DENOISING-GUIDED DEEP REINFORCEMENT LEARNING FOR SOCIAL RECOMMENDATION*

Qihan Du, Li Yu†, Huiyuan Li, Youfang Leng, Ningrui Ou, Junyao Xiang

Renmin University of China, Beijing, China

ABSTRACT

Social recommendation (SR) aims to enhance the performance of recommendations by incorporating social information. However, such information is not always reliable, e.g., some of the friends may share similar preferences with the user on a specific item, while others may be irrelevant to this item due to domain differences. Therefore, modeling all of the user's social relationships without considering the relevance of friends will introduce noises to the social context. To address this issue, in this work, we propose a **D**enoising-guided deep **R**einforcement **L**earning framework **for So**cial recommendation (**DRL4So**). Specifically, the agent (i.e., social denoiser) in our framework automatically masks the user's friends who are irrelevant to the target item; Then, the environment (i.e., recommender) is designed to give rewards to the agent for social denoising without supervised signals; Finally, the two components are jointly trained by DPG to ensure that social denoising correctly guides the recommendation. We conduct extensive experiments on three public datasets, and the results show that DRL4So outperforms existing state-of-the-art SR methods (improving 38.67% and 19.81% in terms of HR@10 and NDCG@10, respectively).

***Index Terms*—** Recommender systems, Reinforcement learning, Social networks, Deep learning.

1. INTRODUCTION

As important online services, recommender systems have received much attention from academia and have been widely adopted in industry. Recently, the rapidly rising social networks have become prevalent in people's daily life [1]. For example, the users share videos to their friends on TikTok or shopping in a group with their friends on Taobao. This brings the idea of social recommendation (SR) [2] which seeks to exploit the available information (e.g., browsing or purchase history) from their friends to infer the users' preferences.

The key issue of SR is how to integrate different social relationships into user-item interaction scenarios to improve the accuracy of recommendations. Lots of conventional SR methods have achieved significant progresses on regularization matrix factorization with user social information [3, 4, 5], but they are limited by modeling social relationships off-line via batch learning without any interactions with the user. To distinguish the influences of different friends during interaction, attention-based methods such as social attention memory network (SAMN) [6] and social enhanced attentive model (SEAM) [1] could be utilized to assign attention coefficients to friends as their importances to the user's current preferences. Meanwhile, graph-based methods such as social graph convolutional networks (SGCN) [7] and hierarchical social graph models (HSGM) [8] could be used to encode connectivity among friends to propagate the user's current preferences.

However, despite the progresses of these efforts, they still suffer from the same drawback: they implicitly assume that all friends are helpful to the user's preferences on the target item. In fact, some friends may share similar preferences with the user on the target item, while others with different domains may be totally irrelevant for recommendation. For example, if the user wants to buy *dresses*, his/her male friends can hardly provide valid suggestions or even negatively affect the user's preferences. Thus, existing methods that involve rigidly assigning attention coefficients and encoding social graphs for all friends will introduce noises in the user's social context, disturbing the recommendation results.

Motivated by the above concern, it is necessary to automatically mask irrelevant friends (i.e., noises) to the target item before modeling social relationships. However, the core challenge of our idea is the lack of supervised signals about which friends are noises to the target item and should be masked. To this end, in this work, we propose a **D**enoising-guided deep **R**einforcement **L**earning framework **for So**cial recommendation (called **DRL4So**) to address it. Specifically, we formalize the noise masking process as a reinforcement learning (RL) task. The social denoiser serves as the agent to observe the user's preferences on the target item and perform actions to decide which friends need to be masked as noises due to the domain differences. The recommender serves as the environment to give rewards to the agent based on the differences in recommendation results before and after denoising without supervised signals. Finally, we adopt DPG [9] training to unify the agent and recommender to ensure a coordinated effort to enhance recommendation performance. Our

† Li Yu is the corresponding author.

*This work is supported by the Fundamental Research Funds for the Central Universities, and the Research Funds of Renmin University of China, grant number: 21XNH178.

contributions can be summarized as follows:

- We propose an RL-based framework to automatically mask noise friends to improve the performance of SR, allowing to perform social denoising without supervised signals. To the best of our knowledge, this is the first work to empower RL to perform social denoising for recommendation.
- Extensive experiments on three public datasets show the DRL4So outperforms existing state-of-the-art SR baseline models (improving 38.67% and 19.81% in terms of HR@10 and NDCG@10, respectively).

2. PROBLEM FORMULATION

Assume a recommender system with the userset U and the itemset V. The basic recommendation problem can be formalized as follows: given the historical items $[v_1, ..., v_t]$ that the user has clicked/liked before time t, we aim at recommending the items that the user most likely clicked/liked at time $t+1$. Let $F \in \mathbb{R}^{|U|\times|V|}$ denote the feedback matrix, where $F_{uv}=1$ means that the user u clicked or liked the item v, and $F_{uv}=0$ otherwise. Let $S \in \mathbb{R}^{|U|\times|U|}$ denote the adjacency matrix in social networks, where $S_{uu_i}=1$ means that there is a social relationship (e.g., friends) between uesr u and user u_i, and $S_{uu_i}=0$ otherwise. Let $N(u)=\{u_i; S_{uu_i}=1\}$ denote the set of friends of user u, and $|N(u)|=n$. The social denoising process we discussed can be modeled as a typical RL task, where the key elements are defined as follows:

- **State.** The state $s_j \in \mathbb{R}^d$ is a dense vector containing the user's current preferences and the target item's information.
- **Action.** The action $a_j \in \mathbb{R}^n$ is a multi-hot vector, where the i-the element $a_j^i = 0$ acts as a noise signal to mask the corresponding friend u_i in the set of friends $N(u)$.
- **Reward.** After the agent takes an action a_j at the state s_j, i.e., masking some noise friends to the target item v_j, the agent receives reward $R(s_j, a_j)$ from the environment.
- **Transition.** Once an action is executed and the reward is received, the agent can transfer to the next state s_{j+1}, and also determine the transition $\{s_j, a_j, R(s_j, a_j), s_{j+1}\}$.

3. PROPOSED FRAMEWORK: DRL4SO

The overall architecture of DRL4So is shown in Fig.1, which contains two major components: the agent (social denoiser) and the environment (recommender). In discrete time step t, the agent takes the historical items $[v_1, ..., v_t]$, the target item v_j and the original set of friends $N(u)$ as inputs, then outputs the denoised set of friends $\widetilde{N}(u)^j$ for item v_j; The environment gives a reward to the agent for denoising and feeds the next target item v_{j+1} from the candidate set into the agent; After the agent processed the last target item v_{j+m-1}, all items in $C(u)$ will be reranked based on the new recommended probability, and the item v_z with the highest new probability will be recommended to the user at time $t+1$. We further elaborate each component in the following.

To derive the available representations, we represent each item $v_i \in \mathcal{V}$ as a low-dimensional embedding vector $e_i \in \mathbb{R}^d$. For convenience, we describe the technical details of DRL4So for a single user u, and it is straightforward to extend the formulas to all users.

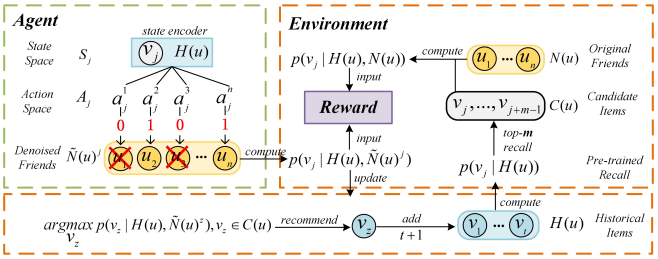

Fig. 1: The Architecture of DRL4So

3.1. Agent (Social Denoiser)

As we mentioned before, the agent aims to mask friends who are irrelevant to the target item.

(1) *State encoder*. We introduce the RNN with gated recurrent unit (GRU) [10] to learn the user's current preferences based on his/her interaction history. With the sequence of historical items vectors $[e_1, e_2, ..., e_t]$ as input, the GRU generates the corresponding hidden states $[h_1, h_2, ..., h_t]$, i.e., $h_t = GRU(h_{t-1}, e_t)$. A linear layer is utilized to merge the user's current preferences h_t with the target item vector e_j to produce the state representation:

$$s_j = W^{s1} h_t + W^{s2} e_j + \phi^s \tag{1}$$

where $W^{s1}, W^{s2} \in \mathbb{R}^{d\times d}, \phi^s \in \mathbb{R}^d$ are the pre-trained parameters, which will be detailed in Section 3.2.

(2) *Actor*. The Actor observes the current state and generates actions to decide which friends of the user are irrelevant to the target item that should be masked as noises. Given a friend $u_i \in N(u)$, and $h_t^{u_i}$ is the preference representation of him/her via the shared state encoder. We feed s_j and $h_t^{u_i}$ into a nonlinear layer, which acts as the policy function [11] to obtain the sub-action a_j^i:

$$a_j^i = \sigma(W^{a1} s_j + W^{a2} h_t^{u_i} + \phi^a) \tag{2}$$

where $\sigma(\cdot)$ is the sigmoid function, and $W^{a1}, W^{a2} \in \mathbb{R}^{n\times d}$, $\phi^a \in \mathbb{R}^n$ are the parameters of the actor network, and n is the size of the friend set. Note that $a_j^i \in \{0, 1\}$ is a scalar, we set $a_j^i = 0$ when $a_t^h \leqq 0.5$ means that friend u_i needs to be masked as noise, otherwise $a_j^i = 1$. Then, the n sub-actions are mapped to an n-dimensional action vector $a_j \in \mathbb{R}^n$ via:

$$a_j = multi\_hot(a_j^i, i, n), i \in [1, n] \tag{3}$$

where $multi\_hot(a_j^i, i, n)$ returns a n-dimensional multi-hot vector where the value of the i-th element is a_j^i.

(3) *Critic*. The Critic estimates the expected return of the action. We utilize a nonlinear layer as an approximator [12] for the action-value function as follows:

$$q(s_j, a_j) = ReLU(W^{q1} s_j + W^{q2} a_j + \phi^q) \tag{4}$$

where $W^{q1} \in \mathbb{R}^{d\times d}, W^{q2} \in \mathbb{R}^{d\times n}, \phi^q \in \mathbb{R}^d$ are the parameters of the critic network.

3.2. Environment (Recommender)

(1) *Pre-trained recall*. In discrete time step t, the environment first recalls m candidate items based on the user's individual preferences h_t to form $C(u)$, i.e., the parameters of the state encoder are pre-trained with the available training data, and jointly updated with the agent during denoising. The recall formula is as follows:

$$score_j = \sigma(e_j^\top \cdot h_t) \tag{5}$$

where the environment scores all items and selects the m items with the highest scores into the set of candidates $C(u)$.

(2) *Reward assignment mechanism*. After the denoised set of friends $\widetilde{N}(u)^j$ is output by the agent, the environment needs to give it a suitable reward. To evaluate the incremental social utility, we first adopt the attention mechanism to aggregate valid social information. The DOT product is applied to compute attention coefficient between the user u and the friend u_i: $q[h_t, h_t^{u_i}] = (h_t)^\top \cdot h_t^{u_i}$. After that, the coefficient of each friend in the denoised set $\widetilde{N}(u)^j$ are normalized by:

$$\alpha_{uu'} = \frac{exp(q[h_t, h_t^{u_i}])}{\sum_{w\in\widetilde{N}(u)^j} exp(q[h_t, h_t^w])} \tag{6}$$

Then, the denoised social context $\widetilde{c}_j \in \mathbb{R}^d$ is as follows:

$$\widetilde{c}_j = \sum_{u_i\in\widetilde{N}(u)^j} \alpha_{uu_i} h_t^{u_i} \tag{7}$$

We further leverage a fusion layer to combine the user's individual preferences and the denoised social context to produce a fused preference representation $\widetilde{f}_j \in \mathbb{R}^d$ of the user:

$$\widetilde{f}_j = \lambda\widetilde{c}_j + (1-\lambda)h_t \tag{8}$$

where $\lambda \in [0,1]$ is an adaptive parameter to tune the balance between two parts. In the aforementioned, in order to ensure that the agent guides the recommendation correctly, we set the reward function according to the increment on the recommended probability of the target item v_j before and after social denoising:

$$\begin{aligned} R(s_j, a_j) =& p(v_j|H(u), \widetilde{N}(u)^j) - p(v_j|H(u), N(u)) \\ =& \sigma(e_j^\top \cdot \widetilde{f}_j) - \sigma(e_j^\top \cdot f_j) \end{aligned} \tag{9}$$

where f_j and $\widetilde{f}_j$ are the fused preference representations of the user before and after denoising, respectively. A positive difference means that the denoising gains positive utility [13, 14]. Meanwhile, as shown in Fig.1, the recommended probability of the target item v_j is updated from $p(v_j|H(u), N(u))$ to $p(v_j|H(u), \widetilde{N}(u)^j)$. After all the items in $C(u)$ were updated, the environment selects the item v_z with the highest new probability $p(v_z|H(u), \widetilde{N}(u)^z)$ and recommends it to the user at time step $t+1$.

3.3. Training Methodology

In the above framework, our DRL4So adopts the Actor-Critic architecture. Since the policy in DRL4So is deterministic, we can utilize DPG [9] to train the parameters of the actor Θ=$\{W^{a1}, W^{a2}, \phi^a\}$ and critic Ψ=$\{W^{q1}, W^{q2}, \phi^q\}$. Specifically, we first sample a mini-batch of N transitions like $\{s_j, a_j, R(s_j, a_j), s_{j+1}\}$. Then, we update the critic network by minimizing the loss as follows:

$$L = 1/N \sum\nolimits_j (y_j - q(s_j, a_j; \Psi))^2 \tag{10}$$

where $y_j = R(s_j, a_j) + \gamma q(s_{j+1}, \mu(s_{j+1}; \Theta); \Psi)$ is the TD-target [15], and γ is the discount factor. The actor network is updated using the sampled policy gradient as follows:

$$\nabla_\Theta J = 1/N \sum\nolimits_j (\nabla_\Theta \mu(s_j; \Theta) \nabla_{a_j} q(s_j, a_j; \Psi)|_{a_j=\mu(s_j;\Theta)}) \tag{11}$$

Finally, the gradient ascent is performed to update the parameters of actor and critic networks respectively : $\Theta \leftarrow \Theta + \Delta\Theta$; $\Psi \leftarrow \Psi + \Delta\Psi$.

4. EXPERIMENT AND ANALYSIS

4.1. Experimental Settings

Datasets. We conduct experiments on three public datasets: LastFM [16], Ciao [17] and Epinions [18]. All datasets have feedback matrix R and social adjacency matrix S. Similar to [19], we convert all observed feedbacks to 1 and remove users and items whose feedback is less than 5. The statistics of three datasets are shown in Table 1.

Table 1: Statistics for LastFM, Ciao and Epinions datasets

Dataset	#users	#items	#feedbacks	#social links
LastFM	1,874	2,828	71,411	25,174
Ciao	7,260	11,166	147,799	110,715
Epinions	23,137	23,585	461,982	372,205

Evaluation Metrics. We evaluate the performance of models with the metrics: **HR@K** (HitRatio) and **NDCG@K** (Normalized Discounted Cumulative Gain). To ensure generality, we set K as a fixed number 5 or 10 for testing [20].

Implementation Details. We randomly divided each dataset into two parts, with 10% for testing and the rest for training. Note that the dimension of the action vector (i.e., the size of the friend set) is n. We empirically set n to fixed values 15,15 and 20 in the three datasets, while truncation and padding [21] are applied to process unsuitable sets. The other hyper-parameters are defined as follows: we set the vector dimension d=128, the discount factor γ=0.95, and the learning rate of actor and critic networks are 0.01 and 0.02, respectively. Our experimenst were completed on NVIDIA RTX3090 GPU with 24GB memory.

4.2. Comparison to Baselines

We compare DRL4So with various state-of-the-art SR baselines, including two RL-based attentive methods SADQN++

Table 2: Overall performance comparison. The best and second results are bolded and underlined, respectively.

Datasets	Model	HR@K(%)		NDCG@K(%)	
		@5	@10	@5	@10
LastFM	POP	3.92	4.35	2.43	3.14
	SoRec	25.21	27.35	19.93	22.27
	SAMN	35.58	38.92	29.44	33.21
	SEAM	37.63	40.17	30.85	35.07
	SGCN	36.86	37.33	29.92	33.81
	HSGM	38.26	40.41	32.06	34.54
	SADQN	42.93	45.17	36.42	39.35
	SADQN++	44.11	46.68	38.34	40.68
	DRL4So-	35.73	37.24	30.06	32.17
	DRL4So	**50.71**	**54.73**	**43.63**	**48.74**
Ciao	POP	3.83	4.06	2.76	3.18
	SoRec	20.79	22.83	15.39	17.01
	SAMN	31.91	32.14	26.57	28.32
	SEAM	32.63	33.26	26.23	28.20
	SGCN	32.51	34.85	25.86	27.89
	HSGM	33.14	34.92	27.11	28.49
	SADQN	41.63	42.11	32.03	34.24
	SADQN++	43.16	45.32	36.28	37.59
	DRL4So-	31.73	33.02	26.51	27.98
	DRL4So	**46.29**	**47.96**	**39.59**	**40.91**
Epinions	POP	3.31	4.22	1.98	2.74
	SoRec	28.03	30.97	21.15	24.81
	SAMN	33.21	35.06	25.58	28.24
	SEAM	33.14	34.03	24.89	28.12
	SGCN	34.46	35.38	24.77	27.79
	HSGM	34.85	36.21	25.62	27.49
	SADQN	42.18	44.33	36.27	38.69
	SADQN++	45.35	46.09	39.41	41.05
	DRL4So-	32.14	34.58	24.21	27.37
	DRL4So	**49.54**	**50.13**	**42.65**	**44.21**

[12] and SADQN [19], two graph-based methods HSGM [8] and SGCN [7], two attention-based methods SEAM [1] and SAMN [6], two conventional methods SoRec[5] and POP. All hyper-parameters of all the baselines are carefully chosen by grid search. From Table 2 we can observe that DRL4So outperforms the baselines on all datasets and metrics, showing the necessity of social denoising in SR. Morever, graph-based and attention-based methods show competitive performance, but both are limited by the introduction of noises when modeling social information; Although RL-based attentive methods do not consider denoising either, they are able to explore user preferences better than non-RL methods benefiting from the ϵ-greedy policy. Finally, the conventional methods significantly inferior to other attentive methods, proving that the attention mechanism does help to model social relationships.

4.3. Model Analysis

Q1: Does social denoiser really helpful?

A1: We designed a variant of DRL4So called DRL4So- to answer this question. In the variant, we remove the social denoiser and retain only the recommender to continuously interact with the user. In Table 2, we find that the performance of DRL4So- decreases disastrously (i.e., degrading to an attention-based model). This ablation study can support the argument that the SOTA performance of DRL4So benefits mainly from the social denoiser.

Table 3: Comparison for different sizes m of the candidate set on HR@10, the best result is bolded.

Size m	LastFM	Ciao	Epinions
m=100	0.5431	0.4620	0.4815
m=200	**0.5473**	0.4648	0.4827
m=500	0.5347	**0.4796**	0.4962
m=1000	0.5218	0.4713	**0.5013**

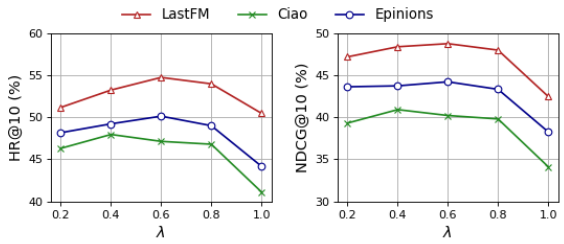

Fig. 2: Impact of λ on the performance of DRL4So.

Q2: What is the effect of the size of candidate set $C(u)$?

A2: The candidate set we designed can fix the maximum episode of the agent to m, and reduce the complexity of the state and action space. We manually set several values $\{100,200,500,1000\}$ to evaluate the effect of m on the final recommendation performance. As shown in Table 3, we find that the optimal size of the candidate set increases as the larger datasets, and the same trend is found on NDCG.

Q3: What is the effect of hyper-parameter λ?

A3: According to the Eq.(8), the λ tunes the relative weight of the user's individual preferences and the social attentive context. The lower λ makes the model focused more on the user's personalized preferences, while the higher λ emphasizes the influence of social networks. In Fig.2, we plot the performance of DRL4So when varing λ. We note that the curve drops significantly when $\lambda = 1$, since this case completely ignores the user's personalized preferences. Generally, the range of $\lambda \in [0.2, 0.6]$ tends to do relatively well in most scenarios, which indicates that this balance is useful.

5. CONCLUSION

In this work, we propose a denoising-guided deep reinforcement learning framework for social recommendation, called DRL4So. Specifically, the social denoiser acts as the agent to automatically mask the noise friends, and the recommender acts as the environment to give rewards to the agent for denoising and refining the recommended probability of items. To the best of our knowledge, this is the first work to empower RL to perform social denoising for recommendation. In the future, we plan to design a social-enhanced environment simulator to extend our work for further improvement.

6. REFERENCES

[1] D Cao, X.N. He, L.H. Miao, G.Y. Xiao, H Chen, and J Xu, "Social-enhanced attentive group recommendation," *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, vol. 33, no. 3, pp. 1195–1209, 2021.

[2] X. Wang, S.C. Hoi, C. Liu, and M. Ester, "Interactive social recommendation," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management (CIKM)*, 2017, pp. 357–366.

[3] B Yang, Y Lei, J.M. Liu, and W.J. Li, "Social collaborative filtering by trust," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 39, no. 8, pp. 1633–1647, 2017.

[4] H Ma, D.Y. Zhou, C Liu, M.R. Lyu, and I King, "Recommender systems with social regularization," in *Proceedings of the 4th ACM international conference on Web Search and Data Mining (WSDM)*, 2011, pp. 287–296.

[5] H Ma, H.X. Yang, M.R. Lyu, and I King, "Sorec: social recommendation using probabilistic matrix factorization," in *Proceedings of the 17th ACM Conference on Information and Knowledge Management (CIKM)*, 2008, pp. 931–940.

[6] C Chen, M Zhang, Y.Q. Liu, and S.P. Ma, "Social attentional memory network: Modeling aspect-and friend-level differences in recommendation," in *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining (WSDM)*, 2019, pp. 177–185.

[7] W.Q. Fan, Y Ma, Q Li, Y He, E Zhao, J.L. Tang, and D.W. Yin, "Graph neural networks for social recommendation," in *Proceedings of the international conference on World Wide Web (WWW)*, 2019, pp. 417–426.

[8] J.L. Yu, H.Z. Yin, J.D. Li, M Gao, Z Huang, and L.Z. Cui, "Enhance social recommendation with adversarial graph convolutional networks," *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 2020.

[9] D Silver, G Lever, N Heess, T Degris, D Wierstra, and M Riedmiller, "Deterministic policy gradient algorithms," in *Proceedings of the 31st International Conference on International Conference on Machine Learning (ICML)*, 2014, pp. 387–395.

[10] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, "Session-based recommendations with recurrent neural networks," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

[11] D.Y. Zhao, L Zhang, B Zhang, L.Z. Zheng, Y.J. Bao, and W.P. Yan, "Mahrl: Multi-goals abstraction based deep hierarchical reinforcement learning for recommendations," in *Proceedings of the 43rd International Conference on Research & Development in Information Retrieval (SIGIR)*, 2020, pp. 871–880.

[12] Y Lei, Z.T. Wang, W.J. Li, H.B. Pei, and Q.Y. Dai, "Social attentive deep q-networks for recommender systems," *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 2020.

[13] J Zhang, B.W. Hao, B Chen, C.P. Li, H Chen, and J.M. Sun, "Hierarchical reinforcement learning for course recommendation in moocs," in *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI)*, 2019, vol. 33, pp. 435–442.

[14] F Llorente, L Martino, J Read, and D Delgado, "A survey of monte carlo methods for noisy and costly densities with application to reinforcement learning," *arXiv preprint arXiv:2108.00490*, 2021.

[15] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. V eness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, and G Ostrovski, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Aug 2015.

[16] I. Cantador, P. Brusilovsky, and T. Kuflik, "Second workshop on information heterogeneity and fusion in recommender systems," in *Proceedings of the fifth ACM conference on Recommender systems (Recsys)*, 2011, pp. 387–388.

[17] J. Tang, H. Gao, H. Liu, and A. Das Sarma, "etrust: Understanding trust evolution in an online world," in *KDD*, 2012, pp. 253–261.

[18] P. Massa and P. Avesani, "Trust-aware recommender systems," in *Proceedings of the 2007 ACM conference on Recommender systems (Recsys)*, 2007, pp. 17–24.

[19] Y Lei, Z.T. Wang, W.J. Li, and H.B. Pei, "Social attentive deep q-network for recommendation," in *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 2019, pp. 1189–1192.

[20] H. Chen, X. Dai, H. Cai, W. Zhang, X. Wang, R. Tang, Y. Zhang, and Y. Yu, "Large-scale interactive recommendation with tree-structured policy gradient," in *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI)*, 2019, vol. 33, pp. 3312–3320.

[21] Y.Q. Qin, P.F. Wang, and C.L. Li, "The world is binary: Contrastive learning for denoising next basket recommendation," in *Proceedings of the 44th International Conference on Research & Development in Information Retrieval (SIGIR)*, 2021, pp. 859–868.