UTS

Studying the effect of Graph Embedding methods on Knowledge Graph Attention Networks for Recommendation

Honours Project
Patrick Songco 10420139
Supervisor: Angela Huo

Abstract

Insufficient attention has been paid to the effect of the design choices when integrating knowledge graphs with user-item interactions in recommender systems. Improving knowledge graph embeddings can encode more useful information and provide a richer representation of the network as well as the users and items, thus, improving recommendation performance. To fill this gap, this paper provides a theoretical analysis and empirical results on the performance of various knowledge graph embedding methods on recommendation accuracy. Specifically, we investigate the performance of three knowledge graph embedding methods on three widely used datasets and show state-of-the-art recommendation performance when compared to our baseline.

Table of Contents

1 Introduction **3**

2 Background **4**

2.1 Recommendation Systems 4

2.2 Knowledge Graph Embeddings 5
2.2.1 Matrix Factorization Models 6
2.2.2 Deep Learning Models 7
2.2.3 Geometric Models 7

3 Research Motivation **8**

4 Methodology **9**

4.1 Task Formulation 9

4.2 KGAT model 9
4.2.1 Embedding Layer 10
4.2.2 Attentive Embedding Propagation Layers 10
4.2.3 Model Prediction 11
4.2.4 Optimization 12

4.3 Experiments 12
4.3.1 Datasets 12
4.3.2 Evaluation Metrics 13
4.3.3 Baseline (TransR) 13
4.3.4 TransD 13
4.3.5 RotatE 14
4.3.6 HAKE 14
4.3.7 Parameter Settings 16

5 Experimental Evaluation **16**

6 Discussion **17**

7 Conclusion and Future Work **19**

References **20**

1 Introduction

In the era of big data, recommendation systems have become an essential tool for consumers to overcome the information overload problem (Edmunds & Morris, 2000; Ricci *et al*., 2015). Traditional recommendation methods such as collaborative filtering (CF) that only consider the user-item interaction matrix experience issues around data sparsity and the cold start problem. These problems relate to the vast number of items available making it harder for a recommender system to learn a users' preference (data sparsity) and the fact that new users and items do not have historical data to aid in generating suitable recommendations (cold start). To overcome these problems, methods such as content-based filtering and a hybrid of both CF and content-based methods have been developed that utilize auxiliary data to enrich the representation of both users and/or items, producing superior recommendations. Consequently, progressively more complex auxiliary data or side information such as social network and knowledge graph information have been combined with increasingly advanced methodologies to effectively capture the non-linear and non-trivial user-item relationships. While these advanced methods have empirically proven to outperform methods that do not exploit the rich value of side information (Wu *et al*., 2020; Sun *et al*., 2019b), insufficient attention has been paid to the effect of the design choices when integrating this auxiliary data. In particular, the choice of knowledge graph embedding (KGE) methods when knowledge graphs are incorporated into recommendation models. While KGE methods have been compared in the past (Wang *et al*., 2017), to our knowledge, the effectiveness of KGE methods in the downstream task of recommendation has yet to be explored. As such, in this paper, we aim to answer the question: **Does improving the expressivity of knowledge graph embedding methods improve recommendation accuracy?**

To answer our question, we perform a comparative study using a state-of-the-art graph neural network-based model, Knowledge Graph Attention Network (KGAT) first proposed by Wang *et al*. (2019) as a baseline. The baseline method fails to consider:

1. The diversity of entities and relations simultaneously.
2. Symmetry/asymmetry, inversion, and composition simultaneously.
3. Semantic Hierarchies.

Therefore, we substitute the baseline KGE method, TransR (Lin *et al*., 2015) with TransD (Ji *et al*., 2015), RotatE (Sun *et al*., 2019b) and HAKE (Zhang *et al*., 2019c); KGE methods that attempt to capture these three properties respectively. Since the KGAT model has been empirically proven to demonstrate state-of-the-art recommendation accuracy on the Amazon Book, Last-FM and Yelp2018 datasets, we empirically show that two of the three considerations produce a model that yields better recommendation accuracy. The rest of this study is organized as follows: Section 2 introduces recommendation systems and knowledge graph embeddings, Section 3 outlines our research motivation, Section 4 describes our methodology, Section 5 evaluates our experimental results, Section 6 provides a discussion on our findings and finally, Section 7 provides some future research directions.

2 Background

2.1 Recommendation Systems

Recommendation systems were first introduced in a technical report by Jussi Karlgren in 1990. The objective was to investigate and formalize digital document recommendation giving users a greater chance of finding documents they did not know to look for. Today, with the exponential growth of online information, recommender systems play a key role in facilitating the decision-making process for consumers and providing an indispensable tool for businesses (Covington et al., 2016; Zhang *et al*., 2019a).

Early recommendation approaches such as collaborative filtering (Bell & Koren, 2007; Sarwar *et al*., 2001), factorization machines (Rendle, 2010) and matrix factorization techniques (Koren *et al*., 2009) focused on recommending similar users or items to a target user by assuming that the target user's preference can be inferred by their interaction history alone. However, the vast number of items offered typically resulted in a sparsely populated interaction matrix making it difficult for these methods to learn a users' preference. This is known as the data sparsity problem. Furthermore, this dependency on a user's interaction history gave rise to problems where new users and items do not have historical data to aid in generating suitable recommendations. This is known as the cold start problem. Consequently, content-based and hybrid content-based/collaborative filtering-based methods were developed to overcome these issues by introducing various side information, producing superior recommendations despite the insufficient user-item historical interaction data.

Initial works focused on incorporating simple auxiliary information such as movie and music genres for better movie or music recommendation (Choi *et al*., 2012; Koenigstein *et al*., 2011). Progressively, researchers experimented on using more diverse side information such as text (Seo *et al*., 2017), image (Niu *et al*., 2018), network features (Guo *et al*., 2015) and knowledge graphs (Catherine & Cohen, 2016) that prompted a shift in the modelling paradigm from memory based or neighbourhood based approaches to model based or representational learning-based approaches (**Figure 1**). This evolution was motivated by the fact that 1) representational learning approaches more effectively capture the non-linear and non-trivial user-item relationships to uncover more complex user behaviour patterns and 2) the rise in popularity and research advances made in deep learning architectures.

Although these deep learning methods have empirically proven to achieve superior recommendation performance (Wu *et al*., 2020), there has been insufficient research into how to effectively incorporate the diverse types of side information (Sun *et al*., 2019b). In particular, the design choices when integrating knowledge graphs.

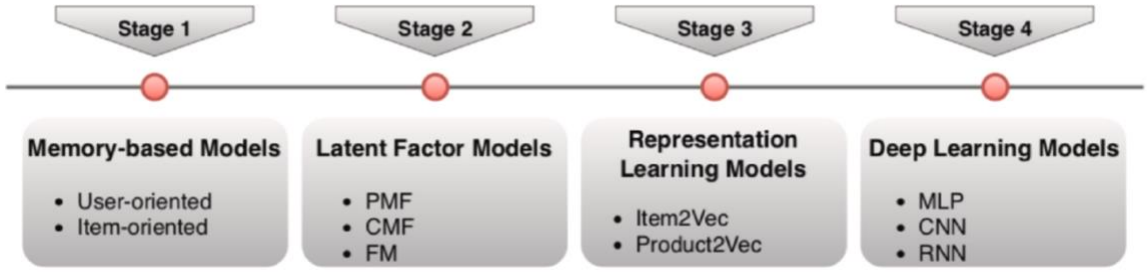


Figure 1: Evolution of recommendation approaches (image adapted from Sun et al. 2019b)

Compared to homogeneous side information such as movie or music genres, knowledge graph information allows a model to integrate heterogeneous information such as higher order associations that help infer subtler user or item relationships. Incorporating knowledge graph information has been achieved in earlier approaches via meta-path based latent features of the network structure (Yu *et al*., 2013) and graph-based latent factors (GraphLF; Catherine & Cohen, 2016) to supplement content-based recommendation. More recently, deep learning methods such as Collaborative Knowledge Base Embedding (CKE; Zhang *et al*.,2016), Knowledge enhanced Translation-based User Preference (KTUP; Cao et al., 2019) and Knowledge Graph Attention Network (KGAT; Wang *et al*., 2019) jointly learn the latent representations of users and items in the collaborative filtering component as well as the items' semantic representation from the knowledge graph in an end to end manner. These methods not only reduce the reliance on the tedious feature engineering process used in meta-path and latent factor-based approaches, but also generate state-of-the-art recommendations through non-linear transformations capturing the highly complex user-item interactions. However, these methods fail to consider the complex relationships the knowledge graph embedding methods attempt to encode.

2.2 Knowledge Graph Embeddings

Knowledge graphs (KGs) are a collection of triplet facts in the form head, relation and tail (denoted as $(\boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t})$). We refer to head & tail facts as entities and are represented as nodes in the KG. These entities are connected by edges in the KG that represent the type of relation between the entities. KGs expressivity as a form of machine readable structured human knowledge has aided downstream machine learning tasks such as question answering (Abujabal *et al*., 2017), natural language processing tasks (Wang *et al*., 2018) and recommendation. Recently, knowledge graph embedding (KGE) methods have been developed to encode the latent semantics implied in the triplet facts into low dimensional dense vectors in continuous vector space. These embeddings attempt to encode aspects such as the symmetry, asymmetry, inversion and composition of entities and relations (**Figure 2**). A comprehensive KGE is able to model these properties, as well as N-to-N relationships, hierarchies, type constraints, transivity, homophily and long-range dependencies. However, there is a tradeoff between capturing all these properties (expressivity) and scalability (model complexity/time to train the model).

Symmetry	<Alice marriedTo Bob>
Asymmetry	<Alice childOf Jack>
Inversion	<Alice childOf Jack> <Jack fatherOf Alice>
Composition	<Alice childOf Jack> <Jack siblingOf Mary> <Alice nieceOf Mary>

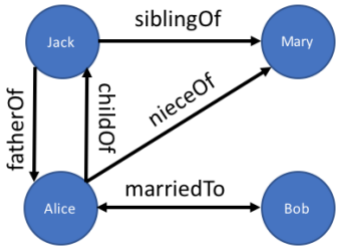


Figure 2: Simple Knowledge Graph Example

Since the introduction of the first KGE model, TransE (Bordes et al., 2013), based on the semantic translation theory (i.e. $\boldsymbol{h} + \boldsymbol{r} \approx \boldsymbol{t}$ holds when $(\boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t})$ is a true triple fact, $\boldsymbol{h} + \boldsymbol{r} \neq \boldsymbol{t}$ otherwise), several KGE models have been developed to aid in the task of link prediction and knowledge graph completion. As discussed earlier, more recently, KGE methods have also been used for downstream tasks that employ representational learning paradigms such as recommendation. These KGE models can be broadly categorized into Matrix Factorization, Deep Learning and Geometric families (**Figure 3**). All the approaches employ some sort of scoring function to determine the plausibility of any triplet fact, $(\boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t})$, which is then used to train the models by comparing the plausibility score with the true labels.

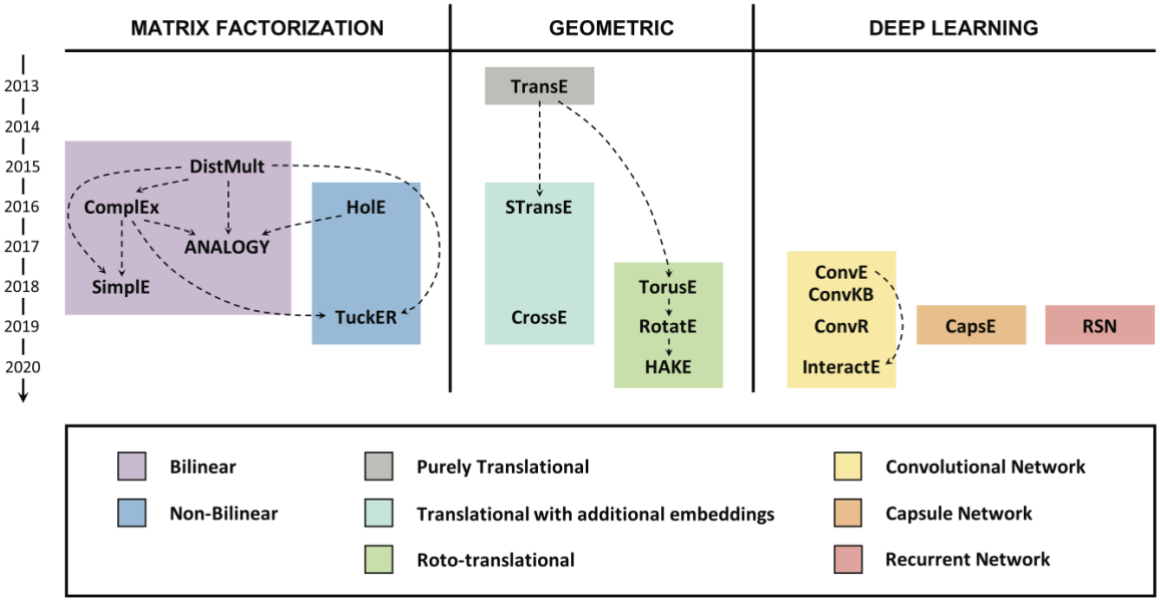


Figure 3: Taxonomy of Knowledge Graph Embedding methods. Dotted arrows indicate that the target method builds on the source method by either generalizing or specializing the definition of its scoring function. (Image adapted from Rossi et al. 2021)

2.2.1 Matrix Factorization Models

Matrix factorization models formulate the problem as a three-way tensor decomposition. It observes the KG input as a three-dimensional adjacency matrix that can be decomposed into a combination of low-dimensional vectors. The scoring function is computed by applying a combination operator, such as the bilinear product, on the embeddings of a facts' head,

relation and tail elements. The benefit of this model family is that they are light and easy to train because they employ few shared parameters. Models such as DistMult (Yang *et al*., 2014), ComplEx (Trouillon *et al*., 2016) and Analogy (Liu *et al*., 2017) have shown to successfully model symmetric and asymmetric relations and empirically proven to outperform TransE in link prediction tasks.

2.2.2 Deep Learning Models

Deep learning models use a range of the existing deep neural network architectures such as convolutional neural networks (ConvE: Dettmers *et al*., 2017; InteractE: Vashishth *et al*., 2019), capsule networks (CapsE: Nguyen *et al*., 2018) and recurrent neural networks (RSN: Guo *et al*., 2019) to learn the KGEs using various layers interspersed with non-linear activation functions. The complexity of these architectures allows the models to perform techniques such as 2D reshaped embeddings, feature permutation, checkered reshaping and circular convolutions to capture entity and relation feature interactions, thus enriching the embedding expressivity. This family of models have empirically proven to outperform existing Matrix Factorization and Geometric models in link prediction tasks; however, they have the disadvantage of taking longer to train and are more prone to overfitting.

2.2.3 Geometric Models

As described earlier, geometric models are based on the semantic translation theory. They interpret relations as geometric operations in the latent space. The simplest example of this principle is in the TransE model (**Figure 4a**). Where the scoring function is determined by the distance between the transformed head embedding vector ($\boldsymbol{h} + \boldsymbol{r}$) and tail embedding vector ($\boldsymbol{t}$) in the latent space (**Figure 4b**). However, by performing translation on entities and relations in the same vector space, TransE fails to model one-to-N, N-to-one, symmetric or transitive relations. Therefore, subsequent geometric models have addressed these weaknesses by performing translation on a hyperplane (TransH: Wang *et al*., 2014), projecting entities from entity space to relation space (TransD: Ji *et al*., 2015), adding relation-specific and entity-specific embeddings (STransE: Nguyen *et al*., 2016; CrossE: Zhang *et al*., 2019b) and finally, performing rotation like transformations in combination with translations (RotatE: Sun *et al*., 2019b; HAKE: Zhang *et al*., 2019c). Together, these methods have expanded the expressivity of KGEs, yet future research is necessary to assess whether these improvements translate to downstream tasks.

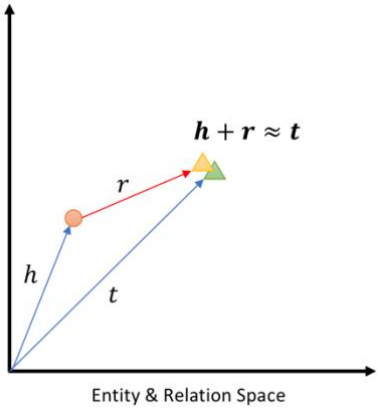


Figure 4a: TransE geometric translation in latent space

$$f(\boldsymbol{h}, \boldsymbol{t}) = -\|\boldsymbol{h} + \boldsymbol{r} - \boldsymbol{t}\|_2^2$$

Figure 4b: TransE score function

3 Research Motivation

In this section, we present the motivation for our experimental approach and how our contribution examines whether improving the expressivity of knowledge graph embedding methods improves recommendation accuracy.

Current state-of-the-art recommendation systems that utilise knowledge graphs as additional side information fail to consider the complex relationships that knowledge graph embedding (KGE) methods attempt to encode. Since the effect of KGE methods when designing models for downstream tasks has gained little research consideration, our study aims to answer the question: Does improving the expressivity of knowledge graph embedding methods improve recommendation accuracy?

Since insufficient attention has been paid to the effect of knowledge graph embedding (KGE) methods when designing models for downstream tasks such as recommendation, this study compares KGE methods in a state-of-the-art graph neural network-based model, Knowledge Graph Attention Network (KGAT), to empirically demonstrate that improving the expressivity of knowledge graph embedding methods lead to improved recommendation accuracy.

The KGAT algorithm was chosen for the study because it is conceptually advantageous to existing Knowledge Graph-based recommendation methods since it explicitly models the high-order connectivity in the knowledge graph in an end-to-end fashion. Additionally, the KGAT model uses TransR as the KGE method, which models entities and relations in distinct spaces, then performs translation in relation space. However, this method fails to consider:

1. The diversity of entities and relations simultaneously.
2. Symmetry/asymmetry, inversion, and composition simultaneously.
3. Semantic Hierarchies.

Therefore, the TransD, RotatE and HAKE KGE methods were explored as these approaches attempt to capture these properties.

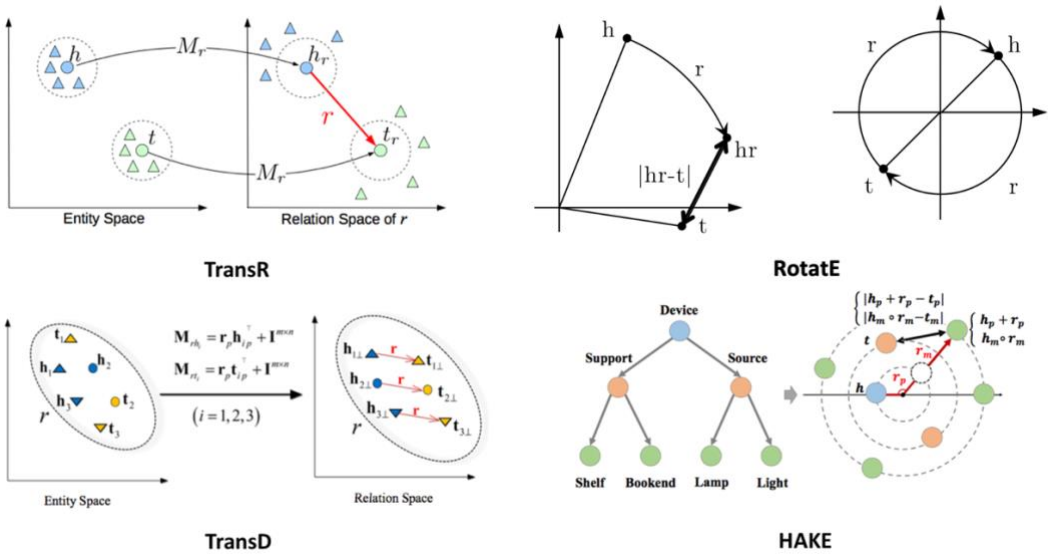


Figure 5: Embedding translations for the different methods. Images adapted from Lin et al. (2015), Ji et al. (2015), Sun et al. (2019b), Zhang et al. (2019b).

All three methods translate embeddings from Entity Space to Relation Space then calculate a distance measure between head and tail entities (**Figure 5**). The methods differ in the way this distance measure (or plausibility score) is calculated (**Figure 6**). Therefore, our experiments modify the KGAT code training on the Amazon Book, Last-FM and Yelp2018 datasets to better understand how these design choices yield better recommendation accuracy.

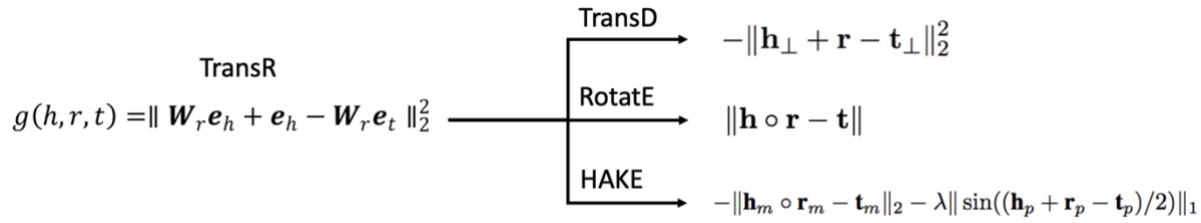


Figure 6: Plausibility score calculations for all methods.

In doing so, we build a model that outperforms our recommendation system baseline and therefore empirically demonstrates the benefit of considering the various expressivities of the KGEs on recommendation accuracy. Furthermore, our research paves the way for future research directions on the design choices when integrating side information to downstream machine learning tasks.

4 Methodology

In this section we outline the methodology of the experiments that were conducted to achieve our aims. Our implementation follows the methodology of the original KGAT implementation as a baseline with adjustments made to the Collaborative Knowledge Graph (CKG) Embedding layer and Attentive Embedding Propagation layers, therefore, we will first introduce the KGAT methodology, then outline the proposed experiments to the CKG Embedding and Attentive Embedding Propagation layers.

4.1 Task Formulation

The Collaborative Knowledge Graph (CKG) as defined in KGAT will be used as input for the model. The CKG is a joint representation of the user-item bipartite graph and the knowledge graph. This unified graph $G = \{(h, r, t) \mid h, t \in \mathcal{E}',\ r \in \mathcal{R}'\}$, where $\mathcal{E}'$ is the set of entities (both users and items) and $\mathcal{R}'$ is the set of all relations (including interaction relations from the user-item bipartite graph). The output for the model is a prediction function that predicts the probability $\hat{y}_{ui}$ that user u would adopt item i.

4.2 KGAT model

The KGAT model framework (**Figure 7**) consists of 3 main components:

1. Embedding layer, one of the layers that will be adapted to various KGE methods.
2. Attentive embedding propagation layer, which further updates the embedding representations with respect to its neighbours.
3. Prediction layer, which aggregates the representations of a user and item from the previous layers to output a predicted matching score.

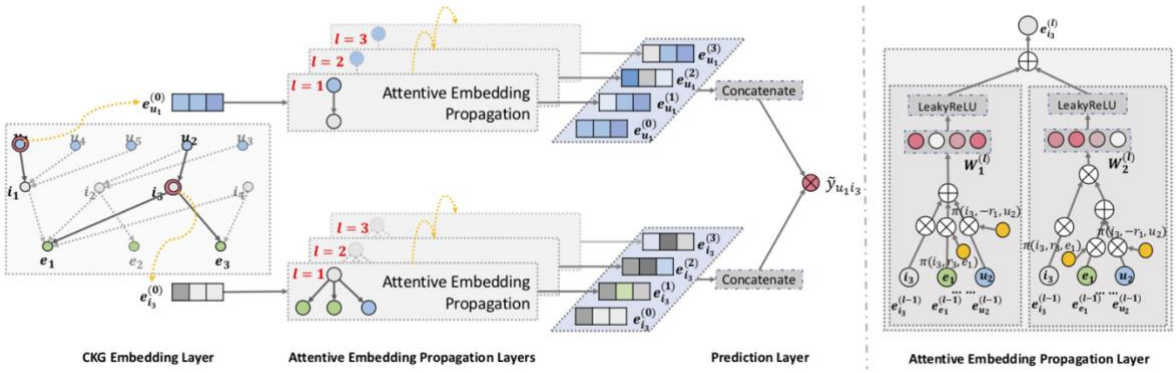


Figure 7: Left subfigure: KGAT model framework. Right subfigure: Attentive propagation layer. Image adapted from Wang et al. (2019).

4.2.1 Embedding Layer

The embedding layer takes input in the form of the collaborative knowledge graph (CKG) and parameterizes the entities and relations as vector representations while considering the underlying graph structure. These vector representations, or embeddings, are learnt via a KGE method. Therefore, to understand the effects of these KGE methods on recommendation performance, we will adapt this layer to employ TransD, RotatE and HAKE KGE methods. To achieve this, we substitute the baseline TransR plausibility score calculation $g(h, r, t)$ with the respective plausibility score calculations of the proposed KGE methods (**Figure 6**). Training of this layer considers the relative order between valid triplets ($\boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t}$) and broken triplets ($\boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t'}$), which are constructed by replacing one entity in a valid triplet randomly. Discrimination between valid and broken triplets are encouraged using a pairwise ranking loss:

$$\mathcal{L}_{KG} = \sum_{(h,r,t,t') \in \mathcal{T}} -\ln \sigma\big(g(h,r,t') - g(h,r,t)\big) \tag{1}$$

Where, $\mathcal{T} = \{(h, r, t, t') \,|(h, r, t) \in G, (h, r, t') \notin G\}$; $\sigma(\cdot)$ is the sigmoid function. However, since RotatE and HAKE use negative sampling loss with self-adversarial training, we modify the loss function accordingly.

4.2.2 Attentive Embedding Propagation Layers

A single layer of the attentive embedding propagation layer consists of three components: information propagation, knowledge-aware attention, and information aggregation. Here we discuss our design choices for each component and how we modified the baseline model to incorporate the plausibility score calculations of the proposed KGE methods.

Information Propagation: This component considers the information propagated between an entity h and its neighbours $\mathcal{N}_h = \{(h, r, t)|(h, r, t) \in G\}$, or ego-network, which denotes the set of triplets where h is the head entity. To characterise the first-order connectivity structure of entity h, a linear combination of h's ego-network is computed:

$$\boldsymbol{e}_{\mathcal{N}_h} = \sum_{(h,r,t) \in \mathcal{N}_h} \pi(h,r,t)\boldsymbol{e}_t \tag{2}$$

Where, $\pi(h,r,t)$ controls the decay factor on each propagation on each edge (h,r,t), indicating how much information is being propagated from t to h conditioned to relation r. $\pi(h,r,t)$ is computed in the knowledge-aware attention component and is the main modification of the attentive embedding propagation layer.

Knowledge-aware Attention: This component computes $\pi(h,r,t)$ via a relational attention mechanism:

$$\pi(h,r,t) = (\boldsymbol{W}_r\boldsymbol{e}_t)^T \tanh\left((\boldsymbol{W}_r\boldsymbol{e}_h + \boldsymbol{e}_r)\right) \tag{3}$$

Where tanh is a non-linear activation function. The attention mechanism suggests which neighbour nodes should be given more attention to capture collaborative signals within the knowledge graph. Equation 3 is modified according to the plausibility score calculations in **Figure 6**.

Information Aggregation: This component was not modified from the original implementation. However, three types of aggregation functions (GCN, GraphSAGE and Bi-Interaction) and multiple stacks of propagation layers are explored in KGAT. In this study, we chose the Bi-Interaction aggregator as it displayed better performance over GCN and GraphSAGE aggregators:

$$f_{Bi-Interaction} = LeakyReLU\left(W_1\left(e_h + e_{N_h}\right)\right) + LeakyReLU\left(W_2\left(e_h \odot e_{N_h}\right)\right) \tag{4}$$

Where, e_h and e_{N_h} are the entity and ego-network representations respectively, $W_1, W_2 \in \mathbb{R}^{d' \times d}$ are trainable weight matrices and $\odot$ denotes the element-wise product. Finally, we will use 3 stacks of the embedding propagation layer since it was shown that 3 layers was sufficient to capture the collaborative signal between higher order relations.

4.2.3 Model Prediction

After the 3 stacks of embedding propagation layers, we get 3 representations for each user, which are then concatenated to obtain a final user (e_u^*) and item (e_i^*) representation:

$$e_u^* = e_u^0 \,\|\, \cdots \,\|\, e_u^3 \tag{5}$$

$$e_i^* = e_i^0 \,\|\, \cdots \,\|\, e_i^3 \tag{6}$$

The probability score is then computed by conducting the inner product of these user and item representations:

$$\hat{y}(u,i) = {e_u^*}^T e_i^* \tag{7}$$

Optimisation in this step uses the Bayesian Personalised Ranking (BPR) loss:

$$\mathcal{L}_{CF} = \sum_{(u,i,j) \in \mathcal{O}} -\ln \sigma\big(\hat{y}(u,i) - \hat{y}(u,j)\big) \tag{8}$$

Where, $\mathcal{O} = \{(u,i,j) \,|(u,i) \in \mathcal{R}'^{+}, (u,j) \in \mathcal{R}'^{-}\}$ denotes the training set, $\mathcal{R}'^{+}$ denotes positive interactions and $\mathcal{R}'^{-}$ denotes negative interactions; $\sigma(\cdot)$ is the sigmoid function.

4.2.4 Optimization

Again, optimization will not be modified from the original implementation. The objective function jointly learns the KGE layer and the prediction layers:

$$\mathcal{L}_{KGAT} = \mathcal{L}_{KG} + \mathcal{L}_{CF} + \lambda \|\ominus\|_2^2 \tag{9}$$

Where, $\ominus = \{E, W_r, \forall l \in \mathcal{R}', W_1^{(l)}, W_2^{(l)}, \forall l \in \{1,2,3\}\}$ is the model parameter set which undergoes L_2 regularization to prevent overfitting.

4.3 Experiments

To examine of the effect of knowledge graph embedding on recommendation accuracy we compared the recommendation accuracy of the modified KGAT algorithms with the original implementation as the baseline. We first describe the datasets, then review the evaluation metrics, finally, we outline the changes to the original KGAT algorithm for each of the experiments.

4.3.1 Datasets

We use the Amazon-book, Last-FM and Yelp2018 datasets for this analysis. The datasets were acquired from the original authors publicly available GitHub repository so that our results are comparable. Due to hardware and compute constraints, we decrease the size of the original datasets. Similar to the original implementation, we randomly select 80% of the interaction history for each user to build the training set. The remaining interaction history for each user will be used as the test set. The dataset statistics are found in **Table 1**.

		Amazon-book	Last-FM	Yelp2018
User-Item Interaction	#Users	5,000	5,000	5,000
	#Items	20,000	20,000	45,537
	#Interactions	85,749	313,483	148,478
Knowledge Graph	#Entities	30,000	30,000	45,999
	#Relations	39	9	42
	#Triplets	733,773	1,516	727,066

Table 1: Dataset Statistics

4.3.2 Evaluation Metrics

To evaluate our models, we used recall@K and ndcg@K to evaluate the accuracy of the top-K recommendations and preference ranking. We set K= 20 and report the average metrics for all users in the test set.

4.3.3 Baseline (TransR)

To demonstrate the effectiveness of considering (1) the diversity of entities and relations simultaneously, (2) symmetry/asymmetry, inversion, and composition simultaneously and (3) semantic hierarchies, we use TransR as the baseline KGE method using the score function:

$$g(h,r,t) = -\|\boldsymbol{M}_r\boldsymbol{h} + \boldsymbol{r} - \boldsymbol{M}_r\boldsymbol{t}\|_2^2 \tag{10}$$

Where, $\boldsymbol{M}_r$ is the relation specific projection matrix and $\boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t}$ are the head, relation, and tail embeddings. This score function will be modified accordingly.

4.3.4 TransD

To compare the TransD KGE to the baseline, we modify the plausibility score calculation $g(h,r,t)$ in the embedding and attentive propagation layers to reflect that of the TransD calculation:

$$g(h,r,t) = -\|\boldsymbol{h}_\perp + \boldsymbol{r} + \boldsymbol{t}_\perp\|_2^2 \tag{11}$$

Where, $\boldsymbol{h}_\perp$ and $\boldsymbol{t}_\perp$are embedding projections from entity space to relation space. Therefore, this implementation requires us to include additional mapping matrices $\boldsymbol{M}_{rh}$ and $\boldsymbol{M}_{rt}$ which are comprised of projection vectors $\boldsymbol{h}_p, \boldsymbol{r}_p, \boldsymbol{t}_p$ and an identity matrix $\boldsymbol{I}^{m\times n}$. Giving:

$$\boldsymbol{M}_{rh} = \boldsymbol{r}_p\boldsymbol{h}_p^\intercal + \boldsymbol{I}^{m\times n} \tag{12}$$

$$\boldsymbol{M}_{rt} = \boldsymbol{r}_p\boldsymbol{t}_p^\intercal + \boldsymbol{I}^{m\times n} \tag{13}$$

$$\boldsymbol{h}_\perp = \boldsymbol{M}_{rh}\boldsymbol{h} \tag{14}$$

$$\boldsymbol{t}_\perp = \boldsymbol{M}_{rt}\boldsymbol{t} \tag{15}$$

The training objective for TransD differs from that of TransR, therefore, the KGE loss function will need to be adjusted accordingly. A margin-based ranking loss is used:

$$\mathcal{L}_{KG} = \sum_{(h,r,t,t')\in\mathcal{T}} [\gamma + g(h,r,t') - g(h,r,t)]_+ \tag{16}$$

Where, $\mathcal{T} = \{(h,r,t,t') \,|(h,r,t) \in G, (h,r,t') \notin G\}$, $[x]_+ = \max(0,x)$, and γ is the margin separating positive and negative triplets.

4.3.5 RotatE

To compare the RotatE KGE to the baseline, we modify the plausibility score calculation in the embedding layer to reflect that of the RotatE calculation:

$$g(h,r,t) = -\|\boldsymbol{h} \circ \boldsymbol{r} - \boldsymbol{t}\| \tag{17}$$

RotatE is motivated by Euler's identity $e^{i\theta} = cos\theta + isin\theta$, which indicates that a unitary complex number can be regarded as a rotation in complex space. As such, the model maps the entities and relations to the complex vector space and defines each relation as a rotation from head entity to tail entity. i.e. given a triplet (h,r,t), we expect that:

$$\boldsymbol{t} = \boldsymbol{h} \circ \boldsymbol{r} \tag{18}$$

Where, $\boldsymbol{h},\boldsymbol{t},\boldsymbol{r} \in \mathbb{C}^k$, $|r_i| = 1$ and $\circ$ is the Hadamard product. Constraining the modulus, r_i, to 1 we get the form $e^{i\theta_{r,i}}$, which corresponds to a counterclockwise rotation by $\theta_{r,i}$ radians about the origin of the complex plane.

The training objective for RotatE differs from that of TransR, therefore, the KGE loss function will need to be adjusted accordingly. A negative sampling loss function with self-adversarial training is used:

$$\mathcal{L}_{KG} = -\log\sigma\big(\gamma - g(h,r,t)\big) - \sum_{i=1}^{n} p(h_i',r,t_i')\log\sigma(g(h,r,t) - \gamma) \tag{19}$$

Where, γ is a fixed margin, $\sigma(\cdot)$ is the sigmoid function and (h_i',r,t_i') is the ith negative triple. Moreover, **Equation 20** is the probability distribution of sampling negative triples, where α is the temperature of sampling.

$$p(h_j',r,t_j'|\,\{(h_i,r_i,t_i)\}) = \frac{\exp(\alpha g\big(h_j',r,t_j'\big))}{\sum_i \exp(\alpha g(h_i',r,t_i'))} \tag{20}$$

4.3.6 HAKE

To compare the HAKE KGE to the baseline, we modify the plausibility score calculation in the embedding layer to reflect that of the HAKE calculation:

$$g(h,r,t) = -\|\boldsymbol{h}_m \circ \boldsymbol{r}_m - \boldsymbol{t}_m\|_2 - \lambda\left\|\sin\left(\frac{\boldsymbol{h}_p + \boldsymbol{r}_p - \boldsymbol{t}_p}{2}\right)\right\|_1 \tag{21}$$

HAKE attempts to model semantic hierarchies in knowledge graphs by introducing two categories of entities: entities at different levels of the hierarchy and entities at the same level of the hierarchy. To achieve this, the method maps entities onto a polar coordinate where the radial and angular coordinate system correspond to the two categories of entities: the modulus part and the phase part. It is well suited to the KGAT architecture because KGAT explicitly models the high-order connectivity in the knowledge graph.

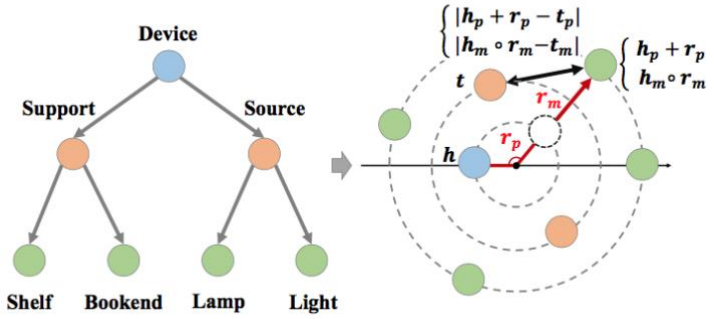


Figure 8: HAKE polar co-ordinate system. Image adapted from Zhang et al. (2019b).

Modulus Part

This part aims to model entities at different levels of the hierarchy. It is inspired by the fact that the moduli can reflect the depth of a tree hierarchy (**Figure 8**). It regards each entity ($\boldsymbol{h}, \boldsymbol{t}$) as a modulus, denoting a level of the hierarchy, and considers each relation ($\boldsymbol{r}$) as a scaling transformation between two moduli. Therefore, the moduli distance function is defined as:

$$d_{r,m}(\boldsymbol{h}_m, \boldsymbol{t}_m) = \|\boldsymbol{h}_m \circ \boldsymbol{r}_m - \boldsymbol{t}_m\|_2 \tag{22}$$

Where, $\boldsymbol{h}_m, \boldsymbol{t}_m$ are entity embeddings in the modulus part, $\boldsymbol{r}_m$ is the relation embedding in the modulus part and $\circ$ is the Hadamard product.

Phase Part

This part aims to model entities at the same level of the hierarchy. It is inspired by the fact points on the same circle or have the same moduli can have different phases. It regards each entity ($\boldsymbol{h}, \boldsymbol{t}$) as a phase and considers each relation ($\boldsymbol{r}$) as a phase transformation. The phase transformation can be formulated as follows:

$$(\boldsymbol{h}_p, \boldsymbol{t}_p) mod\ 2\pi = \boldsymbol{t}_p \tag{23}$$

Where $\boldsymbol{h}_p, \boldsymbol{r}_p, \boldsymbol{t}_p \in [0, 2\pi)^k$. Therefore, the phase distance function is defined as:

$$d_{r,p}(\boldsymbol{h}_p, \boldsymbol{t}_p) = \left\| \sin\left(\frac{\boldsymbol{h}_p + \boldsymbol{r}_p - \boldsymbol{t}_p}{2}\right) \right\|_1 \tag{24}$$

Combining these modulus and phase parts we obtain **Equation 21**. Where, $\lambda \in \mathbb{R}$ is a learnable parameter.

The training objective for HAKE differs from that of TransR, therefore, the KGE loss function will need to be adjusted accordingly. The negative sampling loss function with self-adversarial training used in RotatE (**Equation 19**) is used.

KGE Method	Score Function	Parameters
TransR (Baseline; Lin *et al.*, 2015)	$-\|\|\boldsymbol{M}_r\boldsymbol{h}+\boldsymbol{r}-\boldsymbol{M}_r\boldsymbol{t}\|\|_2^2$	$\boldsymbol{h},\boldsymbol{t}\in\mathbb{R}^d,\boldsymbol{r}\in\mathbb{R}^k,\boldsymbol{M}_r\in\mathbb{R}^{k\times d}$
TransD (Ji *et al.*, 2015)	$-\|\|\boldsymbol{h}_\perp+\boldsymbol{r}-\boldsymbol{t}_\perp\|\|_2^2$	$\boldsymbol{h}_\perp,\boldsymbol{r},\boldsymbol{t}_\perp\in\mathbb{R}^m$
RotatE (Sun *et al.*, 2019b)	$-\|\|\boldsymbol{h}\circ\boldsymbol{r}-\boldsymbol{t}\|\|$	$\boldsymbol{h},\boldsymbol{t},\boldsymbol{r}\in\mathbb{C}^k,\|r_i\|=1$
HAKE (Zhang *et al.*, 2019c)	$-\|\|\boldsymbol{h}_m\circ\boldsymbol{r}_m-\boldsymbol{t}_m\|\|_2-\lambda\left\|\sin\left((\boldsymbol{h}_p\circ\boldsymbol{r}_p-\boldsymbol{t}_p)/2\right)\right\|_1$	$\boldsymbol{h}_m,\boldsymbol{t}_m\in\mathbb{R}^k,\boldsymbol{r}_m\in\mathbb{R}_+^k$ $\boldsymbol{h}_p,\boldsymbol{r}_p,\boldsymbol{t}_p\in[0,2\pi)^k,\lambda\in\mathbb{R}$

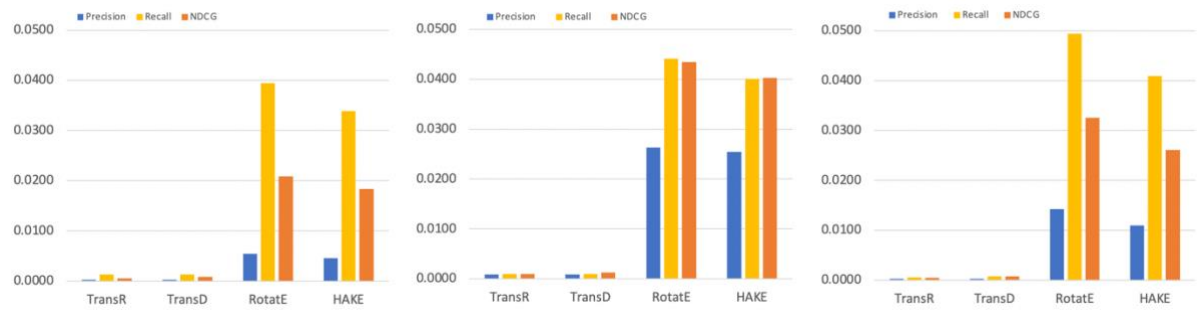
Table 2: Details of knowledge graph embedding models, where ∘ denotes the Hadamard product and σ denotes the sigmoid function.

4.3.7 Parameter Settings

We implement all models in PyTorch in combination with the graph-based module, Deep Graph Library, for the knowledge graph. The embedding size is fixed to 64 for the baseline and TransD models. We set the embedding size to 128 for RotatE and HAKE to account for the phase and modulus parts of the embeddings. We later discuss the effect of increasing this parameter. We optimize all models with Adam optimizer, where the collaborative filtering batch size and knowledge graph batch size is fixed at 1024 and 2048 respectively. Learning rate is set at 0.0001, the coefficient of L_2 normalization at 10^{-5} and dropout rate at 0.1. Similar to the original KGAT implementation, we model third-order connectivity by setting the depth of KGAT L as three with hidden dimensions 64, 32, 16 for all models. However, we set the RotatE and HAKE models' three hidden dimensions to 128, 64, 32 to account for the modulus and phase parts of the embeddings. Additionally, we set the modulus weight, phase weight, gamma, and adversarial temperature to 1.0, 0.5, 12.0 and 1.0 respectively for the RotatE and HAKE model. Early stopping strategy is performed where training is stopped if recall@20 on the validation set does not increase for 10 successive epochs.

5 Experimental Evaluation

The results of our experiment are presented in this section. We compare the recall and ndcg of the KGAT baseline model (TransR) with the modified knowledge graph embeddings methods to determine the effectiveness of the modified models on recommendation accuracy.



(a) Amazon-book ***(b) Last-FM*** ***(c) Yelp2018***

Figure 9: Precision, Recall and NDCG on three datasets

Improved Expressivity vs Recommendation Accuracy. We compare the precision, recall and ndcg of the baseline and modified models in **Figure 9** and **Table 3**. The RotatE and HAKE model outperforms the baseline in all three metrics across all three datasets. In contrast, the TransD model produces comparable results to the baseline. It should be noted that some of the final

accuracy results were reported before training could converge to our early stopping criteria. This was due to timing constraints on the publicly available Google Colab GPU instance. Therefore, these results may have produced a higher recall and ndcg if these models were trained until convergence.

KGE	Amazon Book		Last-FM		Yelp2018	
	recall	ndcg	recall	ndcg	recall	ndcg
TransR	0.0013	0.0006	0.0010	0.0010	0.0006	0.0005
TransD	0.0013	0.0008	0.0010	0.0012	0.0007	0.0007
RotatE	**0.0394**	**0.0209**	**0.0441***	**0.0434***	**0.0494***	**0.0325***
HAKE	0.0339*	0.0184*	0.0401*	0.0403*	0.0409*	0.0261*

Table 3: Overall performance comparison. *Training was stopped early due to compute time restraints

Average Training Time Per Epoch. We report on the average training time per epoch across all models and datasets in **Table 4**. We see that despite RotatE and HAKE having a larger hidden dimension to accommodate for the modulus and phase parts of the embeddings, the average training time per epoch is comparable to the baseline and therefore only adds a relatively small amount of training time for the given performance improvement. However, as touched upon earlier, both RotatE and HAKE took considerably longer to train to convergence compared to the baseline.

KGE	Amazon Book	Last-FM	Yelp2018
	Av. Time/Epoch (s)	Av. Time/Epoch (s)	Av. Time/Epoch (s)
TransR	150.7	60.1	132.4
TransD	156.8	59.1	130.7
RotatE	155.9	62.8	142.5
HAKE	163.3	60.7	135.4

Table 4: Average Training Time Per Epoch comparison.

6 Discussion

In this section we discuss in detail the main findings, report on further experimentation that was conducted on the model's architecture and parameters, and outline some of the challenges.

Our results show that recommendation accuracy can be significantly improved when we increase the expressivity of the learned embeddings. We find that RotatE and HAKE outperform the baseline thus providing more accurate recommendations. These methods project the embeddings onto complex and polar planes respectively and potentially provide an extra dimension with which to encode more useful relationship information and provide a richer representation of the underlying knowledge graph. In addition, the attentive propagation layers in the KGAT model that utilize attentive message passing graph neural network style architectures, synergize with the HAKE knowledge graph embedding model due to the semantic hierarchies that are inherently encoded in the embeddings. This trend was evident across all datasets; however, it should be noted that while RotatE performs slightly better than HAKE, the HAKE model did not complete training due to compute time restraints

and therefore could potentially result in a greater recall/ndcg if the model was able to converge. Conversely, the TransD model performed comparably to the baseline. This may be explained by the fact that TransD is a relatively older knowledge graph embedding method, which may not increase the expressivity of the learned embeddings as effectively as RotatE and HAKE compared to the baseline. Additionally, TransD has shown to underperform in link prediction tasks when compared to RotatE and HAKE (Rossi *et al*., 2021; Dai *et al*., 2020).

To further understand why the recommendation performance was improved by the various Knowledge Graph Embedding methods, we performed some further experimentation by adjusting the model architecture or parameters.

Loss Functions. Following the original implementations of each knowledge graph embedding method, a pairwise loss function was used for the baseline model, whereas a margin-based and negative sampling with self-adversarial training loss was used with the TransD and RotatE/HAKE models respectively. We experimented on the effect of the loss function on the accuracy of the models. We find that substituting the pairwise loss function with either the margin-based or self-adversarial loss functions did not improve the recall and ndcg and we conclude that the loss function was not the reason for the improved recommendation accuracy.

Hidden Dimension Size. During our implementation, the hidden dimension sizes were increased from [64,32,16] to [128,64,32] for the RotatE and HAKE models to incorporate the two axes in the complex and polar coordinates respectively. This resulted in a larger embedding size and may have contributed to the overall increase in recommendation accuracy. As such, we experimented on increasing the hidden dimension size for the TransR baseline model ([128,64,32]) and decreasing the hidden dimension size for the RotatE and HAKE models ([64,32,16]). Our findings show that increasing the hidden dimension size for the baseline did not improve recommendation accuracy, however, decreasing the hidden dimension size for the RotatE and HAKE models reduced the performance of the models by half. However, the recommendation accuracy still outperformed the baseline model.

Challenges. When comparing our results to the existing benchmarks, we find that the overall recall and ndcg across all models and datasets display a lower performance on these metrics. Ideally, we would have liked to see comparable results, however, due to the computational and training time constraints, this was unlikely. The main limiting factor was the amount of RAM made available on the publicly available Google Colab instance. Storing the knowledge graph using Deep Graph Library required a large amount of working memory and therefore, we had to decrease the dataset size across all datasets by 80%. This may account for the overall performance degradation we see when comparing our results to the existing benchmarks. Additionally, the constraints on training time reduced the scope of the experimentation and in some instances, did not allow our models to train until convergence. Concepts such as federated learning (Konečný *et al*., 2016), where the datasets are distributed across multiple devices, and knowledge graph distillation (Deng *et al*., 2021) can be used in the future to potentially overcome these challenges.

7 Conclusion and Future Work

In this paper, we perform a comparative study of knowledge graph embedding methods to understand if considering the diversity of entities and relations simultaneously, symmetry/asymmetry, inversion, and composition simultaneously, and semantic hierarchies improves recommendation accuracy. Through improving the expressivity of knowledge graph embedding methods we conclude that these design considerations can improve downstream tasks such as recommendation. However, the increase in training time needs to be considered and solutions such as federated learning and knowledge graph distillation can be utilized to improve the scalability of the models. Finally, since we only explore the geometric family of knowledge graph embeddings, future research directions may consider matrix factorization and deep learning-based scoring functions as a potential source of recommendation improvement or design choice when integrating side information to downstream machine learning tasks.

References

Abujabal, A., Yahya, M., Riedewald, M., & Weikum, G. (2017). Automated Template Generation for Question Answering over Knowledge Graphs. In Proceedings of the 26th International Conference on World Wide Web, 1191–1200. https://doi.org/10.1145/3038912.3052583

Bell, R., & Koren, Y. (2007). Scalable Collaborative Filtering with Jointly Derived Neighborhood Interpolation Weights. Seventh IEEE International Conference on Data Mining (ICDM 2007), 43–52. https://doi.org/10.1109/ICDM.2007.90

Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., & Yakhnenko, O. (2013). Translating Embeddings for Modeling Multi-relational Data. 1–9.

Cao, Y., Wang, X., He, X., Hu, Z., & Chua, T. (2019). Unifying Knowledge Graph Learning and Recommendation: Towards a Better Understanding of User Preferences. The World Wide Web Conference, 151–161. https://doi.org/10.1145/3308558.3313705

Catherine, R., & Cohen, W. (2016). Personalized Recommendations using Knowledge Graphs: A Probabilistic Logic Programming Approach. Proceedings of the 10th ACM Conference on Recommender Systems, 325–332. https://doi.org/10.1145/2959100.2959131

Choi, S., Ko, S., & Han, Y. (2012). A movie recommendation algorithm based on genre correlations. Expert Systems with Applications, 39(9), 8079–8085. https://doi.org/10.1016/j.eswa.2012.01.132

Covington, P., Adams, J., & Sargin, E. (2016). Deep Neural Networks for YouTube Recommendations. Proceedings of the 10th ACM Conference on Recommender Systems, 191–198. https://doi.org/10.1145/2959100.2959190

Dai, Wang, S., Xiong, N. N., & Guo, W. (2020). A Survey on Knowledge Graph Embedding: Approaches, Applications and Benchmarks. Electronics (Basel), 9(5), 750–. https://doi.org/10.3390/electronics9050750

Deng, & Zhang, Z. (2021). Graph-Free Knowledge Distillation for Graph Neural Networks.

Dettmers, T., Minervini, P., Stenetorp, P., & Riedel, S. (2017). Convolutional 2D Knowledge Graph Embeddings. https://arxiv.org/abs/1707.01476

Edmunds, A., & Morris, A. (2000). The problem of information overload in business organisations: a review of the literature. International Journal of Information Management, 20(1), 17–28. https://doi.org/10.1016/S0268-4012(99)00051-1

Guo, G., Zhang, J., & Yorke-Smith, N. (2015). TrustSVD: collaborative filtering with both the explicit and implicit influence of user trust and of item ratings. In Proceedings of the 29th

AAAI Conference on Artificial Intelligence (AAAI'15). AAAI Press, 123–129. https://dl.acm.org/doi/10.5555/2887007.2887025

Guo, L., Sun, Z., & Hu, W. (2019). Learning to Exploit Long-term Relational Dependencies in Knowledge Graphs. https://arxiv.org/abs/1905.04914

Ji, G., He, S., Xu, L., Liu, K., & Zhao, J. (2015). Knowledge Graph Embedding via Dynamic Mapping Matrix. 687-696. 10.3115/v1/P15-1067. https://www.aclweb.org/anthology/P15-1067/

Karlgren, J. (1990) An Algebra for Recommendations. The Systems Development and Artificial Intelligence Laboratory. Working Paper No 179. https://jussikarlgren.files.wordpress.com/1990/09/algebrawp.pdf

Koenigstein, N., Dror, G., & Koren, Y. (2011). Yahoo! music recommendations: modeling music ratings with temporal dynamics and item taxonomy. Proceedings of the Fifth ACM Conference on Recommender Systems, 165–172. https://doi.org/10.1145/2043932.2043964

Konečný, McMahan, H. B., Ramage, D., & Richtárik, P. (2016). Federated Optimization: Distributed Machine Learning for On-Device Intelligence.

Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix Factorization Techniques for Recommender Systems. Computer (Long Beach, Calif.), 42(8), 30–37. https://doi.org/10.1109/MC.2009.263

Lin, Y., Liu, Z., Sun, M., Liu, Yang., & Zhu, X. (2015). Learning entity and relation embeddings for knowledge graph completion. In Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI'15). AAAI Press, 2181–2187. https://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/viewFile/9571/9523

Liu, H., Wu, Y., & Yang, Y. (2017). Analogical Inference for Multi-Relational Embeddings. https://arxiv.org/abs/1705.02426

Nguyen, D., Sirts, K., Qu, L., & Johnson, M. (2016). STransE: a novel embedding model of entities and relationships in knowledge bases. https://doi.org/10.18653/v1/N16-1054

Nguyen, D., Vu, T., Nguyen, T., Nguyen, D., & Phung, D. (2018). A Capsule Network-based Embedding Model for Knowledge Graph Completion and Search Personalization. https://arxiv.org/abs/1808.04122

Niu, W., Caverlee, J., & Lu, H. (2018). Neural Personalized Ranking for Image Recommendation. Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, 2018-, 423–431. https://doi.org/10.1145/3159652.3159728

Rendle, S. (2010). Factorization Machines. 2010 IEEE International Conference on Data Mining, 995–1000. https://doi.org/10.1109/ICDM.2010.127

Ricci, F., Rokach, L., & Shapira, B. (2015). Recommender Systems: Introduction and Challenges. In Recommender Systems Handbook (pp. 1–34). Springer US. https://doi.org/10.1007/978-1-4899-7637-6_1

Rossi, A., Barbosa, D., Firmani, D., Matinata, A., & Merialdo, P. (2021). Knowledge Graph Embedding for Link Prediction: A Comparative Analysis. ACM Transactions on Knowledge Discovery from Data, 15(2), 1–49. https://doi.org/10.1145/3424672

Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. Proceedings of the 10th International Conference on World Wide Web, 285–295. https://doi.org/10.1145/371920.372071

Seo, S., Huang, J., Yang, H., & Liu, Y. (2017). Interpretable Convolutional Neural Networks with Dual Local and Global Attention for Review Rating Prediction. In Proceedings of the 11th ACM Conference on Recommender Systems, 297–305. https://doi.org/10.1145/3109859.3109890

Sun, Z., Deng, Z.-H., Nie, J.-Y., & Tang, J. (2019a). RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space.

Sun, Z., Guo, Q., Yang, J., Fang, H., Guo, G., Zhang, J., & Burke, R. (2019b). Research commentary on recommendations with side information: A survey and research directions. Electronic Commerce Research and Applications, 37, 100879–. https://doi.org/10.1016/j.elerap.2019.100879

Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., & Bouchard, G. (2016). Complex Embeddings for Simple Link Prediction. https://arxiv.org/pdf/1606.06357.pdf

Vashishth, S., Sanyal, S., Nitin, V., Agrawal, N., & Talukdar, P. (2019). InteractE: Improving Convolution-based Knowledge Graph Embeddings by Increasing Feature Interactions. https://arxiv.org/abs/1911.00219

Wang, G., Zhang, W., Wang, R., Zhou, Y., Chen, X., Zhang, W., Zhu, H., & Chen, H. (2018). Label-free distant supervision for relation extraction via knowledge graph embedding. In Proceedings of the 2018 conference on empirical methods in natural language processing, 2246-2255.

Wang, Q., Mao, Z., Wang, B., & Guo, L. (2017). Knowledge Graph Embedding: A Survey of Approaches and Applications. IEEE Transactions on Knowledge and Data Engineering, 29(12), 2724–2743. https://doi.org/10.1109/TKDE.2017.2754499

Wang, X., He, X., Cao, Y., Liu, M., & Chua, T. (2019). KGAT: Knowledge Graph Attention Network for Recommendation. Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 950–958. https://doi.org/10.1145/3292500.3330989

Wang, Z., Zhang, J., Feng, J., & Chen, Z. (2014). Knowledge Graph Embedding by Translating on Hyperplanes. In Proceedings of AAAI, 1112- 1119. https://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/view/8531

Wu, S., Sun, F., Zhang, W., & Cui, B. (2020). Graph Neural Networks in Recommender Systems: A Survey. https://arxiv.org/abs/2011.02260

Yang, B., Yih, W., He, X., Gao, J., & Deng, L. (2014). Embedding Entities and Relations for Learning and Inference in Knowledge Bases. https://arxiv.org/abs/1412.6575

Yu, X., Ren, X., Sun, Y., Sturt, B., Khandelwal, U., Gu, Q., Norick, B., & Han, J. (2013). Recommendation in heterogeneous information networks with implicit user feedback. In Proceedings of the 7th ACM conference on Recommender systems, 347–350. https://doi.org/10.1145/2507157.2507230

Zhang, F., Yuan, N., Lian, D., Xie, X., & Ma, W. (2016). Collaborative Knowledge Base Embedding for Recommender Systems. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 353–362. https://doi.org/10.1145/2939672.2939673

Zhang, S., Yao, L., Sun, A., & Tay, Y. (2019a). Deep Learning Based Recommender System: A Survey and New Perspectives. ACM Computing Surveys, 52(1), 1–38. https://doi.org/10.1145/3285029

Zhang, W., Paudel, B., Zhang, W., Bernstein, A., & Chen, H. (2019b). Interaction Embeddings for Prediction and Explanation in Knowledge Graphs. Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, 96–104. https://doi.org/10.1145/3289600.3291014

Zhang, Z., Cai, J., Zhang, Y., & Wang, J. (2019c). Learning Hierarchy-Aware Knowledge Graph Embeddings for Link Prediction. In Proceedings of the 2020 AAAI Conference on Artificial Intelligence. https://arxiv.org/abs/1911.09419