# A Knowledge-Enhanced Deep Recommendation Framework Incorporating GAN-based Models

Deqing Yang*‡, Zikai Guo*, Ziyi Wang*, Junyang Jiang†, Yanghua Xiao†§ ¶ and Wei Wang†
*School of Data Science, Fudan University, Shanghai Key Laboratory of Data Science, Shanghai, China
*Email: {yangdeqing, 14307130162, 17210980022}@fudan.edu.cn
‡CETC Big Data Research Institute Co. Ltd., Guizhou, China
†School of Computer Science, Fudan University. Email: {15307130102, shawyh, weiwang1}@fudan.edu.cn
§Alibaba Group, Zhejiang, China
¶Shanghai Institute of Intelligent Electronics & Systems, Shanghai, China.

***Abstract*—Although many researchers of recommender systems have noted that encoding user-item interactions based on DNNs promotes the performance of collaborative filtering, they ignore that embedding the latent features collected from external sources, e.g., knowledge graphs (KGs), is able to produce more precise recommendation results. Furthermore, CF-based models are still vulnerable to the scenarios of sparse known user-item interactions. In this paper, towards movie recommendation, we propose a novel knowledge-enhanced deep recommendation framework incorporating GAN-based models to acquire robust performance. Specifically, our framework first imports various feature embeddings distilled not only from user-movie interactions, but also from KGs and tags, to constitute initial user/movie representations. Then, user/movie representations are fed into a generator and a discriminator simultaneously to learn final optimal representations through adversarial training, which are conducive to generating better recommendation results. The extensive experiments on a real Douban dataset demonstrate our framework's superiority over some state-of-the-art recommendation models, especially in the scenarios of sparse observed user-movie interactions.**

***Keywords*-recommendation, embedding, knowledge graph, GAN, adversarial training, user-item interactions**

## I. Introduction

Recommender systems have played a significant role of web services in the era of big data. As the most popular recommendation class, CF-based methods suffer from the problem of *data sparsity* because they infer user preferences and item characteristics based on observed user-item interactions, such as ratings, review and purchase records. Thus the *cold-start* users/items not observed in any interaction can not be learned well, resulting in failed recommendation. In recent years, inspired by deep learning's (DL for short) power on computer vision, image and natural language processing, more and more researchers also employ deep neural networks (DNNs for short) in recommender systems for different goals, such as alleviating the problem of data sparsity [1], [2]. Although these DL-based models exhibit powerful performance, they still have some drawbacks as follows.

Deqing Yang is corresponding author. This paper is supported by National Key R&D Program of China No.2017YFC0803700, by National Key Basic Research Program of China No.2015CB358800, by National Key R&D Program of China No.2017YFC1201200, by Chinese NSFC Project (No.61472085, No.U1509213, No.U1636207), by STCSMs key project (No.15JC1400900, No.16511102102, No.16JC1420401), STCSMs R&D Program under Grant No.16JC1420400.

1) In many of DL-based recommendation models, user/item representations are still generated based on observed user-item interactions, resulting in restricted performance in the scenarios of sparse user-item interactions.

2) For those models incorporating knowledge graphs (KGs for short) as auxiliary information to promote recommendation [5], [6], they pay more attention to composing an item's representation through embedding its mapped entity in KGs. Comparatively, in these models, a user is only represented by the fusion of his/her favorite/clicked/watched items' representations rather than knowledge embeddings. Such single type of user embeddings can not characterize a user sufficiently, thus restricts final recommendation performance. In fact, some sources other than KGs can also be utilized to enrich user representations, such as tag information.

3) Wang et al. proposed IRGAN [7] based on generative and adversarial networks (GANs for short) [8], in which a generator and a discriminator try to beat each other for accomplishing information retrieval tasks. Although the authors have demonstrated IRGAN's capability for item recommendation task, they only fed the models with user/item representations initialized in random. Thus a costly pre-training is inevitable.

To address above problems, we propose a novel GAN-based recommendation framework towards a specific recommendation task, i.e, Douban (https://movie.douban.com) movie recommendation. At first, in order to enrich user/movie representations, our framework incorporates various feature vectors, i.e., knowledge embeddings distilled from KGs and tag embeddings distilled from tag corpus, respectively. Specifically, a movie's knowledge embedding is the combination of its entity embedding and context embedding which are both generated through a KG embedding algorithm. On the other hand, a user's knowledge embedding is the average of his/her historical favorite movies' knowledge embeddings. Furthermore, tag embeddings encode co-occurrence relationships between tags, indicating more correlations between users and movies. Therefore, appending tag embeddings to knowledge embeddings is effective, which is justified by our experiments. Then, for the users observed in training set and candidate movies, their representations are simultaneously fed into a generator $G$ and a discriminator $D$ constructed based on IRGAN. Through adversarial training, $G$ and $D$ adjust user/movie representations to the optimal vectors which are

978-1-5386-9159-5/18/$31.00 ©2018 IEEE
DOI 10.1109/ICDM.2018.00187

conducive to final recommendation.

To the best of our knowledge, this is the first work of feeding GAN-based models with meaningful representations generated by DNNs, to address the challenge of the recommendation with sparse user-item interactions. The main contributions of this paper are summarized as follows:

1) We propose a recommendation framework which incorporates various meaningful embeddings distilled from different sources rather than randomly initialized representations, and then feed GAN-based models with the embeddings for accomplishing the recommendation in the scenarios of sparse user-item interactions.

2) The results of extensive experiments not only demonstrate our framework's robust advantage over the state-of-the-art recommendation models, but also justify the necessity of incorporating various meaningful embeddings for better model learning.

The rest of this paper is organized as follows. We introduce the details of our recommendation framework in Section 2. Section 3 and Section 4 show our experiment results and related work, respectively. At last, we conclude our work in Section 5.

## II. Solution

### A. The Framework

Our framework is illustrated in Fig. 1, which consists of two phases. The first one is fusing various embeddings distilled from different sources, into initial user/movie representations. The second one is feeding user/movie representations to a generator and a discriminator which are designed based on IRGAN [7], to issue an adversarial training. At last, the outputs from the generator or the discriminator are used to accomplish top-N movie recommendation.

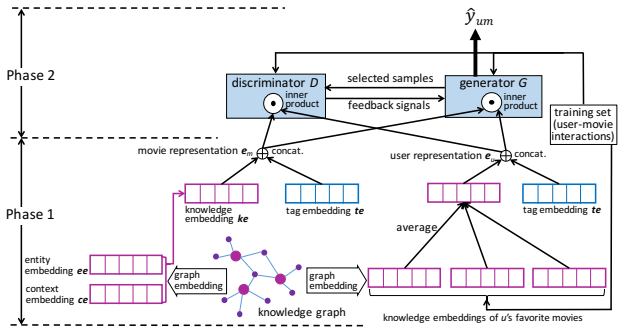



Fig. 1. GAN-based recommendation framework incorporating movie knowledge embeddings and tag embeddings.

*The First Phase:* In the first phase, various embeddings are distilled from different sources in order to fully represent users/movies. At first, we import KG's knowledge to generate feature embeddings for movies. Specifically, we get a movie's *entity embedding*, denoted as $\boldsymbol{ee}$, though a KG embedding algorithm so long as the movie can be mapped into a KG entity. Meanwhile, the neighbor nodes of a movie's entity node in a KG are generally the movie's attributes, e.g., directors, actors and genres. These attributes embody latent relationships across different movies as illustrated in Fig. 2, implying that the embeddings of attribute nodes, namely *attribute embeddings*, also contribute to a movie's representation as its entity embedding. Thus, a movie's *context embedding*, denoted as $\boldsymbol{ce}$, is the weighted sum of its attribute embeddings. Then, we aggregate $\boldsymbol{ee}$ and $\boldsymbol{ce}$ as a movie's *knowledge embedding*, namely $\boldsymbol{ke}$. The detailed process of this step is introduced in Section II-B. In addition, every user and movie in Douban have a set of descriptive tags, such as movie genres, directors and countries. Therefore, we further apply Word2Vec [9] tool to generate a vector for each tag through inputting all users/movies' tag sets. Then, a user/movie's *tag embedding*, denoted as $\boldsymbol{te}$, is the average of his/her/its all tags' vectors. When a movie's $\boldsymbol{ke}$ and $\boldsymbol{te}$ are both generated, we concatenate these two vectors as the movie's representation. On the other hand, a user's knowledge embedding is the average knowledge embedding of the movies which are the user's favorite movies found from user-movie interactions. If we can not know a user's favorite movies from observed user-movie interactions, we set the user's knowledge embedding as the average of all movies' knowledge embeddings. At last, a user's representation is the concatenation of his/her tag embedding and knowledge embedding.

*The Second Phase:* In this phase, a generator $G$ and a discriminator $D$ are constructed to learn optimal representations of the users found from training set and candidate movies. The learned optimal user/movie representations are very crucial to achieve precise recommendation. $G$ and $D$ are simultaneously fed with user/movie representations which are generated in the first phase. These two models play the same roles as in IRGAN [7] during the process of adversarial training. In generic IRGAN, $G$ and $D$ are both initialized in random and pre-trained, which is generally costly. In our framework, the step of pre-training can be skipped since the representations fed into $G$ and $D$ are more indicative than randomly initialized vectors. Meanwhile, the observed user-movie interactions are provided to the two models for labeling true positive samples of adversarial training. The final score $\hat{y}_{um}$ generated by $G$ indicates the extent to which a candidate movie $m$ would be preferred by a given user $u$. The detailed process of this phase is presented in Section II-C.

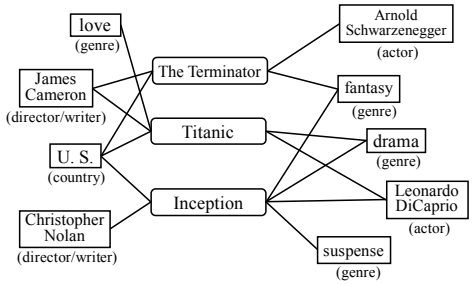



Fig. 2. A toy example of movie(entity)'s knowledge graph. The latent relationship/similarity between two different movies can be found through their shared attributes, e.g., director, actor or genre.

### B. Distilling Movie's Knowledge Embeddings

At first, we construct a sub-KG involving Douban movies by extracting relevant knowledge from an open Chinese KG, namely CN-DBpedia[1] [10]. In the KG, a movie is generally regarded as an entity. A movie's neighbor nodes are its attributes

[1] http://kw.fudan.edu.cn/cndbpedia/

including director, actor, genre and etc. Fig. 2 illustrates a toy example of such KG, which is often recognized as a *heterogeneous information network* (HIN for short). From the graph, we can find some relationships between movies through their shared attributes. For example, we believe that movie $Titanic$ and $Inception$ are two related or similar movies because they both star actor $DiCaprio$. Hence the fans of $DiCaprio$ may like these two movies simultaneously. In terms of KG topology, two similar movie nodes are linked through the paths traversing their shared attribute nodes. The more such paths exist, the more similar the two movies are. In this paper, we use Metapath2Vec [11] as KG embedding algorithm where each node (including movie/entity and attribute nodes) is represented as an embedding vector, since it has been proved to be a powerful state-of-the-art algorithm of HIN embedding. The algorithm ensures that, if two movies/entities are similar in terms of topology, their embeddings are close in the vector space.

Besides representing a movie with its entity embedding, we further use its contextual information, i.e, its attribute embeddings. Since a movie's attributes are usually very related to it w.r.t. semantics and logic, the usage of attribute embeddings could complement the identifiability of the movie. To calculate a movie's context embedding, we first average attribute embeddings in terms of attribute category. Specifically, suppose a movie $m$'s attribute category set is $A = \{actor, director, writer, genre, country\}$, for any category $a \in A$, $a$'s embedding w.r.t. $m$ is calculated as

$$\boldsymbol{e}_m^a = \frac{1}{|N_m^a|} \sum_{e_i \in N_m^a} \boldsymbol{e}_i \tag{1}$$

where $N_m^a$ is the group of $m$'s attribute nodes of which the category is $a$, and $\boldsymbol{e}_i$ is the embedding of attribute node $i$ learned by Metapath2Vec algorithm. Then, $m$'s context embedding is computed as weighted sum of all $\boldsymbol{e}_m^a$s, i.e.,

$$\boldsymbol{ce}_m = \sum_{a \in A} w_a \boldsymbol{e}_m^a \tag{2}$$

where $w_a$ is the normalized weight of attribute category $a$.

The quantification of $w_a$ is based on an intuition that, an attribute contributes less to seeking related movies if a member of this attribute connects more movies. For example, the movies starring a certain actor are more related than the movies belonging to a certain genre. More users recognize their favorite movies in terms of actors instead of genres. In other words, common actors indicate the similarity between two movies more evidently than common genres. Based on this intuition, we compute $w_a$ as follows. Given an attribute category $a$, a member node of $a$ is denoted as $i_a$, and the movie involving $i_a$ as $m_{i_a}$. The average number of movies involving $i_a$ is represented as $\overline{|\{m_{i_a}\}|}$. A larger $\overline{|\{m_{i_a}\}|}$ implies that $i_a$ connects more movies in the HIN. Then, we have

$$w_a = \frac{1}{Z} \times \frac{\overline{n_a}}{\log \overline{|\{m_{i_a}\}|}} \tag{3}$$

where $\overline{n_a}$ is the average number of $a$'s members involved by a movie and $Z$ is a normalization factor.

When a movie's $\boldsymbol{ee}$ and $\boldsymbol{ce}$ are obtained, we compute its knowledge embedding as

$$\boldsymbol{ke} = \alpha \boldsymbol{ee} + (1 - \alpha) \boldsymbol{ce} \tag{4}$$

where $\alpha \in (0, 1)$ is a coefficient to tune the balance between a movie's entity embedding and its context embedding.

For a user, we average the knowledge embeddings of his/her favorite movies which are found from observed user-movie interactions, as the user's knowledge embedding. For both movies and users, their representations are the concatenations of their knowledge embeddings and tag embeddings, which are the inputs of the models in the next phase.

### C. Learning Optimal Representations by GAN-based Models

In this phase, two models, i.e., a generator and a discriminator, adjust user/movie representations into the optimal values through adversarial training, in order to achieve precise recommendation. Specifically, they are both fed with user representations and movie representations to play a minimax game, which is indeed a process of adversarial training as GAN. This principle was first imported to information retrieval by IRGAN [7] which was designed for the task of searching relevant documents for a given query. In fact, movie recommendation can be easily mapped into this task if we regard a given user as a query, and regard candidate movies as candidate documents.

At first, given a user $u$, $u$'s *true favorite* movie distribution is defined as a conditional probability $p_{true}(m|u, r)$ where $r$ denotes the relevance between $u$ and the movie $m$. Then we construct the following two models:

*Generator G:* It tries to generate relevant (favorite) movies from the candidate pool for a given user, according to a conditional probability $p_\theta(m|u, r)$. In other words, it aims to let $p_\theta(m|u, r)$ approximate the true distribution $p_{true}(m|u, r)$ as much as possible.

*Discriminator D:* It tries to discriminate real-relevant user-movie pairs $< u, m >$ from irrelevant pairs, according to the probability $f_\phi(u, m)$. It is indeed a binary classifier in which real relevant pairs are positive samples and irrelevant pairs are negative samples.

*1) Optimization Objectives:* The goal of co-training $G$ and $D$ is to learn the parameters in $p_\theta(m|u, r)$ and $f_\phi(u, m)$, i.e., $\theta$ and $\phi$. During the process of training, $G$ and $D$ have adversarial goals to beat each other. The generator tries to select relevant user-movie pairs that look like the ground-truth, in order to confuse the discriminator. Whereas the discriminator tries to draw an optimal distinction between the ground-truth pairs and the ones selected by its opponent generator. Therefore, we have the following global objective:

$$\begin{aligned} \mathcal{O} = \min_\theta \max_\phi \sum_{n=1}^{N} \big\{ & \mathbb{E}_{m \sim p_{true}(m|u_n, r)} \big[ \log P(m|u_n) \big] \\ & + \mathbb{E}_{m \sim p_\theta(m|u_n, r)} \big[ \log \big( 1 - P(m|u_n) \big) \big] \big\} \end{aligned} \tag{5}$$

where $P(m|u_n)$ estimates the probability of movie $m$ being preferred by the given user $u_n$, and is computed by sigmoid function of $f_\phi$:

$$P(m|u) = \sigma(f_\phi(u, m)) = \frac{1}{1 + exp(-f_\phi(u, m))} \tag{6}$$

According to this global objective, the discriminator's objective is to maximize the log-likelihood of correctly distinguishing the real-relevant pairs and the relevant paris selected by current optimal $p_{\theta^*}(m|u,r)$. Thus the optimal parameters of the discriminator $D$ are learned by

$$\phi^* = \arg\max_{\phi} \sum_{n=1}^{N} \{\mathbb{E}_{m\sim p_{true}(m|u_n,r)}\big[\log \sigma(f_\phi(u_n,m))\big] + \mathbb{E}_{m\sim p_{\theta^*}(m|u_n,r)}\big[\log\big(1-\sigma(f_\phi(u_n,m))\big)\big]\} \tag{7}$$

This maximization is solved by stochastic gradient descent.

In contrast, the generator $G$ intends to let $p_\theta(m|u,r)$ fit with $p_{true}(m|u,r)$. Based on $p_\theta(m|u,r)$, $G$ randomly selects samples (user-movie pairs) from candidate pool to confuse $D$. Thus, when keeping $f_\phi(u,m)$ fixed after its maximization, the optimal parameters of $G$ is learned by the following minimization:

$$\begin{aligned}\theta^* &= \arg\min_{\theta} \sum_{n=1}^{N} \{\mathbb{E}_{m\sim p_{true}(m|u_n,r)}\big[\log \sigma(f_\phi(u_n,m))\big] \\ &+ \mathbb{E}_{m\sim p_\theta(m|u_n,r)}\big[\log\big(1-\sigma(f_\phi(u_n,m))\big)\big]\} \\ &= \arg\max_{\theta} \sum_{n=1}^{N} \{\mathbb{E}_{m\sim p_\theta(m|u_n,r)}\big[\log\big(1+exp(f_\phi(u_n,m))\big)\big]\end{aligned} \tag{8}$$

As in GAN, this minimization can not be solved by gradient descent directly, so we can instead use policy gradient based reinforcement learning [7].

Then, similar to IRGAN, we use softmax function to compute $p_\theta$ as

$$p_\theta(m_k|u,r) = \frac{exp(s_\theta(u,m_k))}{\sum_m exp(s_\theta(u,m))} \tag{9}$$

where $m_k$ is the selected movie by the generator to confuse the discriminator, and $s_\theta(u,m)$ quantifies the chance of $m$ being selected w.r.t. $u$.

*2) Feeding with User/Movie Representations:* In order to train $G$ and $D$, we use the following score function to compute $f_\phi(u,m)$ in Eq. 6∼8 and $s_\theta(u,m)$ in Eq. 9,

$$s(u,m) = \boldsymbol{e}_u \cdot \boldsymbol{e}_m + b_m \tag{10}$$

where $\boldsymbol{e}_u$ and $\boldsymbol{e}_m$ are $u$'s and $m$'s representation, respectively. $b_m$ is the bias for movie $m$. According to above objective functions, both $\boldsymbol{e}_u$, $\boldsymbol{e}_m$ and $b_m$ are the parameters of $G$ and $D$, which are learned/adjusted during adversarial training. At the beginning of adversarial training, $\boldsymbol{e}_u$ and $\boldsymbol{e}_m$ are just the representations generated in the first phase. At the end of adversarial training, $\boldsymbol{e}_u$ and $\boldsymbol{e}_m$ are optimal for the downstream recommendation. As illustrated in Fig. 1, $\boldsymbol{e}_u$ and $\boldsymbol{e}_m$ are fed into $G$ and $D$ separately. Hence them are adjusted by $G$ and $D$ separately during adversarial training, since $G$ and $D$ have different objectives. In generic IRGAN, $G$ and $D$ are fed with user/movie representations initialized in random. We will prove in the experiment section that the meaningful representations generated in the first phase are more effective and efficient than random representations for adversarial training.

As introduced in Section III-A, we divide all samples (user-movie pairs) into training set and test set to demonstrate our framework towards movie recommendation. The observed real-relevant pairs are the positive samples in the training set, and the unobserved real-relevant pairs are the positive samples in the test set which are the targets to be predicted by the models. According to the principle of adversarial training, both $G$ and $D$ try to score observed/unobserved positive user-movie pairs higher than negative pairs. Therefore, we can use either $G$'s score function $p_\theta(m|u,r)$ or $D$'s score function $f_\phi(u,m)$ as the final output score $\hat{y}_{um}$, to indicate whether $m$ deserves being recommended to $u$. In our experiments, we select $G$ which has nearly the same performance as $D$ in fact.

## III. Evaluation

In this section, we conduct extensive experiments aiming to answer the following research questions:

*RQ1*: Does our framework have better performance than previous recommendation models especially in the scenarios of sparse observed interactions?

*RQ2*: With respective to the learning of $G$ and $D$, are knowledge embeddings and tag embeddings more efficient than the ones initialized in random or based on user-item interactions?

*RQ3*: Is it useful to incorporate more embeddings to represent users/movies, for the goal of promoting final recommendation performance?

### A. Experiment Settings

*1) Dataset Description:* The knowledge data of CN-DBpedia can be fetched through open APIs in the website of Knowledge Works. The Douban dataset is published on *https://doi.org/10.7910/DVN/WCXPQA*, in which there are about 42,000 movies and 5,000 users, including movie/user tags and users' rating scores on movies. In order to implement our framework, we collected the attributes of all these movies from CN-DBpedia. In total, we collected 89,909 unique tags for the Douban dataset.

A movie is identified as a user's favorite movie if the user rates the movie with a score of 4 or 5, which is treated as a positive sample of user-movie interaction. For a given user, we randomly selected popular movies which have not been rated by the user as disliked (negative) movie. For better evaluating the performance of recommendation, for each test user we collected his/her negative movies of which the number is the same as the number of his/her positive movies, i.e., at least 8 positive movies and 8 negative movies.

*2) Sample Division:* Since user/movie representations are generated based on KGs and tags, instead of user-movie interactions, we believe our framework works better than other DL-based recommendation models when no or very few user-movie interactions have been observed. To justify it, we divided all interactions into training set (as observed pairs) and test set (as unobserved pairs). The small training set simulates the scenario of sparse known interactions. To be specific, we randomly selected some interactions from the whole dataset as training set according to a certain proportion, denoted as $r$, and the rest records were used as test set. A smaller $r$ indicates fewer interactions can be observed. According to this setting, CF-based recommendation models can only utilize the known interactions in training set.

*3) Evaluation Metrics:* We used three popular metrics for evaluating top-N recommendation, i.e., Prec. (Precision), AP (Average Precision), nDCG (Normailzed Discounted Cumulative Gain), to evaluate the performance of all compared methods. In the following results we report performance scores of top-N recommendations averaged on all test users in test set.

*4) Baselines:* We compare our framework with some new and popular recommendation models, to justify our framework's superiority.

*COS*: This is a straightforward baseline in which a user/movie representation is composed by the same process as our framework, i.e, it is the concatenation of a knowledge embedding and a tag embedding. Then, a given user's preference for a candidate movie can be inferred directly based on the cosine distance between their representation vectors instead of feeding them into GAN-based models.

*NFM*: It is Neural Factorization Machine [3] which was proposed for prediction under sparse settings. NFM seamlessly combines the linearity of FM [12] in modeling second-order feature interactions and the non-linearity of neural network in modeling higher-order feature interactions. According to FM's basic idea, only user-movie interactions are inputted into NFM.

*NCF*: Neural Collaborative Filtering [4] is a DNN-based CF framework, consisting of a GMF (generalized matrix factorization) layer and an MLP (multi-layer perceptron). Both GMF and MLP are fed with user/item latent vectors which are initialized in random. Then their outputs are fused through a NeuMF layer to generate a predicted score. All parameters of the networks are learned based on observed user-item interactions.

*IRGAN*: This is the generic IRGAN model [7] where user/movie representations are initialized in random instead of meaningful embeddings as ours. Thus a pre-training is critical for IRGAN to gain better performance.

*KGAN*: This is a weak variant of our framework, in which user/movie representations are only their knowledge embeddings.

At last, our proposed framework is denoted as *KTGAN* in which the generator and the discriminator are both fed with user/movie representations constituted by knowledge embeddings and tag embeddings.

### B. Experiment Results

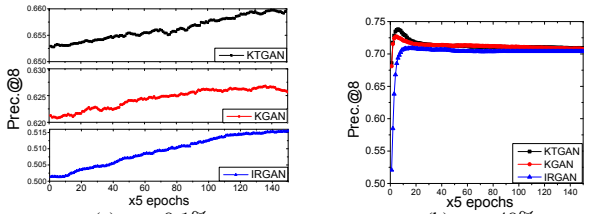
All experiments were conducted on a workstation of which the GPU is NVIDIA GeForce GTX 1080Ti and RAM is 32G. The DNN framework was implemented by tensorflow+python. When implementing KTGAN, we set $\alpha = 0.9$ and embedding dimension to 100 which were both learned through tuning experiments[2].

*1) Recommendation Performance:* As we stated before, our framework has robust performance in the scenarios of sparse user-movie interactions since users/movies are both represented based on knowledge embeddings and tag embeddings, rather than historical interactions. Table I lists all competitors' performance of the three metrics when $r$=0.1%. We concern this scenario because there are only less than 3% users being observed in training set, which is an extreme scenario of sparse user-movie interactions. The results give RQ1 an affirmative answer. KTGAN outperforms COS, justifying GAN-based models' powerful performance on learning latent relationships between users and movies. NFM and NCF also perform very weakly because they can only utilize very few interactions to infer user preferences, especially for the cold-start users. We also answer yes to RQ3 according to KTGAN's advantage over KGAN.

TABLE I
TOP-N MOVIE RECOMMENDATION RESULTS WHEN $r = 0.1\%$.

| Model | Precision | | | Average Precision | | | nDCG | | |
|---|---|---|---|---|---|---|---|---|---|
| | @3 | @5 | @8 | @3 | @5 | @8 | @3 | @5 | @8 |
| COS | 0.568 | 0.535 | 0.521 | 0.478 | 0.409 | 0.354 | 0.581 | 0.562 | 0.529 |
| NFM | 0.560 | 0.532 | 0.511 | 0.476 | 0.408 | 0.355 | 0.576 | 0.551 | 0.533 |
| NCF | 0.556 | 0.530 | 0.515 | 0.476 | 0.407 | 0.358 | 0.579 | 0.538 | 0.555 |
| IRGAN | 0.556 | 0.525 | 0.510 | 0.476 | 0.404 | 0.354 | 0.577 | 0.550 | 0.533 |
| KGAN | 0.723 | 0.678 | 0.628 | 0.661 | 0.583 | 0.506 | 0.738 | 0.703 | 0.663 |
| KTGAN | **0.759** | **0.719** | **0.657** | **0.701** | **0.631** | **0.545** | **0.771** | **0.741** | **0.693** |

Furthermore, we exploited how more observed user-movie interactions help the models perform better. To this end, we evaluated all models' performance as $r$ increases gradually from 0.1% to 40%. Due to space limitation, we only report all models' performance scores on top-8 movie recommendation in Fig. 3. The results of other top-N recommendations also support the same conclusions. From the figure, we find that all models enhance their performance as $r$ rises up. As $r$ steps close to 40%, all baselines catch up or approach KTGAN/KGAN, implying that these CF-based deep models can utilize interactions to infer user preference precisely and quickly. KTGAN's superiority over KGAN in all scenarios also affirms RQ3 further.

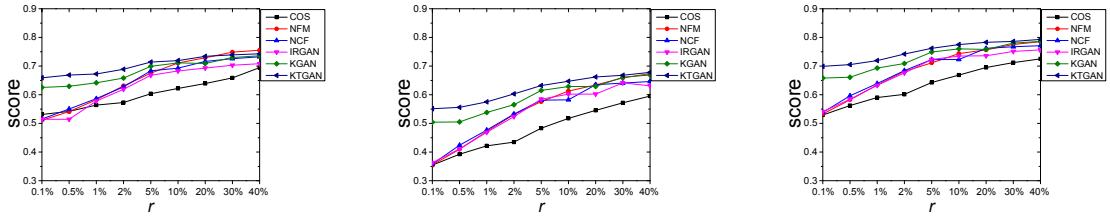


(a) $r = 0.1\%$ (b) $r = 40\%$

Fig. 4. Learning curves of the generator in terms of Prec.@8 when $r = 0.1\%$ and $r = 40\%$. KTGAN has the best learning capability because it is fed with more meaningful user/movie representations.

*2) Model Learning:* Adversarial training is unstable although it has been widely regarded as effective solution. Therefore, we further compare the learning trend of different GAN-based models, i.e., IRGAN, KGAN and KTGAN, to justify the meaningful embeddings (in KGAN/KTGAN) are more useful for learning $G$ and $D$ than randomly initialized inputs (in IRGAN). To let the models fed with random inputs arrive their optimal states faster, we set IRGAN's learning rate as 5 times of the one of KGAN/KTGAN. Fig. 4 shows the generator's learning curves of the three frameworks in terms of Prec.@8 when $r$=0.1%, from which we find the generator's performance varies as epoch number increases from 1 to 750 epochs. The results show that KTGAN's curve has a higher start point (in 0 epoch) and stay higher consistently than its

[2]We have tested different settings of embedding dimension in our experiments and found that they have the nearly same performance. So we only report the results of 100 dimension.




(a) Prec.@8 (b) AP@8 (c) nDCG@8

Fig. 3. Performance comparisons of top-8 Douban movie recommendation. The results show that our proposed KTGAN are superior to its competitors particularly when observing rare user-movie interactions.

rivals, resulting in better performance in the scenario of so sparse interactions. In contrast, IRGAN's curve stays at the lowest position, and can not arrive its optimal state even when epoch number has been 750, although it has a much bigger learning rate. We also compared the competitors' learning curves in the scenario of sufficient interactions ($r$=40%), and the results are displayed in Fig. 4. The results show that although IRGAN can arrive its optimal state much faster (within 60 epochs) than the scenario of $r$=0.1%, it still falls behind with KGAN/KTGAN. Hence, we give an affirmative answer to RQ2.

*3) Discussions:* 1) The outperformance of KGAN and KTGAN in the scenarios of sparse interactions is attributed to representing users and movies based on knowledge embeddings and tag embeddings rather than observed user-movie interactions, which inspires us to alleviate the cold-start problem by resorting to external sources such as KGs.

2) The reason of KTGAN's advantage over KGAN is two-fold. First, besides knowledge embeddings, tag embeddings supply more indicative information to mine latent relationships between users/movies. Second, the knowledge embeddings of those cold-start users having no known favorite movies, are generated as global average from which less preference information can be fetched. Thus, KTGAN's advantage is more apparent when $r$ is very small.

3) IRGAN starts its adversarial training with randomly initialized representations far away from the optimal point. Comparatively, knowledge embeddings and tag embeddings are more close to the optimal representations since they encode sufficient latent relationships between users/movies. Thus, IRGAN's learning efficiency is much lower than KGAN/KTGAN. Especially in the scenario of sparse user-movie interactions (when $r$ is very small), too few interactions can not ensure the sufficient learning of $G$ and $D$ if they only start from a random state. It inspires us to initialize user/item representations with meaningful information rather than randomness.

## IV. Related Work

Many researchers in recommender systems have resorted to DNN to promote various recommendation tasks. The authors in [5] proposed a deep knowledge-aware network (DKN) based on CNN and attention network for news recommendation. In addition, traditional recommendation algorithms such as CF and matrix factorization, have been proved being improved by DNN-based models. [1] proposed a novel AutoEncoder (AE) framework for CF. [4] integrated generalized matrix factorization model (GMF) and multiple-layer perceptron (MLP) to predict CF-based implicit feedback.

In addition, many researchers found that abundant entities in KGs can be used as auxiliary information to overcome cold-start problem. Noia et al. [15] utilized knowledge from KGs to measure movie similarity for improving recommendation performance. More than using movie attributes in KGs, Zhang et al. [6] further fed text and image knowledge of movies in deep learning framework to achieve powerful recommendation performance. As our work, DKN [5] utilizes the embeddings of entities and entity contexts from KGs to predict the implicit feedbacks of news recommendation, but it does not incorporate tag embeddings.

## V. Conclusion

We propose a GAN-based recommendation framework incorporating tag embeddings and knowledge embeddings to generate meaningful user/item representations. The experiments prove that incorporating more indicative embeddings is more effective and efficient than random representations on model learning, especially ensures robust performance in the scenarios of sparse user-item interactions.

## References

[1] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie, "Autorec: Autoencoders meet collaborative filtering," in *Proc. of WWW*, 2015.
[2] H. Wang, N. Wang, and D.-Y. Yeung, "Collaborative deep learning for recommender systems," in *Proc. of KDD*, 2015.
[3] X. He and T. S. Chua, "Neural factorization machines for sparse predictive analytics," in *Proc. of SIGIR*, 2017.
[4] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T. S. Chua, "Neural collaborative filtering," in *Proc. of WWW*, 2017.
[5] H. Wang, F. Zhang, X. Xie, and M. Guo, "Dkn: Deep knowledge-aware network for news recommendation," 2018.
[6] F. Zhang, N. J. Yuan, D. Lian, X. Xie, and W. Y. Ma, "Collaborative knowledge base embedding for recommender systems," in *Proc. of KDD*, 2016.
[7] J. Wang, L. Yu, W. Zhang, Y. Gong, Y. Xu, B. Wang, P. Zhang, and D. Zhang, "Irgan: A minimax game for unifying generative and discriminative information retrieval models," in *Proc. of SIGIR*, 2017.
[8] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. of NIPS*, 2014.
[9] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *arXiv:1301.3781*, 2013.
[10] B. Xu, Y. Xu, J. Liang, C. Xie, B. Liang, W. Cui, and Y. Xiao., "Cn-dbpedia: A never-ending chinese knowledge extraction system," in *Proc. of ICIEA*, 2017.
[11] A. Swami, A. Swami, and A. Swami, "metapath2vec: Scalable representation learning for heterogeneous networks," in *Proc. of KDD*, 2017.
[12] S. Rendle, "Factorization machines," in *Proc. of ICDM*, 2010.
[13] X. Dong, L. Yu, ZhonghuoWu, Y. Sun, L. Yuan, and F. Zhang, "A hybrid collaborative filtering model with deep structure for recommender systems," in *Proc. of AAAI*, 2017.
[14] P. Vincent, H. Larochelle, Y. B. I. Lajoie, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *JMLR*, vol. 11, pp. 3371 - 3408, 2010.
[15] N. T. D, M. R, O. V. C, R. D, and Z. M, "Linked open data to support content-based recommender systems," in *Proc. of ICSS*, 2012.