as stated below. The ultimate decision, however, depends on whether o_t, the linear combination of the N real values, exceeds a sliding-window threshold $thre_t$, the $\frac{n_t+1}{p_t+1}$ percentile of the list $\{o_i\}_{i\in[t-1]}$, where p_t is the number of historical positive samples and n_t is the number of historical negative samples. If yes, then the ultimate decision is 1, i.e., the algorithm will recommend the most recent $\mathcal{K}$ users that the currently served user has mentioned to that user. If not, then the ultimate decision is 0, i.e., the algorithm will randomly select $\mathcal{K}$ users for recommendation to the currently served user. As can be seen in the ablation study, setting $thre_t$ to be $\frac{n_t+1}{p_t+1}$, which considers datasets' imbalanced property, instead of the normal threshold 0.5 improves the performance of our MA-DGNN remarkably in all real datasets.

Reward. To update each node of DGNN according to their different contributions for the ultimate decision, we customize different rewards for each node at each round. Specifically, at each round t, our algorithm will output a real value $o^0_{t,i} \in [0, 1]$ for each user $i \in [N]$ and we thus sample a binary value $o^1_{t,i} \sim Bernoulli(o^0_{t,i})$. If the mentioned user at round t was mentioned by user i_t in user i_t's most recent $\mathcal{K}$ rounds, then we say that the behavior at round t is a positive sample and the ground truth at round t is 1; otherwise, the behavior at round t is a negative sample and the ground truth is 0. Since in real datasets, the number of negative samples far outweighs the number of positive samples, we scale the terms in rewards to handle the issue of imbalanced datasets. Specifically, first, denote the ground truth at round t to be G_t. Then, for the reward of agent $i \in [N]$ at round t, we set

$$r_{t,i} = \begin{cases} 1 - \frac{p_t}{1+n_t+p_t}, \text{ if } o^1_{t,i} = G_t = 0; \\ -1, \text{ if } o^1_{t,i} = 0, G_t = 1; \\ -(1 - \frac{p_t}{1+n_t+p_t}), \text{ if } o^1_{t,i} = 1, G_t = 0; \\ 1, \text{ if } o^1_{t,i} = G_t = 1. \end{cases} \tag{1}$$

The design of $r_{t,i}$ is to give a bigger reward or punishment for the prediction of the rarer positive samples. Otherwise, if $r_{t,i} \in \{-1, 1\}$ for all t and i, great chances are that the learning of rarer positive samples will be quickly offset by the major negative samples and the performance of MA-DGNN might hurt much accordingly.

3.2.2 Our customized cooperative MARL algorithm. We regard each user as an agent and design a graph-based MARL algorithm to make a comprehensive decision for each user. The minimax multi-agent deep deterministic policy gradient (M3DDPG) approach [18] is adopted to update our proposed algorithm, for the purpose of learning robust policies. Below we illustrate the structure of our method in detail.

Centralized critic and decentralized actors with minimax optimization. We take a centralized critic which absorbs the concatenation of all agents' states and actions (decentralized actors). The critic, then, outputs the value vector which evaluates the state-action pair of each agent, so as to update the policy network.

Since during the training of MARL, agents' policies might be very sensitive to the changes in other agents' strategies, which could slow down the speed of policy loss's convergence as well as make the decision-making

process out-of-order. In order to learn more robust policies, we are the first ones in the field of mention recommendation who follow [18] to update the policy and critic networks considering the worst situation, i.e. optimizing the reward of each agent while assuming other agents perform adversarially. Formally, we customize minimax objectives for each agent to update the policy and critic networks at round t, where minimax is a fundamental concept in game theory and can be applied to general decision-making under uncertainty, prescribing a strategy that minimizes the possible losses for a worst-case scenario, as stated in [18]. Specifically, the minimax objectives of the policy network and the critic network for agent i are

$$\max_{\boldsymbol{\mu}} \min_{a^0_{j\neq i}} Q_i^{\boldsymbol{\mu}}(\mathbf{s}_t, a^0_1, \ldots, a_i, \ldots, a^0_N)|_{a_i=\boldsymbol{\mu}_i(\mathbf{s}_t)}, \tag{2}$$

and

$$\begin{aligned} \max \; -\mathcal{L}_{t,i} &= -\left(Q_i^{\boldsymbol{\mu}}\left(\mathbf{s}_t, a_{t,1}, \ldots, a_{t,N}\right) - y_i\right)^2, \\ y_i &= r_{t,i} + \gamma Q_i^{\boldsymbol{\mu}'}\left(\mathbf{s}_{t+1}, a'^{\star}_1, \ldots, a'_i, \ldots, a'^{\star}_N\right), \\ a'_i &= \boldsymbol{\mu}'_i\left(\mathbf{s}_{t+1}\right), \\ a'^{\star}{}_{j\neq i} &= \arg\min_{a'_{j\neq i}} Q_i^{\boldsymbol{\mu}'}\left(\mathbf{s}_{t+1}, a'_1, \ldots, a'_N\right), \end{aligned} \tag{3}$$

respectively. In (2) and (3), $\mathbf{s}_t$ is the concatenation of states for all agents at round t, $a_{t,i}(i \in [N])$ is the action of agent i at round t, $\boldsymbol{\mu}$ is the policy of all agents, $\boldsymbol{\mu}_i(\mathbf{s_t})$ is the i-th component of $\boldsymbol{\mu}(\mathbf{s_t})$, and $\boldsymbol{\mu}'$ is the target policy network of M3DDPG [18].

To derive $a^0_{j\neq i}$ and $a'^{\star}{}_{j\neq i}$ for each agent $i \in [N]$, we adopt the end-to-end solution that approximates the Q function by a locally linear function and replaces the inner-loop minimization in (2) and (3) with a 1-step gradient descent. Below is the calculation procedure of $a^0_{j\neq i}$ and $a'^{\star}{}_{j\neq i}$:

$$\begin{aligned} a_k &= \boldsymbol{\mu}_k\left(\mathbf{s}_t\right), \quad \forall 1 \leq k \leq N, \\ a^0_j &= a_j + \epsilon^0_j, \quad \forall j \neq i, \\ \epsilon^0_{j\neq i} &= \arg\min_{\epsilon^0_{j\neq i}} Q_i^{\boldsymbol{\mu}}\left(\mathbf{s}_t, a_1 + \epsilon^0_1, \ldots, a_i, \ldots, a_N + \epsilon^0_N\right) \\ &\approx -\alpha_j \nabla_{a_j} Q_i^{\boldsymbol{\mu}}\left(\mathbf{s}_t, a_1, \ldots, a_N\right). \\ a'_k &= \boldsymbol{\mu}'_k\left(\mathbf{s}_{t+1}\right), \quad \forall 1 \leq k \leq N, \\ a'^{\star}_j &= a'_j + \epsilon'_j, \quad \forall j \neq i, \\ \epsilon'_{j\neq i} &= \arg\min_{\epsilon'_{j\neq i}} Q_i^{\boldsymbol{\mu}'}\left(\mathbf{s}_{t+1}, a'_1 + \epsilon'_1, \ldots, a'_i, \ldots, a'_N + \epsilon'_N\right) \\ &\approx -\alpha_j \nabla_{a'_j} Q_i^{\boldsymbol{\mu}'}\left(\mathbf{s}_{t+1}, a'_1, \ldots, a'_N\right). \end{aligned} \tag{4}$$

After obtaining the policy loss, i.e., $-\min_{a^0_{j\neq i}} Q_i^{\boldsymbol{\mu}}(\mathbf{s}_t, a^0_1, \ldots, a_i, \ldots, a^0_N\,|_{a_i=\boldsymbol{\mu}_i(\mathbf{s}_t)})$, and the critic loss $\mathcal{L}_{t,i}$ for each $i \in [N]$, we linearly combine the policy losses for all agents and the critic losses for all agents, respectively, with the combination coefficients described in the below credit assignment part.

Delayed aggregation graph policy network. To model the dynamic and sparse communications between users and make a comprehensive decision, we adopt the delayed aggregation graph neural network (DGNN) [33] as the policy network of MARL. This method was never used in recommendation systems and we novelly integrate it into our elaborately designed MA-DGNN which results in great experimental performances on various datasets. Thanks to different levels of information diffusion along the multi-hop neighbors in DGNN, the relationship between agents with sparse interactions might be well inferred from the flowing information among other agents with varying distances. Below, we introduce DGNN in detail.

Aggregation GNNs [7] are information processing frameworks which operate on network data in a decentralized manner, utilizing useful information through repeated exchanges with neighbors. Different from normal aggregation GNNs that deal with fixed graph signals defined over fixed graph support, Tolstaya et al. [33] extend them to tackle time-varying graph processes set up over time-varying graph support.

Use $(i,j) \in \mathcal{E}_n$ to represent that j might send data to i at time n. With these denotations, Tolstaya et al. [33] then describe the graph support G_n by a graph shift operator $S_n \in \mathbb{R}^{N\times N}$ where $[S_n]_{ij}$ might not be zero only if $(j,i) \in \mathcal{E}_n$ or if $i=j$, indicating the sparsity of the graph. More specifically, denote $\left[x_{(n-1)}\right]_j = x^{\top}_{j(n-1)}$ to be the state of node j at time $n-1$, then through the property that $[S_n]_{ij}$ might not be zero only if $(j,i) \in \mathcal{E}_n$ or if $i=j$, we obtain:

$$[S_n x_{n-1}]_i = \sum_{j=i, j\in\mathcal{N}_{in}} [S_n]_{ij} \left[x_{n-1}\right]_j, \mathcal{N}_{\text{in}} = \{j : (i,j) \in \mathcal{E}_n\}. \tag{5}$$

$$\mathcal{N}^k_{in} = \left\{ j' \in \mathcal{N}^{k-1}_{j(n-1)} : j \in \mathcal{N}_{in} \right\}, \mathcal{H}_{in} = \bigcup_{k=0}^{K-1} \left\{ x_{j(n-k)} : j \in \mathcal{N}^k_{in} \right\}. \tag{6}$$

Based on the locality of (5), delayed aggregation GNNs build a sequence of recursive k-hop neighborhood aggregations as shown in (6), (7), and (8). Specifically speaking, define a sequence of signals $y_{kn} \in \mathbb{R}^{N\times p}$ with $y_{0n} = x_n$ and

$$y_{kn} = S_n y_{(k-1)(n-1)}. \tag{7}$$

Then it holds that $y_{kn} = (S_n S_{n-1} \dots S_{n-k+1})\, x_{n-k}$. Therefore, (7) is characterizing the diffusion of x_{n-k} through the series of time varying networks from S_{n-k+1} to S_n. Next, aggregate y_{kn} for $k \in \{0, 1, \cdots, K-1\}$ to form the nested states along the multi-hop neighbors:

$$\mathcal{Z}_{in} = \left[[y_{0n}]_i ; [y_{1n}]_i ; \dots ; \left[y_{(K-1)n}\right]_i \right], \tag{8}$$

where the $k+1$-st element of z_{in}, $[y_{kn}]_i = [S_n S_{n-1} \dots S_{n-k+1} x_{n-k}]_i$, is an average of $x_{j(n-k)}$ of k-hop neighbors $j \in \mathcal{N}^k_{in}$ at time $n-k$.

Specifying that z_{in} has a regular temporal structure due to its nested aggregation property, Tolstaya et al. [33] model z_{in} by a convolutional neural network with length L, that is, for $l \in [L]$, set

$$\mathcal{Z}^{(\ell)}_{in} = \sigma^{(\ell)} \left(\mathbf{H}^{(\ell)} \mathcal{Z}^{(\ell-1)}_{in} \right) \tag{9}$$

where $\sigma^{(\ell)}$ is a nonlinearity operator and $\mathbf{H}^{(\ell)}$ is a bank of small-support filters shared across all nodes with learnable parameters.

To sum up, the delayed aggregation GNN architecture, characterized by (5)-(9), comprises a local parameterization of the policy $\pi(\mathcal{H}_{in}, \mathbf{H})$ which captures the dynamic and sparse interactions in a network and allows for distant communications through the multi-hop information diffusion.

Customized experience replay. To (i) absorb experience from users with larger similarity to other users and better decision quality for other users, and (ii) attach greater importance to the newer samples and the rarer positive samples, during the sampling process in MARL's replay buffer, we are the first one to take all of the following factors into account: (i) the time when a sample occurs; (ii) whether the user feedback is positive or negative; (iii) the times that the target user in a sample mentions a person; (iv) the times that the target user is mentioned by other users; (v) the averaged performance that the target user assists on other users' decisions[6]; and (vi) the averaged similarity between the target user's state and other users' states. We calculate the aforementioned six factors $z^0_{i,1}, z^0_{i,2}, \cdots, z^0_{i,6}$ for each sample $i \in [N_{buffer}]$. With regard to each factor $j \in [6]$, we perform the softmax function on $z^0_{1,j}, z^0_{2,j}, \cdots, z^0_{N_{buffer},j}$ and obtain $z_{1,j}, z_{2,j}, \cdots, z_{N_{buffer},j}$. Then for each sampling process, for the i's

[6]Assume user i is the target user of a certain sample in replay buffer, and j is the time when the sample occurs. Then, the averaged performance that the target user assists on other users' decisions is characterized by $(\sum_{k=1}^{j} r_{k,i})/j$.

sample ($i \in [N_{buffer}]$) in the replay buffer, the corresponding sampling probability is $\frac{\sum_{j=1}^{6} z_{i,j}}{6}$. Note that to better approximate most of the actions and the state-action pairs, it is necessary to update MA-DGNN with a rich variety of samples. However, since samples provided by the above method might demonstrate a strong tendency regarding the six indexes, the diversity of the training samples could be far from enough. Therefore, to boost the diversity of experience replay, we take the above sampling method for a part of the training samples and the other part is obtained by state clustering. Specifically, we cluster the samples in the buffer into 10 groups according to their state and select samples from each group uniformly at random with an equal amount. Furthermore, during the sampling process, we novelly remove a certain amount of samples according to the degree of samples' non-stationarity. Specifically, we measure the standard error of the true labels for samples in each group and obtain the averaged value, $std_{avg,t}$, of the 10 standard errors. If $std_{avg,t}$ is larger than 80% of the historical $\{std_{avg,i}\}_{i\in[t-1]}$, then we judge that there might exist obvious non-stationarity and thus remove the oldest $\min\{100, std_{avg,t}/100\}$ percent of samples to keep track of the latest preferences of users.

Credit assignment. Recall that $\mathbf{s}_t$ is the concatenation of states for all agents at round t, $a_{t,i}(i \in [N])$ is the action of agent i, and $\boldsymbol{\mu}$ is the policy of all agents. For each agent $i \in [N]$, the policy loss is $Q_i^{\boldsymbol{\mu}}(\mathbf{s}_t, a_t) \ln \pi(a_{t,i}|\mathbf{s}_t)$, where π is the probability of obtaining $a_{t,i}$ under state $\mathbf{s}_t$ and policy $\boldsymbol{\mu}$. In the critic network and the policy network, we assign different weights to the critic loss and the policy loss of different agents and sum the losses to perform unified back-propagation. The weight coefficients are the elements of the i-th row of the user-user utility matrix, just as stated below.

3.2.3 Design and update of user-user utility matrix. To make a comprehensive decision and perform the credit assignment in MARL, we construct a user-user utility matrix to obtain the weights of linear combinations. Since there are N^2 parameters in the utility matrix which require much time to be correctly approximated, we assume the utility matrix can be factorized into two small matrices, A and B, with dimensions $N \times d$ and $d \times N$, respectively, where d is a small number relative to N. To estimate the two matrices, the neural matrix factorization method is utilized with its loss composed of the following two parts:

- $-[\frac{p_t+1}{n_t+1} G_t \log(o_t)) + (1 - G_t) \log(1 - o_t))]$, where $o_t \in [0, 1]$ is the linearly weighted output of the graph-based MARL algorithm. Here, p_t and n_t serve to alleviate the issue of the seriously imbalanced data.
- $\frac{\mu}{2} \left(\Sigma_{ij} s_{ij} \left\| a_i - a_j \right\|_2^2 + \Sigma_{ij} s_{ij} \left\| b_i - b_j \right\|_2^2 \right)$, where s_{ij} is the similarity between user i's state and user j's state, a_i is the i-th row of the A matrix, and b_i is the i-th column of the B matrix. We use this term to encourage that the more similar between user i's state and user j's state, the more similar between user i's relationship to others and user j's relationship to others.

Note that although there might exist a great deal of promising matrix factorization methods [1, 1, 24**? ?**] and other algorithms appropriate to update the utility matrix, our MA-DGNN with the above neural matrix factorization mechanism has already achieved remarkable performances in extensive experiments. Here we leave the study of other approaches that optimize the utility matrix as the future work.

4 DISCUSSION

In this section, we illustrate several aspects that need to be noted for our proposed MA-DGNN algorithm.

- **Q:** Why do not we adopt the Gumbel-softmax sampling trick for DDPG to conduct discrete actions?
  A: The difference between our method and the Gumbel-softmax sampling trick is that we compare the aggregated output with coefficients derived by neural matrix factorization with a sliding-window threshold instead of performing sampling according to each value output by each agent. For our MA-DGNN, if we adopt Gumbel-softmax sampling, then the neural matrix factorization technique could not be implemented due to the lack of the loss derived by the aggregated outputs and thus the credit assignment mechanism

also could not be carried out, which significantly degrades MA-DGNN's performance in the experimental evaluation. Therefore, we do not consider the Gumbel-softmax sampling trick.

- **Q:** What are the motivation and advantages of each component in MA-DGNN?
  A: (1) **DGNN:** The preferences of each user are dynamic, so are the interactions between users, which are very similar to the phenomenon of population evolution in a complex network. Based on this, we associate the DGNN (delayed aggregation graph neural network) algorithm, which is very suitable for modeling the population evolution of complex networks. DGNN is expected to capture the changes of each individual and of the relationships between individuals adaptively. Also, it is hoped that by using DGNN, the decisions of individuals with sparse interactions can be derived from individuals with richer interaction information.
  (2) **MADDPG:** When using a centralized evaluation function, the decision mechanism for each intelligence is influenced not only by the external dynamic environment but also by the changing decision mechanisms for other intelligence, which may make the decision updating process very unstable. In order to make the training process as stable and high-quality as possible, MADDPG assumes that all other intelligence make the worst decisions when optimizing each intelligence and expects the current intelligence to optimize its own decisions to the best under such a worst-case scenario. In summary, MADDPG is expected to help the algorithm keep optimizing in a good quality action space stably, rather than accidentally fall into a bad action space that is hard to get out of.
  (3) **The customized replay buffer:** Sample selection in MARL has a significant impact on the effectiveness of the algorithm. The time of appearance of samples, the positivity or negativity samples, the number of historical appearances of corresponding users, the similarity between users, and the quality of each user's assisted decision-making for other users are all factors that should be taken into account when performing sample selection. In this regard, we develop a prioritized experience replay mechanism based on six factors to select suitable samples for training. In addition, we also perform sample selection based on state clustering to promote sample diversity. Besides these, we also measure the non-stationarity of the recent environment based on the concentration of the sample pools in the same class and eliminate a certain amount of old samples according to the degree of non-stationarity. The carefully modulated sampling strategy results in real-time, high-quality policy updates, and we can see the clear advantages of this mechanism in ablation learning.
  (4) **Neural matrix factorization:** To tackle the credit assignment in MARL, we construct a user-user utility matrix and update it in real-time with a supervised learning algorithm. Due to the sparse information of some user-user interactions, it is difficult to accurately estimate the corresponding elements of the utility matrix using a normal deep neural network. In this regard, we adopt the neural matrix factorization that can cope well with the problem of missing data by learning hidden factors, and develop a loss function for mitigating the positive and negative sample imbalance while exploiting the similarity of user states to train neural matrix factorization, and we can see the great improvement of this technique from the latter-stated ablation study section.

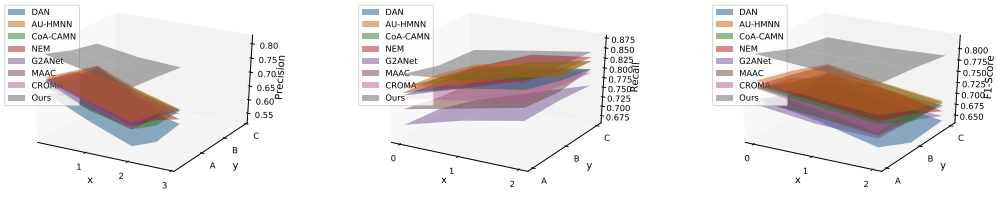

Fig. 3. Comparison results on the 9 synthetic datasets with $\mathcal{K}$=3.

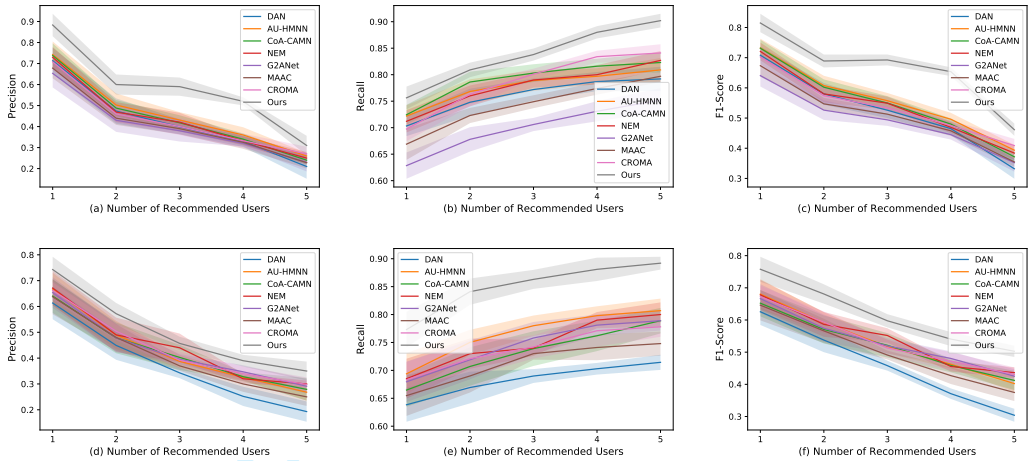

Fig. 4. Precision, recall, and F1-Score in different number of recommended users. (a)(b)(c): performances on the Twitter dataset; (d)(e)(f): performances on the Sina Weibo dataset. Algorithms are run for 3 times.

Table 2. Overall performance on the Twitter dataset. We repeat each experiment 3 times and report the average scores and standard errors (displayed in the brackets). The best results are given in bold. The higher the metric, the better performance of the model.

Model	Twitter					
	Prec.	Recall	F1.	MRR	Hits@3	Hits@5
DAN	0.7237 (0.0441)	0.6231 (0.0182)	0.6696 (0.0319)	0.6402 (0.0284)	0.7103 (0.0298)	0.6287 (0.0281)
AU-HMNN	0.7482 (0.0458)	0.6721 (0.0197)	0.7081 (0.0341)	0.6948 (0.0311)	0.7354 (0.0339)	0.6842 (0.0302)
CoA-CAMN	0.7383 (0.402)	0.6449 (0.0193)	0.6893 (0.0327)	0.6695 (0.0259)	0.7346 (0.0291)	0.6541 (0.0251)
NEM	0.7314 (0.0386)	0.6756 (0.0207)	0.7024 (0.0305)	0.6894 (0.0298)	0.7233 (0.0327)	0.6803 (0.0247)
G2ANet	0.6533 (0.0442)	0.6539 (0.0248)	0.6284 (0.0327)	0.6795 (0.0334)	0.6406 (0.0356)	0.6665 (0.0292)
MAAC	0.6781 (0.0476)	0.6376 (0.0297)	6689 (0.0385)	0.6548 (0.0304)	0.6735 (0.0374)	0.6461 (0.0312)
CROMA	0.7080 (0.0437)	0.6621 (0.0228)	0.6974 (0.0336)	0.6832 (0.0314)	0.7027 (0.0355)	0.6725 (0.0261)
MA-DGNN	**0.8862** (0.0397)	**0.7464** (0.0209)	**0.8103** (0.0318)	**0.7749** (0.0294)	**0.8179** (0.0308)	**0.7592** (0.0254)

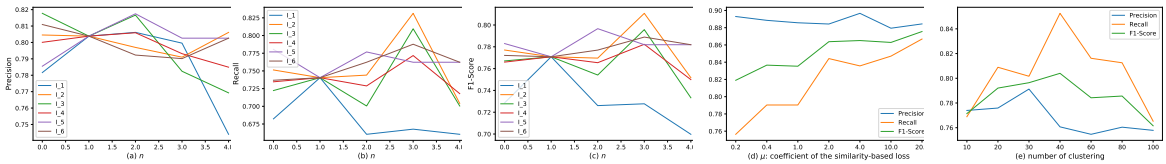

Fig. 5. Hyper-parameter analysis on the Twitter dataset. (a)(b)(c): analysis of six indexes in replay buffer; (d) analysis of μ, the coefficient of the similarity-based loss in neural matrix factorization; (e) analysis of the number of sample clusters in replay buffer.

5 EXPERIMENTS

In this section, we conduct extensive experiments on both the real-world and synthetic datasets to answer the following questions:

Table 3. Overall performance on the Sina Weibo dataset.

Model	Sina Weibo					
	Prec.	Recall	F1.	MRR	Hits@3	Hits@5
DAN	0.7749 (0.0498)	0.6502 (0.0312)	0.7071 (0.0428)	0.6443 (0.0391)	0.8341 (0.0418)	0.6551 (0.0301)
AU-HMNN	0.8012 (0.0547)	0.7002 (0.0384)	0.7473 (0.0458)	0.6975 (0.0429)	0.9116 (0.0428)	0.7147 (0.0312)
CoA-CAMN	0.7881 (0.0479)	0.6751 (0.0301)	0.7272 (0.0392)	0.6702 (0.0433)	0.8703 (0.0402)	0.6829 (0.0274)
NEM	0.7803 (0.0519)	0.6875 (0.0349)	0.7310 (0.0472)	0.6987 (0.0445)	0.8291 (0.0462)	0.7263 (0.0329)
G2ANet	0.7329 (0.0581)	0.6705 (0.0362)	0.7835 (0.0481)	0.6806 (0.0468)	0.83.49 (0.0479)	0.7025 (0.0337)
MAAC	0.7532 (0.0558)	0.6634 (0.0364)	0.80.02 (0.0477)	0.6720 (0.0459)	0.8627 (0.0481)	0.6973 (0.0309)
CROMA	0.7838 (0.0476)	0.6728 (0.0317)	0.8269 (0.0401)	0.6809 (0.0424)	0.8783 (0.0417)	0.7023 (0.0316)
MA-DGNN	**0.8879** (0.0513)	**0.7853** (0.0313)	**0.8335** (0.0418)	**0.7851** (0.0421)	**0.9341** (0.0441)	**0.8195** (0.0308)

Table 4. Overall performance on the Twitter dataset for users with sparse interactions.

Model	Twitter					
	Prec.	Recall	F1.	MRR	Hits@3	Hits@5
LambdaMart	0.5218 (0.0435)	0.5031 (0.0385)	0.5122 (0.0402)	0.4894 (0.0463)	0.6649 (0.0441)	0.4984 (0.0369)
DAN	0.4828 (0.0387)	0.4595 (0.0394)	0.4708 (0.0389)	0.4398 (0.0384)	0.6702 (0.0405)	0.4452 (0.0406)
AU-HMNN	0.5084 (0.0401)	0.4792 (0.0413)	0.4933 (0.0409)	0.4423 (0.0428)	0.6996 (0.0416)	0.4608 (0.0387)
CoA-CAMN	0.4934 (0.0426)	0.4813 (0.0374)	0.4872 (0.0399)	0.4294 (0.0393)	0.6603 (0.0385)	0.4378 (0.0391)
NEM	0.4915 (0.0387)	0.4736 (0.0415)	0.4824 (0.0401)	0.4401 (0.0418)	0.6302 (0.0406)	0.4484 (0.0375)
G2ANet	0.4618 (0.0372)	0.4495 (0.0384)	0.4556 (0.0379)	0.4375 (0.0375)	0.6297 (0.0394)	0.4457 (0.0417)
MAAC	0.4814 (0.0421)	0.4602 (0.0398)	0.4706 (0.0403)	0.4473 (0.0369)	0.6583 (0.0371)	0.4552 (0.0433)
CROMA	0.5097 (0.0386)	0.4831 (0.0416)	0.4960 (0.0396)	0.4571 (0.0414)	0.6589 (0.0428)	0.4638 (0.0396)
Ours	0.6618 (0.0378)	0.6487 (0.0409)	0.6552 (0.0392)	0.5725 (0.0391)	0.7495 (0.0397)	0.5784 (0.0387)

Table 5. Overall performance on the Sina Weibo dataset for users with sparse interactions.

Model	Sina Weibo					
	Prec.	Recall	F1.	MRR	Hits@3	Hits@5
LambdaMart	0.5572 (0.0375)	0.5381 (0.0391)	0.5475 (0.0389)	0.4891 (0.0385)	0.7281 (0.0445)	0.5084 (0.0397)
DAN	0.5193 (0.0435)	0.4794 (0.0398)	0.4986 (0.0417)	0.4615 (0.0391)	0.6983 (0.0375)	0.4672 (0.0381)
AU-HMNN	0.5382 (0.0394)	0.4823 (0.0371)	0.5087 (0.0384)	0.4732 (0.0402)	0.7292 (0.0381)	0.4823 (0.0418)
CoA-CAMN	0.5294 (0.0403)	0.4685 (0.0348)	0.4971 (0.0381)	0.4489 (0.0375)	0.6827 (0.0416)	0.4509 (0.0394)
NEM	0.5271 (0.0391)	0.4703 (0.0361)	0.4971 (0.0386)	0.4632 (0.0329)	0.6494 (0.0427)	0.4639 (0.0402)
G2ANet	0.4895 (0.0324)	0.4617 (0.0413)	0.4752 (0.0409)	0.4583 (0.0423)	0.6582 (0.0395)	0.4595 (0.0379)
MAAC	0.4931 (0.0457)	0.4763 (0.0381)	0.4846 (0.0421)	0.4672 (0.0374)	0.6802 (0.0418)	0.5387 (0.0342)
CROMA	0.5374 (0.0413)	0.4894 (0.0362)	0.5123 (0.0392)	0.4826 (0.0387)	0.6855 (0.0373)	0.5491 (0.0371)
Ours	0.6707 (0.0394)	0.6182 (0.0328)	0.6434 (0.0371)	0.6094 (0.0342)	0.7507 (0.0379)	0.6561 (0.0421)

RQ1 How does MA-DGNN perform on various datasets for mention recommendation, compared with existing baselines?

RQ2 How do different configurations of hyper-parameters and different indexes in the prioritized replay buffer influence the overall performance of our proposed algorithm?

RQ3 What are the influences of each component in MA-DGNN, such as the credit assignment procedure, the mimimax optimization of the critic losses, etc.?

Table 6. Ablation study on the Twitter dataset.

Metric	Dataset	MA-DGNN	$thre_t = 0.5$	w/o CER	w/o SBL	w/o CRA	w/o MMO
F1-Score	Twitter	**0.8103**	0.6708	0.7506	0.7834	0.7681	0.7792
	Sina	**0.8335**	0.7129	0.7439	0.7902	0.7792	0.7875
Hit@3	Twitter	**0.8179**	0.6951	0.7428	0.7826	0.7606	0.7693
	Sina	**0.9341**	0.8017	0.8517	0.9021	0.8693	0.8838
Hit@5	Twitter	**0.6803**	0.6175	0.6189	0.6695	0.6417	0.6579
	Sina	**0.8195**	0.6982	0.7473	0.7928	0.7686	0.7802

5.1 Experiment Setup

5.1.1 Dataset. We evaluate our proposed MA-DGNN and the selected baseline models on both the real-world and synthetic datasets.

Real world datasets. We adopt two real datasets, the Twitter public dataset [**?**] and the crawled Sina Weibo dataset[7], from the two most popular social network platforms, Twitter and Sina Weibo. The statistics of the real-world datasets are shown in Table 7.

Table 7. The dataset statistics.

	Twitter	Sina Weibo	Synthetic
#Twitters / #Weibo	200, 465	56, 673	60, 000
#Images	200, 465	60, 652	0
#Users	15, 539	2, 000	2, 000
#Avg. Mention per User	39.45	36.23	41.77
#Avg. Mentioned per User	7.51	6.97	10.32

Synthetic datasets. As for the construction of synthetic datasets, for $i \in [N]$ and $t \in [T]$, we set the feature vector of user i at time t to be: $\theta_{i,t} = \frac{(cos(u_{i,1}t),\cdots,cos(u_{i,d}t))}{||(cos(u_{i,1}t),\cdots,cos(u_{i,d}t))||_2}$, where d is the dimension of the feature vectors and $\{u_{i,j}|i \in [N], j \in [d]\}$ are the constant coefficients in different components.

For each $t \in [T]$, we randomly generate a tweet tw_t for the currently served user i_t. The feature vector of tw_t is set to be $\frac{v_t}{||v||_t}$, where $v_{t,j} \sim Uniform([0, 1])$ for $\forall j \in [d]$. At this time, the user mentioned by user i_t is $\arg\max_{j\in[N]} \theta_{j,t}^{\top} tw_t - ||\theta_{j,t} - \theta_{i_t,t}||_2$, i.e., the more similar to user i_t a user is and the larger projection length on the vector tw_t a user owns, the more likely he/she is to be mentioned by user i_t at time t.

To demonstrate the robustness of our proposed MA-DGNN, we construct synthetic datasets with different degrees of non-stationarity and different distributions of users' appearing frequency. Specifically, with regard to users' varying preferences, i.e., the non-stationarity, we classify the synthetic datasets into 3 categories:

$$\begin{cases} (a) : category\ I : for\ i \in [N], j \in [d], u_{i,j} \sim Uniform([0, \frac{10}{3T}]); \\ (b) : category\ II : for\ i \in [N], j \in [d], u_{i,j} \sim Uniform([0, \frac{20}{3T}]); \\ (c) : category\ III : for\ i \in [N], j \in [d], u_{i,j} \sim Uniform([0, \frac{10}{T}]). \end{cases}$$

Regarding users' appearing frequency, we also classify the synthetic datasets into 3 categories: (a) category A: at each time $t \in [T]$, each user is uniformly randomly served, i.e., each user issues a tweet with the same probability

[7]We crawl the Sina Weibo according to [**?**] and process the crawled dataset through the same procedures as the literature [21].

at each round; (b) category B: users are served in a totally randomized way. (c) category C: we randomly select 10 percent of users to issue approximately 60 percent of tweets. And in the rest rounds, the rest users are served in a totally randomized way.

We cross-link categories I, II, III with categories A, B, C and thus construct 9 ($= 3 \times 3$) synthetic datasets.

5.1.2 Baselines. The existing mention recommendation models can be categorized into two groups: the supervised learning-based (SL-based) group and the multi-agent reinforcement learning-based (MARL-based) group. As shown in Table 2,3, we consider DAN [22], AU-HMNN [13], CoA-CAMN [21] and NEM [39] as representative SL-based baselines, and incorporate each of them with the actor-critic method [8] for fair comparison with our reinforcement learning approach. For MARL-based methods, we choose G2ANet [20], MAAC [15] and CROMA [9] as baselines. Below, we describe our selected baselines in brief:

(i) **DAN.** Dual-attention (DAN) is an attention method introduced in [22] which models the relationship between objects from different perspectives.

(ii) **AU-HMNN.** AU-HMNN is proposed in [13], which incorporates the textual information of query tweets and user histories. This is a state-of-the-art approach to the mention recommendation task.

(iii) **CoA-CAMN.** To use textual and visual information, Renfeng et al. [21] propose a novel cross-attention memory network to perform the mention recommendation task for multimodal tweets.

(iv) **NEM.** Yi et al. [39] construct a heterogeneous mention network according to different relationships among different entities, and then infer a unified low dimensional embedding for users and messages by taking the network structure and vertex content information into account.

(v) **G2ANet.** Yong et al. [20] model the relationship between agents by a complete graph and propose a novel game abstraction mechanism based on the integration of a two-stage attention network (G2ANet) with the multi-agent reinforcement learning training mechanism.

(vi) **MAAC.** MAAC [15] is an actor-critic algorithm that trains decentralized policies in multi-agent settings, using centrally computed critics that share an attention mechanism which selects relevant information for each agent at every timestep. This attention mechanism enables more effective and scalable learning in complex multi-agent environments when compared to recent approaches.

(vii) **CROMA.** CROMA [9] is a novel cooperative multi-agent approach to mention recommendation, which incorporates dozens of more historical tweets than earlier approaches.

(viii) **LambdaMart.** LambdaMart [2] is a Listwise-learning-to-rank algorithm that transforms the ranking problem into a regression decision tree problem. As a classic learning-to-rank method, it is still widely applied in the search engine and recommendation systems of major Internet companies.

5.1.3 Evaluation Metrics. Following previous works [9, 20, 39], we take the Precision, Recall, F1-score [29], and Mean Reciprocal Rank (MRR) [25] as evaluation metrics for the highest-ranked result. Furthermore, Hits@3 and Hits@5 [13] are reported to denote the percentage of correct results recommended from the top-$\mathcal{K}$ results. Higher values of these metrics indicate better performance of the mention recommendation model.

5.1.4 Implementation Details. In this work, we set the discounted factor γ of MDP to be 0.99, the length of *SeqConv*'s input N_{list} to be 10, the perturbations α_i $(i \in [N])$ during minimax optimization to be 10^{-5}, the number of hops in the decentralized actors to be 2, and the number of sample clusters in the customized replay buffer to be 20. The coefficient of the similarity-based loss (i.e., μ) in the neural matrix factorization component is set to be 0.4. Moreover, the learning rates for the actor network, the critic network, and the neural matrix factorization mechanism are set to be 10^{-5}, 10^{-4}, and 10^{-4}, respectively. Our model implemented on PyTorch is available for reviewers[9] and will be publicly available upon the acceptance of this work.

[8] We take the supervised method as the policy network and the critic network of actor-critic.

[9] Code: https://filedropper.com/d/s/dLEu015PoqARaoZOWSIgu9J3LKYbXL .

5.2 Overall Performance (RQ1)

We first perform each mention recommendation model on two real-world datasets, i.e., Twitter and Sina Weibo. The results are presented in Table 2,3, and Figure 4, from which we can obtain the following observations:

(i) MA-DGNN significantly outperforms all the baseline models, especially for the Precision metric. Such improvement validates the effectiveness of applying MARL with the delayed aggregation graph neural network, which ensures the personalized recommendation for each target user and the in-time adjustment for users' dynamic preferences.

(ii) As for the performance analysis of other baselines, SL-based methods generally outweigh MARL-based methods except MA-DGNN, which forms a vivid contrast to the superior performance of MA-DGNN and consistently demonstrates the effectiveness of our proposed components (DGNN, the customized priority replay buffer, and the minimax optimization technique) in MA-DGNN to improve over traditional MARL methods. Therefore, MA-DGNN enables a better MARL-policy learning process and gains significant performance improvement.

To further evaluate MA-DGNN and baselines under different degrees of non-stationarity and user-appearing frequency, we construct 9 synthetic datasets and report the Precision, Recall and F1-Score for each model. The results are shown in Figure 3, where our MA-DGNN shows absolute competence against other baselines in all 9 synthetic datasets, especially when the user appearance is non-uniform and the environment is highly non-stationary.

In Figure 4, we show the Precision, Recall, and F1-score of different algorithms with the number of recommended users ranging from one to five. From Figure 4, the performances of MA-DGNN rank the highest regarding all the three metrics in all real datasets. Especially for the recall metric, however we change the number of users between one and five, the average recall rate of our MA-DGNN is consistently above 0.75, which shows absolute competence compared with other baseline methods.

5.3 Sparse Users Analysis

To test the advantage of dealing with sparse users' interactions for MA-DGNN, on the Twitter and Sina Weibo datasets, we only select users whose Ag. Mention times is less than 20 percent of the mean times or users whose Ag. Mentioned times is less than 20 percent of the mean times. We train MA-DGNN and all baselines on the selected users and the performances are shown in Table 4 and 5. As can be seen in Table 4 and 5, our MA-DGNN still beats all other baselines whether on the Twitter dataset or the Sina Weibo dataset, which shows the strong capability of MA-DGNN to tackle with users with less interaction information.

5.4 Hyper-parameter analysis (RQ2)

In this section, we test the influence of different configurations of hyper-parameters on the overall performance in the Twitter dataset. Recall that μ is the coefficient of the similarity-based loss in our neural matrix factorization module and we define cn to be the number of sample clustering in the replay buffer.

(i) Set $cn = 20$ and $\mu = 0.2, 0.4, 1, 2, 4, 10, 20$. Results under different values of μ are shown in Figure 5 (d).

(ii) Set $\mu = 0.4$ and $cn = 10, 20, 30, 40, 60, 80, 100$. Results under different values of cn are shown in Figure 5 (e)[10].

From Figure 5 (d)(e), we can see that the overall performances of MA-DGNN are positively correlated to the coefficient of the similarity-based loss and applying an appropriate number of sample clusters in the replay buffer also improves MA-DGNN significantly

Next, we study the influence of different indexes in the prioritized replay buffer on the overall performance in detail. For each $m \in [6]$ and each $n \in [6]$, we first fix m and n, and then set the sampling probability for the i's sample to be $\frac{\sum_{j=1}^{6} z_{i,j}+(n-1)z_{i,m}}{6}$. By varying $m \in [6]$ and $n \in [6]$, we can obtain the performances of our proposed model under different proportions of the six indexes in the replay buffer (See Figure 5 (a)(b)(c) for the Twitter

[10]Although MA-DGNN is somewhat sensitive to cn, it has consistently great performances which outweigh the second best method in a wide range of cn in the Twitter dataset. Therefore, we do not need to overly worry about the setting of cn.

dataset and Figure 5 (a)(b)(c) for the Sina Weibo dataset with the number of recommended user set to be 1). From Figure 5, we can see that increasing the proportions of the indexes (ii)(v)(vi) could contribute to improving the performance regarding the recall and F1-score metrics. The performance with regard to Precision also does not hurt much when we pay more attention to indexes (ii)(v)(vi). These demonstrate the importance of whether the user feedback is positive or negative, the averaged performance that a user assists on other users' decisions, and the averaged similarity between a user's state and other users' states in boosting the prioritized experience replay mechanism.

5.5 Ablation study (RQ3)

For the ablation study, we test our method when $thre_t = 0.5$ and in the lack of the customized experience replay (CER) mechanism, the similarity-based loss in neural matrix factorization (SBL), the credit assignment procedure (CRA), and the mimimax optimization of the critic losses (MMO), separately, on both the Twitter dataset and the Sina Weibo dataset. As shown in Table 6, no matter for F1-score, hits@3, or hits@5, replacing $thre_t = \frac{n_t+1}{p_t+1}$ with $thre_t = 0.5$ and the lack of any component bring varying degrees of reduction in algorithm performance, which consistently demonstrates the Precision of algorithm design and the importance of tacit cooperation between different components. The obvious performance deterioration when replacing $thre_t = \frac{n_t+1}{p_t+1}$ with $thre_t = 0.5$ and when lacking the customized experience replay component specifically show the effectiveness and efficiency of our design of $thre_t$, and our selected six indexes and the non-stationary detection mechanism with state clustering in the replay buffer.

6 CONLUSION

Mention recommendation is an important tool in social networking platforms to boost users' experience. To deal with the dynamic and sparse interactions between users which are seldom considered by existing literature, we apply multi-agent reinforcement learning with customized prioritized experience replay to model users' varying relationship and employ a delayed aggregation graph neural network as the decentralized actor. A user-user utility matrix optimized by neural matrix factorization with a similarity-based loss is designed to help make a comprehensive decision and perform credit assignment. In the experimental evaluation, apart from two real-world datasets, we also construct various synthetic datasets with different degrees of non-stationarity and distributions of users' appearing frequency. Extensive experiments demonstrate the effectiveness of our design in characterizing users' relationship and our method which has expressive advantages over existing methods. In the future, we plan to improve the explainability of MA-DGNN and its application in more specific scenarios, such as scenarios where different users' social circles are extremely imbalanced.

REFERENCES

[1] Sanjeev Arora, Nadav Cohen, Wei Hu, and Yuping Luo. 2019. Implicit regularization in deep matrix factorization. *Advances in Neural Information Processing Systems* 32 (2019), 7413–7424.

[2] Christopher JC Burges. 2010. From ranknet to lambdarank to lambdamart: An overview. *Learning* 11, 23-581 (2010), 81.

[30]]crawl Lei Chen. [n. d.]. A tool for crawling Sina Weibo. https://github.com/dataabc/weiboSpider.

[4] Kevin Dabérius, Elvin Granat, and Patrik Karlsson. 2019. Deep Execution-Value and Policy Based Reinforcement Learning for Trading and Beating Market Benchmarks. *Available at SSRN 3374766* (2019).

[5] Zhaoyun Ding, Xueqing Zou, Yueyang Li, Su He, Jiajun Cheng, Fengcai Qiao, and Hui Wang. 2016. Mentioning the optimal users in the appropriate time on Twitter. In *Asia-Pacific Web Conference*. Springer, 464–468.

[6] Laura Fontanesi, Sebastian Gluth, Mikhail S Spektor, and Jörg Rieskamp. 2019. A reinforcement learning diffusion decision model for value-based decisions. *Psychonomic bulletin & review* 26, 4 (2019), 1099–1121.

[7] Fernando Gama, Antonio G Marques, Alejandro Ribeiro, and Geert Leus. 2019. Aggregation graph neural networks. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 4943–4947.

[30]]githubcroma Tao Gui, Peng Liu, Qi Zhang, Liang Zhu, Minlong Peng, Yunhua Zhou, and Xuanjing Huang. [n. d.]. Pytorch implementation of Mention Recommendation in Twitter with Cooperative Multi-Agent Reinforcement Learning. https://github.com/mritma/CROMA.

[9] Tao Gui, Peng Liu, Qi Zhang, Liang Zhu, Minlong Peng, Yunhua Zhou, and Xuanjing Huang. 2019. Mention recommendation in Twitter with cooperative multi-agent reinforcement learning. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 535–544.

[10] Taehyun Ha, Seunghee Han, Sangwon Lee, and Jang Hyun Kim. 2017. Reciprocal nature of social capital in Facebook: an analysis of tagging activity. *Online Inf. Rev.* 41, 6 (2017), 826–839.

[11] Huang Hu, Xianchao Wu, Bingfeng Luo, Chongyang Tao, Can Xu, Wei Wu, and Zhan Chen. 2018. Playing 20 question game with policy-based reinforcement learning. *arXiv preprint arXiv:1808.07645* (2018).

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

[13] Haoran Huang, Qi Zhang, Xuanjing Huang, Haoran Huang, Qi Zhang, and Xuanjing Huang. 2017. Mention Recommendation for Twitter with End-to-end Memory Network.. In *IJCAI*. 1872–1878.

[14] Jeff Huang, Katherine Thornton, and Efthimis N. Efthimiadis. 2010. Conversational tagging in twitter. In *HT'10, Proceedings of the 21st ACM Conference on Hypertext and Hypermedia, Toronto, Ontario, Canada, June 13-16, 2010*, Mark H. Chignell and Elaine G. Toms (Eds.). ACM, 173–178.

[15] Shariq Iqbal and Fei Sha. 2019. Actor-attention-critic for multi-agent reinforcement learning. In *International Conference on Machine Learning*. PMLR, 2961–2970.

[16] Bo Jiang, Ying Sha, and Lihong Wang. 2015. Predicting user mention behavior in social networks. In *Natural Language Processing and Chinese Computing*. Springer, 146–158.

[17] Quanle Li, Dandan Song, Lejian Liao, and Li Liu. 2015. Personalized mention probabilistic ranking–recommendation on mention behavior of heterogeneous social network. In *International conference on web-age information management*. Springer, 41–52.

[18] Shihui Li, Yi Wu, Xinyue Cui, Honghua Dong, Fei Fang, and Stuart Russell. 2019. Robust multi-agent reinforcement learning via minimax deep deterministic policy gradient. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 4213–4220.

[19] Feng Liu, Ruiming Tang, Xutao Li, Yunming Ye, Haokun Chen, Huifeng Guo, and Yuzhou Zhang. 2018. Deep Reinforcement Learning based Recommendation with Explicit User-Item Interactions Modeling. *CoRR* abs/1810.12027 (2018). arXiv:1810.12027 http://arxiv.org/abs/1810.12027

[20] Yong Liu, Weixun Wang, Yujing Hu, Jianye Hao, Xingguo Chen, and Yang Gao. 2020. Multi-agent game abstraction via graph attention neural network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 7211–7218.

[21] Renfeng Ma, Qi Zhang, Jiawen Wang, Lizhen Cui, and Xuanjing Huang. 2018. Mention recommendation for multimodal microblog with cross-attention memory network. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 195–204.

[22] Hyeonseob Nam, Jung-Woo Ha, and Jeonghee Kim. 2017. Dual attention networks for multimodal reasoning and matching. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 299–307.

[23] Soumajit Pramanik, Mohit Sharma, Maximilien Danisch, Qinna Wang, Jean-Loup Guillaume, and Bivas Mitra. 2019. Easy-Mention: a model-driven mention recommendation heuristic to boost your tweet popularity. *International Journal of Data Science and Analytics* 7, 2 (2019), 131–147.

[24] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. 2018. Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In *Proceedings of the eleventh ACM international conference on web search and data mining*. 459–467.

[25] Dragomir R Radev, Hong Qi, Harris Wu, and Weiguo Fan. 2002. Evaluating Web-based Question Answering Systems.. In *LREC*. Citeseer.

[26] Muhammad Rehan Raza, Carlos Natalino, Peter Öhlen, Lena Wosinska, and Paolo Monti. 2018. A slice admission policy based on reinforcement learning for a 5G flexible RAN. In *2018 European Conference on Optical Communication (ECOC)*. IEEE, 1–3.

[27] Saiph Savage, Andrés Monroy-Hernández, Kasturi Bhattacharjee, and Tobias Höllerer. 2015. Tag Me Maybe: Perceptions of Public Targeted Sharing on Facebook. *CoRR* abs/1509.01095 (2015). arXiv:1509.01095

[28] Mohit Sewak. 2019. Policy-Based Reinforcement Learning Approaches. In *Deep Reinforcement Learning*. Springer, 127–140.

[29] Guy Shani and Asela Gunawardana. 2011. Evaluating recommendation systems. In *Recommender systems handbook*. Springer, 257–297.

[30]]seqconv Shobrook. [n. d.]. SeqConv, a PyTorch implementation of a LSTM-based graph convolutional operator. https://pypi.org/project/seq-conv/.

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

[32] Liyang Tang, Zhiwei Ni, Hui Xiong, and Hengshu Zhu. 2015. Locating targets through mention in Twitter. *World Wide Web* 18, 4 (2015), 1019–1049.

[33] Ekaterina Tolstaya, Fernando Gama, James Paulos, George Pappas, Vijay Kumar, and Alejandro Ribeiro. 2020. Learning decentralized controllers for robot swarms with graph neural networks. In *Conference on Robot Learning*. PMLR, 671–682.

[34] Beidou Wang, Can Wang, Jiajun Bu, Chun Chen, Wei Vivian Zhang, Deng Cai, and Xiaofei He. 2013. Whom to mention: expand the diffusion of tweets by@ recommendation on micro-blogging systems. In *Proceedings of the 22nd international conference on World Wide Web*. 1331–1340.

[35] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 1225–1234.

[36] Kai Wang, Weiyi Meng, Shijun Li, and Sha Yang. 2019. Multi-Modal Mention Topic Model for mentionee recommendation. *Neurocomputing* 325 (2019), 190–199.

[37] Xuesong Wang, Yang Gu, Yuhu Cheng, Aiping Liu, and CL Philip Chen. 2019. Approximate policy-based accelerated deep reinforcement learning. *IEEE transactions on neural networks and learning systems* 31, 6 (2019), 1820–1830.

[38] Tengyu Xu and Yingbin Liang. 2021. Sample complexity bounds for two timescale value-based reinforcement learning algorithms. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 811–819.

[39] Feng Yi, Bo Jiang, and Jianjun Wu. 2020. Heterogeneous information network embedding for mention recommendation. *IEEE Access* 8 (2020), 91394–91404.

[40] Mengran Yu and Shiliang Sun. 2020. Policy-based reinforcement learning for time series anomaly detection. *Engineering Applications of Artificial Intelligence* 95 (2020), 103919.

[41] Xinshi Zang, Huaxiu Yao, Guanjie Zheng, Nan Xu, Kai Xu, and Zhenhui Li. 2020. Metalight: Value-based meta-reinforcement learning for traffic signal control. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 1153–1160.

[42] Xiangyu Zhao, Long Xia, Liang Zhang, Zhuoye Ding, Dawei Yin, and Jiliang Tang. 2018. Deep reinforcement learning for page-wise recommendations. In *Proceedings of the 12th ACM Conference on Recommender Systems*. 95–103.

[43] Xiangyu Zhao, Liang Zhang, Zhuoye Ding, Long Xia, Jiliang Tang, and Dawei Yin. 2018. Recommendations with negative feedback via pairwise deep reinforcement learning. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1040–1048.

[44] Guanjie Zheng, Fuzheng Zhang, Zihan Zheng, Yang Xiang, Nicholas Jing Yuan, Xing Xie, and Zhenhui Li. 2018. DRN: A deep reinforcement learning framework for news recommendation. In *Proceedings of the 2018 World Wide Web Conference*. 167–176.

[45] Ge Zhou, Lu Yu, Chu-Xu Zhang, Chuang Liu, Zi-Ke Zhang, and Jianlin Zhang. 2015. A novel approach for generating personalized mention list on micro-blogging system. In *2015 IEEE international conference on data mining workshop (ICDMW)*. IEEE, 1368–1374.