Rethinking the Influence of Knowledge Graph Embedding Towards Graph Recommendation

Zijian Wu
School of Computing
zijian.wu@u.nus.edu

Yuexin Li
School of Computing
e0718954@u.nus.edu

Yuwen Li
School of Computing
e0724274@u.nus.edu

Jingnan Zheng
School of Computing
e0718957@u.nus.edu

Abstract

To provide a more accurate, diverse, and explainable recommendation, it is compulsory to go beyond modeling user-item interactions and take side information into account. Our work mainly builds on the work of Knowledge graph attention network (KGAT), a paper investigating the utility of knowledge graph, which breaks down the independent interaction assumption by linking items with their attributes. It creates a hybrid structure of KG and user-item graphs to learn high-order relations –an essential factor for successful recommendation. KGAT explicitly models the high-order connectivities in knowledge graph (KG) in an end-to-end fashion, and recursively propagates the embeddings from a node's neighbors (which can be users, items, or attributes) to refine the node's embedding, and employs an attention mechanism to discriminate the importance of the neighbors. To better understand and intensify the utility of the knowledge graph on the KGAT model, we conduct studies on the learning of knowledge graph embedding in the pipeline. Besides TransR – a widely used parameterization method the paper employed, we try abundant methods including TransE, TransD, TransH, ComplEx, and DistMult on the original three public benchmarks with different information aggregation methods. Based on empirical results, we compare the performances among different methods and analyze the possible reason behind, focusing on the mechanism of these methods.

1 Introduction

Recommendation System plays a key role in modern information services, such as search engines, E-commerce platforms, and social medias. The successful application of recommendation systems hugely benefits from the tremendous number of user-item (UI) interactions that are easily obtained by recording the user behavior data. One of the commonly used models in recent years is collaborative filtering (CF), which evaluates similarities between users and items (user/item based CF), or learns latent vector representations for users and items (model-based CF). However, CF-based methods fail to model the side information of items, such as the profile of a product, the director of a movie, or the genre of a book. In other words, this type of methods overlook higher-order relations of items, which makes them deficient in discovering intuitively relevant items for users. Hence, they are less likely to provide high-quality recommendations in the increasingly complex scenarios.

To address the limitation, a solution is to take the graph of item side information, *aka.* knowledge graph into account to construct the predictive model. We term the hybrid structure of knowledge graph

Course Project for Neural Networks and Deep Learning II (CS5260 2022), NUS Computing.

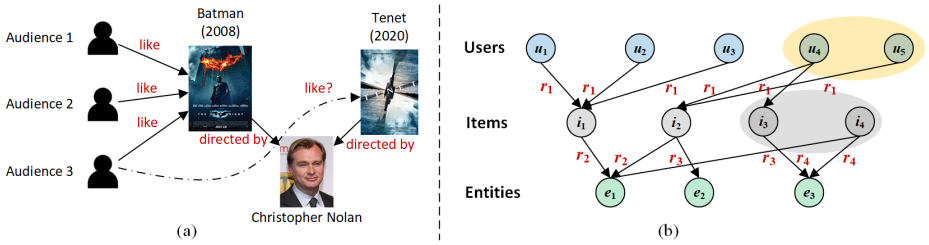

Figure 1: (a) A simple example of how side information may be utilized in graph recommendation with the help of KG. (b) An illustration of a CKG.

and user-item graph as collaborative knowledge graph (CKG). The key to successful recommendation is to fully exploit the high-order relations in CKG, *e.g.,* the long-range connectivities. Efforts have been made to leverage the CKG structure for recommendation, which can be roughly categorized into two types, path-based [1, 2], and regularization based [3, 4].

Nevertheless, the recommendation model cannot work on KG directly. The utilization effect of KG relies heavily on how the KG is embedded in the continuous latent space. Many KG embedding methods have been proposed to model the complex higher-order relationships between the entities, and they can be mainly divided into two categories[5]. One is translation-based methods [6–9] and the other is bilinear methods[10, 11]. To the best of our knowledge, no work has comprehensively studied the effect of KG embedding methods for recommendation task, even though many works have been done on knowledge-aware recommendation. Hence, it is important to extend current work and investigate how these KG embedding methods may affect the recommendation performance.

The paper KGAT proposes a model that can exploit high-order information in knowledge graph in an efficient, explicit, and end-to-end manner. The model of Knowledge Graph Attention Network (KGAT) is equipped with two designs to leverage CKG structure for recommendation: 1) recursive embedding propagation, which updates a node's embedding based on the embeddings of its neighbors. 2) attention-based aggregation, which employs the neural attention mechanism to learn the weight of each neighbor during a propagation.

For our project, We further integrate six KG embedding methods into KGAT, such that these KGAT variants is capable of discover high-order side information from different perspectives. We comprehensively evaluate the prediction performance and convergence speed on three widely used recommendation datasets, Amazon-book, Last-FM and Yelp2018, and provide insightful guidance for future use. To summarize, the contributions of our work are mainly three-fold: (1) **Novelty**. We are the first to investigate how different KG embedding methods help explore higher-order relations under knowledge-aware graph recommendation setting. (2) **Models**. We integrate different KG embedding methods into a state-of-the-art graph recommendation framework KGAT to establish its variants. (3) **Experiments and Guidance**. We conduct extensive experiments to evaluate the recommendation quality and model convergence of these KG embedding methods, and further provide suggestions of choosing KG embedding methods for future work.

2 Preliminaries

In this section, we introduce two basic graph structures related to our task, the UI interaction graph and the knowledge graph. Then we integrate these two types of graphs together to form a collaborative knowledge graph (CKG) and define the graph recommendation task based on CKG.

User-Item Interaction Graph. A UI interaction graph is essentially a bipartite graph since the users can only interact with the items. Denote the UI interaction graph as $\mathcal{G}_{\text{UI}} = (\mathcal{U}, \mathcal{I}, \mathcal{Y})$, where $\mathcal{U}, \mathcal{I}, \mathcal{Y}$ are the sets of users, items, and UI interaction directed edges, respectively. $\{interact\}$ is the only one type of edge in $\mathcal{Y}$. Hence, an interaction between a user $u \in \mathcal{U}$ and an item $i \in \mathcal{I}$ is represented as (*u, interact, i*).

Knowledge Graph. The side information is utilized in recommendation via an additional KG $\mathcal{G}_{\text{KG}}$—a knowledge base of real world facts. A fact is represented by a triplet (h, r, t), which means a head entity h has a relation r with a tail entity t. Take Figure 1(a) as an example, the fact that

the movie Tenet is directed by Christopher Nolan can be represent by a triplet (*Tenet, DirectedBy, ChristopherNolan*). Formally, we define the set of facts in $\mathcal{G}_{\text{KG}}$ as $\{(h,r,t)|h,t \in \mathcal{E}, r \in \mathcal{R}\}$, where $\mathcal{E}$ and $\mathcal{R}$ are the sets of entities and relations. It should be noted that the item set $\mathcal{I}$ in $\mathcal{G}_{\text{UI}}$ is a subset of $\mathcal{E}$ (i.e., $\mathcal{I} \subseteq \mathcal{E}$).

Collaborative Knowledge Graph. A CKG $\mathcal{G}$ is simply a combination of the UI interaction graph $\mathcal{G}_{\text{UI}}$ and the knowledge graph $\mathcal{G}_{\text{KG}}$, as it is shown in Figure 1(b). Such a unified graph enables the model to explore higher-order relations between items in addition to user behaviors for more accurate recommendation, while a single UI interaction graph fails to address. Specifically, the CKG $\mathcal{G} = \{(h,r,t)|h,r \in \mathcal{E}', r \in \mathcal{R}'\}$, where $\mathcal{E}' = \mathcal{E} \cup \mathcal{U}$ and $\mathcal{R}' = \{\textit{interact}\} \cup \mathcal{R}$.

Task Definition. The graph recommendation task is formulated as follows: given a CKG $\mathcal{G}$, the graph recommendation model is expected to output the score $\tilde{y}_{ui}$ (i.e., the possibility that u is interested in i) for each UI pair (u,i) where $u \in \mathcal{U}$ and $i \in \mathcal{I}$, such that the true (u,i) pairs have higher scores than the false ones.

3 Revisiting KG Embedding

Given a triplet (h,r,t) composed of two entities $h,t \in \mathcal{E}$ and a relation $r \in \mathcal{R}$, KG embedding methods define different energy functions $g(h,r,t)$ to measure the reliability of the triplet. The energy score is expected to be lower for a golden triplet and higher for an incorrect triplet. We explore six KG embedding methods and will introduce each of them in this section.

3.1 Translation-based Methods

Let $\mathbf{h}, \mathbf{r}, \mathbf{t}$ denote the embeddings of h, r, t. Translation-based embedding methods force the summation of head embedding and relation embedding to be close to tail embedding (e.g., $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$) for all fact triplets in their proposed embedding spaces.

TransE. TransE[6] simply assumes that the entities and relations are in the same space, and uses the squared euclidean distance as dissimilarity measure to form the energy function:

$$g(h,r,t) = \|\mathbf{h} + \mathbf{r} - \mathbf{t}\|_2^2 = \|\mathbf{h}\|_2^2 + \|\mathbf{r}\|_2^2 + \|\mathbf{t}\|_2^2 - 2\left(\mathbf{h}^\top \mathbf{t} + \mathbf{r}^\top (\mathbf{t} - \mathbf{h})\right) \tag{1}$$

However, TransE can only handle 1-to-1 relationship, while fails to handle 1-to-n, n-to-1 and n-to-n relationships. Take 1-to-n relationship as a counterexample. For a fixed pair of head entity and relation, there can be many tail entities in 1-to-n scenario. But TransE will force all tail entities to equal to the summation of the head embedding and the relation embedding. This drawback of TransE will be solved by subsequent methods.

TransH. TransH [7] is developed to overcome flaws in TransE when dealing with relations with mapping properties of reflexitve/1-to-n/n-to-1/n-to-n. In TransH, each relation is characterized by two vectors, the norm vector $\mathbf{w}_r$ of the hyperplane , and the translation vector $\mathbf{d}_r$ on the hyperplane. For a given triplet (h,r,t), the projections of h and t on the hyperplane are expected to be connected by the translation vector $\mathbf{d}_r$ with low error. This simple method overcomes the flaws of TransE in dealing with reflexive/1-to-n/n-to-1/n-to-n relations while keeping the model complexity almost the same as that of TransE.

For a relation r, we position the relation-specific translation vector $\mathbf{d}_r$ in the relation-specific hyperplane $\mathbf{w}_r$ rather than in the same space of entity embedding. By restricting $\|\mathbf{w}_r\| = 1$, it is easy to get

$$\mathbf{h}_\perp = \mathbf{h} - \mathbf{w}_r^\top \mathbf{h} \mathbf{w}_r, \quad \mathbf{t}_\perp = \mathbf{t} - \mathbf{w}_r^\top \mathbf{t} \mathbf{w}_r \tag{2}$$

Then the energy function is

$$g(h,r,t) = \|\mathbf{h}_\perp + \mathbf{d}_r - \mathbf{t}_\perp\|_2^2 \tag{3}$$

TransR. Taking a step forward from TransE and TransH, TransR[8] models the entities and relations in two distinct spaces, i.e., an *entity space* and multiple *relation spaces*. This formulation lies on the fact that an entity may have multiple aspects, and various relations focus on different aspects of entities. The computation of TransR's energy function is performed on the projected entity embedding in the relation space:

$$g(h,r,t) = \|\mathbf{M}_r \mathbf{h} + \mathbf{r} - \mathbf{M}_r \mathbf{t}\|_2^2 \tag{4}$$

where $\mathbf{M}_r$ is the projection matrix for relation r. These projection matrices provides sufficient flexibility on addressing multiple aspects of entities w.r.t different relations and handling 1-to-n/n-to-1/n-to-n relationships.

TransD. Unlike TransE/TransR/TransH, TransD[9] uses not one but *two* vectors to represent the embeddings of entities and relations . The projection matrices for head and tail entities w.r.t. relation r are calculated as follows:

$$\mathbf{M}_{rh} = \mathbf{I} + \mathbf{r}_p \mathbf{h}_p^\top, \quad \mathbf{M}_{rt} = \mathbf{I} + \mathbf{r}_p \mathbf{t}_p^\top \tag{5}$$

where the identity matrix $\mathbf{I}$ is the initialization of the projection matrices and $\mathbf{h}_p, \mathbf{r}_p, \mathbf{t}_p$ are used to construct mapping matrix dynamically. This leads to the corresponding energy function:

$$g(h, r, t) = \|\mathbf{M}_{rh}\mathbf{h} + \mathbf{r} - \mathbf{M}_{rt}\mathbf{t}\|_2^2 \tag{6}$$

There are mainly three advantages of TransD. The two vector-based dynamic projection matrix considers the diversity of both the relations and entities, while maintaining less trainable parameters. Moreover, by reorganizing the terms in formula (6), TransD can get rid of matrix-vector multiplication and involves vector-vector multiplication only, making it suitable for large-scale knowledge graphs.

3.2 Bilinear Methods

Bilinear methods, instead, model a relation r as a bilinear product between entity embeddings[5].

ComplEx. ComplEx [10] follows the commonly-used dot product decomposition. However, the traditional decomposition is often used to approximate real symmetric matrices, while ComplEx is interested in anti-symmetric cases. Since the real space cannot represent anti-symmetric scenarios, ComplEx chooses to do decomposition in the complex space, where one embedding $\mathbf{x} \in \mathbb{C}^K$ is composed of a real component $\text{Re}(\mathbf{x})$ and an imaginary component $\text{Im}(\mathbf{x})$. In ComplEx's setting, given a single type of binary relationship and a set of n entities, the latent score matrix $\mathbf{X} \in \mathbb{R}^{n \times n}$ can be represented as:

$$\mathbf{X} = \text{Re}\left(\mathbf{E}\mathbf{W}\bar{\mathbf{E}}^\top\right), \tag{7}$$

where $\mathbf{W} \in \mathbb{C}^{n \times n}$ is the diagonal matrix of eigenvalues with decreasing modulus, $\mathbf{E} \in \mathbb{C}^{n \times n}$ is a unitary matrix of eigenvectors, $\bar{\mathbf{E}}$ is its complex conjugate. Equation 7 only keeps the real part to ensure a real score.

ComplEx further makes an assumption that the $\mathbf{E}\mathbf{W}\bar{\mathbf{E}}^\top$ has a low rank of $K \ll n$, which has been justified by previous researches. Then only the first K values of $\mathbf{W}$ are non-zero and we can make adjustment $\mathbf{W} \in \mathbb{C}^{K \times K}$ and $\mathbf{E} \in \mathbb{C}^{n \times K}$. In this case, given a head embedding $\mathbf{h} \in \mathbb{C}^K$ and a tail embedding $\mathbf{t} \in \mathbb{C}^K$, their corresponding score is:

$$\mathbf{X}_{ht} = \text{Re}\left(\mathbf{h}^\top \mathbf{W} \bar{\mathbf{t}}\right). \tag{8}$$

As for multi-relational data, we just need to calculate the score matrices $\mathbf{X}_r$ for all relations $r \in \mathcal{R}$. Then the final energy function of ComplEx is:

$$\begin{aligned} g(h, r, t) &= \text{Re}\left(< \mathbf{r}, \mathbf{h}, \bar{\mathbf{t}} >\right) = \text{Re}\left(\Sigma_{k=1}^K \mathbf{r}_k \mathbf{h}_k \bar{\mathbf{t}_k}\right) \\ &= \langle \text{Re}(\mathbf{r}), \text{Re}(\mathbf{h}), \text{Re}(\mathbf{t}) \rangle + \langle \text{Re}(\mathbf{r}), \text{Im}(\mathbf{h}), \text{Im}(\mathbf{t}) \rangle \\ &\quad + \langle \text{Im}(\mathbf{r}), \text{Re}(\mathbf{h}), \text{Im}(\mathbf{t}) \rangle - \langle \text{Im}(\mathbf{r}), \text{Im}(\mathbf{h}), \text{Re}(\mathbf{t}) \rangle . \end{aligned} \tag{9}$$

DistMult. Instead of use complex metrics, DistMult [11] reduces the number of relations parameters by using a diagonal matrix only and models entities and relations as vectors in $\mathbb{R}^d$. It uses an entry-wise product ($\odot$) to measure compatibility between head and tail entities, while using logistic loss for training the model. The learned entity representations, $\mathbf{h}', \mathbf{t}' \in \mathbb{R}^d$ can be written as:

$$\mathbf{h}' = f(\mathbf{W}\mathbf{h}), \quad \mathbf{t}' = f(\mathbf{W}\mathbf{t}), \tag{10}$$

where $f(\cdot)$ can be a linear or non-linear function, and $\mathbf{W}$ is a parameter matrix, which can be randomly initialized or initialized using some pre-trained vectors. The energy function for DistMult can be expressed as:

$$g(h, r, t) = \mathbf{r}^\top (\mathbf{h}' \odot \mathbf{t}') \tag{11}$$

Since the entry-wise product is symmetric, DistMult is not suitable for anti-symmetric relations.

4 KGAT

We now present the enhanced KGAT with various knowledge graph embedding components model, which exploits the influence of different knowledge graph embedding components towards high-order relations representation in an end-to-end fashion. Figure 2 illustrates the model framework with three main components.

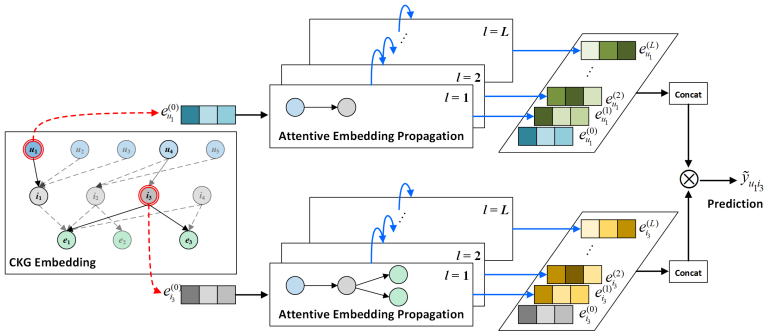

Figure 2: An illustration of the KGAT model.

4.1 Embedding Layers

Embedding layer exploits high-order connectivities and victories each node by preserving the structure of CKG. It provides an effective way to parameterize entities and relations as vector representations while preserving the graph structure. However, currently, different knowledge graph embedding methods can only model a subset of all relation types efficiently. For instance, TransE has a relatively satisfying expression effect for combinational relations, while ComplEx is better at modeling asymmetric relations. Hence, Our proposed work explores the impact of different knowledge graph embedding methods on the expressiveness of the model.

Both the training of translation-based and bilinear-based knowledge graph embedding layers adopt negative sampling strategy following the settings in [12], considering the relative order between valid triplets and broken ones, and optimise their difference through a pairwise ranking loss:

$$\mathcal{L}_{\text{KG}} = \sum_{(h,r,t,t')\in\mathcal{T}} -\ln \sigma(g(h,r,t') - g(h,r,t)), \tag{12}$$

where $\mathcal{T} = \{(h,r,t,t')|(h,r,t) \in \mathcal{G}, (h,r,t') \notin \mathcal{G}\}$, and (h,r,t') is a broken triplet constructed by negative sampling strategy.

4.2 Attentive Embedding Propagation Layers

We now consider how to recursively aggregates and propagates embeddings from a node's neighbors following message passing protocol and update its vectorized representation. In order to propagate information, considering an entity h, we use $\mathcal{N}_h$ to denote its neighborhood. To capture entity h's first order connectivity structure information, we compute the linear combination from h's neighbor domain:

$$\mathbf{e}_{\mathcal{N}_h} = \sum_{(h,r,t)\in\mathcal{N}_h} \pi(h,r,t)\mathbf{e}_t, \tag{13}$$

where $\pi(h,r,t)$ controls the attention weights along each propagation edge, which is formulated as follows:

$$\pi(h,r,t) = \text{SoftMax}((\mathbf{W_r e_t})^\top \tanh((\mathbf{W_r e_h} + \mathbf{e_r}))). \tag{14}$$

The final phase is to aggregate the learned entity $\mathbf{e}_h$ and neighbor domain representation as the new representation of entity h. Three types of aggregators $f(\cdot)$ are implemented:

GCN Aggregator receives two representations up and applies a graph kernel transformation:

$$f_{\text{GCN}} = \text{LeakyReLU}(\mathbf{W}(\mathbf{e}_h + \mathbf{e}_{\mathcal{N}_h})), \tag{15}$$

where $\mathbf{W} \in \mathbb{R}^{d' \times d}$ are the trainable GCN weight matrices to distill useful information from neighbor domain, and d' is the hidden size.

GraphSage Aggregator concatenates two representations together and applies a nonlinear transformation:

$$f_{\text{GraphSage}} = \text{LeakyReLU}(\mathbf{W}(\mathbf{e}_h || \mathbf{e}_{\mathcal{N}_h})), \tag{16}$$

where $||$ denotes concatenation operation.

Bi-Interaction Aggregator follows the definition in [12] to take two kinds of feature interactions into consideration:

$$f_{\text{Bi-Interaction}} = \text{LeakyReLU}(\mathbf{W}(\mathbf{e}_h + \mathbf{e}_{\mathcal{N}_h})) + \text{LeakyReLU}(\mathbf{W}(\mathbf{e}_h \odot \mathbf{e}_{\mathcal{N}_h})), \tag{17}$$

where $\mathbf{W}_1, \mathbf{W}_2 \in \mathbb{R}^{d' \times d}$ are the trainable weight matrices and $\odot$ denotes the element-wise product.

4.3 Prediction Layer

Propagating information after L layers, the model receives high order information and obtains multiple representations for user node u and item node i. To capture information from different connectivities, we concatenate the representations at each layer into the final representation, as follow:

$$\mathbf{e}_u^* = \mathbf{e}_u^{(0)} || ... || \mathbf{e}_u^{(L)}, \mathbf{e}_i^* = \mathbf{e}_i^{(0)} || ... || \mathbf{e}_i^{(L)}. \tag{18}$$

Finally, we simply conduct inner product of user and item representations to preditc their matching score:

$$\hat{y}(u, i) = {\mathbf{e}_u^*}^\top \mathbf{e}_i^*. \tag{19}$$

4.4 Optimization

To optimize the graph collaborative filtering based recommendation model, we adopt BPR loss [13] to assign higher prediction values for observed ones than unobserved ones:

$$\mathcal{L}_{\text{CF}} = \sum_{(u,i,j) \in \mathcal{O}} - \ln \sigma(\hat{y}(u, i) - \hat{y}(u, j)), \tag{20}$$

where $\mathcal{O} = \{(u, i, j) | (u, i) \in \mathcal{R}^+, (u, j) \in \mathcal{R}^-\}$, denotes the training set, $\mathcal{R}^+$ indicates the observed interactions while $\mathcal{R}^-$ is the sampled unobserved interaction set.

Finally, the joint optimization object for knowledge graph and collaborative filtering is defined as:

$$\mathcal{L}_{\text{KGAT}} = \mathcal{L}_{\text{KG}} + \mathcal{L}_{\text{CF}} + \lambda ||\Theta||_2^2, \tag{21}$$

where Θ is all parameters defined in the model and L_2 regularization controlled by λ is adopted to prevent overfitting.

5 Experiment

We evaluate six embedding methods on three datasets (amazon-book, last-fm and yelp2018) with three aggregation methods (bi-interaction, graphsage and GCN). Results of three metrics (precision, recall and NDCG) are reported. In this section, we will first introduce the datasets, then the metrics, and finally give the performance report.

5.1 Datasets

In our experiment, we use three popular benchmarks. Amazon-book [14] is for product recommendation. Last-FM is a music listening dataset from online music systems. Yelp2018 is the business dataset for 2018 Yelp challenge. We adopt the 10-core setting to ensure stability, which means we only retain users and items with at least 10 interactions between them. Details of the three datasets are shown in Table 1.

Table 1: Datasets

		Amazon-book	Last-FM	Yelp2018
User-Item Interaction	#Users	70,679	23,566	45,919
	#Items	24,915	48,123	45,538
	#Interactions	847,733	3,034,796	1,185,068
Knowledge Graph	#Entities	88,572	58,266	90,961
	#Relations	39	9	42
	#Triplets	2,557,746	464,567	1,853,704

5.2 Metrics

We use three commonly-used metrics in evaluating recommendation system: precision, recall and NDCG (Normalized Discounted Cumulative Gain). Since neither precision or recall considers the rank of the recommended items, we introduce NDCG here. Given k recommended items, each item i has a score rel_i, DCG is calculated as:

$$DCG_k = \sum_{i=1}^{k} \frac{2^{rel_i} - 1}{log_2(i+1)}. \tag{22}$$

Then we can calculate NDCG:

$$NDCG_k = \frac{DCG_k}{IDCG_k}, \tag{23}$$

where $IDCG$ is ideal DCG, which means the returned rel_i matches the real rank. The range of NDCG is from 0 to 1. Closer to 1 means better recommendation.

5.3 Performance

The comparison results are reported in Table 2, 3 and 4. We use early stop to prevent overfitting and the best epoch number is also reported. From the tables, we can see that TransR performs best on Amazon-Book, ComplEx performs best on Last-FM, while TransD and TransR tie for the first place on Yelp2018.

For translation-based methods, TransE can only handle 1-to-1 relationship, so it is not surprising that TransE gets a poor performance. As for TransH, although TransH covers 1-to-n, n-to-1 and n-to-n relationships, it assumes that entity and relation are in the same space, which fails to consider different aspects of an entity. As a result, TransH also get a poor performance. TransR and TransD both cover 1-to-n, n-to-1 and n-to-n relationships and assume that entity and relation are in different spaces. In TransR, the mapping from entity space to relation space is relation dependent, while in TransD, it depends on both entity and relation. From the results, it seems that relation-dependent mapping is enough in most cases.

For bilinear methods, ComplEx performs better than DistMult in all three datasets. This meets our expectation since DistMult is not applicable for anti-symmetric relations, while ComplEx can handle anti-symmetric relations.

Different embedding methods have different emphases. Different datasets have different characteristics. Choosing an embedding method for a dataset requires case-by-case analysis. However, there are some rules which can be concluded from our experiments. First, 1-to-n, n-to-1 and n-to-n relationships are common in most datasets, which means embedding methods should better handle all these cases. Second, anti-symmetric relation is not rare. A good embedding method should cover both symmetric and anti-symmetric relations. Third, different relations focus on different aspects of entities, so assuming entity and relation are in different spaces is a good choice.

To better illustrate the efficiency of different embedding methods, we draw the best epoch number in Figure 3. We can see that TransE and TransH are among the fastest in most cases, while TransR and DistMult are among the slowest. Considering their performances, we think TransD is an economic choice for these three datasets.

Table 2: Experiment Results of Different Embedding Methods on Amazon-book.

		Best Epoch	Precision@20	Precision@100	Recall@20	Recall@100	NDCG@20	NDCG@100
Bi-interaction	TransE	120	0.0144	0.0069	0.1376	0.2994	0.0739	0.1105
	TransH	60	0.0147	**0.0071**	0.1404	0.3066	0.0753	0.1129
	TransR	300	**0.0150**	**0.0071**	**0.1443**	**0.3089**	**0.0758**	**0.1132**
	TransD	210	0.0142	0.0069	0.1377	0.3042	0.0716	0.1093
	ComplEx	280	0.0146	0.0070	0.1404	0.3025	0.0738	0.1107
	DistMult	310	0.0144	0.0069	0.1397	0.3003	0.0733	0.1102
GCN	TransE	40	0.0144	0.0069	0.1382	0.3006	0.0746	0.1114
	TransH	60	**0.0149**	**0.0071**	0.1412	0.3084	0.0757	0.1136
	TransR	360	**0.0149**	**0.0071**	**0.1439**	**0.3106**	**0.0759**	**0.1138**
	TransD	170	0.0141	0.0070	0.1371	0.3069	0.0707	0.1091
	ComplEx	130	0.0147	0.0070	0.1413	0.3032	0.0745	0.1113
	DistMult	260	0.1445	0.0068	0.1400	0.2974	0.0738	0.1096
GraphSAGE	TransE	50	0.0142	0.0069	0.1363	0.2997	0.0732	0.1102
	TransH	80	0.0145	0.0070	0.1389	0.3046	0.0740	0.1115
	TransR	440	**0.0147**	**0.0071**	**0.1425**	**0.3094**	**0.0749**	**0.1128**
	TransD	360	0.0143	0.0069	0.1375	0.3041	0.0727	0.1103
	ComplEx	180	0.0146	0.0069	0.1406	0.3008	0.0742	0.1106
	DistMult	280	0.0144	0.0068	0.1400	0.2969	0.0740	0.1098

Table 3: Experiment Results of Different Embedding Methods on Last-FM.

		Best Epoch	Precision@20	Precision@100	Recall@20	Recall@100	NDCG@20	NDCG@100
Bi-interaction	TransE	120	0.0340	0.0170	0.0846	0.1665	0.0734	0.1002
	TransH	100	0.0344	0.0170	0.0848	0.1661	0.0742	0.1007
	TransR	460	0.0336	0.0169	0.0851	0.1665	0.0725	0.0995
	TransD	250	0.0339	0.0172	0.0859	0.1700	0.0736	0.1015
	ComplEx	650	**0.0359**	**0.0178**	**0.0933**	**0.1782**	**0.0787**	**0.1072**
	DistMult	650	0.0346	0.0171	0.0885	0.1702	0.0745	0.1017
GCN	TransE	100	0.0341	0.0171	0.0847	0.1674	0.0735	0.1006
	TransH	90	0.0341	0.0170	0.0842	0.1668	0.0735	0.1004
	TransR	530	0.0336	0.0168	0.0851	0.1660	0.0723	0.0991
	TransD	270	0.0341	0.0173	0.0854	0.1697	0.0737	0.1015
	ComplEx	520	**0.0360**	**0.0179**	**0.0930**	**0.1790**	**0.0788**	**0.1077**
	DistMult	430	0.0344	0.0172	0.0879	0.1716	0.0741	0.1020
GraphSAGE	TransE	100	0.0336	0.0170	0.0838	0.1673	0.0719	0.0992
	TransH	110	0.0338	0.0170	0.0837	0.1671	0.0730	0.1003
	TransR	460	0.0335	0.0169	0.0848	0.1662	0.0721	0.0989
	TransD	180	0.0340	0.0173	0.0854	0.1706	0.0733	0.1012
	ComplEx	510	**0.0362**	**0.0179**	**0.0938**	**0.1793**	**0.0793**	**0.1080**
	DistMult	650	0.0346	0.0171	0.0888	0.1707	0.0750	0.1022

Table 4: Experiment Results of Different Embedding Methods on Yelp2018.

		Best Epoch	Precision@20	Precision@100	Recall@20	Recall@100	NDCG@20	NDCG@100
Bi-interaction	TransE	70	0.0162	0.0099	0.0664	0.1964	0.0423	0.0793
	TransH	70	0.0161	**0.0100**	0.0662	0.1975	0.0421	0.0794
	TransR	70	0.0163	**0.0100**	0.0674	**0.1989**	0.0430	0.0803
	TransD	30	**0.0167**	**0.0100**	**0.0688**	0.1987	**0.0443**	**0.0811**
	ComplEx	60	0.0160	0.0098	0.0662	0.1949	0.0421	0.0786
	DistMult	160	0.0162	0.0099	0.0667	0.1963	0.0428	0.0796
GCN	TransE	100	0.0164	0.0100	0.0676	0.1986	0.0429	0.0799
	TransH	60	0.0163	0.0100	0.0675	0.1986	0.0429	0.0800
	TransR	130	**0.0166**	**0.0101**	**0.0685**	**0.2004**	0.0438	**0.0812**
	TransD	80	0.0165	0.0100	0.0682	0.1993	**0.0440**	**0.0812**
	ComplEx	160	0.0162	0.0099	0.0670	0.1971	0.0426	0.0795
	DistMult	190	0.0163	0.0099	0.0671	0.1970	0.0427	0.0795
GraphSAGE	TransE	80	0.0162	0.0099	0.0665	0.1973	0.0427	0.0798
	TransH	60	0.0162	0.0100	0.0667	0.1987	0.0427	0.0801
	TransR	260	0.0165	**0.0101**	0.0682	**0.2004**	0.0431	0.0807
	TransD	170	**0.0166**	**0.0101**	**0.0690**	0.1999	**0.0440**	**0.0811**
	ComplEx	80	0.0163	0.0099	0.0672	0.1973	0.0427	0.0795
	DistMult	240	0.0159	0.0098	0.0671	0.1965	0.0425	0.0792

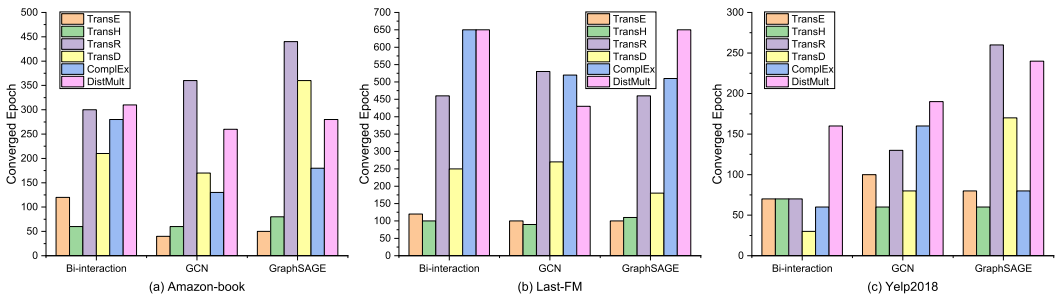

Figure 3: Convergence speed comparison of KGAT with different KG embedding methods and aggregation layers.

6 Conclusions

Our work builds on the work of KGAT to further explore the utility of knowledge graph embedding methods. KGAT explores high-order connectivity with semantic relations in CKG for knowledge-aware recommendation. It devised a new framework KGAT, which explicitly models the high order connectivities in CKG in an end-to-end fashion. At its core is the attentive embedding propagation layer, which adaptively propagates the embeddings from a node's neighbors to update the node's representation. We continue to study the effect of using different knowledge graph embedding methods on the performance and conduct analysis based on the mechanisms of these different methods. TransR is proved to be the best among all the KG embedding layers in our experiments.

References

[1] Sun, Z., J. Yang, J. Zhang, et al. Recurrent knowledge graph embedding for effective recommendation. In *Proceedings of the 12th ACM Conference on Recommender Systems*, pages 297–305. 2018.

[2] Wang, H., F. Zhang, J. Wang, et al. Ripplenet: Propagating user preferences on the knowledge graph for recommender systems. In *Proceedings of the 27th ACM international conference on information and knowledge management*, pages 417–426. 2018.

[3] Huang, J., W. X. Zhao, H. Dou, et al. Improving sequential recommendation with knowledge-enhanced memory networks. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 505–514. 2018.

[4] Wang, X., D. Wang, C. Xu, et al. Explainable reasoning over knowledge graphs for recommendation. In *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, pages 5329–5336. 2019.

[5] Zhang, Y., Q. Yao, J. T. Kwok. Bilinear scoring function search for knowledge graph learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2022.

[6] Bordes, A., N. Usunier, A. Garcia-Duran, et al. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26, 2013.

[7] Wang, Z., J. Zhang, J. Feng, et al. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 28. 2014.

[8] Lin, Y., Z. Liu, M. Sun, et al. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, page 2181–2187. AAAI Press, 2015.

[9] Ji, G., S. He, L. Xu, et al. Knowledge graph embedding via dynamic mapping matrix. In *Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing*, pages 687–696. 2015.

[10] Trouillon, T., J. Welbl, S. Riedel, et al. Complex embeddings for simple link prediction. In *International conference on machine learning*, pages 2071–2080. PMLR, 2016.

[11] Yang, B., W.-t. Yih, X. He, et al. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*, 2014.

[12] Wang, X., X. He, Y. Cao, et al. Kgat: Knowledge graph attention network for recommendation. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 950–958. 2019.

[13] Rendle, S., C. Freudenthaler, Z. Gantner, et al. Bpr: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618*, 2012.

[14] He, R., J. McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*, pages 507–517. 2016.