Neural Computing and Applications (2022) 34:3097–3115
https://doi.org/10.1007/s00521-021-06573-8

ORIGINAL ARTICLE

Poisoning attacks against knowledge graph-based recommendation systems using deep reinforcement learning

Zih-Wun Wu[1] · Chiao-Ting Chen[2] · Szu-Hao Huang[3]

Received: 14 March 2021 / Accepted: 21 September 2021 / Published online: 8 October 2021

Abstract
In recent years, studies have revealed that introducing knowledge graphs (KGs) into recommendation systems as auxiliary information can improve recommendation accuracy. However, KGs are usually based on third-party data that may be manipulated by malicious individuals. In this study, we developed a poisoning attack strategy applied on a KG-based recommendation system to analyze the influence of fake links. The aim of an attacker is to recommend specific products to improve their visibility. Most related studies have focused on adversarial attacks on graph data; KG-based recommendation systems have rarely been discussed. We propose an attack model corresponding to recommendations. In the model, the current recommended status and a specified item are analyzed to estimate the effects of different attack decisions (addition or deletion of facts), thereby generating the optimal attack combination. Finally, the KG is contaminated by the attack combination so that the trained recommendation model recommends a specific item to as many people as possible. We formulated the process into a deep reinforcement learning method. Conducting experiments on the movie and the fund data sets enabled us to systematically analyze our poisoning attack strategy. The experimental results proved that the proposed strategy can effectively improve an item's ranking in a recommendation list.

Keywords Poisoning attacks · Adversarial attacks · Knowledge graph-based recommendation · Deep reinforcement learning

1 Introduction

Recently, recommendation systems have become increasingly valuable. Recommendation systems are commonly used in real-world applications such as music, film, and online shopping. These systems have attracted attention from numerous scholars; introducing a knowledge graph (KG) into a recommendation system as auxiliary information can improve the system's recommendation accuracy. Therefore, KG-based recommendation systems have gradually become popular. A KG is essentially a heterogeneous network constructed from real-world facts; it has strong problem analysis capabilities in terms of extracting knowledge from fact associations. Such graphs have played a crucial role in many practical programs, such as those for information extraction and question answering. In a KG recommendation system, KG embedding (KGE) technology is applied to account for knowledge rules when entities and relationships are being transformed into vectors. This technology has good recommendation performance and can explain the advantages of recommendation results.

A disadvantage of recommendation systems is that they attract the attention of attackers. Attack research has mainly focused on changing the interaction records between people and objects on traditional recommendation systems, such as the injection of forged transaction or

✉ Szu-Hao Huang
szuhaohuang@nycu.edu.tw

Zih-Wun Wu
zihwun07534.iim07g@nctu.edu.tw

Chiao-Ting Chen
rsps971130.cs09@nycu.edu.tw

[1] Institute of Information Management, National Yang Ming Chiao Tung University, Hsinchu, Taiwan 30010

[2] Department of Computer Science, National Yang Ming Chiao Tung University, Hsinchu, Taiwan 30010

[3] Department of Information Management and Finance, National Yang Ming Chiao Tung University, Hsinchu, Taiwan 30010

evaluation records. Although the results of research on KG-based recommendation systems have made such systems more popular in the related industry, little is known about the robustness of adversarial attacks. Research has revealed that deep learning models are easily affected by perturbations and make wrong decisions, making their practical application questionable. Most KGs in such recommendation systems are established using public data sources; for example, they are established to obtain information on entities using web crawlers. However, if a malicious person forges information to change the collected content (i.e., the KG structure is changed), then the recommendation system makes recommendations according to the attacker's intention. In other words, attacking the recommendation system can give the attacker some commercial benefits. For example, the system may actively recommend certain products to make the products reach more users, increasing the likelihood of them being purchased. As mentioned, it is necessary to analyze the vulnerability of a recommendation system with respect to its KG.

In this study, we focused on the problem of poisoning attacks in KG-based recommendation systems. As the demand for deep learning models increases, researchers in different fields (e.g., those using graph models) have gradually realized the importance of this topic. In the past 2 years, relevant researchers have proposed the use of adversarial attacks against graph models. However, the training process of a graph model treats each node as a discrete feature, and the connection effect engenders information propagation; hence, operating a single node affects many other nodes. Consequently, attackers cannot use gradient-based algorithms to explore the weaknesses of graph models. Furthermore, existing graph-based adversarial attack methods cannot be directly applied to KGs. One reason is that KGs have heterogeneous structures but require that the graph nodes and links be of only one type. Moreover, the KG links must be trained because different link relationships have diverse influences. Finally, attacking the top-k recommendation system is different from attacking one based on node classification or link prediction. In such attacks, the matching degree of multiple items (entities) to users must be considered. According to our review of the relevant literature, few reports on adversarial attacks have discussed KG models, and research in which such models are applied for practical recommendation tasks is lacking.

This study aims to attack a KG-based recommendation system using reinforcement learning approaches. Attackers influence the KG by adding or forging facts, thereby increasing the exposure of a product and making the recommendation system attempt to recommend it to more people. However, during our research, we discovered that the number of possible attack combinations was excessively high, and each attack combination must interact with the recommendation system under a poisoning attack setting. The correct attack effect of all combinations could not be obtained. In addition, we noted that the learning process of the KG-based recommendation model involved using a transductive learning setting; specifically, the test node (item) was included in the training graph. Similar to most graph model attacks, we focused on poisoning attacks. Companies can gain commercial benefits and promote products by adding or deleting certain facts and understand the future direction of product improvement. Hence, the recommendation system provider may also be an attacker. Accordingly, we simulated the selection of wrong links (to add or delete facts) to rewrite the KG structure as a continuous decision-making process. We developed a deep Q-learning algorithm [1], which is a classic reinforcement learning method, to obtain scores for a continuous decision. Each decision is based on a set of triples. Because of the attack on the KG, entities and relationships could not be separately evaluated. The proposed model considers the joint effect of choosing different link combinations on the recommended model. Additionally, through the KGE training process, we revealed that most nodes and links in the KG would not be significantly affected by the operation. Therefore, we used transfer learning to reduce the convergence time required for KGE.

In our experiments, we used the MovieLens-1M data set [2] and real fund trading records to validate the effectiveness and extensibility of the proposed attack method. The MovieLens-1M data set is a public resource for movie viewing records and contains anonymous movie ratings. The real fund trading records contain fund characteristics and transaction records. We measured hit ratio indicators to prove that in the proposed strategy, only a few changes must be made to the product characteristics of the graph to change the recommendation results of the target item and improve a product's visibility. The purpose of this study was to explore reasonable attack methods for recommendation systems based on KGs. According to our review of the literature, this topic has not been examined by previous studies. The main contributions of this research are as follows:

- We present a strategy for attacking recommendation systems based on KGs by polluting the KGs, thereby recommending specific items to more people than before the attack.
- We developed a deep Q-learning algorithm to transform the process of selecting the most effective attack combination into a step-by-step training process of deep reinforcement learning. We propose two reward mechanisms for identifying the combination that maximizes the attack effect in the continuous decision process.

– Our attack strategy has greater effectiveness and practicability than do those proposed by existing research on adversarial attacks based on KGs, as demonstrated by the experimental results. The adversarial examples of the KG are transferable, as revealed by a black box experiment.
– The application of our proposed strategy to mutual fund recommendation systems confirmed that manipulating the KG can increase the visibility of specific products and bring commercial benefits to attackers.

2 Related work

This section is divided into three parts. The related literature on KGE is introduced first. Second, we discuss the relevant literature on recommendation systems and adversarial machine learning.

2.1 Deep learning for a KG

A KG, which is virtually a heterogeneous network, consists of entities (nodes) with different physical meanings and multiple relationships (edges) between them in the real world. It is usually displayed and stored using a discrete triple relationship (head, relationship, tail), which represents the relationship between the front and back entities. These relationships are sometimes called facts because the constructed knowledge is obtained from the real world. In the past few years, numerous studies have analyzed KGE techniques. Such techniques are mainly used to obtain the vector representation of a triple in the hope that the low-dimensional vector can express the potential association of entities and relationships [3, 4]. One well-known method involves transforming a distance model, such as TransE [5], which treats the probability of triples as the distance in the vector space and uses a distance-based scoring function to update the embedded vector. Another method involves the use of a semantic matching model, such as ComplEx [6], which evaluates the possibility of triples through semantic analysis of entities and relationships. In addition, NAM [7], a semantic matching model, uses a deep learning architecture to connect head entity embedding and relationship embedding to the input layer and provides scores based on the similarity between the output of the model and tail entity embedding. The KG attacked in this study mainly uses the concept of semantic matching.

2.2 Recommendation system

Due to the substantial development of the electronic industry, recommendation technology has become a pertinent research topic, and many recommendation algorithms have been developed [8, 9]. The main task of a recommendation system is to estimate customer preferences for products and then recommend suitable products to users to increase sales. In recent years, researchers have begun to harness the nonlinear transformation of neural network technology in recommendation systems to mine hidden knowledge from limited data in order to obtain more accurate predictions. In general, modeling approaches for recommendation systems focus on the side information between user information, item characteristics, and unique knowledge in some fields, and the correspondence between user items is accurately and effectively identified to generate a recommendation list [10, 11]. Researchers have recently discussed the recommendation problem of predicting a consumer's next purchase on the basis of a series of behaviors. Recurrent neural networks are commonly used for shopping order analysis to capture the latent features between contexts in order to solve this problem [12, 13]. Reinforcement learning agents have been applied in a variety of domains though there still exist several limitations. In order to solve one of the difficulties, Mnih et al. [1] develop a deep Q-network that can learn policies from high-dimensional successfully. Reinforcement learning (RL) algorithm is also used in unknown dynamics to solve Markov jump time linear system problem. He et al. [14] propose a RL algorithm that can parallelly compute the corresponding N-coupled algebraic Riccati equations while ignoring the dynamic trait of the MJLSs. Based on deep reinforcement learning, which treats recommendation as a sequential decision making process and captures the interactions between the users and items, [15] adopt a scheme called "Actor-Critic" that can consider both the dynamic adaption and long-term rewards. Zhao et al. [16] develop an optimal advertising algorithm to consider both positive and negative influence of ads on user experience of recommended items, and this can continuously update its advertising strategies and maximize reward in the long run.

In this study, we focused on the KG method. With the success of graph models, researchers have used users' explicit or implicit feedback on a project to construct graphs. In graph models, recommendation tasks are considered as link prediction problems. The tasks involve predicting the possibility that a connection exists between the item and the user; predicting the existence of a link means that the user will like the item [17–19]. However, this approach entails making recommendations based on similar objects (products) and user behaviors (purchases) and ignores other information. For example, in scientific literature recommendations, information on conference venues and authors have different degrees of importance. Therefore, the use of KGs for generating recommendations

has become common because they can structure auxiliary information adequately [20]. KGE-related methods provide knowledge for extracting potential features that correspond to entities from different relationships to identify similar users for target users; thus, the recommendation results are usually helpful [21, 22]. He et al. [23] introduced a translation method to establish a recommendation model. Wang et al. [24] proposed RippleNet based on the assumption that the propagation of preferences for entities on a graph is similar to the propagation of ripples on water. In this network, the influence of entities at different distances is considered. Wang et al. [25] proposed MKR, which can be divided into a KGE module and recommendation module. The KGE module learns potential representations of users and items, and the recommendation module learns representations of item-related entities with semantically matched KGE models. Wang et al. [26] proposed KGCN, which simulates the final representation of candidate items by embedding aggregates of entities and neighbors in a KG. The ability to aggregate information from different sources is an advantage of using a KG. Auxiliary information, such as social or trust relationships, can be added to the recommendation system, thereby improving the recommendation effect and alleviating cold start and sparsity problems [27, 28]. However, research has ignored the vulnerability crisis caused by the addition of KGs.

2.3 Adversarial attack

Initial studies on adversarial machine learning have focused on eye-catching deep learning applications such as image classification and object recognition [29–31]. Szegedy et al. [32] first declared that those image classification models are sensitive. Although they are well trained, simple adversarial examples can severely degrade their performance. Two classification methods are commonly used for adversarial attacks. One is based on the execution time of the attack: The attacker designs the test input of the model. This involves adding a small perturbation to the input to cause the prediction model to malfunction. This is called an evasion attack [33]. The other method, examined in this research, is a poisoning attack. Such an attack enables the insertion of adversarial samples into the training set. This attack is designed to deceive the model during the training process and cause it to misjudge a specific test sample [34]. Various adversarial attack strategies have subsequently been proposed, and attacks have even been tested in the physical world to prove the vulnerability of deep neural networks in various situations [35–37]. Moreover, the other classification method is based on information obtained by the attacker. If the attacker has any data (such as parameters and architecture) on the victim's model (that is, the model being attacked), then the attack is called a white box attack. Otherwise, it is called a black box attack.

Recently, because graphic data have become the core feature of many high-impact applications, research on the robustness of graph data has been increasing. The characteristics of graph data pose challenges in terms of directly transferring the attack method of the image to the graph data. For example, the search space of the adversarial sample against the graph structure is discrete [38]. Attackers deceive most related models by modifying structural information, such as adding or deleting some edges or modifying node features [39]. For training graph embedding tasks, Bojcheski and Günnemann [40] proposed an embedding method based on factorization and an iterative gradient method for data poisoning attacks. Under the topic of graph clustering, Chen et al. [41] discussed the performance variation when noise is injected into the bidirectional graph of domain name system queries. In the link prediction task, Chen et al. [42] used the fast gradient step method to conduct evasion attacks on a graph convolutional network architecture. Zugner et al. [43] discussed GCN linearization [44] and proposed an attack system that perturbs a given edge and node feature based on a proxy model and greedily chooses the most likely of these to change the perturbation of classification results. Dai et al. [45] proposed an attack method termed RL-S2V based on reinforcement learning. This attack decomposes the selection of edges into two nodes. For KG attacks, because of the heterogeneity of KGs and the fact that the connections of KGs have different meanings, the attack method cannot be directly applied to graph data. Zhang et al. [46] proposed a poisoning attack that calculates the perturbation benefit score that can be used to add or delete facts independently through the calculation of the target fact gradient. Attackers can choose large scores to add or delete facts in order to manipulate the rationality of any target facts in the KG. The attack ignores the influence of fake links. Xian et al. [47] proposed a method to assume the importance of links in different structures in order to develop a robust link prediction attack. Another study calculates the optimal data to explore the robustness and the performance of data poisoning attacks, injecting the fake users to attack the recommendation systems [48]. A systematic study was firstly proposed to poisoning attack deep learning-based recommendation systems that users are recommended targeted items [49]. Fang et al. [50] and Chen et al. [48] have proposed the use of poisoning attacks on recommendation models, but they have mainly been aimed at attacks on user–item bipartite graphs or collaborative filtering methods based on neighborhood information. Such attacks involve the injection of false user interaction data into the recommendation models. Tang et al. [51] proposed Adversarial Multimedia

Recommendation, which uses adversarial learning to establish a robust multimedia recommendation system.

Several methods are available for performing adversarial attacks on KGs. In [52], SAShA, an attack method for leveraging semantic features extracted from a KG, was introduced. In [53], generating fake users as an optimization problem increased the impact of adversarial attacks. In [54], the attacker launched a data poisoning attack on a recommendation system through the injection of fake users such that the attacker's desired recommendations would be made. With carefully crafted user-item interaction data, the attacker can trick a system into recommending a target item to as many normal users as possible. Current methods mainly focus on generating fake users to attack recommendation systems. In this paper, we suggest that the modification of existing KGs is worth investigating. In addition to applying fake users or items in recommendation models, we directly modified an existing KG to determine the differences before and after attacks. By evaluating the effects of different KG modifications on framework performance, we can reveal the importance of different entities and relationships in a KG. This study also holds value for future applications. For example, commercial benefits can be obtained, and the robustness of recommendation systems can be enhanced. Therefore, our proposed framework is worthy of exploration.

3 Poisoning attacks against KG-based recommendations

In this section, we discuss the problem setting and recommendation model mainly for white box attacks.

3.1 Problem definition

First, we assume that the primary objective of the attack on the recommendation system is to promote a specific item i to as many people as possible. We can hypothesize that the system recommends K products to each user. We can use the general indicator $hit(i)$ to indicate the proportion of users who had the target item in their top-K recommendations after the attack. The $hit(i)$ variable is the hit rate of the target item i and can be defined as follows:

$$hit(i) = \frac{\sum_{u \in U_i} H_{ui}}{|U_i|}, \qquad H_{ui} = \begin{cases} 1, & i \in \hat{L_u} \\ 0, & otherwise \end{cases}. \tag{1}$$

where $\hat{L_u}$ denotes the top-K item list that the system recommends to user u. U_i is the user group that does not have an interaction record for item i during the training process, that is, target users, and $|U_i|$ is the number of target users. The attacker's task is to maximize the hit rate. A hit rate increase means that the characteristics of the target product satisfy the user's preference more than other products do. To achieve this goal, the attacker changes the structure of the KG mainly by changing the fact that the target commodity is the head entity. As shown in Fig. 1, the attacker affects the recommendation system by attacking the KG. However, under the premise of having limited resources and preventing the attack from being discovered, changes to the structure are limited; we assume that the attacker can, at most, increase or decrease N facts. On the basis of these considerations, this attack problem can be explained using the following equation:

$$\begin{aligned} \text{maximize} \quad & hit(i) \\ \text{subject to} \quad & \left|(h_i, r_j, t_j)\right| \leq N \\ & (h_i, r_j, t_j) \in C_i \end{aligned} \tag{2}$$

where $\left|(h_i, r_j, t_j)\right|$ is the number of attack triples, $C_i = \{h_i\} \times \{(r_j, t_j)\}$ is the set of all the add and delete candidates of the target item i, $\{h_i\}$ and $\{(r_j, t_j)\}$ are the set of head entities of the target item i and all possible tail entity pairs in the KG ($\mathcal{KG}$), respectively, and $\times$ is the Cartesian product.

Finally, the attacker may possess varying levels of knowledge about the training data (i.e., $\mathcal{KG}$, target recommendation model $\mathcal{RS}$, and model parameter θ after training). As usual, according to the attack method and the attacker's knowledge, the attack can be considered to be either a white or black box attack. In a white box attack, the attacker understands the algorithm and the parameters used by the deep learning model. Furthermore, the attacker can interact with the system while generating adversarial examples. By contrast, in a black box attack, the attacker does not have detailed information on the model, and the attacker can commonly receive assistance from a surrogate model to make inferences on the weakness of the target model. Herein, we can assume that the attacker possesses information on the recommendation algorithm and the KG used by the given recommendation system. The triples of the KG can be obtained, and a KG can be constructed by the attacker. In this study, we primarily focused on white box attacks. Finally, through our experiments, we demonstrated that our methods can also be used in black box attack settings.

3.2 Victim model

We first consider white box attacks herein. The chosen model is MKR proposed by Wang et al. [25], a multi-task feature learning method for KG recommendation

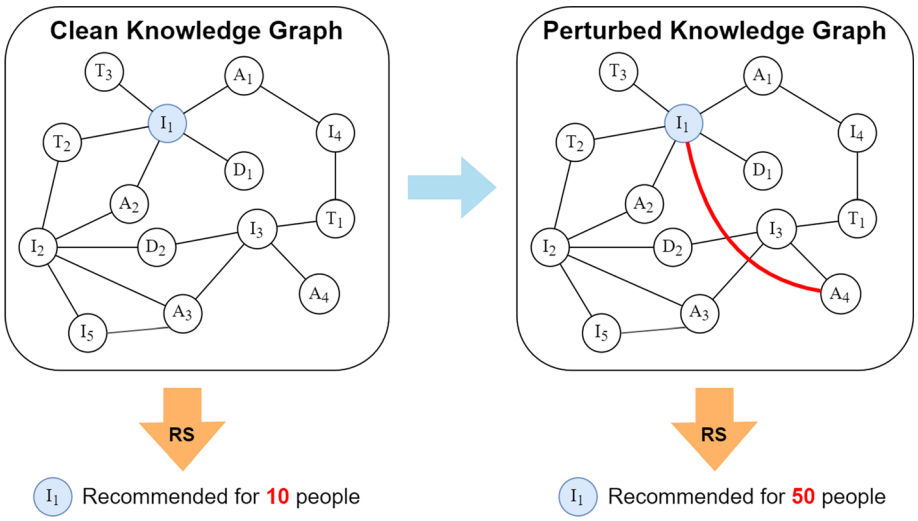

Fig. 1 Example of a poisoning attack on a KG-based recommendation system. The goal of the system is to predict the likelihood of interaction between the entity and the user. Here, entity I_1 is the target product. The attacker aims to modify the prediction of the recommendation system for entity I_1 by modifying the edge

enhancement. MKR is an architecture that incorporates KGE technology to its design to facilitate recommendation tasks. Items in the recommendation system may be associated with one or more entities. Therefore, the item and its equivalent entities may have similar adjacent structures in the recommendation system and KG, and they may share similar features in the task-specific latent feature space. The overall training process involves taking users and products as inputs, using deep neural networks to extract user features, and using cross-compression units to interactively extract product features from the potential representations of items and entities. Then, the extracted features are fed together into a deep neural network to output the recommendation prediction probability. In the KGE task, neural networks are employed to collect features from the facts between the head and relationship. The KGE module uses semantic matching models, the scoring function, and the real tail to supervise the predicted tail representation of the optimized output. In the study by Wang et al. [25], results demonstrated that with the assistance of a KG, the accuracy of real movie data sets could be improved. KGs have been confirmed to have favorable performance in actual recommendation scenarios. In addition, the MKR model provided highly accurate movie recommendations at the beginning of this study; therefore, we chose it as the target of the white box attack. Our attack was mainly on the KGE state. Subsequently, we made the recommendation network output the desired results.

3.3 System overview

Figure 2 presents the overall structure of our system. Under a white box attack scenario, we first obtain a training mechanism for the recommendation model and obtain the KG triples. The recommended system is the environment shown in the figure. The attacker's goal is to identify N added or deleted facts to change the KG. Because of the presence of excessive attack combinations and the limitations of poison attacks, we could not use brute force solutions to test all possible combinations to determine the best attack combination. Therefore, we set the attack strategy as an optimization problem that can make continuous decisions and use continuous optimization to perform a discrete search for potential modifications. To construct a well-trained recommendation system, we used a deep Q-learning algorithm [1] and designed a Q network, which is responsible for estimating the attack effect under different KG structures and attack actions. After each attack, to change the KG, the attacker provides a perturbed KG to the recommendation system. The recommendation system is trained to obtain recommendation result changes and provide feedback on the effect of attack actions. Through this process, the attacking system continuously accumulates information, stores it in replay memory, updates the Q network, and finally selects the optimal attack combination for the attacker.

To further explain our proposed framework, we present an example of its operation before and after an attack. As shown in Fig. 3, in the rating records, the user liked several movies in which Tom Hanks starred. This indicates that the recommendation system may recommend other movies starring Tom Hanks to the user. If the original relations in the KG are learned, the recommendations of the user will be similar to those in Fig. 3. However, by learning to add fake facts to the KG, as shown in Fig. 4, we can ensure that an unpopular movie with no relation to Tom Hanks is presented in the recommendation list, thus increasing that movie's visibility. Furthermore, corresponding commercial profits can be obtained. We can also delete facts, with similar effects.

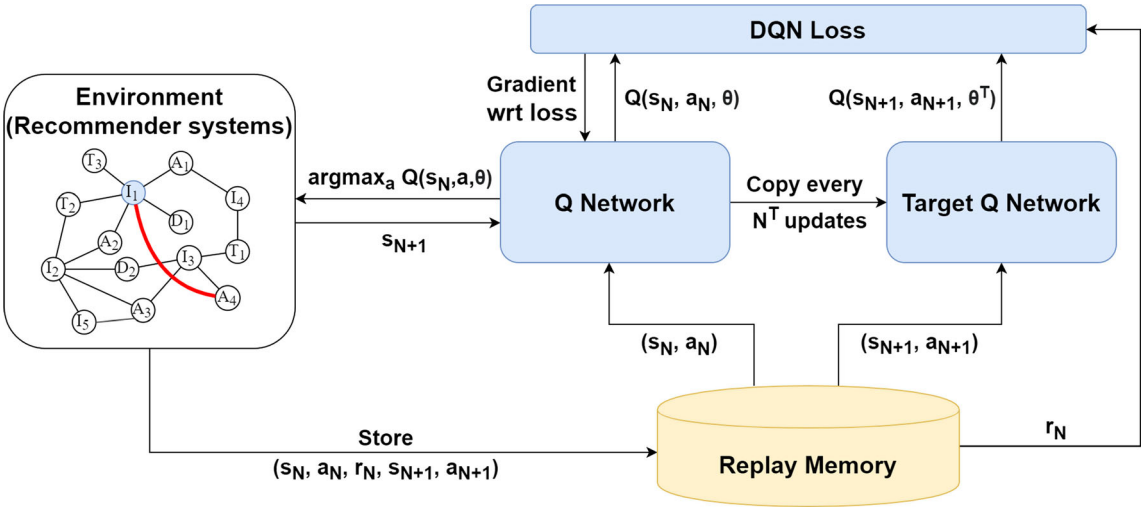

Fig. 2 Overview of our system

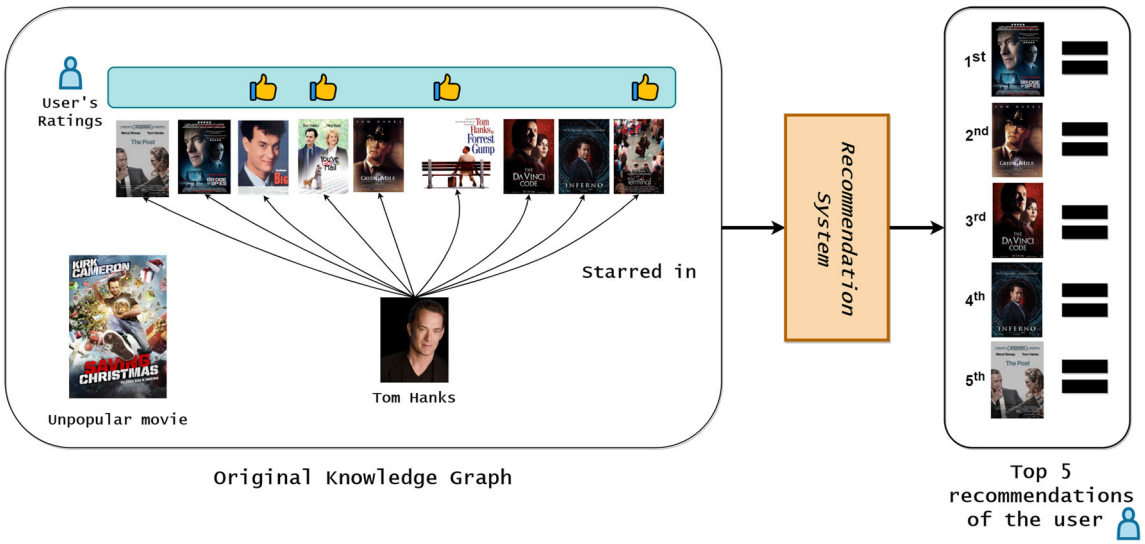

Fig. 3 Toy example of our proposed framework before attack

3.4 Necessity of deep reinforcement learning

According to the research goals established herein, two problems were discovered in the research process. The first problem pertained to the existence of an excessive number of possible attack combinations. Assume that an entity can add or delete 1000 facts in the KG; then, if the attacker can change five facts, they have approximately 8 trillion combinations to choose from. An attack model cannot directly determine which five facts combine to produce the optimal attack effect. The second problem was that in the KGE training method, the influence on embedding cannot be ignored when a knowledge rule is changed. The target product is affected by the different relationships between adjacent entities. As mentioned, we adopted a poisoning attack scenario. Each attack combination must interact with the recommendation system to reveal whether the attack was successful, making it impossible for us to obtain all combinations' correct attack effect. Therefore, to overcome these two challenges, we transformed the process of selecting the most effective attack combination into a continuous decision optimization problem. We divided the behavior of changing the structure into multiple time choices. The attack model proposed in this study can make predictions based on the current recommendation situation and the specified product, interact with its environment after adding or deleting one fact at a time, and attack N times continuously. Through the use of deep reinforcement learning and the training method of maximizing rewards, our attack model can identify the combination that can maximize the attack effect after examining the impact of the attack action.

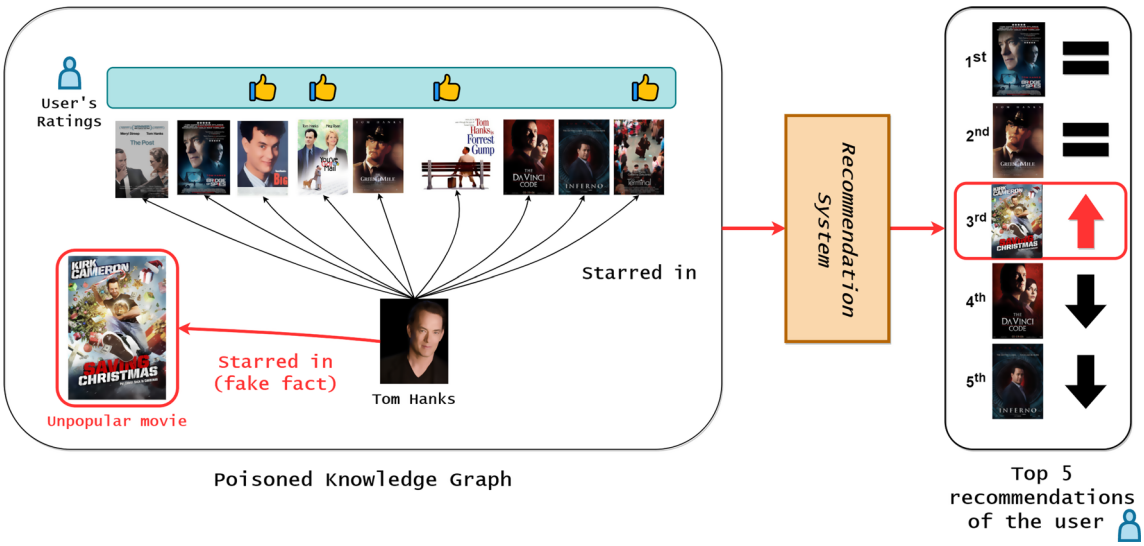

Fig. 4 Toy example of our proposed framework after attack

4 DQN framework for poisoning attacks

Given the pollution-free KG $\mathcal{KG}$ and the target recommendation model $\mathcal{RS}$, we can model the attack process as a limited Markov decision process (MDP). The sample trajectory of this MDP is as follows: $(s_1, a_1, r_1, \ldots, s_N, a_N, r_N, s_{N+1})$, where s_N represents the state of the KG observed at the nth time point and a_N and r_N represent the actions and rewards, respectively, selected by the agent at the nth time point. The framework of the Q network is illustrated in Fig. 5. The network inputs the current state of the KG and the executed attack actions into the framework, and the attacker executes the attack with the greatest effect. Each component design in our proposed DQN framework is detailed in the following sections. Based on the finite Markov decision assumption, the future depends on the present, and each choice depends on the state of the current KG. Therefore, we can assume that wrong link selection pollutes the state of the KG $\widehat{\mathcal{KG}}$ (i.e., the entity embedding of the target project has changed). The changes in the recommendation list can be estimated through entity embedding or changes in the polluted KG so that the optimal attack combination can be identified through continuous decision-making after the influence of different behaviors is considered.

4.1 Design of state

Let a KG-based recommendation system be the environment, and let the agent observe the embedding obtained through KG training. Through the complex links in the KG, we can understand the influence of certain entities on the

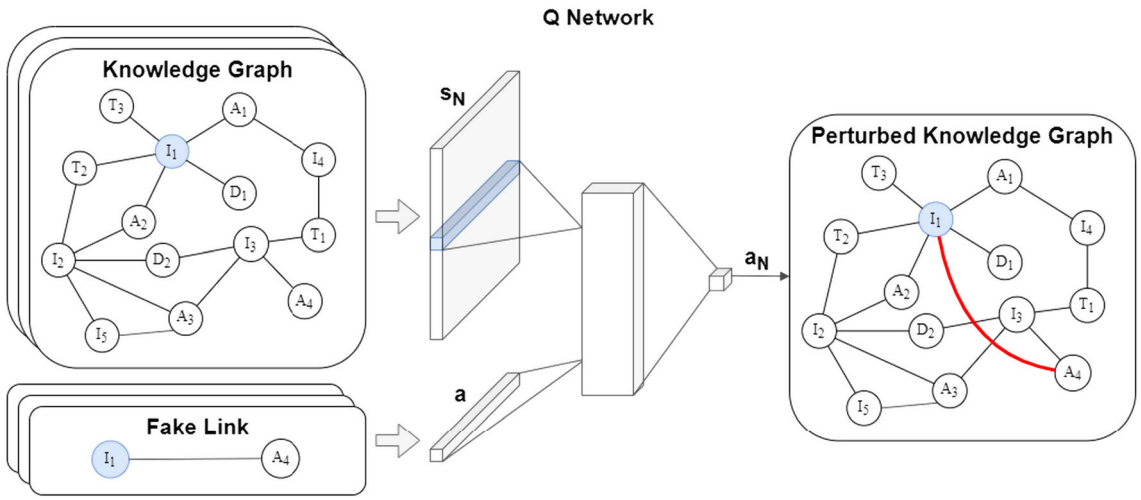

Fig. 5 DQN framework

recommendation effect. For example, a movie starring Tom Hanks is loved by most people. Therefore, the target product i added to the link with Tom Hanks may be recommended to more people. As mentioned, our agent observes a local KG structure centered on the target commodity i. For the target commodity i, its corresponding entity embeds $\mathbf{e_i}$ as a seed in the KG and then extends 1-hop along with the link triples. We can obtain the connected triples (h, r, t) as candidate facts for deletion. Next, we can gather all possible tail entity pairs in the KG and remove the existing tail entities of the target entity. The retained collection can be regarded as a preliminary fact that can be added.

We can obtain the vector representation of the triples added and deleted and merge them into a matrix to represent the KG structure with the target item i as the seed. The state can be defined as follows:

$$\begin{aligned} &\text{state}(\widehat{\mathcal{KG}}_N) = \mathbf{e_i^h} \parallel \mathbf{r_j} \parallel \mathbf{e_j^t} \parallel Y, \\ &Y = \begin{cases} 1, & (h_i, r_j, t_j) \in \mathcal{KG} \\ 0, & (h_i, r_j, t_j) \notin \mathcal{KG} \end{cases}. \end{aligned} \tag{3}$$

where $\widehat{\mathcal{KG}}_N$ is a partially modified graph (some edges had been added or removed from $\mathcal{KG}$ at time N) and Y is the label representing whether a triplet exists.

4.2 Design of action

The agent's responsibility in the poisoning attack task is to determine the addition or deletion of specific facts. We can use continuous decision-making to determine the combination of attacks; therefore, we choose a single action to execute attacks at a single time. In our attack action space, a behavior must be selected for execution. In short, we do not have the option to not implement an attack. We can apply a discrete action space and then let the agent choose. Our action space input design is similar to the state design. We use a triplet entity, relationship embedding, and action tags to represent different acts of adding or deleting facts. We can express the action as follows:

$$\begin{aligned} &\text{action}(\widehat{\mathcal{KG}}_N) = \mathbf{e_i^h} \parallel \mathbf{r_j} \parallel \mathbf{e_j^t} \parallel A, \\ &A = \begin{cases} -1, & (h_i, r_j, t_j) \in \mathcal{KG} \\ 1, & (h_i, r_j, t_j) \notin \mathcal{KG} \end{cases}. \end{aligned} \tag{4}$$

where $\widehat{\mathcal{KG}}_N$ is a partially modified graph (some edges have been added or removed from $\mathcal{KG}$ at time N) and A is the attacker's action against the triple. If the graph already has a triple, the attacker performs the delete action so that it is negative. Conversely, if the graph does not have a triple, one is added to the graph; thus, it is a positive number.

4.3 Design of the reward function

The attacker's aim is to deceive the target recommendation model. In our preliminary experiment, only nonzero rewards were received at the end of the MDP; no rewards were received in the intermediate steps of the modification, which would make the training process considerably slow. Moreover, we were concerned with the value and reward of operations performed in this state, and we expected to obtain the value of each action, but this was not possible. This is because we could choose too many actions at the same time. Moreover, every time an action is chosen, recommendation model training must be conducted, which cannot be completed in a limited time. On the basis of the aforementioned limitations and the mechanism of the recommendation system, we designed two schemes for rewards based on the attack effect and judged the reward by comparing the recommended changes before and after the attack.

In the first reward design, the output of the recommendation model is used to predict the likelihood of the user interacting with the item. For the purpose of the attack, we hope that when a KG changes, the recommendation model provides each target user with a higher predicted result of the specific project interaction than that before the attack. Because a higher probability value means that the user is more likely to purchase the target product, that product is more likely to appear in the recommendation list. In this case, the operation selected by the agent receives a positive reward; otherwise, it is negatively rewarded. This is called a score reward, which can be expressed as follows:

$$\text{score}(\text{state}(\widehat{\mathcal{KG}}_N), \text{action}(\widehat{\mathcal{KG}}_N)) = \sum_{u \in U_i, v=i} \widetilde{y}_{u,v} - \hat{y}_{u,v} \tag{5}$$

where $\hat{y}_{u,v} = \sigma(f_{\mathcal{RS}}(u, v))$ represents the prediction results of the pollution-free KG recommendation system and $\widetilde{y}_{u,v}$ represents the prediction results of the contaminated KG recommendation system after new training.

The second reward is based on our attack targets and evaluation indicators. A high recommended ranking of a specific item is desirable; therefore, the most direct approach can be used to consider whether each target user's ranking of the target item has improved. When the selected action improves the ranking, a positive reward is obtained, termed the rank reward; the formula can be expressed as follows:

$$\text{rank}(\text{state}(\widehat{\mathcal{KG}}_N), \text{action}(\widehat{\mathcal{KG}}_N)) = -\sum_{u \in U_i, v=i} \widetilde{L}_{u,v} - \hat{L}_{u,v} \tag{6}$$

where $\hat{L}_{u,v} = \text{sort}(\{\hat{y}_{u,v}\})$ represents the ranking position of target products i in the recommendation list given to users u before the attack and $\widetilde{L}_{u,v}$ represents the ranking position after the attack. A lower ranking after the attack indicates a better attack effect. The observation recommendation system can reveal that ranking rewards include scoring rewards. Due to changes in users and target product prediction possibilities, the recommendation list changes slightly, but no guarantee of a substantial change can be made because only changes between users and target products are considered. When users are more likely to interact with other commodities, it is difficult for target commodities to be interchanged with them. The advantage of observing ranking directly is that for ranking, not only users and target products but also the likelihood scores of users and other products are considered. A positive reward can be obtained only when the ranking is higher than that before the attack; therefore, the attack system substantially increases the product ranking.

The most crucial part of reinforcement learning is identifying the best policy for reward maximization. Because our problem setting is a discrete optimization problem, our first intuition would be to use Q-learning because DQN exhibits high sampling efficiency and stable performance for discrete operational spaces. Q-learning is an off-policy optimization related to the Bellman optimality equation, and a value-based method is employed in DQN. The optimal action–value function is the maximum value for all strategies. This is expressed as follows:

$$Q^*(s_N, a_N) = \max_{\pi} Q^{\pi}(s_N, a_N) = \mathbb{E}[r(s_N, a_N) + \gamma \max_{a_{N+1}} Q^*(s_{N+1}, a_{N+1}) | s_N, a_N] \tag{7}$$

where γ is the discount factor; the best attack effect in the future is considered. Because our system focuses on limiting the structural modifications an attacker can make, we do not consider the attack capabilities of an excessively large attack combination. We hope to gain a certain level of attack power during a limited attack. Therefore, discount

Algorithm 1 DQN framework for poisoning attacks against a KG-based recommendation system

Input: $\mathcal{KG}$: pollution-free KG; $\mathcal{RS}$: pre-trained recommendation system; $\widehat{\mathcal{RS}}$: recommendation system based on $\widehat{\mathcal{KG}}$; Q: action–value function; $\mathcal{D}$: replay memory; θ: randomly initialized weights of policy network; θ^T: copy of θ; N: modification budget; N_r: maximum replay memory size; N_b: training batch size; N^T: target network replacement time; ϵ: probability epsilon γ: reward discount factor

Output: $\widehat{\mathcal{KG}}$: perturbed KG

for episode = 1,M **do**
  Initialize $\mathcal{KG}$, $\mathcal{RS}$ and state s_1
  for n = 1,N **do**
    With probability ϵ, select a random action a_n; otherwise, select $a_n = \max Q(s_n, a, \theta)$
    if $(s_n, a_n) \notin \mathcal{D}$ **then**
      Replace $\widehat{\mathcal{KG}}$ by inserting or removing link a_n to/from $\widehat{\mathcal{KG}}$
      Observe the next state s_{n+1} from environment $\widehat{\mathcal{RS}}$ given $\widehat{\mathcal{KG}}$ and receive reward r_n
    else
      Get next state s_{n+1} and reward r_n from $\mathcal{D}$
    end if
    Select $a_{n+1} = \max Q(s_{n+1}, a, \theta^T)$
    Add transition tuple $(s_n, a_n, r_n, s_{n+1}, a_{n+1})$ to $\mathcal{D}$ and set $s_{n+1} = s_n$, replacing the oldest tuple if $|\mathcal{D}| \geq N_r$
    if $|\mathcal{D}| > N_b$ **then**
      Sample random minibatch of N_b tuple $(s_j, a_j, r_j, s_{j+1}, a_{j+1})$ from $\mathcal{D}$
      if $j == N$ **then**
        $y_j = r_j$
      else
        $y_j = r_j + \gamma Q(s_{j+1}, a_{j+1}, \theta^T)$
      end if
      Perform a gradient descent step with loss $\|y_j - Q(s_j, a_j, \theta)\|^2$
    end if
    Replace target weights $\theta^T = \theta$ every N^T steps
  end for
end for
return $\widehat{\mathcal{KG}}$;

factors are used to reduce the impact of relying on distant behavior.

The training method for the dual DQN algorithm was adjusted for our attack purpose. First, the input of the model is the state of a single attack action (i.e., addition or removal of links), and the output is the expected attack effect of a single action in this state. In general, the DQN algorithm requires only states as its input, and it outputs sufficient probability values as executable operands. However, because of the nature of the tasks accentuated in this study, numerous operations could be performed simultaneously. If the aforementioned method is used to output all action probability values simultaneously, our model parameters would be too large to converge. During network training, a ϵ-greedy policy is used. In this policy, attack behaviors can be randomly chosen; this is often called exploration. Additionally, with this policy, the current model parameters can be used to find the best operation; this is known as exploitation. As the training time increases, the possibility of choosing an exploration behavior becomes smaller, thereby increasing the stability of the training process. After a period of training, our model would become closer to the optimal strategy. In addition, owing to the characteristics of the model input and each selected attack measure, we must retrain our environment, which is the target recommendation system. To accelerate the training time of the model, memory recall is applied $(s_N, a_N, r_N, s_{N+1}, a_{N+1})$. Thus, when we choose the same action in the same state, we no longer interact with the environment. We assume that the agent would observe the same state in the next moment and use random sampling to train the model, thereby reducing the time spent waiting for the agent to choose actions during training. The loss function of the training process is presented as follows:

$$(\theta) = \mathbb{E}[(r(s_N, a_N) + \gamma \max_{a_{N+1}} Q(s_{N+1}, a_{N+1}, \theta) - Q(s_N, a_N, \theta))^2]. \tag{8}$$

Algorithm 1 presents the proposed DQN framework for conducting poisoning attacks against a KG-based recommendation system. By following the algorithm, our proposed framework can execute effective poisoning attacks against a KG-based recommendation system. Due to the notations of this paper are various, we present a symbol Table 1 below to sum up the used symbols in this paper.

5 Experiments

We conducted experiments to evaluate the model proposed in this paper. First, the data sets and data preprocessing methods used in the experiment are introduced. Then, the evaluation index used to compare the model with other methods is defined. Finally, four experiments are discussed, and the effectiveness of the proposed model is confirmed.

5.1 Data sets and experimental settings

We used two real data sets for our experiments. The first, the MovieLens-1M data set [2] provided by MKR, was used for the first to third experiments. We determined that under similar experimental settings and model structures, problems arose in which manipulating the structure of KGs affected the visibility of commodities. As mentioned, MovieLens-1M is a common benchmark data set in the recommendation field. Additionally, the main method of constructing KGs entails the use of Internet resources. MKR is no exception; information for each movie is collected using the Microsoft Satori repository. The Internet enables the rapid collection of sufficient relationships to establish a graph structure, but in the collection process, information may be obtained that is not related to a movie, such as an actor's family background. Websites can be easily attacked, and malicious edits of movie information can be made.

The second data set, comprising fund transaction records, customer information, and fund information, was provided by an industry–academia partnership fund sales agency. We reviewed and conducted our fourth experiment

Table 1 Symbol table

K	The length of the recommendation list
U	Target user set
u	Target user
I	Target item set
i	Target item
L_u	Top-K of user u
H_{ui}	The indicator to show whether item i is in L_u
N	The limited amount to add or delete facts
C_i	The add and delete candidates of the target item i
h_i	All the head entity of item i
(r_j, t_j)	All the possible tail entity pairs in KG
Y	The indicator to show whether (h,r,t) exist in KG
$\mathbf{e_i}$	The embedding of the entity (the target item i)
$\mathbf{r_j}$	The embedding of the relation of (r_j, t_j)
$\mathbf{e_j}$	The embedding of the tail entity of (r_j, t_j)
A	The indicator of the attacker's action
$y_{u,v}^{\sim}$	The prediction results after attacks
$y_{u,v}^{\wedge}$	The prediction results before attacks
$L_{u,v}^{\sim}$	The ranking positions of target item i after attacks
$L_{u,v}^{\wedge}$	The ranking positions of target item i before attacks

on this data set. In the demonstration of different application scenarios in the fourth experiment, we used the data interval from January 2018 to December 2018. The data sets for 2018 contain over 1 million fund transaction records, including purchases, redemptions, conversions, mergers, and other operations. We retained only purchase and redemption transactions that were directly related to the model. We used fund information as the basis for constructing a KG. The basic data set statistics are presented in Table 2.

We used the hit rate $hit(i)$ as the main indicator for evaluating the performance of attack strategies. The $hit(i)$ variable is the proportion of target users whose K-recommended list includes the target item. Target users are those who have not rated the target items in the training set. Therefore, we present the indicator as Hit@ K, where K is equal to 1, 3, 5, and 10. If the hit rate after the attack is higher than that before the attack, then the average ranking of the target user group has improved. In other words, compared with the recommendation status before the attack, the target item has been recommended to more people to achieve our attacker's aim.

To demonstrate the performance of our model, we considered two methods of selecting the target item. The first was to randomly select the target product from all the products in the data sets. In our experiment, we uniformly and randomly sampled the item set and used it as the target item (i.e., normal item). In the second method, the attacker could also promote unpopular items. That is, when the targeted recommendation system recommends something to all users, the target item does not appear in anyone's top ten most likely recommendations. This may be caused by the low average rating of the item in the training data. The products selected in our second approach were called unpopular items. Following Eq. 1, the final average hit rate is denoted as:

$$\frac{1}{|I|}\sum_{i\in I} hit(i) = \frac{\sum_{u\in U_i} H_{ui}}{|I||U_i|} \tag{9}$$

where I denotes the set of target items and $|I|$ denotes the number of target items. To equitably compare the performance levels of the proposed frameworks to baseline values, we ran the training 10 times, randomly selecting 1 target item each time, to determine our model's average performance. This supported the premise that our model is effective in dealing with a specific target item and that it can also learn attack strategies related to different items.

Table 2 The data description of the data set

Data set	Users	Items	Interactions	KG triples
MoviseLens-1M	6036	2347	753,772	20,195
Fund	90,218	2368	698,140	6312

In addition to the hit ratio, we used the other metric to compare the performance of our proposed frameworks to baselines levels, namely precision. Precision is the ratio of correctly predicted positive samples to the total predicted positive samples. High precision means the false positive rate is low. In a recommendation system, precision is the ratio of the accurate recommended items to all the recommended items. By comparing the values of these metrics after the framework is applied with those at baseline, we can comprehensively evaluate these methods.

Because of the attack settings applied in this study, the training and test sets were not our concern. After a recommendation system is trained, the attacker can construct a surrogate model to execute attacks. By learning information from the surrogate model, the attacker can devise an attack strategy. Subsequently, this strategy can be applied to a well-trained recommendation system. In the Discussion section, we elaborate on the reduced performance of the recommendation system and on the main focus of our work-the performance of the target item.

5.2 Performance evaluation of the proposed framework under different attack budget settings

The first experiment verified the existence of the problem raised at the beginning of this paper. Recommendation systems assisted by KGs are vulnerable to poisoning attacks. We also proved the effectiveness of our poisoning attack strategy. We first considered the white box attack. Because of the limited availability of attack methods under our study settings, we considered the following methods scenarios for comparison with our proposed attack strategy.

- *None* The recommendation system is trained under a clean KG and has not been contaminated by any attackers. It also represents the hit rate of the target item in the recommended model.
- *Random attack* The triples with the target item as the head entity are randomly selected to be added or deleted on the KG. An attacker using this method first determines the proportion of interference facts that can be added and deleted under the related attack interference level.
- *Zhang* [46] In this attack technique, the best embedding shift vector for each target item is calculated to complete the attack target. We changed the attack technology for our attack scenario. The function of calculating the rationality of facts is replaced with a recommendation model to predict the possibility of user interaction with items.

The attack strategy based on deep Q-learning continuous decision-making was used in this study. Our default parameters were set as follows. The maximum memory size for replay was 500, the training batch size was 50, and the Q-network parameters were copied to the target network every 20 epochs. To examine our experimental hypothesis and obtain attack combinations within an acceptable time, we set the number of attack episodes to 200. Furthermore, to reduce the time spent retraining the recommended model, we set the training epoch to 10 when interacting with the environment in the continuous decision-making process. For all attack strategies, we also limited the number of times an attack behavior could be performed by an attacker to 3, 5, and 7; attack behaviors included injection or deletion. The attack budget refers to the number of times the attacker executes the behavior of changing the graph structure. For example, $N = 3$ means that the attacker can add or delete a total of three sets of triples. When the attack budget is larger, the attack is easier to identify. Because an attacker typically endeavors to avoid being discovered, we discussed the case of a small attack budget. During the final retraining of the recommended model, all perturbations affected the training set simultaneously.

The attack results for this experiment are presented in Table 3 and each value in bold font means that it outperforms the other models in the comparison. In the first column of the table, the experimental data are categorized into three groups, representing the budget limit assigned to the attacker in the same group. The first row of each group in the table represents the normal training mode of the target system (i.e., when it is not under attack). Thus, the sampling index percentages in the unpopular group were all 0%. The unpopular items were not recommended to anyone. The second to fifth lines represent the four attack strategies. Each column in the table represents a technical evaluation index; we used Hit@1, Hit@3, Hit@5, and Hit@10 to present our attack results. First, the group with a budget of 5 was analyzed. The second row represents the attacker selecting five attack perturbations in random order. For Hit@1, Hit@3, and Hit@5, the hit rates were reduced by 0.0132%, 0.0983%, and 0.2098%, respectively, whereas that for Hit@10 increased by approximately 0.08%. Our comparison revealed that a change in the structure of the KG affected the ordering of the target commodities, but the random selection method could not achieve our attack goal. The third row presents the poisoning attack strategy proposed by Zhang et al. for revealing attack perturbation. Examining the values of Hit@1 and Hit@3 revealed that the effect was still lower than that in normal training. The fourth row presents the proposed poisoning attack strategy with the KG. For the reward design, the score reward method was used. We found that the effect of Hit@10 was 0.4213% greater than None, which means that on average, the target item group was recommended to 25 people. However, the values for Hit@1, Hit@3, and Hit@5 were not as expected. We posit that because the target customer's expected scores for the target product improved, the score reward method did not consider that the product was ranked at the top of the recommendation list. The fifth row presents the rank reward method; the effect for this method was superior to those for other methods. Consider, for example, Hit@10; in normal training, the target product was recommended to 36 people on average, and after the fifth attack method, the average target product was recommended to 139 people, an increase of over 100 people. This may be because the relative ranking between the target item and other items is considered more directly than the score. Therefore, the agent actively promoted the order of the target items in the recommendation list so as to recommend a specific product to more people and increase product awareness.

Table 3 reveals that our proposed attack strategy was superior to that of our control group under three different budget constraints. We analyzed Hit@1 when the attack budget was 3, 5, and 7, and the attacker's random interference caused a reverse attack effect. The attack strategy proposed by Zhang et al. achieved the attack goals when the attack budget was 3 but failed to do so when the attack budget was 5 or 7. In our proposed attack strategy, the ranking reward is taken as an example. The hit rate of normal target items was higher than that of the general training recommendation system under any attack budget. Consider, for example, $N = 7$; the hit rate of the normal target items almost doubled. In other words, in our attack method, the target item was recommended to twice as many users as those before the attack. In our attack strategy, the orders of target items and other commodities are considered, enabling the identification of an attack combination that can maximize the benefits to the attacker. Table 2 shows that as the attack budget N increased, the hit rate of our proposed poisoning method increased overall. However, when we compared the performance under the budgets of 3 and 5 with that under the budgets of 5 and 7, we revealed that the improvement was relatively small. We believe that this was limited by the KG training method; the embedded representation of entities and relationships is affected by neighbors. Although the methods proposed in this paper account for the influence of entities on each other, we still had to restrain the experimental settings of the recommendation system. The target entity was still influenced by the stability of KGE training. Increasing the number of connections could provide the entity with more traction in training.

In addition to the hit rate, we compared our proposed frameworks' performance levels in precision with those at

Table 3 Compare the impact of different attack budgets on normal target items and unpopular target items

N	Attack	Normal items (%)				Unpopular items (%)			
		Hits@1	Hits@3	Hits@5	Hits@10	Hits@1	Hits@3	Hits@5	Hits@10
3	None	0.0583	0.1937	0.3252	0.6122	0.0000	0.0000	0.0000	0.0000
	Random	0.0282	0.1920	0.4083	1.1184	0.0018	0.0144	0.0490	0.1951
	Zhang [46]	0.0956	0.4349	0.8723	1.9437	0.0000	0.0018	0.0338	0.2318
	DQN-score	0.0056	0.0704	0.2525	0.8781	0.0000	0.0116	0.0663	0.3717
	DQN-rank	**0.0991**	**0.7189**	**1.5011**	**3.1581**	**0.0020**	**0.0254**	**0.1137**	**0.4794**
5	None	0.0583	0.1937	0.3252	0.6122	0.0000	0.0000	0.0000	0.0000
	Random	0.0132	0.0983	0.2098	0.6927	0.0000	0.0000	0.0118	0.0703
	Zhang [46]	0.0368	0.1899	0.4830	1.4629	0.0000	0.0306	0.1014	0.4665
	DQN-score	0.0056	0.0787	0.2352	1.0335	0.0020	0.0257	0.1079	0.4128
	DQN-rank	**0.1174**	**0.5099**	**1.0291**	**2.3043**	**0.0038**	**0.0834**	**0.2522**	**0.7956**
7	None	0.0583	0.1937	0.3252	0.6122	0.0000	0.0000	0.0000	0.0000
	Random	0.0266	0.2471	0.6787	1.7362	0.0000	0.0036	0.0205	0.1799
	Zhang [46]	0.0266	0.2052	0.4467	1.3847	0.0000	0.0379	0.1275	0.5012
	DQN-score	0.0921	0.4739	0.9649	1.9893	**0.0134**	0.0893	0.2098	0.6463
	DQN-rank	**0.1257**	**0.5494**	**1.0033**	**2.0891**	0.0059	**0.0943**	**0.2516**	**0.6581**

baseline. Unlike the hit rate, precision is usually used to evaluate the proportion of recommended items in the top K set that are relevant. We utilized Precision@ 1, 3, 5, and 10, when the attack budget N was 3,5, and 7. The performances on the target items are presented in Table 4. The strategy of Zhang [46] had strong attack ability when $N = 3$ on normal target items; however, under the other settings, the performances of Zhang [46] failed to make true impacts on the target items. On the contrary, our proposed methods DQN-rank had overall made significant impacts under all the settings. The performances of DQN-rank almost doubled than those of the others. Also, as previously stated, when the number of attacks increased, the strength of our proposed methods also enhanced. By measuring the performances of the attack methods with precision, we can analyze how the methods perform in all the predicted recommended items. To sum up, we can see that our proposed frameworks are effective on both normal and unpopular target items. In summary, our proposed frameworks are effective in managing both normal and unpopular target items. Additionally, the use of the other metric revealed that our proposed frameworks perform adequately under different conditions.

5.3 Performance evaluation of black box attack settings

In the previous experiments, we demonstrated the performance of our method and the possibility of different attack budgets in the white box setting. In this experiment, we explored whether an attacker can attack the recommendation system under the black box design. We also attempted to determine whether the contaminated KG is transferable. In the black box setting, the attacker is not acquainted with the algorithm and parameters of the target recommendation system. Considering the factors of interest, our recommendation system restricting black box attacks is still based on a KG. The attacker does not know that the algorithm of the target recommendation system is realistic. When a recommendation system is under attack, it is generally established that the recommendation results should be promoted within an appropriate time frame. In this case, it is difficult to determine the structure and parameters of the model. Thus, we believe black box attacks are a pertinent topic of discussion. The following is a brief introduction to the two recommendation systems:

- *KGCN* [26] KGCN is an architecture for KG convolutional networks for recommendation systems. When the embedded representation of an entity is being calculated, multihop neighborhood information is combined according to the user's preference, and the recommendation results are calculated. KGCN is the first method in which a nonspectral GCN method is applied to a KG recommendation system.
- *RippleNet* [24] A method similar to a memory network is used to introduce preference propagation and iteratively expands along the links in a KG to stimulate a user's potential interest in a set of knowledge entities. Multiple "ripples" activated by the user's historic activities are superimposed to form a user's preference distribution for recommendation items.

To determine transferability, we used the same combination of ten items used in the first experiment in the sampling combination of normal and unpopular target items.

Table 4 Compare the precision of different attack budgets on normal target items and unpopular target items. P is the abbreviation of Precision

N	Attack	Normal items (%)				Unpopular items (%)			
		P@1	P@3	P@5	P@10	P@1	P@3	P@5	P@10
3	None	0.0420	0.0466	0.0469	0.0441	0.0000	0.0000	0.0000	0.0000
	Random	0.0203	0.0461	0.0589	0.0806	0.0013	0.0035	0.0071	0.0141
	Zhang [46]	0.0689	0.1045	0.1258	0.1401	0.0000	0.0004	0.0049	0.0167
	DQN-score	0.0040	0.0169	0.0364	0.0633	0.0000	0.0028	0.0096	0.0268
	DQN-rank	**0.0715**	**0.1728**	**0.2165**	**0.2277**	**0.0014**	**0.0061**	0.0164	**0.0346**
5	None	0.0420	0.0466	0.0469	0.0441	0.0000	0.0000	0.0000	0.0000
	Random	0.0095	0.0236	0.0303	0.0499	0.0000	0.0000	0.0017	0.0051
	Zhang [46]	0.0265	0.0456	0.0696	0.1055	0.0000	0.0074	0.0146	0.0336
	DQN-score	0.0040	0.0189	0.0339	0.0745	0.0014	0.0062	0.0156	0.0298
	DQN-rank	**0.0843**	**0.1225**	**0.1484**	**0.1661**	**0.0027**	**0.0200**	**0.0364**	**0.0574**
7	None	0.0420	0.0466	0.0469	0.0441	0.0000	0.0000	0.0000	0.0000
	Random	0.0192	0.0594	0.0979	0.1252	0.0000	0.0009	0.0030	0.0130
	Zhang [46]	0.0192	0.0493	0.0644	0.0998	0.0000	0.0091	0.0184	0.0361
	DQN-score	0.0664	0.1139	0.1391	0.1434	**0.0097**	0.0215	0.0303	0.0466
	DQN-rank	**0.0906**	**0.1320**	**0.1447**	**0.1506**	0.0043	**0.0227**	**0.0363**	**0.0474**

First, we used our proposed continuous decision poisoning attack strategy to attack the forged KG generated by MKR. Subsequently, we replaced the fake KG with a clean one before retraining different recommendation systems based on the KG. This is because the process of forging the KG involved no interaction with KGCN and RippleNet; interaction was prevented in order to simulate the black box setting involving an unclear model architecture.

The experimental results are presented in Table 5. The first column of this table represents different recommendation models based on KGs. The training mechanisms of KGCN and RippleNet were not used to generate the polluted KG; therefore, the increase in the hit rate after the attack was not obvious. However, we could still transfer attacks to KGCN and RippleNet using the polluted KG. With a DQN-Ranke attack strategy as an example, after performing five disturbing actions, the attacker increased the Hit@10 of the normal target item from 0.7346 to 1.5654% under the KGCN model structure. In addition, under the RippleNet model structure, Hit@10 increased from 0.7541 to 0.9243%. We believe that these results were obtained because the KG-based recommendation system affected the relevance of users and items based on the perspective of assistance. Mining related attributes between items through links in a KG causes different recommendation systems to create similar blind spots for user preferences. This influences the generation of personalized ranking lists for users. Therefore, the polluted KG generated by the MKR training method can effectively overcome KGCN and RippleNet attacks.

Observing the unpopular items, we could determine that the same item group still has a certain hit rate in KGCN and RippleNet. We believe that this is related to the training method. When calculating the likelihood of interaction between an item and a user, the framework uses the user's preferences to aggregate information for calculating embedded representations. The main purpose of embedding is not to obtain training stability for the entire KG. We also revealed KGCN to be more reliable than the other models for data poisoning methods related to unpopular items because the hit rate after the attack of the KGCN model structure was lower than that observed for RippleNet. For example, after the KGCN model structure was retrained, Hit@10 increased by 0.4365%, whereas the hit rate of the RippleNet model structure increased by 0.8814%. However, RippleNet was relatively stable for normal items. In addition, we believe that considering the user's preference distribution mainly enables an evaluation of whether an item is related to the user's historical activities. The false link established by the DQN-score attack makes a single item closely resemble user preferences. Therefore, the DQN-Score strategy has strong attack ability in black box attack scenarios involving false KGs. The experimental results demonstrate that a contaminated KG generated by a recommendation model can deceive other recommendation models.

5.4 Performance evaluation of poisoning attacks against mutual fund recommendation

In the third experiment, we explored the scalability of our proposed attack strategy in various scenarios. In the experiment, we used the data sets provided by the industry–academic partnership fund sales organization to conduct

Table 5 Under the black box attack, the impact of the transferability of the pollution knowledge graph on normal target items and unpopular target items

Model	Attack	Normal items (%)				Unpopular items (%)			
		Hits@1	Hits@3	Hits@5	Hits@10	Hits@1	Hits@3	Hits@5	Hits@10
MKR	None	0.0583	0.1937	0.3252	0.6122	0.0000	0.0000	0.0000	0.0000
	Random	0.0132	0.0983	0.2098	0.6927	0.0000	0.0000	0.0118	0.0703
	Zhang [46]	0.0368	0.1899	0.4830	1.4629	0.0000	0.0306	0.1014	0.4665
	DQN-score	0.0056	0.0787	0.2352	1.0335	0.0020	0.0257	0.1079	0.4128
	DQN-rank	**0.1174**	**0.5099**	**1.0291**	**2.3043**	**0.0038**	**0.0834**	**0.2522**	**0.7956**
KGCN	None	0.0017	0.0340	0.1195	0.7346	0.0000	0.0308	0.0865	0.3915
	Random	0.0000	0.0108	0.0709	0.5719	0.0020	0.0097	0.0292	0.2343
	Zhang [46]	0.1655	0.2751	0.3692	0.8555	0.0054	0.0346	0.0604	0.2719
	DQN-score	**0.5609**	**0.9306**	**1.2806**	**2.2598**	**0.0163**	**0.0963**	**0.2854**	**0.8280**
	DQN-rank	0.2667	0.5668	0.8557	1.5654	0.0038	0.0822	0.2460	0.7687
RippleNet	None	0.0017	0.0345	0.1230	0.7541	0.0000	0.0222	0.0570	0.2384
	Random	0.0038	0.0309	0.1123	0.5184	0.0126	0.0495	0.1136	0.4026
	Zhang [46]	0.0282	0.1475	0.2758	0.7613	0.0129	0.0841	0.1695	0.4884
	DQN-score	**0.0398**	**0.1616**	**0.3041**	**0.9887**	**0.0714**	**0.2402**	**0.4624**	**1.1198**
	DQN-rank	0.0022	0.0583	0.2143	0.9243	0.0020	0.0266	0.1220	0.5388

experiments. We applied different experimental settings for personalized fund recommendations on real-world data. Most clients consider mutual fund transactions as long-term investments. In general transactions, peoples' concentrated funds are managed and actively invested by professional financial managers. Therefore, this is a common choice for ordinary investors to make investments; however, one problem is that the frequency of active transactions is usually low. In this experiment, we mainly used the same white box attack settings as those in our first experiment. Moreover, this application scenario was markedly different from the movie recommendation scenario. At the end of each month, personalized fund recommendations typically focus on providing fund recommendations for the next month. Such recommendations rely on a record of the previous state and predict the funds that the customer will purchase the following month. By contrast, the evaluation method of movie recommendation results involves extracting part of the results at the same recording time to test whether the recommended model can restore the transaction record.

We used a data interval from January 2018 to December 2018. For time-series data, a rolling test is often used in model training to evaluate the effect of the model at different time points. We conducted 6 months of training and had a 1-month rolling test window. We used the transaction records within the 6 months to obtain the mutual fund sets purchased during the period and connect them to the fund information to build a KG. Therefore, the structures of KGs created during different training periods would differ. In the training stage, when we searched for perturbation attacks, we generated negative samples for each customer in the training data from the collection of funds that they did not purchase. Therefore, the negative samples collected during different training sessions were different. In the testing phase, which was our attack result evaluation phase, we considered only customers who had actual transactions in the training phase and the testing phase to become our target customers. The final list of recommended funds contained all funds that had been traded during the training phase. Finally, to bolster the comparison benchmark, we randomly selected ten mutual funds that had been traded every month throughout the year to evaluate the attack effect.

In this experiment, we used the optimal strategy under the reward method proposed in the previous experiment for testing and evaluation. On the basis of a unified comparison of the first, second, and third experiments, we believe that the effect of the poisoning attack strategy of ranking rewards was relatively stable. In the different periods, we analyzed the relationship between denomination currency, risk type, fund group type, and investment type; these factors are related to the fund set during the training of time information. We created a KG using these factors. For the recommendation system of mutual funds, we also assumed that the attacker wished to promote a specific mutual fund. In practice, some funds are mainly recommended by sales organizations; hence, attackers recommend the funds to

numerous customers through recommendation system attacks. In contrast to the movie recommendation system attack, the attack here had a future effect.

We present the recorded attack results in Table 6. Regarding our target items, we randomly sampled ten target items and made recommendations to customers in different periods. Proving the effectiveness of recommendations is beyond the scope of this study. We attempted to identify perturbation triplets with attack effects and change the recommendation results of the recommendation system to make the attack successful. The first row of the table shows the conventional training mode of the target system under the use of fund data (i.e., the system was not under attack). The second to fourth lines show the three attack strategies. Our table is divided into six test periods. The effect of our attack strategy was expressed using a rolling test. Table 6 shows that the proposed poisoning attack strategy limited continuous decision reinforcement learning, and the effect of disturbing actions on others was strong. Furthermore, the effect was more stable than that in our control group. First, we examined the hit rate before the occurrence of attacks in each month (None). Each month's hit rate was different from those of the other months. One reason is that the transaction records we collected and trained were different. In addition, our KG structure differed for the different periods. This demonstrates that even for the same mutual fund, the optimal perturbation combination would differ in different periods. Next, consider, for example, the July 2018 period. When the attacker was restricted from performing five attacks to adjust the KG, we found that the Hit@10 value for the randomly selected interference attack was lower after the attack than that before it. The strategy proposed by Zhang improved the Hit @1 rate but did not achieve random attack goals. The attack strategy proposed in this paper steadily improved the hit rate of the target mutual fund group with different recommendation numbers. On average, our attack strategy doubled or tripled the attack effect. During our experiment, we also identified attacks in September 2018 and October 2018. Our Hit @1, Hit @3, and Hit @5 attacks improved, as previously described, but Hit @10 rates decreased by 0.2841% and 0.0905%; this did not meet our expectations. In response to this phenomenon, we examined the number of transactions of our target items during training and compared them with other attack methods. We identified highly popular funds in ten target items. In other words, modifying the graph structure for such commodities would easily cause adverse effects. Although our numerical results revealed that the attack failed, our proposed attack strategy still has advantages over other methods.

6 Conclusions

We devised a poisoning attack strategy using deep Q-learning that transforms the process of selecting the most effective attack combination into step-by-step training involving deep reinforcement learning. Subsequently, we proposed two reward mechanisms to identify the optimal combination of perturbations. Attackers influence KG-based recommendation systems by polluting the corresponding KG, thereby recommending specific items to more people than before the attack. Experimental results obtained using data sets with different attributes demonstrate that in the KG-based recommendation system, the evaluation value of the poisoning attack strategy proposed

Table 6 Performance comparison of different attack strategies applied to fund transaction record data

Attack	2018/ 07 (%)				2018/ 08 (%)				2018/ 09 (%)			
	Hit@1	Hit@3	Hit@5	Hit@10	Hit@1	Hit@3	Hit@5	Hit@10	Hit@1	Hit@3	Hit@5	Hit@10
None	0.2329	0.4976	1.2738	4.5432	0.7830	1.0735	1.8505	6.0607	0.4355	1.8436	3.3128	**6.9499**
Random	0.4970	1.4622	2.3707	4.3045	0.8618	2.0715	3.0406	5.5285	0.5385	1.8358	3.1467	5.9618
Zhang [46]	0.4628	1.4383	2.4596	4.9810	0.6188	1.8391	2.9157	5.8013	0.9224	2.2455	3.4269	6.2198
DQN-Rank	**0.5310**	**1.5486**	**2.5691**	**5.4148**	**1.4502**	**2.9073**	**4.2074**	**7.4981**	**1.3204**	**3.0829**	**4.2809**	6.6658
Attack	2018/ 10 (%)				2018/ 11 (%)				2018/ 12 (%)			
	Hit@1	Hit@3	Hit@5	Hit@10	Hit@1	Hit@3	Hit@5	Hit@10	Hit@1	Hit@3	Hit@5	Hit@10
None	0.7674	1.9678	3.4135	**6.6922**	0.3588	1.6122	2.9557	5.6723	0.6083	1.6027	2.6739	6.0431
Random	0.4575	1.5157	2.6203	5.6173	0.5021	1.3461	2.2765	4.6982	0.5973	2.0918	3.6480	7.2666
Zhang [46]	0.8546	2.1114	3.2968	6.4979	0.5574	1.6205	2.9860	6.4356	0.7491	2.5908	4.3579	8.3397
DQN-Rank	**0.8956**	**2.2594**	**3.5338**	6.6017	**1.0752**	**2.3697**	**3.6212**	**6.5537**	**1.6355**	**4.1913**	**6.1398**	**10.0971**

in this paper was higher than that before the attack by 1.6921%. This strategy has validity and practicability advantages over other methods discussed in counterattack research. Black box experiments showed that the adversarial examples of KGs are transferable. Results from our mutual fund experiments reveal that the manipulation of a KG can increase the visibility of specific products in practice and bring commercial benefits to attackers. Some possible future directions are worth discussing. In addition, certain aspects of the attack strategy still require enhancement. For example, in attack strategies, more realistic attack methods can be examined, such as adding new products. In particular, future research can develop effective defense algorithms on the basis of the poisoning attack strategy proposed by the present study. For example, defense mechanisms such as using adversarial training modules can improve the robustness of KG-based recommendation systems to prevent malicious attacks.

Acknowledgements This work was supported in part by the Ministry of Science and Technology, Taiwan, under Contract MOST 110-2221-E-A49 -101 and Contract MOST 110-2622-8-009 -014 - TM1; and in part by the Financial Technology (FinTech) Innovation Research Center, National Yang Ming Chiao Tung University.

Declarations

Conflict of interest All authors certify that they have no affiliations with or involvement in any organization or entity with any financial interest or non-financial interest in the subject matter or materials discussed in this manuscript.

References

1. Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A, Riedmiller M, Fidjeland AK, Ostrovski G et al (2015) Human-level control through deep reinforcement learning. Nature 518:529–533
2. Harper F Maxwell, Konstan Joseph A (2015) The movielens datasets: history and context. ACM Trans Interact Intell Syst (TiiS) 5:1–19
3. Wang Q, Mao Z, Wang B, Guo L (2017) Knowledge graph embedding: a survey of approaches and applications. IEEE Trans Knowl Data Eng 29:2724–2743
4. Ji S, Pan S, Cambria E, Marttinen P, Yu Philip S (2020) A survey on knowledge graphs: representation, acquisition and applications. arXiv preprint arXiv:2002.00388
5. Bordes A, Usunier N, Garcia-Duran A, Weston J, Yakhnenko O (2013) Translating embeddings for modeling multi-relational data. In NIPS, pp 2787–2795
6. Trouillon T, Welbl J, Riedel S, Gaussier É, Bouchard G (2016) Complex embeddings for simple link prediction. In: Proceedings of the 33nd international conference on machine learning (ICML), pp 2071–2080
7. Liu Q, Jiang H, Evdokimov A, Ling Z-H, Zhu X, Wei S, Hu Y (2016) Probabilistic reasoning via deep learning: neural association models. arXiv preprint arXiv:1603.07704
8. Bai X, Wang M, Lee I, Yang Z, Kong X, Xia F (2020) Scientific paper recommendation: a survey. arXiv e-prints
9. Zhang S, Yao L, Sun A, Tay Y (2019) Deep learning based recommender system: a survey and new perspectives. ACM Comput Surv (CSUR) 52:1–38
10. Zhang L, Luo T, Zhang F, Wu Y (2018) A recommendation model based on deep neural network. IEEE Access 6:9454–9463
11. He X, Liao L, Zhang H, Nie L, Hu X, Chua TS (2017) Neural collaborative filtering. In Proceedings of the 26th international conference on world wide web, pp 173–182
12. Hidasi B, Karatzoglou A, Baltrunas L, Tikk D (2016) Session-based recommendations with recurrent neural networks. In: 4th International conference on learning representations (ICLR)
13. Quadrana M, Karatzoglou A, Hidasi B, Cremonesi P (2017) Personalizing session-based recommendations with hierarchical recurrent neural networks. In: Proceedings of the eleventh ACM conference on recommender systems, pp 130–137
14. He S, Zhang M, Fang H, Liu F, Luan X, Ding Z (2020) Reinforcement learning and adaptive optimization of a class of Markov jump systems with completely unknown dynamic information. Neural Comput Appl 32(18):14311–14320
15. Liu F, Tang R, Li X, Zhang W, Ye Y, Chen H, Guo H, Zhang Y (2018) Deep reinforcement learning based recommendation with explicit user-item interactions modeling. arXiv preprint arXiv:1810.12027
16. Zhao X, Gu C, Zhang H, Liu X, Yang X, Tang J (2019) Deep reinforcement learning for online advertising in recommender systems. arXiv preprint arXiv:1909.03602
17. Zhao W, Wu R, Liu H (2016) Paper recommendation based on the knowledge gap between a researcher's background knowledge and research target. Inf Process Manag 52:976–988
18. Xia F, Liu H, Lee I, Cao L (2016) Scientific article recommendation: exploiting common author relations and historical preferences. IEEE Trans Big Data 2:101–112
19. Huang Z, Chung W, Ong T-H, Chen H (2002) A graph-based recommender system for digital library. In: Proceedings of the 2nd ACM/IEEE-CS joint conference on digital libraries, place, pp 65–73
20. Guo Q, Zhuang F, Qin C, Zhu H, Xie X, Xiong H, He Q (2020) A survey on knowledge graph-based recommender systems. arXiv preprint arXiv:2003.00911
21. Di Noia T, Ostuni VC, Tomeo P, Di Sciascio E (2016) Sprank: semantic path-based ranking for top-n recommendations using linked open data. ACM Trans Intell Syst Technol (TIST) 8:1–34
22. Zhang Y, Ai Q, Chen X, Wang P (2018) Learning over knowledge-base embeddings for recommendation. arXiv preprint arXiv:1803.06540
23. He R, Kang W-C, McAuley J (2017) Translation-based recommendation. In: Proceedings of the eleventh ACM conference on recommender systems, pp 161–169
24. Wang H, Zhang F, Wang J, Zhao M, Li W, Xie X, Guo M (2018) Ripplenet: propagating user preferences on the knowledge graph for recommender systems. In: Proceedings of the 27th ACM international conference on information and knowledge management (CIKM), pp 417–426
25. Wang H, Zhang F, Zhao M, Li W, Xie X, Guo M (2019) Multi-task feature learning for knowledge graph enhanced recommendation. In: The World Wide Web conference, pp 2000–2010
26. Wang H, Zhao M, Xie X, Li W, Guo M (2019) Knowledge graph convolutional networks for recommender systems. In: The World Wide Web conference, pp 3307–3313
27. Zhenzhen X, Jiang H, Kong X, Kang J, Wang W, Xia Feng (2016) Cross-domain item recommendation based on user similarity. Comput Sci Inf Syst 13:359–373
28. Niu J, Wang L, Liu X, Yu S (2016) FUIR: fusing user and item information to deal with data sparsity by using side information in recommendation systems. J Netw Comput Appl 70:41–50

29. Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2014) Generative adversarial nets. Adv Neural Inf Process Syst pp 2672–2680
30. Xie Q, Hovy E, Luong M-T, Le QV, (2020) Self-training with noisy student improves ImageNet classification. In: 2020 IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 10684–10695
31. Chivukula AS, Liu W (2019) Adversarial deep learning models with multiple adversaries. IEEE Trans Knowl Data Eng 31:1066–1079
32. Szegedy C, Zaremba W, Sutskever I, Bruna J, Erhan D, Goodfellow I, Fergus R (2014) Intriguing properties of neural networks. In: 2nd International conference on learning representations (ICLR)
33. Goodfellow IJ, Shlens J, Szegedy C (2015) Explaining and harnessing adversarial examples. In: 3rd International conference on learning representations (ICLR)
34. Shafahi A, Huang WR, Najibi M, Suciu O, Studer C, Dumitras T, Goldstein T (2018) Poison frogs! Targeted clean- label poisoning attacks on neural networks. In Proceedings of the 32nd Conference on Neural Information Processing Systems, Montreal, Canada, pp 6103–6113
35. Carlini N, Wagner D (2017) Towards evaluating the robustness of neural networks. In: 2017 IEEE symposium on security and privacy (SP), pp 39–57
36. Athalye A, Engstrom L, Ilyas A, Kwok K (2018) Synthesizing robust adversarial examples. In: Proceedings of the 35th international conference on machine learning (ICML), pp 284–293
37. Sharif M, Bhagavatula S, Bauer L, Reiter MK (2016) Accessorize to a crime: real and stealthy attacks on state-of-the-art face recognition. In: Proceedings of the 2016 ACM sigsac conference on computer and communications security, pp 1528–1540
38. Sun L, Wang J, Yu PS, Li B (2018) Adversarial attack and defense on graph data: a survey. arXiv preprint arXiv:1812.10528
39. Jin W, Li Y, Xu H, Wang Y, Tang J (2020) Adversarial attacks and defenses on graphs: a review and empirical study. arXiv preprint arXiv:2003.00653
40. Bojchevski A, Günnemann S (2019) Adversarial attacks on node embeddings via graph poisoning. In: Proceedings of the 36th international conference on machine learning (ICML), pp 695–704
41. Chen Y, Nadji Y, Kountouras A, Monrose F, Perdisci R, Antonakakis M, Vasiloglou N (2017) Practical attacks against graph-based clustering. In: Proceedings of the 2017 ACM SIGSAC conference on computer and communications security, pp 1125–1142
42. Chen J, Shi Z, Wu Y, Xu X, Zheng H (2018) Link prediction adversarial attack. arXiv preprint arXiv:1810.01110
43. Zügner D, Akbarnejad A, Günnemann S (2018) Adversarial attacks on neural networks for graph data. In: Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining, pp 2847–2856
44. Kipf TN, Welling M (2017) Semi-supervised classification with graph convolutional networks. In: 5th International conference on learning representations (ICLR)
45. Dai H, Li H, Tian T, Huang X, Wang L, Zhu J, Song L (2018) Adversarial attack on graph structured data. In: Proceedings of the 35th international conference on machine learning (ICML), pp 1123–1132
46. Zhang H, Zheng T, Gao J, Miao C, Su L, Li Y, Ren K (2019) Data poisoning attack against knowledge graph embedding. In: Proceedings of the 28th international joint conference on artificial intelligence (IJCAI), pp 4853–4859
47. Xian X, Wu T, Qiao S, Wang W, Wang C, Liu Y, Xu G (2021) DeepEC: Adversarial attacks against graph structure prediction models. Neurocomputing 437:168–185
48. Chen L, Xu Y, Xie F, Huang M, Zheng Z (2021) Data poisoning attacks on neighborhood-based recommender systems. Trans Emerg Telecommun Technol 32(6):e3872
49. Huang H, Mu J, Gong NZ, Li Q, Liu B, Xu M(2021) Data poisoning attacks to deep learning based recommender systems. arXiv preprint arXiv:2101.02644
50. Fang M, Yang G, Gong NZ, Liu J (2018) Poisoning attacks to graph-based recommender systems. In: Proceedings of the 34th annual computer security applications conference, pp 381–392
51. Tang J, Du X, He X, Yuan F, Tian Q, Chua T-S (2019) Adversarial training towards robust multimedia recommender system. IEEE Trans Knowl Data Eng 32:855–867
52. Anelli VW, Deldjoo Y, Noia T Di, Di Sciascio E, Merra FA. Semantics-aware shilling attacks against collaborative recommender systems via knowledge graphs
53. Tang J, Wen H, Wang K (2020) Revisiting adversarially learned injection attacks against recommender systems. In: Fourteenth ACM conference on recommender systems, pp 318–327
54. Fang M, Gong NZ, Liu J (2020) Influence function based data poisoning attacks to top-n recommender systems. In: Proceedings of the web conference 2020, pp 3019–3025