Knowledge-Based Systems 257 (2022) 109903

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

TKGAT: Graph attention network for knowledge-enhanced tag-aware recommendation system

Beilun Wang [a,c], Haoqing Xu [a], Chunshu Li [a], Yuchen Li [b], Meng Wang [a,*]

[a] School of Computer Science and Engineering, Southeast University, Nanjing, 218889, Jiangsu, China
[b] School of Artificial Intelligence, Southeast University, Nanjing, 218889, Jiangsu, China
[c] Key Laboratory of Computer Network and Information Integration (Southeast University), Ministry of Education, China

ARTICLE INFO

Article history:
Received 19 April 2022
Received in revised form 13 September 2022
Accepted 13 September 2022
Available online 19 September 2022

Keywords:
Tag-aware recommendation
Recommendation systems
Knowledge graph
Social network

ABSTRACT

In recent practices, sparsity problems often arise in recommendation systems, resulting in weak generalization ability. To alleviate this problem, tag-aware recommendation systems (TRS) leverage personalized tags to enhance the modeling of user preferences and item characteristics. However, current tag-aware methods suffer from arbitrary user behaviors as they limit the additional information only to user tags. In this paper, we investigate a more general scenario, namely Knowledge-enhanced Tag-aware Recommendation System (KTRS) which involves auxiliary knowledge compared with TRS. Correspondingly, we propose a novel recommendation model for KTRS, called TKGAT. It firstly constructs a collaborative recommendation graph and then learns heterogeneous representation via an multi-layer multi-head attention mechanism. Experiments conducted on real-world datasets demonstrate that the proposed system outperforms the state-of-the-art recommendation methods, and show effectiveness of the auxiliary knowledge.

1. Introduction

With the exponential growth of web contents, recommendation techniques that present the users with what might interest them, are attracting increasing research attention [1–5]. In practice, recommendation systems have been widely used in various application fields, including search engines, E-commerce, content creation platforms, etc., with the intention of providing accurate suggestions for users [6–8]. To analyze the user preferences, traditional approaches, such as Collaborative Filtering method and its variants [9–11], construct historical item interactions (e.g., click, purchase, review, etc.) of users as an adjacency matrix. They then provide recommendations based on the similarity between user preference vectors. Such traditional recommendation systems suffer from sparsity problems [12]. For example, online E-commerce websites may have millions of users and items in their database. However, the connection between users and items is relatively sparse [13]. Each user may purchase and comment on only tens or hundreds of items, which are much less than the total amount of items. Therefore, finding proper candidate recommended items for each user will encounter high variance and randomness.

Many efforts have been made to overcome the sparsity problem in traditional recommendation systems. In recent years, the researchers introduce *tags* from social tagging systems into the recommendation systems, i.e. the Tag-aware Recommendation System (TRS). In the TRS, users give personalized tags to the items they interact with. These tags reflect users' preferences and item characteristics [14,15]. Therefore, they enrich information for item recommendation [16] and relieve the sparsity problem. To exploit the personalized tags, a common practice is to extract user and item representation vectors through a tag-involved deep neural network. They then learn to calculate recommendation scores in a supervising approach. For example, HTRM [17] uses LSTM to model user tagging behaviors. TNAM [18] applies attention mechanism in order to fuse user-related tags and item-related tags together. TGCN [19] utilizes GAT method [20] on a user-tag-item graph to learn the embedding vectors end-to-end.

Although the personalized tags provide supplementary information for recommendation systems, they also bring about inherent issues due to the arbitrariness of user behaviors as illustrated in the left part of Fig. 1. As shown in the example, users may assign contradictory tags to similar items, due to external factors such as judgments made by their friends. The contradictory tags provide opposite semantics, which pulls the representation vectors in opposite directions, causing local optimal problems. Additionally, some of the tags assigned by a user may reflect not the objective item properties but the user's subjective preference that is hardly significative to other users.

* Corresponding author.
E-mail addresses: beilun@seu.edu.cn (B. Wang), haoqing_xu@outlook.com (H. Xu), senseli.cn@gmail.com (C. Li), 213190968@seu.edu.cn (Y. Li), meng.wang@seu.edu.cn (M. Wang).

https://doi.org/10.1016/j.knosys.2022.109903
0950-7051/© 2022 Elsevier B.V. All rights reserved.

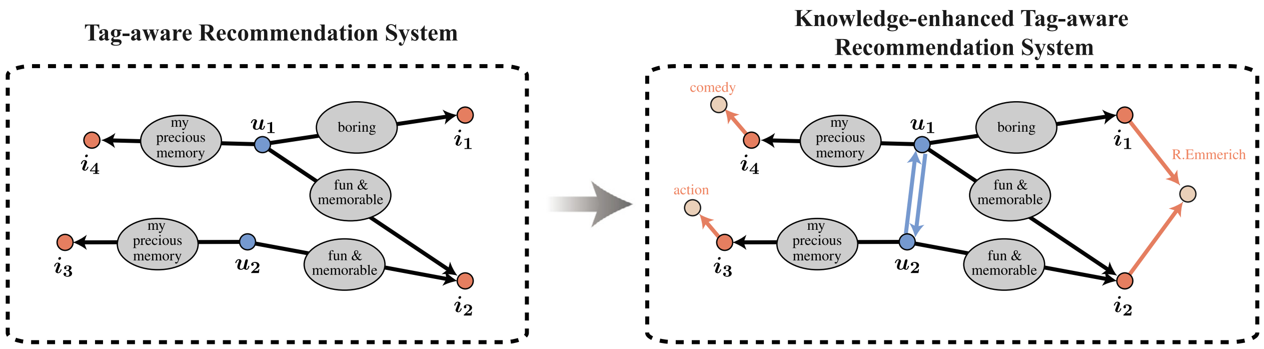


Fig. 1. An example comparison for TRS and KTRS for movie recommendation. Black arrows stand for tag assignments. Blue and orange arrows stand for auxiliary knowledge: friend relationships and knowledge graph, respectively. In TRS, model learn from tags only. Therefore, the contradictory tags would negatively affect the recommendation performance for the user (u_1) due to conflict semantics. Also, The item with the same tag (i_3, i_4) would tend to have similar representations. However, auxiliary knowledge involved in KTRS reveals that it is not the case. User u_1 has fun with his friend, which makes him assign "fun" tag to the movie, and the movies of their precious memory turn out to be different types.

The subjective preference will be propagated into items and other users related with these items, causing suboptimal recommendation performance for other users. These issues severely limit the performance of TRS methods.

Inspired by recent practices in recommendation systems, we aim to leverage reliable auxiliary knowledge to address the aforementioned issues of TRS. The AI community has seen the success of auxiliary knowledge in alleviating the sparsity problem in recommendation systems [21,22]. For example, DiffNet++ [23] uses multi-level attention mechanism to diffuse interests from items and social networks. KGAT [12] leverage GAT [20] to aggregate information from knowledge graphs. In real-world applications such as E-commerce and content creation platforms, the service providers have easy access to social networks, item knowledge graphs and user tags. However, existing studies has not considered the task setting where tags and auxiliary knowledge are jointly modeled in the recommendation system. Therefore, in this paper, we investigate a new task called Knowledge-enhanced Tag-aware Recommendation System (KTRS), which combines auxiliary knowledge with tag-aware recommendation systems. The auxiliary knowledge is expected to provide reliable and objective information for both users and items [24]. Thus, the potentially arbitrary behaviors of users will have less influence in KTRS than in TRS. Fig. 1 provides a comparison: the reliable auxiliary knowledge entities could anchor the semantics of the movie representations [25]. They help explain the contradictory tags and uncover the subjective tags.

Compared with knowledge-based recommendation systems, in KTRS, tags are tightly coupled with auxiliary knowledge in KTRS by narrowing the learning gap between the auxiliary knowledge and the user–item interactions. The information provided by the social network and knowledge graph is relatively separated since they only focus on one side of recommendation system (either users or items) [21]. In knowledge-based recommendation systems, the separated information is aggregated only by user–item interactions, which are insufficient to capture the correlational pattern of user preference information and auxiliary knowledge. By contrast, tagging interactions could combine the diluted information and capture the useful information for modeling user preference [16,19]. For example, if a user assigns the name of his favorite actor as his personalized tag to a movie that he watched, the system can easily know that the user is attracted to the movie due to the actor. Thus, the actor preference of this user will be highlighted and be aggregated from the knowledge graph to user–item interactions. To summarize, tags and auxiliary knowledge supplement each other in KTRS to overcome the problems of their own.

However, in order to fully exploit the huge potential of KTRS, there still exist two challenges to be resolved:

- **Modeling of tags and knowledge**. Current tag-aware recommendation methods do not consider auxiliary knowledge, making them have little extendibility. Also, since tagging interactions consist of three aspects of entities (users, items, and tags), recommendation methods using auxiliary knowledge do not have a simple way to model such kind of relationship. Therefore, how to model both tagging interactions and auxiliary knowledge in a united framework is remained to be researched.
- **High heterogeneity in graphs**. In KTRS, heterogeneous neighbors, such as users, items, tags, knowledge entities, etc., are simultaneously involved. Current tag-aware recommendation methods only focus on simple graphs with users, items, and tags, or just user–item bipartite graphs. A more effective way to extract useful information from heterogeneous nodes and relations is required for KTRS to prevent models from overfitting problems.

To overcome these challenges, we propose a novel end-to-end recommendation model for knowledge-enhanced tag-aware recommendation systems called Tag-aware Knowledge Graph Attention Network (TKGAT). In order to model the tags and auxiliary knowledge in a comprehensive framework, we construct the Collaborative Recommendation Graph (CRG) with user–item interactions, user tagging, and auxiliary knowledge. In the CRG, to take advantage of tag information, we treat tags as special relations between users and items. Tag embeddings are shared among tags with the same descriptions from different users to reduce the model complexity. To extract these heterogeneous information in CRG, we design an attention-based graph neural network to adaptively learn the useful information from heterogeneous neighbors. We further regularize the node and relation embeddings to maintain inherent semantics within the representations. Our contributions in this paper can be summarized as follows:

- We focus on a new recommendation scenario, Knowledge-enhanced Tag-aware Recommendation System (KTRS), that absorbs the advantage of knowledge graph based methods into TRS and thus addresses sparsity and arbitrariness problems.
- We construct Collaborative Recommendation Graph (CRG) using user–item interactions, user tagging and auxiliary knowledge to connect the three aspects of information, providing contextual semantics for each component graph to learn their representations.
- We develop a novel tag-aware recommendation model called TKGAT that utilizes a multi-layer multi-head attention

mechanism to aggregate node representations in CRG. We further use pretrained word vectors to force a prior semantic on tag nodes and regularize knowledge graph embeddings to avoid semantic drift.
- We conduct real-world experiments on two public datasets and compare our proposed method with four baselines. TKGAT performs great improvement over state-of-the-art methods in top-K recommendation task.

2. Related work

2.1. Social tagging and tag-aware recommendation system

Social tagging is a powerful tool to reflect users' preferences and help both businesses and consumers evaluate the quality of a product. Tags are successfully used in many fields. For example in computer vision, TelecomNet [26] is a tag-based weakly-supervised deep hashing framework for image retrieval. Many recent researches show that social tagging information also has significant influence on personalized recommendation systems. Commonly, neural network-based models transform tags into a sparse feature space and extract information by deep networks. CFA [27] follows the Collaborative Filtering method and uses the stack autoencoder to extract user representations from tags. TRSDL [28] and DSPR [29] leverage MLPs to map item and user representations into a feature space, then use RNN or deep-semantic similarities to learn from recommendation data. HDLPR [30] further develops DSPR by autoencoder with reconstruction errors. HTRM [17] uses LSTM to learn from user tagging behaviors. TNAM [18] introduces attention mechanisms into the model in order to analyze the information between user-related tags and item-related tags. TGCN [19] uses GAT [20] on a collaborative tag graph and a prediction layer to learn the embedding vectors end-to-end.

These methods try to address the ambiguity and redundancy problems of tags in social tagging systems with their own approaches. However, their performance could still be impaired by the arbitrariness of user tags. In this paper, we introduce auxiliary knowledge to fully address this issue. User tags therefore become a "bridge" to connect user–item interactions and auxiliary knowledge and at that time the model can take full advantage of the tagging information.

2.2. Recommendation with auxiliary knowledge

Introducing auxiliary knowledge to recommendation methods is a common practice to avoid cold start problems [21,22]. In traditional recommendation systems, there are mainly two kinds of auxiliary knowledge: knowledge graphs and social networks. The knowledge graph is only concerned with items, whilst the social network is only concerned with users. KGCN [31] and KGAT [12] are the two pioneering studies in knowledge-enhanced recommendation systems. They both introduce the knowledge graph into the original user–item bipartite graph and train the GCN or GAT model on the augmented graph. KGAT adopts GAT on a collaborative knowledge graph to aggregate information from the knowledge graph into user and item embeddings. KGCN provides another scoring function to put more emphasis on relations that are more consistent with user's preference. Following KGCN and KGAT, a number of studies are proposed to address particular applicational issues. For example, MKGAT [32] discusses multi-modal learning and designs a multi-modal GNN for knowledge-enhanced recommendation systems. ATBRG [33] extracts and prunes a multi-layer subgraph for the target item and user's previous behaviors to figure out the underlying relationship between targets and user behaviors. One advantage of knowledge-enhanced recommendation system is that the model could learn more information about items since knowledge graphs may contain more features and connections between items. On the other hand, social network based recommendation system utilizes a user network, which puts more emphasis on the relationship between users [34–36]. For example, GraphRec [37], DiffNet++ [23] and DANSER [38] all use attention mechanism to differentiate the influence of users' friends. GraphRec applies multi-layer MLPs on the concatenated representation vectors, which come from user–item bipartite graph and social network, to enhance the feature interactions. DiffNet++ uses a multi-level attention network to update node representations on a unified graph that contains both user–item graph and social network. As a recent practice, ESRF [39] tries to modify the given social network via the autoencoder mechanism in order to obtain better neighbors for learning. Social network enhanced recommendation system could easily extract users' characteristics and propagate interests between friends [40,41].

These methods introduce the approach of exploiting auxiliary knowledge in recommendation systems. Many studies have achieved state-of-the-art recommendation performance. However, these methods do not take social tagging into consideration. Hence, they cannot efficiently utilize the information provided by tags.

Recent recommendation systems take multimodal auxiliary knowledge into consideration. For example, UMPR [42] extract both textual and visual preferences for users to boost the performance on restaurant recommendation and product recommendation. In addition, dynamic recommendation systems [43] are also attracting a growing number of researchers. Dynamic embeddings [44] are commonly used in these systems to model the evolving attributes of users and items. A typical paradigm for dynamic recommendation is knowledge tracing [45] that recommends tailored items for users based on tracking their previous interactions. There are many GNN-based methods to solve this problem, such as GKT [46], Bi-CLKT [47], and JKT [48]. Handling tag-aware recommendation in these scenarios are not formally considered in this paper and we leave them for the future work.

3. Problem formulation

In a tag-aware recommendation system, the historical interactions (e.g., clicks or reviews) between users and items can be recorded as a user–item bipartite graph. In this paper, we denote the user–item graph as $\mathcal{G}_I \subset \{(u,i) \mid u \in \mathcal{U}, i \in \mathcal{I}\}$, where $\mathcal{U}, \mathcal{I}$ are the set of users and items, respectively. A tuple (u, i) is in the graph $\mathcal{G}$ if and only if there exists at least one interaction record between user u and item i. The user tagging graph contains user-tag-item triplets and we denote it as $\mathcal{G}_T \subset \{(u,t,i) \mid u \in \mathcal{U}, i \in \mathcal{I}, t \in \mathcal{T}\}$. $\mathcal{T}$ is the total set of tags. Also, triplet (u, t, i) is in $\mathcal{G}_T$ if and only if user u assigns a tag t on item i. The key objective of recommendation system task is to predict a score $y_{ui} \in [0, 1]$ about a given user u and a given item i. The higher the score is, the more likely the user will be interested in the item.

4. Methodology

The general framework of our proposed method is illustrated in Fig. 2. Our method consists of four components: (1) Collaborative recommendation graph, which fulfills the user–item bipartite graph to involve more information for predicting; (2) Embedding regularization, which forces node and relation embeddings to have an inherent semantic relationship for stable training; (3) Attentive aggregation layer, which integrates heterogeneous information into user or item embedding vectors with attention mechanism; (4) Predicting layer, which exploits the aggregated representation vectors to output the predicted recommendation scores.

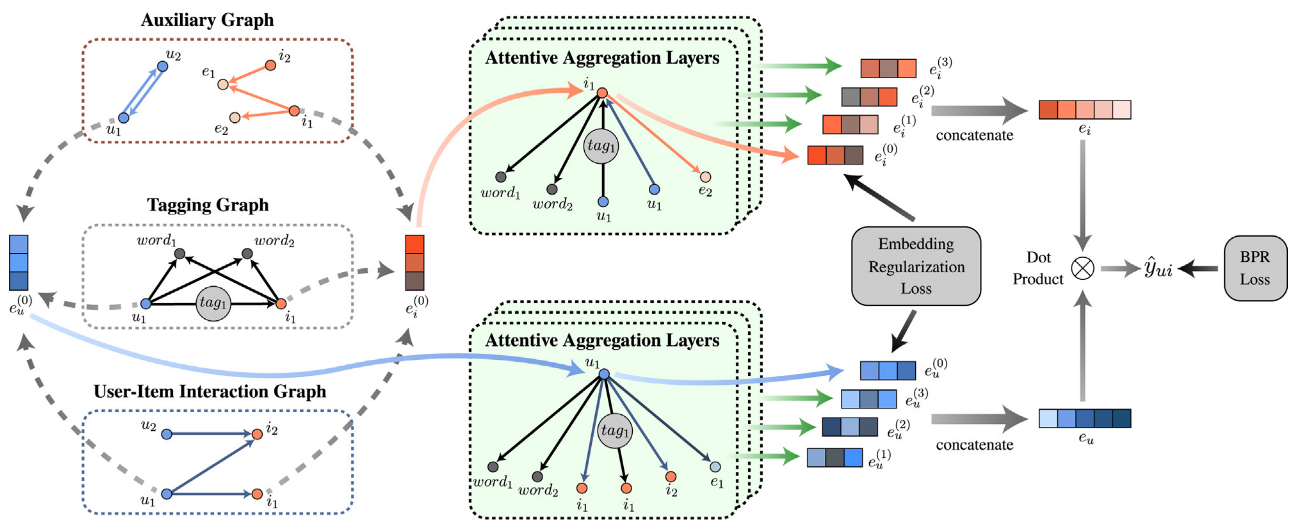
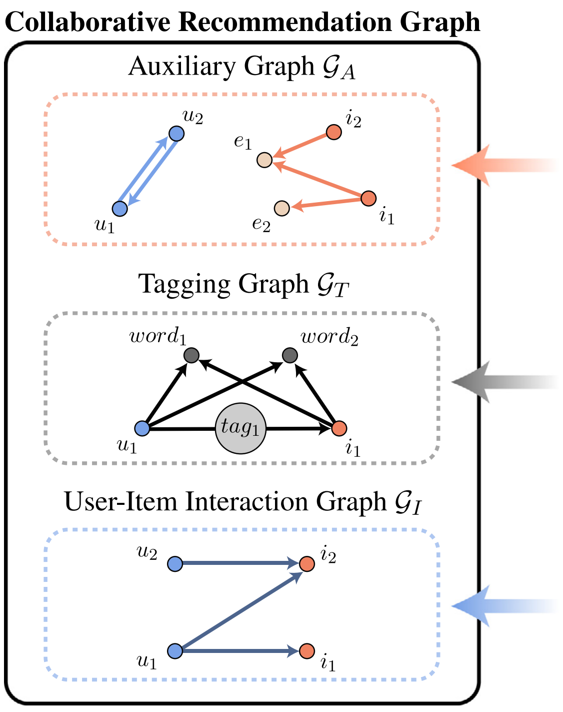


Fig. 2. The framework of the proposed method.


Collaborative Recommendation Graph
Auxiliary Graph $\mathcal{G}_A$
Tagging Graph $\mathcal{G}_T$
User-Item Interaction Graph $\mathcal{G}_I$
u_1
u_2
i_1
i_2
e_1
e_2
$word_1$
$word_2$
tag_1

Auxiliary Knowledge		
u_1	$friend$	u_2
i_1	r_1	e_1
i_1	r_2	e_2
i_2	r_1	e_1

User Tagging		
u_1	tag_1	i_1

Tag Description		
tag_1	$word_1$	$word_2$

User-Item Interaction Records		
u_1	$Interaction$	i_1
u_1	$Interaction$	i_2
u_2	$Interaction$	i_2

Fig. 3. An construction example for collaborative recommendation graph.

4.1. Collaborative recommendation graph

We stack user–item interaction graph $\mathcal{G}_I$, user tagging graph $\mathcal{G}_T$ and auxiliary graph $\mathcal{G}_A$ (including social network and knowledge graph) as a three-layer collaborative graph. Fig. 3 shows an example to construct the CRG. Since all the three graphs have the same user entities and item entities, we share their embeddings among the three graphs. In the following part of this section, we will introduce the three component graphs in detail.

4.1.1. User-item interaction graph $\mathcal{G}_I$

User-item interactions are the basics of recommendation systems. Typically, the interaction graph $\mathcal{G}_I$ is a user–item bipartite graph as introduced in Section 3. The interaction graph is the only graph that will be separated into train-valid-test sets since these interactions are labels in recommendation system tasks.

4.1.2. Tagging graph $\mathcal{G}_T$

The original user tagging data is user-tag-item triplets as described in Section 3. The key challenge is how to integrate tags into the whole collaborative graph. Inspired by [19] where the authors treat all the tags as nodes in the graph, we treat tags as relations which is more natural in tag-aware recommendation systems. With tag relations, the user and the tagged item can directly influence each other and only each other through the tag, unlike the situation of making tag nodes where all the related users and items will be influenced.

Meanwhile, to avoid overfitting due to too many additional tag relations in the graph, we involve pretrained word nodes in tagging graph. Specifically, for any tag, we pick out all the words in the tag text description. Then we treat each word as a node and connect the related users and items with the word nodes generated from this tag. Take the data in Fig. 3 as an example, user u_1 attaches a tag tag_1 to item i_1. The tag has a description of two words, $word_1$ and $word_2$. Therefore, in the Tagging Graph $\mathcal{G}_T$, we connect u_1 with i_1 using relation "tag_1". Also, two word nodes are added and connected to u_1, i_1, since u_1, i_1 are related with tag_1.

Note that tags could have overlapped words so there will not be so many word nodes. In fact, in the experiments, word nodes have the same amount or less than tags. In addition, we initialize these word nodes with their corresponsive pretrained word vectors. For example, we use GloVe [49] as the word vectors in experiments. We also initialize tag relation embeddings by averaging all the word vectors about this tag and then apply the relation transformation $\boldsymbol{W}_r$. The relation transformation matrix $\boldsymbol{W}_r$ is described in Section 4.2 in detail. By these additional word nodes, the learning signal will be propagated in a much larger subgraph so that central nodes will be anchored by other nodes, making training procedure more stable.

4.1.3. Auxiliary graph $\mathcal{G}_A$

The auxiliary graph $\mathcal{G}_A$ contains auxiliary entities and relations related with the recommendation system. We denote it as $\mathcal{G}_A \subset \{(h, r, t) \mid h, t \in \mathcal{E}, r \in \mathcal{R}\}$ where $\mathcal{E}$ stands for the entity set and $\mathcal{R}$ stands for the relation set. Commonly the auxiliary graph is constructed from collected knowledge graph or social network. Knowledge graph provides the side information about items while social network provides that of users. The auxiliary graph can be collected either from open database (for knowledge graphs) or from private data of model users (for social networks). These auxiliary entities and relations complement the ambiguous semantics in tags since they are more reliable and precise. For example, the user tag "apple" could both refer to a kind of fruit or a brand. With a connection between the tagged item and auxiliary entity "Cell phone", the ambiguity can be eliminated.

4.2. Embedding regularization

Knowledge graph embedding uses a learnable vector to represent each node in the knowledge graph. It is generally used to efficiently capture the structural information. In our method, we apply TransR [50] as the embedding method, which learns the entity and relation embedding by scoring function:

$$f(h, r, t) = \|\boldsymbol{W}_r\boldsymbol{e}_h + \boldsymbol{e}_r - \boldsymbol{W}_r\boldsymbol{e}_t\|_2^2, \tag{1}$$

where h, r, t stand for head entity, relation, tail entity, respectively. $\boldsymbol{e}_h \in \mathbb{R}^d$, $\boldsymbol{e}_r \in \mathbb{R}^{d_r}$, $\boldsymbol{e}_t \in \mathbb{R}^d$ are their corresponding embedding vectors. $\boldsymbol{W}_r \in \mathbb{R}^{d_r \times d}$ is the transformation matrix for relation r. A higher score indicates a higher probability of the existence of the triplet (h, r, t). TransR utilizes a pairwise ranking loss for training:

$$\mathcal{L}_{KGE} = \sum_{(h,r,t,t')\in\mathcal{T}} -\ln\sigma\left(f(h, r, t') - f(h, r, t)\right), \tag{2}$$

where $\mathcal{T}$ is the training set, (h, r, t') is the negative training triplet that does not exist in the graph, and $\sigma(\cdot)$ is the sigmoid function. In our model, we use this loss to regularize the embedding. Through this regularization, the information within the word nodes and pretrained tag relations can be merged into user and item representations [19].

4.3. Attentive aggregation layer

In order to aggregate user preferences and item information from neighbors, we construct the attentive aggregation layer to analyze the learnability between one node and its neighbors. Given the input of the entity and relation embeddings, we compute the attention score as

$$\alpha(h, r, t) = \tanh(\boldsymbol{W}_a\boldsymbol{W}_r\boldsymbol{e}_t)^\top \tanh(\boldsymbol{W}_a(\boldsymbol{W}_r\boldsymbol{e}_h + \boldsymbol{e}_r)), \tag{3}$$

where $\boldsymbol{W}_a \in \mathbb{R}^{d_a \times d_r}$, d_a is the attention dimension. Then, we can collect first-order structural information $\boldsymbol{e}_{\mathcal{N}_h}$ by summing up the neighboring embeddings of head entity h:

$$\boldsymbol{e}_{\mathcal{N}_h} = \frac{\sum_{(r,t)\in\mathcal{N}_h} \exp(\alpha(h, r, t))\boldsymbol{e}_t}{\sum_{(r,t)\in\mathcal{N}_h} \exp(\alpha(h, r, t))}, \tag{4}$$

where $\mathcal{N}_h = \{(r, t) \mid (h, r, t) \in \mathcal{G}\}$ is the neighbor set of head entity h. Finally, we aggregate the first-order neighboring information using the bi-interaction aggregation function:

$$\begin{aligned} g_{Bi-Interaction}(\boldsymbol{e}_h, \boldsymbol{e}_{\mathcal{N}_h}) = {} & \text{LeakyReLU}\left(\boldsymbol{W}_{aggr}^1(\boldsymbol{e}_h + \boldsymbol{e}_{\mathcal{N}_h})\right) + \\ & \text{LeakyReLU}\left(\boldsymbol{W}_{aggr}^2(\boldsymbol{e}_h \odot \boldsymbol{e}_{\mathcal{N}_h})\right), \end{aligned} \tag{5}$$

where $\boldsymbol{W}_{aggr}^1, \boldsymbol{W}_{aggr}^2 \in \mathbb{R}^{d' \times d}$ is the learnable transformation matrix, $\odot$ is the element-wise product, d' is the dimension of aggregated embedding vectors (or called entity representations, which is the output dimension of this layer). In the experiments we set $d' = d/2$ to force a high-order feature extraction in the attention. To enhance the robustness of the attention layer, we introduce the K-head attention strategy as follows:

$$\begin{aligned} \alpha^k(h, r, t) &= \tanh(\boldsymbol{W}_a^k\boldsymbol{W}_r\boldsymbol{e}_t)^\top \tanh(\boldsymbol{W}_a^k(\boldsymbol{W}_r\boldsymbol{e}_h + \boldsymbol{e}_r)) \\ \boldsymbol{e}_{\mathcal{N}_h}^k &= \frac{\sum_{(r,t)\in\mathcal{N}_h} \exp(\alpha^k(h, r, t))\boldsymbol{e}_t}{\sum_{(r,t)\in\mathcal{N}_h} \exp(\alpha^k(h, r, t))}, \quad k = 1, 2, \ldots, K. \end{aligned} \tag{6}$$

Stack all the structure information heads $\boldsymbol{e}_{\mathcal{N}_h}^k$ in the aggregation function, we formulate the multi-head aggregation function as

$$\begin{aligned} & g_{Bi-Interaction}(\boldsymbol{e}_h, \boldsymbol{e}_{\mathcal{N}_h}) = \\ & \quad \text{concat}\Big(\text{LeakyReLU}\left(\boldsymbol{W}_{aggr}^1(\boldsymbol{e}_h + \boldsymbol{e}_{\mathcal{N}_h}^k)\right) + \\ & \qquad \text{LeakyReLU}\left(\boldsymbol{W}_{aggr}^2(\boldsymbol{e}_h \odot \boldsymbol{e}_{\mathcal{N}_h}^k)\right), \\ & \qquad k = 1, 2, \ldots, K \Big), \end{aligned} \tag{7}$$

where $\boldsymbol{W}_{aggr} \in \mathbb{R}^{d'/K \times d}$ and d' is divisible by K. We stack multiple attentive layers to capture high-order structural information, thus obtaining more supervision signals from distant nodes. To be explicit, in the lth layer, denoting the representation vector from the last layer as $\boldsymbol{e}^{(l-1)}$, we can recursively compute the entity representation as

$$\boldsymbol{e}_h^{(l)} = g_{Bi-Interaction}(\boldsymbol{e}_h^{(l-1)}, \boldsymbol{e}_{\mathcal{N}_h}^{(l-1)}). \tag{8}$$

4.4. Prediction layer

Assuming there are L layers of attentive aggregation, we concatenate all the entity representations for user u and item i in these layers to construct a final representation $\boldsymbol{e}_u$ and $\boldsymbol{e}_i$:

$$\begin{aligned} \boldsymbol{e}_u &= \text{concat}(\boldsymbol{e}_u^{(0)}, \boldsymbol{e}_u^{(1)}, \ldots, \boldsymbol{e}_u^{(L)}), \\ \boldsymbol{e}_i &= \text{concat}(\boldsymbol{e}_i^{(0)}, \boldsymbol{e}_i^{(1)}, \ldots, \boldsymbol{e}_i^{(L)}). \end{aligned} \tag{9}$$

To predict the recommendation score between user u and item i, we conduct the inner product as follows:

$$\hat{y}_{u,i} = \boldsymbol{e}_u^\top \boldsymbol{e}_i. \tag{10}$$

We use the BPR loss to train the prediction model, which is defined as:

$$\mathcal{L}_{pred} = \sum_{(u,i,i')\in\mathcal{T}} -\ln\sigma(\hat{y}_{u,i} - \hat{y}_{u,i'}). \tag{11}$$

Here, $\mathcal{T}$ is the training set and i' is the negative item that user u never interacts with.

Table 1
Parameter complexity of components in TKGAT. Vertex set $\mathcal{V}$ includes users $\mathcal{U}$, items $\mathcal{I}$, and entities in the auxiliary graph $\mathcal{E}$. d, d_r, and d_a are the embedding, relation, and attention dimension, respectively. L is the number of attention layers and K is the number of attention heads.

Model component	Parameter complexity
Graph embeddings	$O\left(\lvert\mathcal{V}\rvert d+\lvert\mathcal{R}\cup\mathcal{T}\rvert d_r d\right)$
Attentive layers	$O\left(Ld'd+Kd_a d_r\right)$
Prediction layer	$O(1)$

4.5. Learning strategy

For both embedding loss and prediction loss, we train the model in a supervised manner. Positive samples are generated from training data and negative samples are generated by negative sampling, a common practice in graph neural network models, as well as in recommendation systems [10,18,30,51]. In negative sampling, edges that are unobserved in training set are considered to be a potential negative sample. With a sampling rate that indicates the ratio between negative samples and positive ones, unobserved edges are randomly selected for each user. In our experiments, we fix the sampling rate as $1:1$.

Since our model has both embedding loss and prediction loss, a simple way is to add them together for training. However that will involve hyper-parameter tuning to balance the two losses. We here use a two-step train strategy to stabilize the learning. Firstly we train the model on prediction loss and then on embedding loss for each single epoch. In traditional CNN-based training in computer vision where pretrained CNN models are used, the last fully-connected layer for prediction is trained before fine-tuning the CNN backbone. Here we adopt the two-step fine-tuning to benefit the training since there exists a similar pretrained structure in our model.

4.6. Complexity analysis

In order to provide a fair comparison, we list the parameter size of each component in Table 1. In practice, the parameters of graph embeddings contribute to the most of the model size when the number of relations grows. The high complexity is due to TransR assigns a unique $\boldsymbol{W}_r$ for each relation. It is possible to use other embedding methods like TransE [52] and modify the embedding regularization loss accordingly. Other part of TKGAT, including attentive layers and prediction layer, is independent from data scale.

5. Experiments

In order to thoroughly evaluate our proposed method TKGAT and our claims, we consider the following three research questions (RQs):

- **RQ1:** Does the proposed TKGAT outperform the state-of-the-art recommendation methods in real-world tag-aware recommendation tasks?
- **RQ2:** How do the hyper-parameters in TKGAT influence the recommendation performance? How can the hyper-parameters be tuned for better performance?
- **RQ3:** What role do the tagging graph play in knowledge-enhanced tag-aware recommendation systems? What if we remove the additional word nodes?

We will discuss these research questions in Sections 5.2–5.4 in detail. In Section 5.1, we introduce the detailed settings about the experiments. In Section 5.5, we show some visual results for case study.

Table 2
Dataset statistics. "Auxiliary edges" includes edges in the knowledge graph and edges in the social network.

Dataset	#Users	#Items	#Tags	#Interactions	#Auxiliary edges
MovieLens	2113	10109	13222	855598	467001
Last.FM	1877	17617	11946	92801	25217
Delicious	1741	61861	3508	339744	15328

5.1. Experiment setup

5.1.1. Datasets

We conduct experiments on three real-world datasets: MovieLens [53], Last.FM [54], and Delicious. The datasets are released in HetRec 2011 [55]. We filter out the 5-core subgraph of each dataset and follow Chen et al. [19] to remove infrequent tags. The statistics is summarized in Table 2.

- **MovieLens:** An extension of MovieLens10M dataset published by GroupLens.[1] Only those users with both rating and tagging information have been maintained. Auxiliary knowledge is the knowledge graph about movies, including genres, directors, actors, locations, etc.
- **Last.FM:** A music artist listening dataset from Last.fm[2] online music system. The dataset includes about 2000 users and their social network for auxiliary. The social network is constructed by communications between users.
- **Delicious:** A bookmark tagging dataset collected in Delicious[3] social bookmarking system. Users discover and share website bookmarks in this system. User contacts form a social network among the 2000 users in the dataset.

We choose these three datasets due to the following two reasons: (1) the amount of users and items vary among the three datasets, providing comparisons on different data scales. (2) they cover two types of auxiliary knowledge, knowledge graph and social network, respectively. As a result, they can provide a relatively fair comparison platform for TKGAT and the chosen baselines. We split the user–item interactions into three subsets with a ratio of $7:1:2$ for training, validation and test. The split is shared among methods.

5.1.2. Baselines

We compare the proposed TKGAT method with 7 baseline methods as follows:

- **TGCN** [19]**:** A state-of-the-art tag-aware recommendation method. It models tags as nodes and use GCN to aggregate information from interactions.
- **DeepFM** [56]**:** A recommendation method that combines factorization machines [11] and deep neural network for feature learning.
- **NGCF** [57]**:** An improved NCF model that encodes the collaborative signal with embedding propagation.
- **KGAT** [12]**:** A state-of-the-art KG-enhanced recommendation model. It applies knowledge-aware graph attention network on collaborative graph for recommendation.
- **CFKG** [58]**:** A model that utilizes TransE [52] on a unified graph including users, items, entities, relations and predicts $(user, interaction, item)$ triplets as recommendation.
- **FM** [11]**:** Classic factorization machine method for recommendation system. It uses second-order feature interactions to predict recommendation scores.

[1] https://grouplens.org/
[2] http://www.last.fm/
[3] http://del.icio.us/

- **NFM** [10]**:** A state-of-the-art factorization model that extract features with a neural network. We use one hidden layer suggested in the original paper [10].

5.1.3. Metrics

The recommendation performance is evaluated by three commonly used metrics: Precision, Recall and Normalized Discounted Cumulative Gain (NDCG) [59]. All the metrics are calculated in the top-K recommendation results, as used in many recommendation system-related model evaluations [12,19,57]. We evaluate the methods on each user to obtain the user-wise evaluation and then take average among all the candidate users for the final performance results.

5.1.4. Parameter settings

All the models are implemented by Python and Pytorch. Experiments are conducted on a Linux server with four RTX2080Ti GPUs and two Intel Xeon Silver 4216 CPUs. We train all the model with batch size 2048 for prediction loss training and 4096 for embedding loss training. Learning rate is tuned from $\{0.001, 0.01, 0.05\}$ just to prevent loss explosion. Dropout rate is set to 0.1 if applicable. The entity embedding size is set to 100 and relation embedding size is set to 50. For TKGAT, we use 3 aggregation layer with output size 100, 50, 25. We fix the number of attention heads to 5 and attention dimension to 50. We use early stopping for all the models. Any model that does not improve its validation performance in 50 epochs will be stopped.

5.2. Overall performance comparison (RQ1)

We compare our proposed TKGAT with the 7 baseline methods mentioned in Section 5.1.2. The 3 datasets, MovieLens, LastFM, and Delicious, include knowledge graph and social networks, respectively. For the knowledge graph based methods, we reconstruct the social network into a one-relation knowledge graph as input. Also, we follow [12] that adds the knowledge graph in the input features. The top-K personalized recommendation evaluation results are illustrated in Fig. 4. TKGAT outperforms baselines in both knowledge-enhanced dataset and social network-enhanced dataset.

Compare between the three datasets, MovieLens has a less amount of items and a much larger amount of interactions than LastFM and Delicious. As a result, LastFM and Delicious dataset is much more sparse than MovieLens, which leads to a harder learning procedure for the models. From the performance figure, we can verify this fact since a performance drop can be seen for each method from MovieLens to LastFM and Delicious. The improvement in Recall is due to the less size of test set in LastFM and Delicious.

NFM, the improvement of FM, behaves better than FM in MovieLens while suffers from sparse problem in LastFM. Knowledge graph based methods, KGAT and ECFKG, show better performance in MovieLens than in LastFM. They even perform worse than classic FM method in LastFM. This indicates that they cannot handle recommendation tasks with social networks only. The supervision signal provided by user–item interactions and social network is too sparse to learn a general knowledge graph based model, so these model may encounter overfitting problems in social network based recommendation tasks. Even if in MovieLens, TKGAT, utilizing tagging information, significantly performs better than KGAT and ECFKG. In addition, we train TKGAT from sketch without word nodes in Ablation Study (Section 5.4). The results also show better performance than baselines in MovieLens. As a result, we proved by experiments that tags do play an important role in recommendation model learning. In addition, TGCN, as a state-of-the-art tag-aware recommendation method, performs worse than TKGAT is most of the cases (only be better in Recall of MovieLens and NDCG of LastFM). This suggests that the introduced auxiliary information is capable to boost the learning and effectiveness of tag-aware recommendation systems.

5.3. Hyper-parameter efficiency (RQ2)

Since there are many hyper-parameters in TKGAT, we wonder what kind of effect they will have on recommendation performance, positive or negative. In our early experiments, we vary the batch size and learning rate for TKGAT, but the performance is affected little. This indicates that batch size and learning rate within a proper range will not affect the performance, so in the following experiments we use the same setting of batch size and learning rate. Further, we mainly choose two hyper-parameters to study their effects. Firstly, since we utilize multi-layer attention aggregation, it is vital to carefully choose the number of layers. In each layer, the output representation vector only includes the information from one-hop neighborhood of the central node. As a result, a higher number of layers could aggregate more information from far-distance neighbors while the scale of parameters would meanwhile grow, causing overfitting problem. Secondly, in collaborative recommendation graph, we treat tags as relations and use attention mechanism to merge relation information into node representations. We argue that relation and attention dimension may influence the recommendation performance since they control the information capacity of each relation and attention head.

In Overall Performance Comparison (Section 5.2), we set TKGAT to have three layers of attention aggregation of dimension 100, 50, and 25. Here we discard one or two layers from the top, keeping other settings the same. The test performance of the two datasets in shown in Table 3. From the table we see a slight performance drop in 3-layer TKGAT compared with 1-layer and 2-layer ones in MovieLens and LastFM. In MovieLens, 3-layer TKGAT maintains a second place in the three settings, while in LastFM 3-layer TKGAT drops to the third place. This can be explained by the less complexity in LastFM dataset than MovieLens, as we analyzed in Section 5.2. The 3-layer TKGAT achieves better performance in Delicious, which has much more items and is harder for learning. This indicates that, in view of the scale and complexity in real-world applications, TKGAT with more attention aggregation layers would achieve better performance in most large-scale tag-aware recommendation tasks. Nevertheless, the largest performance drop in the table does not exceed 4% and there is not any layer setting that maintains the best place among the three metrics, so the number of layers would not have a crucial impact on the performance.

The relation embedding size and attention embedding size are both set to 50 in Overall Performance Comparison. We decrease them by half two times to study the influence of different scales of dimension. Table 4 shows the results. For dataset MovieLens, we find a gradual improvement of performance with the increase of dimension. Comparing 10-dimensional TKGAT with 50-dimensional TKGAT, all the three metrics increases about 4%. In LastFM, 25-dimensional TKGAT takes the first place on all the metrics, followed by 50-dimensional. In Delicious, we notice a large performance drop between 10- and 25-dimensional TKGAT. This may due to the much larger amount of items, which requires more parameters to learn a better model. In the comparison, we observe that a too small relation and attention dimension setting could cause bad performance. A proper setting of relation and attention embedding size can be chosen by validation.

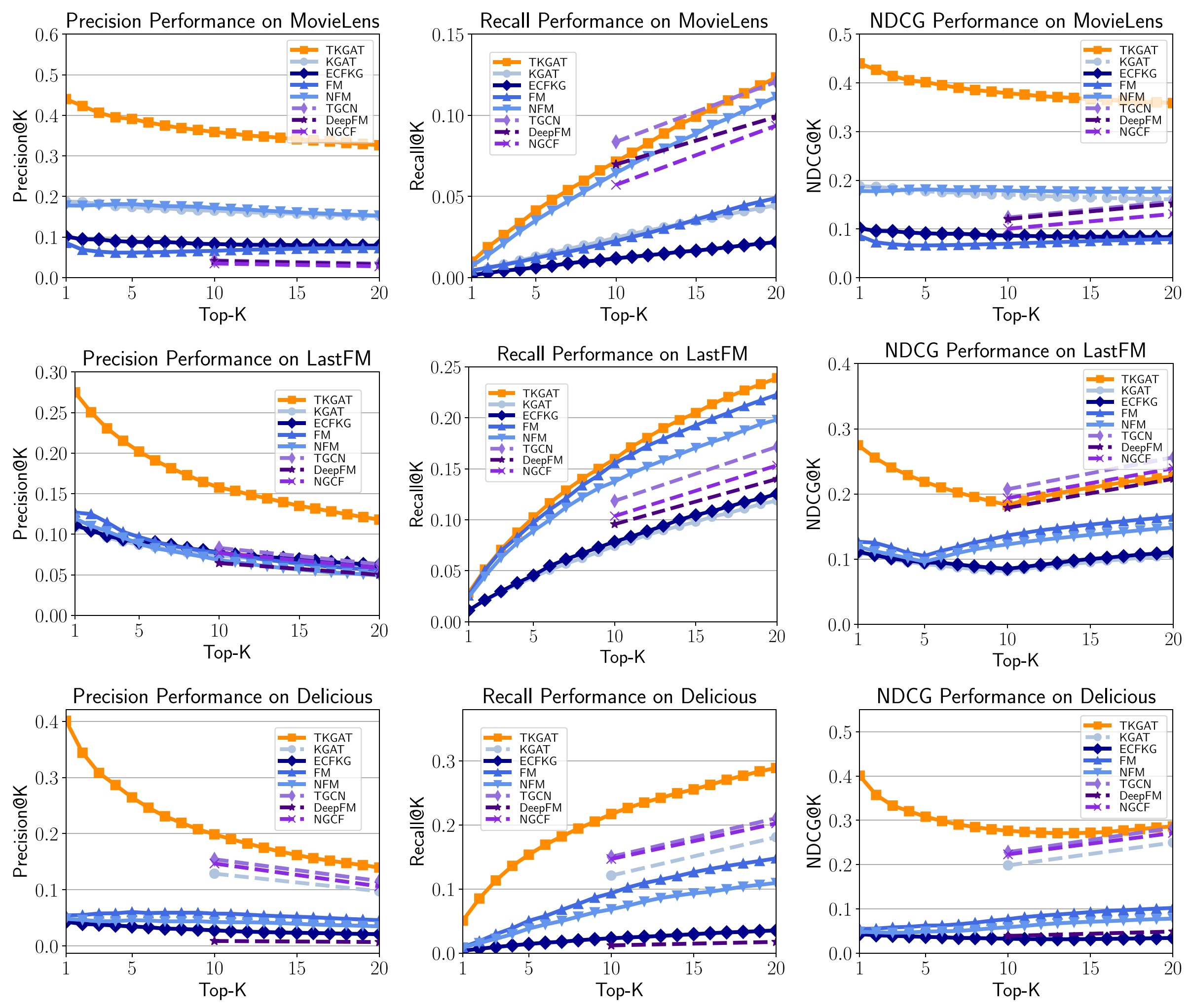


Fig. 4. Overall performance comparison on MovieLens, LastFM, and Delicious of TKGAT and 7 baselines. We vary K from 1 to 20 for the top-K recommendation evaluation on test set. The results drawn with dashed lines are from the original paper of TGCN.

Table 3
Performance evaluation of TKGAT with different attention aggregation layer settings. The number "100-50-25" indicates that the output dimension of the layers is 100, 50, and 25 from bottom to top. The bold score is the highest one of the column and underlined score is the second highest one.

Attention layer size	MovieLens					
	Precision@10	Precision@20	Recall@10	Recall@20	NDCG@10	NDCG@20
1 Layer: 100	**0.3646**	**0.3287**	0.0708	0.1209	**0.3836**	**0.3607**
2 Layers: 100–50	0.3501	0.3161	**0.0718**	0.1208	0.3676	0.3471
3 Layers: 100-50-25	0.3591	0.3265	0.0715	**0.1234**	0.3787	0.3586
Attention layer size	LastFM					
	Precision@10	Precision@20	Recall@10	Recall@20	NDCG@10	NDCG@20
1 Layer: 100	0.1587	**0.1186**	0.1607	**0.2401**	0.184	0.2279
2 Layers: 100–50	**0.1588**	0.1185	**0.1610**	0.2398	**0.1866**	**0.2300**
3 Layers: 100-50-25	0.1576	0.1180	0.1596	0.2390	0.1833	0.2272
Attention layer size	Delicious					
	Precision@10	Precision@20	Recall@10	Recall@20	NDCG@10	NDCG@20
1 Layer: 100	0.1770	0.1269	0.1930	0.2625	0.2449	0.2563
2 Layers: 100–50	0.1818	0.1312	0.1966	0.2725	0.2499	0.2631
3 Layers: 100-50-25	**0.1988**	**0.1397**	**0.2172**	**0.2890**	**0.2766**	**0.2863**

5.4. Ablation study (RQ3)

In the collaborative recommendation graph constructed by us, the tagging graph contains both tagging interactions and word nodes. In this section, we will discuss about the additional word nodes. To figure out the effect of these nodes, we train TKGAT from sketch (hereinafter referred to as "TKGAT (w/o)") and record the learning procedure. In detail, we remove all the word nodes and the connections with other entities. Also, we use random initialization on tag embeddings, instead of the average of pre-trained word vectors. In fact, train TKGAT from sketch is quite the same as train TKGAT in a dataset where we do not have access to

Table 4
Performance evaluation of TKGAT with different relation and attention embedding sizes. Also, the bold score is the highest one of the column and underlined score is the second highest one.

Relation/attention embedding size	MovieLens					
	Precision@10	Precision@20	Recall@10	Recall@20	NDCG@10	NDCG@20
10	0.3481	0.3142	0.0693	0.1191	0.3661	0.3451
25	0.3590	0.3234	0.0709	0.1220	0.3754	0.3540
50	**0.3591**	**0.3265**	**0.0715**	**0.1234**	**0.3787**	**0.3586**
Relation/attention embedding size	LastFM					
	Precision@10	Precision@20	Recall@10	Recall@20	NDCG@10	NDCG@20
10	0.1548	0.1159	0.1570	0.2345	0.1790	0.2218
25	**0.1611**	**0.1193**	**0.1631**	**0.2416**	**0.1878**	**0.2310**
50	0.1576	0.1180	0.1596	0.2390	0.1833	0.2272
Relation/attention embedding size	Delicious					
	Precision@10	Precision@20	Recall@10	Recall@20	NDCG@10	NDCG@20
10	0.1325	0.0985	0.1490	0.2108	0.1836	0.1974
25	0.1785	0.1258	0.1966	0.2648	0.2505	0.2604
50	**0.1988**	**0.1397**	**0.2172**	**0.2890**	**0.2766**	**0.2863**

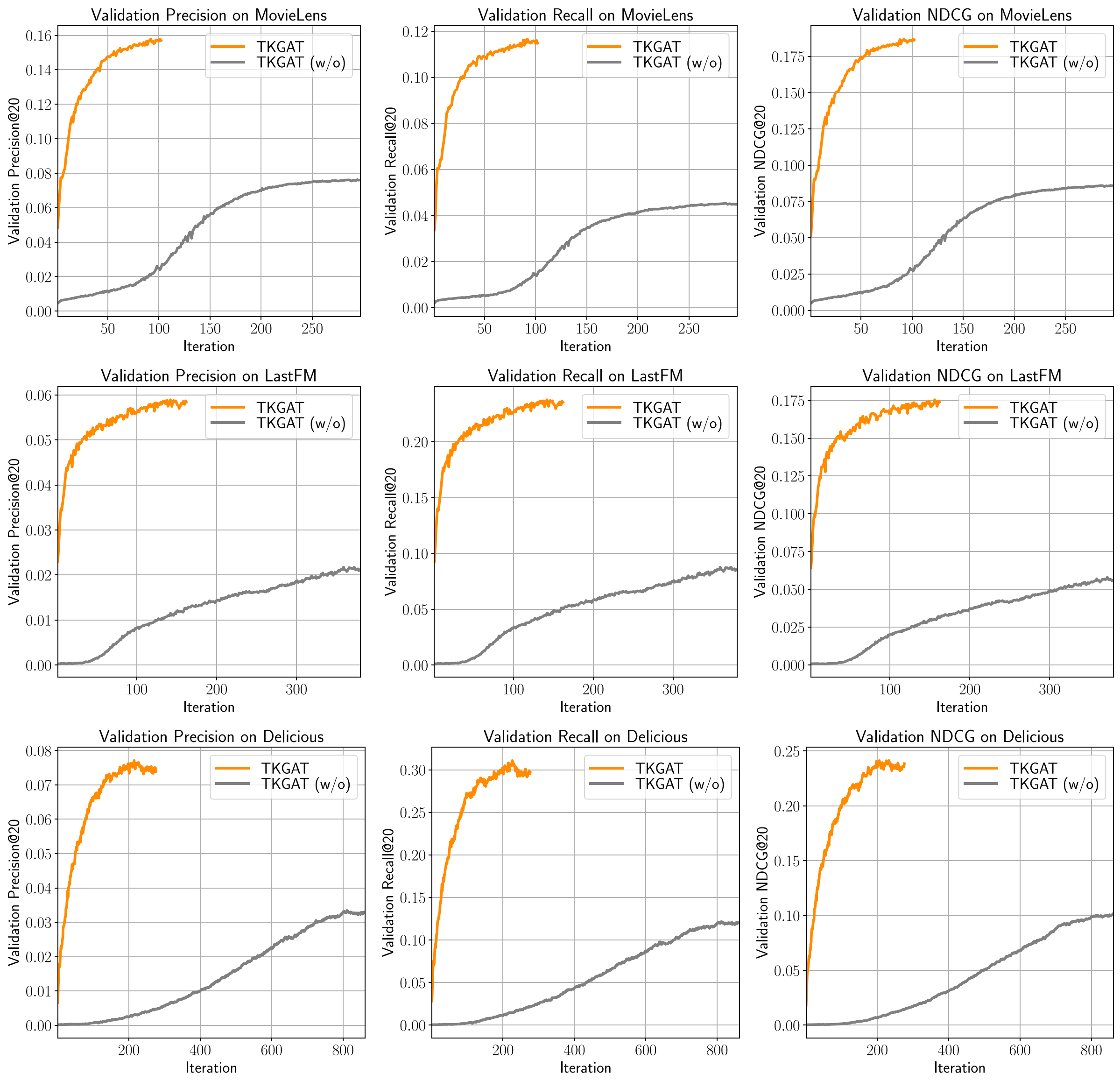


Fig. 5. Ablation comparisons. TKGAT (w/o) stands for the TKGAT model without pretrained word nodes and with randomly initialized tag embeddings. The precision, recall, and NDCG are evaluated on the validation set and recorded after each epoch.

tag descriptions. Other settings are remain the same as in Overall Performance Comparison.

Fig. 5 shows the validation performance during training iterations. We find that there are several epoches in the beginning of

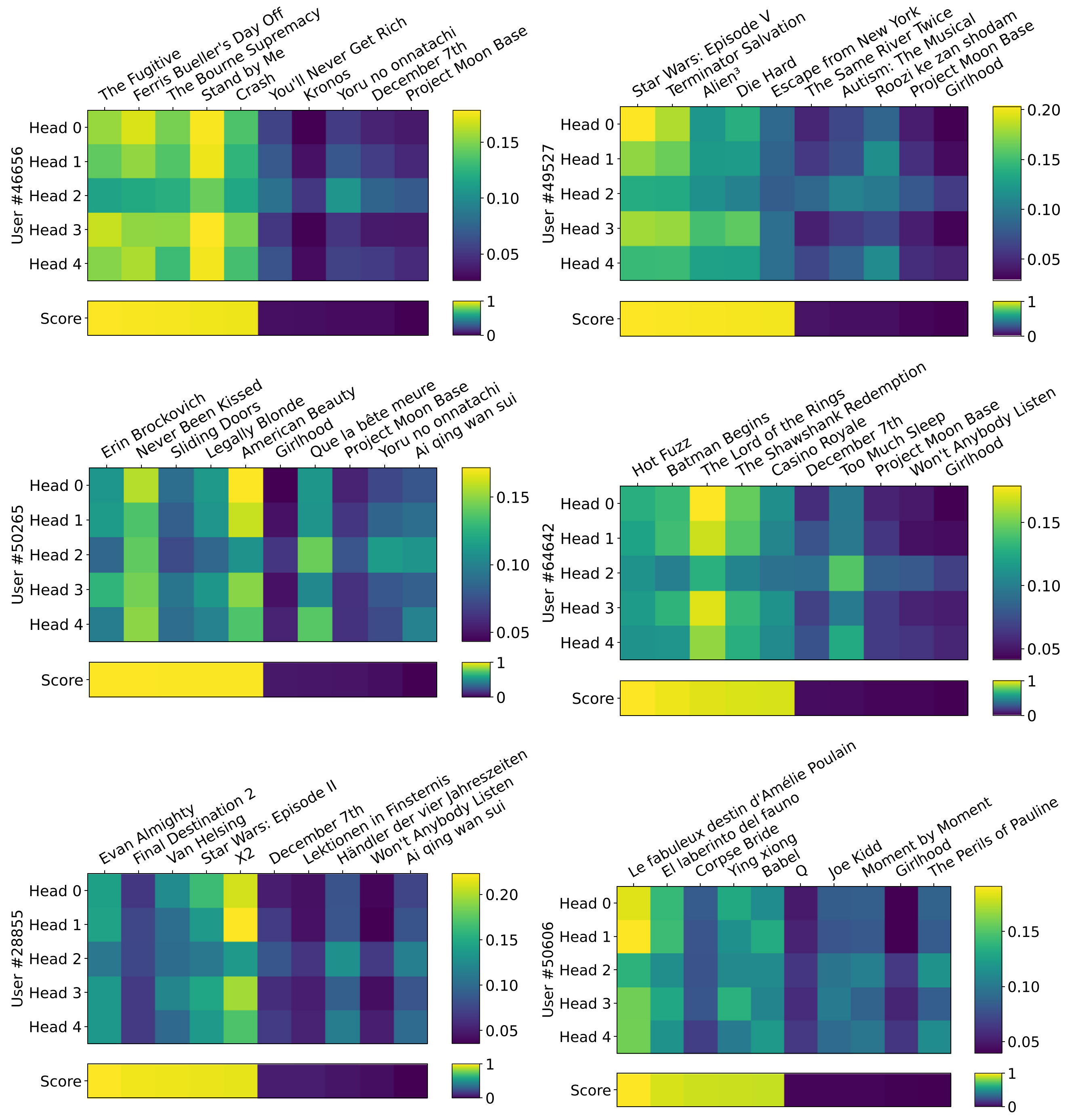


Fig. 6. Comparisons between attention scores and recommendation scores of four users. In the subfigure of each user, the heatmap above shows the attention scores of the 5 attention heads (Head 0 to 4) with respect to 10 movies, including top-5 and bottom-5 recommended by TKGAT. The headmap labeled as "Score" shows the recommendation score $\hat{y}_{u,i}$ calculated by the prediction layer.

training that TKGAT (w/o) learns slowly. After these trial epoches, TKGAT (w/o) rapidly improves the validation performance until it reaches the final fine-tuning procedure. However, with word nodes and pretrained word vectors, TKGAT obtains a much higher performance in the first several epoches. TKGAT could finish its fine-tuning procedure within 200 epoches. Through this figure, we observe that the word nodes and pretraining strategy could significantly accelerate the training progress of TKGAT. The total epoches could be cut down to at least a half via the constructed tagging graph.

We also test TKGAT (w/o) and the result is shown in Table 5. In general, training TKGAT from sketch will eventually obtain a worse model than normal TKGAT. However, TKGAT (w/o) still achieves better results than baselines in Overall Performance Comparison on MovieLens. In LastFM, it looks like TKGAT (w/o) is trapped by local minima and suffers from overfitting. As a result, the word nodes and pretrain word vectors help our model improve the ability of generalization and prevent from overfitting.

5.5. Case study: MovieLens

In order to provide a visual understanding of the effectiveness of our proposed model, we visualize the attention layer in the

Table 5
Test Performance of TKGAT with or without word nodes and pretraining.

Dataset		TKGAT	TKGAT(w/o)
MovieLens	Precision@20	0.3265	0.2338
	Recall@20	0.1234	0.0830
	NDCG@20	0.3586	0.2484
LastFM	Precision@20	0.1180	0.0437
	Recall@20	0.2390	0.0885
	NDCG@20	0.2272	0.0773
Delicious	Precision@20	0.1397	0.0007
	Recall@20	0.2890	0.0012
	NDCG@20	0.2863	0.0011

prediction of MovieLens as shown in Fig. 6. We use TKGAT to provide recommendation scores of movies for the 6 selected users. We pick the top-5 and bottom-5 recommended movies (movies with largest-5 and smallest-5 recommendation scores) for each user. The recommendation scores are normalized among the 10 movies and shown in the bottom of each subfigure. We calculate the attention score of the 5 heads from each user to the movies and show in the upper heatmap of each subfigure. From the heatmap, we surprisingly find that the attention scores have the same tendency with recommendation scores: for the most recommended movies, the attention scores tend to be larger while for the least recommended movies, the attention scores become smaller. Since the selected users do not necessarily have interactions with the chosen movies, the attention scores are actually independent from the predicted recommendation scores. This proves that the attention layer does learn the user preferences from the tags and other relations.

6. Conclusion and future work

In this paper, we propose a new task called Knowledge-enhanced Tag-aware Recommendation System (KTRS) to address the problems of sparsity and arbitrariness in traditional tag-aware recommendation systems. In this new scenario, we further propose a tag-aware graph attention network based model called TKGAT. In our model, we design the Collaborative Recommendation Graph (CRG) including user–item interactions, user tagging and auxiliary knowledge in order to bridge the three aspects of data. Multi-head attention aggregation layers and embedding regularization are utilized to extract latent representations of users and items for the final recommendation score prediction. Experimental results on real-world datasets show that our model significantly outperforms the state-of-the-art tag-aware recommendation models.

In the future, we will address the potential limitations of our method. (1) When the number of tags grows, the embedding size will meanwhile grows. This might cause memory explosion in applications that have huge amount of tags. Space efficiency can be further researched for large-scale recommendation systems. (2) Online recommendation is a topic attracting increasing research attention. In this scenario, taggings and interactions may update in a timeline. How to learn the recommendation model with online updatable auxiliary knowledge is remain to be discussed. (3) At present, we only consider the text auxiliary knowledge about users and items. There exist many other kinds of information like images, locations, audio or video materials. These can be fully utilized in future researches.

CRediT authorship contribution statement

Beilun Wang: Conceptualization, Methodology, Writing – original draft. **Haoqing Xu:** Software, Investigation, Writing – original draft. **Chunshu Li:** Writing – review & editing. **Yuchen Li:** Visualization. **Meng Wang:** Validation, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by National Natural Science Foundation of China [Grant No. 61906040]; the Natural Science Foundation of Jiangsu Province [Grant Numbers BK20190335, BK20190345]; National Natural Science Foundation of China [Grant Numbers 61906037, 61972085]; the Fundamental Research Funds for the Central Universities; Jiangsu Provincial Key Laboratory of Network and Information Security [Grant No. BM2003201], Key Laboratory of Computer Network and Information Integration of Ministry of Education of China [Grant No. 93K-9].

References

[1] D. Kong, Y. Fan, Y. Du, S. Hu, Y. Liu, Q. Li, Personalized recommendation algorithm based on the chance discovery in social network services, in: 2018 5th IEEE International Conference on Cloud Computing and Intelligence Systems, CCIS, IEEE, 2018, pp. 719–723.
[2] F. Durao, P. Dolog, A personalized tag-based recommendation in social web systems, in: Adaptation and Personalization for Web, Vol. 2, 2009, p. 40.
[3] S. Zhang, L. Yao, A. Sun, Y. Tay, Deep learning based recommender system: A survey and new perspectives, ACM Comput. Surv. 52 (1) (2019) 1–38.
[4] P. Covington, J. Adams, E. Sargin, Deep neural networks for youtube recommendations, in: Proceedings of the 10th ACM Conference on Recommender Systems, 2016, pp. 191–198.
[5] R. Ying, R. He, K. Chen, P. Eksombatchai, W.L. Hamilton, J. Leskovec, Graph convolutional neural networks for web-scale recommender systems, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 974–983.
[6] B. Hu, C. Shi, W.X. Zhao, P.S. Yu, Leveraging meta-path based context for top-N recommendation with a neural co-attention model, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 1531–1540.
[7] H. Zhao, Q. Yao, J. Li, Y. Song, D.L. Lee, Meta-graph based recommendation fusion over heterogeneous information networks, in: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2017, pp. 635–644.
[8] F. Zhang, N.J. Yuan, D. Lian, X. Xie, W.-Y. Ma, Collaborative knowledge base embedding for recommender systems, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 353–362.
[9] X. He, Z. He, J. Song, Z. Liu, Y.-G. Jiang, T.-S. Chua, NAIS: Neural attentive item similarity model for recommendation, IEEE Trans. Knowl. Data Eng. 30 (12) (2018) 2354–2366.
[10] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, T.-S. Chua, Neural collaborative filtering, in: Proceedings of the 26th International Conference on World Wide Web, 2017, pp. 173–182.
[11] S. Rendle, Z. Gantner, C. Freudenthaler, L. Schmidt-Thieme, Fast context-aware recommendations with factorization machines, in: Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2011, pp. 635–644.
[12] X. Wang, X. He, Y. Cao, M. Liu, T.-S. Chua, KGAT: Knowledge graph attention network for recommendation, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 950–958.
[13] Z. Sun, Q. Guo, J. Yang, H. Fang, G. Guo, J. Zhang, R. Burke, Research commentary on recommendations with side information: A survey and research directions, Electron. Commer. Res. Appl. 37 (2019) 100879.
[14] H. Liu, Resource recommendation via user tagging behavior analysis, Cluster Comput. 22 (1) (2019) 885–894.
[15] C. Chen, X. Zheng, Y. Wang, F. Hong, D. Chen, Capturing semantic correlation for item recommendation in tagging systems, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 30, no. 1, 2016.
[16] B.M. Shoja, N. Tabrizi, Tags-aware recommender systems: a systematic review, in: 2019 IEEE International Conference on Big Data, Cloud Computing, Data Science & Engineering, BCD, IEEE, 2019, pp. 11–18.

[17] J. Bao, S. Ren, F. Ding, HTRM: A hybrid neural network algorithm based on tag-aware, in: 2021 IEEE International Conference on Smart Internet of Things, SmartIoT, IEEE, 2021, pp. 160–165.
[18] R. Huang, N. Wang, C. Han, F. Yu, L. Cui, TNAM: A tag-aware neural attention model for top-N recommendation, Neurocomputing 385 (2020) 1–12.
[19] B. Chen, W. Guo, R. Tang, X. Xin, Y. Ding, X. He, D. Wang, TGCN: Tag graph convolutional network for tag-aware recommendation, in: Proceedings of the 29th ACM International Conference on Information & Knowledge Management, 2020, pp. 155–164.
[20] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, 2017, arXiv preprint arXiv:1710.10903.
[21] Q. Guo, F. Zhuang, C. Qin, H. Zhu, X. Xie, H. Xiong, Q. He, A survey on knowledge graph-based recommender systems, IEEE Trans. Knowl. Data Eng. (2020).
[22] S. Wu, F. Sun, W. Zhang, B. Cui, Graph neural networks in recommender systems: A survey, 2020, arXiv preprint arXiv:2011.02260.
[23] L. Wu, J. Li, P. Sun, R. Hong, Y. Ge, M. Wang, DiffNet++: A neural influence and interest diffusion network for social recommendation, IEEE Trans. Knowl. Data Eng. (2020).
[24] S. Ji, S. Pan, E. Cambria, P. Marttinen, S.Y. Philip, A survey on knowledge graphs: Representation, acquisition, and applications, IEEE Trans. Neural Netw. Learn. Syst. (2021).
[25] J. Chicaiza, P. Valdiviezo-Diaz, A comprehensive survey of knowledge graph-based recommender systems: Technologies, development, and contributions, Information 12 (6) (2021) 232.
[26] W. Zhao, C. Xu, Z. Guan, X. Wu, W. Zhao, Q. Miao, X. He, Q. Wang, TelecomNet: Tag-based weakly-supervised modally cooperative hashing network for image retrieval, IEEE Trans. Pattern Anal. Mach. Intell. (2021) 1, http://dx.doi.org/10.1109/TPAMI.2021.3114089.
[27] Y. Zuo, J. Zeng, M. Gong, L. Jiao, Tag-aware recommender systems based on deep neural networks, Neurocomputing 204 (2016) 51–60.
[28] N. Liang, H.-T. Zheng, J.-Y. Chen, A.K. Sangaiah, C.-Z. Zhao, TRSDL: Tag-aware recommender system based on deep learning–intelligent computing systems, Appl. Sci. 8 (5) (2018) 799.
[29] Z. Xu, C. Chen, T. Lukasiewicz, Y. Miao, X. Meng, Tag-aware personalized recommendation using a deep-semantic similarity model with negative sampling, in: Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, 2016, pp. 1921–1924.
[30] Z. Xu, T. Lukasiewicz, C. Chen, Y. Miao, X. Meng, Tag-aware personalized recommendation using a hybrid deep model, in: AAAI Press/International Joint Conferences on Artificial Intelligence, 2017.
[31] H. Wang, M. Zhao, X. Xie, W. Li, M. Guo, Knowledge graph convolutional networks for recommender systems. corr abs/1904.12575 (2019), 2019, arXiv preprint arXiv:1904.12575.
[32] R. Sun, X. Cao, Y. Zhao, J. Wan, K. Zhou, F. Zhang, Z. Wang, K. Zheng, Multi-modal knowledge graphs for recommender systems, in: Proceedings of the 29th ACM International Conference on Information & Knowledge Management, 2020, pp. 1405–1414.
[33] Y. Feng, B. Hu, F. Lv, Q. Liu, Z. Zhang, W. Ou, ATBRG: Adaptive target-behavior relational graph network for effective recommendation, in: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020, pp. 2231–2240.
[34] J. Tang, X. Hu, H. Gao, H. Liu, Exploiting local and global social context for recommendation, in: IJCAI, Vol. 13, Citeseer, 2013, pp. 2712–2718.
[35] J. Tang, S. Wang, X. Hu, D. Yin, Y. Bi, Y. Chang, H. Liu, Recommendation with social dimensions, in: Thirtieth AAAI Conference on Artificial Intelligence, 2016.
[36] H. Ma, D. Zhou, C. Liu, M.R. Lyu, I. King, Recommender systems with social regularization, in: Proceedings of the Fourth ACM International Conference on Web Search and Data Mining, 2011, pp. 287–296.
[37] W. Fan, Y. Ma, Q. Li, Y. He, E. Zhao, J. Tang, D. Yin, Graph neural networks for social recommendation, in: The World Wide Web Conference, 2019, pp. 417–426.
[38] Q. Wu, H. Zhang, X. Gao, P. He, P. Weng, H. Gao, G. Chen, Dual graph attention networks for deep latent representation of multifaceted social effects in recommender systems, in: The World Wide Web Conference, 2019, pp. 2091–2102.
[39] J. Yu, H. Yin, J. Li, M. Gao, Z. Huang, L. Cui, Enhance social recommendation with adversarial graph convolutional networks, IEEE Trans. Knowl. Data Eng. (2020).
[40] J. Tang, X. Hu, H. Liu, Social recommendation: A review, Soc. Netw. Anal. Min. 3 (4) (2013) 1113–1133.
[41] W. Fan, Q. Li, M. Cheng, Deep modeling of social relations for recommendation, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 32, no. 1, 2018.
[42] C. Xu, Z. Guan, W. Zhao, Q. Wu, M. Yan, L. Chen, Q. Miao, Recommendation by users' multimodal preferences for smart city applications, IEEE Trans. Ind. Inf. 17 (6) (2021) 4197–4205, http://dx.doi.org/10.1109/TII.2020.3008923.
[43] X. Li, M. Zhang, S. Wu, Z. Liu, L. Wang, S.Y. Philip, Dynamic graph collaborative filtering, in: 2020 IEEE International Conference on Data Mining, ICDM, IEEE, 2020, pp. 322–331.
[44] G. Xue, M. Zhong, J. Li, J. Chen, C. Zhai, R. Kong, Dynamic network embedding survey, Neurocomputing 472 (2022) 212–223.
[45] Q. Liu, S. Shen, Z. Huang, E. Chen, Y. Zheng, A survey of knowledge tracing, 2021, arXiv preprint arXiv:2105.15106.
[46] H. Nakagawa, Y. Iwasawa, Y. Matsuo, Graph-based knowledge tracing: Modeling student proficiency using graph neural net, in: Proceedings of the International Conference on Learning Representations, ICLR, 2019, pp. 1–8.
[47] X. Song, J. Li, Q. Lei, W. Zhao, Y. Chen, A. Mian, Bi-CLKT: Bi-graph contrastive learning based knowledge tracing, Knowl.-Based Syst. 241 (2022) 108274, http://dx.doi.org/10.1016/j.knosys.2022.108274, URL: https://www.sciencedirect.com/science/article/pii/S0950705122000880.
[48] X. Song, J. Li, Y. Tang, T. Zhao, Y. Chen, Z. Guan, JKT: A joint graph convolutional network based deep knowledge tracing, Inform. Sci. 580 (2021) 510–523.
[49] J. Pennington, R. Socher, C.D. Manning, Glove: Global vectors for word representation, in: Empirical Methods in Natural Language Processing, EMNLP, 2014, pp. 1532–1543, URL: http://www.aclweb.org/anthology/D14-1162.
[50] Y. Lin, Z. Liu, M. Sun, Y. Liu, X. Zhu, Learning entity and relation embeddings for knowledge graph completion, in: Twenty-Ninth AAAI Conference on Artificial Intelligence, 2015.
[51] S. Rendle, C. Freudenthaler, Z. Gantner, L. Schmidt-Thieme, BPR: Bayesian personalized ranking from implicit feedback, 2012, arXiv preprint arXiv:1205.2618.
[52] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, O. Yakhnenko, Translating embeddings for modeling multi-relational data, Adv. Neural Inf. Process. Syst. 26 (2013).
[53] F.M. Harper, J.A. Konstan, The movielens datasets: History and context, Acm Trans. Interact. Intell. Syst. (Tiis) 5 (4) (2015) 1–19.
[54] T. Bertin-Mahieux, D.P. Ellis, B. Whitman, P. Lamere, The million song dataset, 2011.
[55] I. Cantador, P. Brusilovsky, T. Kuflik, 2Nd workshop on information heterogeneity and fusion in recommender systems (HetRec 2011), in: Proceedings of the 5th ACM Conference on Recommender Systems, in: RecSys 2011, ACM, New York, NY, USA, 2011.
[56] H. Guo, R. Tang, Y. Ye, Z. Li, X. He, DeepFM: A factorization-machine based neural network for CTR prediction, 2017, arXiv preprint arXiv:1703.04247.
[57] X. Wang, X. He, M. Wang, F. Feng, T.-S. Chua, Neural graph collaborative filtering, in: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2019, pp. 165–174.
[58] Q. Ai, V. Azizi, X. Chen, Y. Zhang, Learning heterogeneous knowledge base embeddings for explainable recommendation, Algorithms 11 (9) (2018) 137.
[59] K. Järvelin, J. Kekäläinen, Cumulated gain-based evaluation of IR techniques, ACM Trans. Inform. Syst. (TOIS) 20 (4) (2002) 422–446.