Entity-Agnostic Representation Learning for Parameter-Efficient Knowledge Graph Embedding

Mingyang Chen[1*], Wen Zhang[2*], Zhen Yao[2], Yushan Zhu[1], Yang Gao[4], Jeff Z. Pan[5], Huajun Chen[1,3,6†]

[1]College of Computer Science and Technology, Zhejiang University
[2]School of Software Technology, Zhejiang University
[3]Donghai Laboratory
[4]Huawei Technologies Co., Ltd.
[5]School of Informatics, The University of Edinburgh
[6]Alibaba-Zhejiang University Joint Institute of Frontier Technologies
{mingyangchen, zhang.wen, yz0204, yushanzhu, huajunsir}@zju.edu.cn,
frank.gaoyang@huawei.com, j.z.pan@ed.ac.uk

Abstract

We propose an entity-agnostic representation learning method for handling the problem of inefficient parameter storage costs brought by embedding knowledge graphs. Conventional knowledge graph embedding methods map elements in a knowledge graph, including entities and relations, into continuous vector spaces by assigning them one or multiple specific embeddings (i.e., vector representations). Thus the number of embedding parameters increases linearly as the growth of knowledge graphs. In our proposed model, Entity-Agnostic Representation Learning (EARL), we only learn the embeddings for a small set of entities and refer to them as reserved entities. To obtain the embeddings for the full set of entities, we encode their distinguishable information from their connected relations, k-nearest reserved entities, and multi-hop neighbors. We learn universal and entity-agnostic encoders for transforming distinguishable information into entity embeddings. This approach allows our proposed EARL to have a static, efficient, and lower parameter count than conventional knowledge graph embedding methods. Experimental results show that EARL uses fewer parameters and performs better on link prediction tasks than baselines, reflecting its parameter efficiency.

1 Introduction

Recently, many knowledge graphs (KGs) (Pan et al. 2017), including Freebase (Bollacker et al. 2008), NELL (Carlson et al. 2010), Wikidata (Vrandečić and Krötzsch 2014), and YAGO (Tanon, Weikum, and Suchanek 2020) have been used as the knowledge resource for a myriad of applications in the field of natural language processing (Xiong et al. 2020; Yu et al. 2022), as well as in the study of computer vision (Huang et al. 2020; Chen et al. 2021b).

Typically, knowledge graphs contain a large number of factual triples in the form of *(head entity, relation, tail entity)*, or *(h, r, t)* for short. A triple reflects a specific connection (i.e., relation) between two entities / concepts. However, since the incompleteness of KGs (Chen et al. 2022c), a fundamental problem is knowledge graph completion (Bordes et al. 2013; Sun et al. 2019; Wiharja et al. 2020). Many knowledge graph embedding (KGE) methods (Ji et al. 2022) have been proposed and are theoretically and empirically shown to be effective forsolving the problem of knowledge graph completion. Conventionally, KGE methods map entities and relations from a KG into continuous vector spaces and predict missing links based on the computations of vector representations to complete knowledge graphs.

In KGE methods, entities and relations are often mapped into vectors with a specific dimension. For example, TransE (Bordes et al. 2013) maps both entities and relations to the same d-dimensional vector space, namely $\mathbb{R}^d$, and the total embedding matrix is in $\mathbb{R}^{(|\mathcal{E}|+|\mathcal{R}|)\times d}$, where $|\mathcal{E}|$ and $|\mathcal{R}|$ are number of entities and relations, respectively. In practice, since $|\mathcal{E}|$ is much larger than $|\mathcal{R}|$, the number of embedding parameters scales linearly to the number of entities.

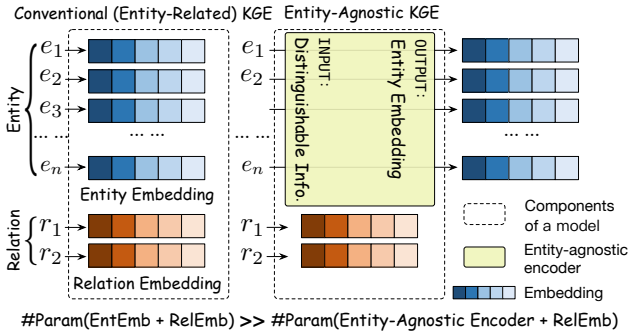

Figure 1: A conventional KGE model (left) learns a specific embedding for each entity, and the model's components (i.e., entity embeddings) are related to entities. An entity-agnostic KGE model (right) learns an encoding approach for transforming entities' distinguishable information to their embeddings, avoiding maintaining a large embedding matrix.

*These authors contributed equally.
†Corresponding Author.

As shown in Figure 1, we call conventional KGE methods *entity-related KGE*, meaning that the components of a KGE model (i.e., entity embeddings) are related to entities.

Hence, the storage space costs for maintaining embeddings of entity-related KGE methods can be huge; e.g., 500-dimensional RotatE (Sun et al. 2019) maintains 123 million parameters for YAGO3-10 (Mahdisoltani, Biega, and Suchanek 2015). Such inefficient linearly scaling space costs for entity-related KGE methods bring several challenges to real-world KG applications. For example, many deep learning models, including KGE models, are expected to be applied on edge devices (Howard et al. 2017), and colossal parameter space costs may weaken the feasibility. Furthermore, some studies explore adapting federated learning (McMahan et al. 2017) to KGs and training KGE models decentralized (Peng et al. 2021; Chen et al. 2021a), and the number of parameters significantly increases communication costs in the federated learning scenario. To this end, we argue that *an entity-agnostic KGE method with a stable and relatively low parameter count, as well as independent of the number of entities*, is essential for solving the above issues and achieving efficient knowledge graph embedding.

In this paper, we propose a novel knowledge graph embedding method named Entity-Agnostic Representation Learning, **EARL**, in the sense that the components of EARL are *not* mapped to entities, enabling the number of model parameters to not linearly scale up when the number of entities increases, as shown in Figure 1. Specifically, instead of learning a specific embedding for each entity as traditional KGE methods, we encode *distinguishable information* of entities to represent them, and the encoding process is universal and *entity-agnostic*. First, in EARL, we only learn embeddings for a small set of entities and refer to them as *reserved entities*. Then, we design the following three kinds of distinguishable information for encoding the embeddings for the full set of entities. 1) *ConRel*: connected relation information can make an entity distinguishable since the semantics of a relation's head or tail entities is often distinct and stable; 2) *kNResEnt*: we also use the status of connected relation to retrieve k-nearest reserved entity information for an entity to improve the distinguishability; 3) *MulHop*: we incorporate multi-hop neighbor information that different entities often have various ones. For the model design, we propose *relational features* for entities to reflect the status of their connected relations for encoding ConRel and retrieving kNResEnt. Based on the above ConRel and kNResEnt encoding, a GNN is used to consider MulHop information and finally output embeddings for entities.

·We conduct an extensive empirical evaluation to show the effectiveness of our proposed EARL. We train EARL on various KG benchmarks with different characteristics, and the results illustrate that we use fewer parameters and obtain better performance than baselines. The contributions of our work are summarised as follows:

- We point out the problem of entity-related KGE methods and emphasize the importance of exploring entity-agnostic representation learning for KGs.
- We propose a novel KGE method, EARL, which uses an entity-agnostic encoding process to encode entity embeddings based on their distinguishable information.
- We conduct comprehensive experiments and show that our model is more parameter-efficient than baselines and achieves competitive performance.

2 Related Work

2.1 Knowledge Graph Embedding

For applying KGs to downstream tasks, including question answering (Yasunaga et al. 2021; Xu et al. 2022; Hu et al. 2022), search (Pan, Taylor, and Thomas 2009), recommendation (Zhang et al. 2016), and some other in-KG tasks like link prediction (Bordes et al. 2013; Zhang et al. 2019), lots of studies are devoted to designing methods for mapping entities and relations of a KG into continuous vector spaces and remaining the inherent semantics in the KG. From the view of designing KGE models, they can be divided into various types (Wang et al. 2017; Zhang et al. 2022; Ji et al. 2022).

Conventional KGE methods are mainly categorized into translational distance models and semantic matching models. TransE (Bordes et al. 2013) is a classic translational distance model, which assumes that the relation is a translation vector from the head entity to the tail entity for a true triple. RotatE (Sun et al. 2019) defines the relation as a rotation between entities in the complex vector space and can capture various relation patterns. For semantic matching models, DistMult (Yang et al. 2015) calculates the score for a triple by capturing the interactions between entity embeddings. ComplEx (Trouillon et al. 2016) extend DistMult to map embeddings to the complex vector space for handling asymmetric relations.

Adapting graph neural networks (GNNs) to embed knowledge graphs has recently gained massive attention (Baek, Lee, and Hwang 2020; Chen et al. 2022b; Geng et al. 2022). Typically, R-GCN (Schlichtkrull et al. 2018) is developed for multi-relational data and uses different transformation weights for various relations. CompGCN (Vashishth et al. 2020) leverages entity-relation composition operations from score functions of other conventional KGE methods (e.g. TransE) for message passing.

However, these methods do not consider the efficiency of parameters, or data compression (Pan et al. 2014; Zhu et al. 2018) and simplification (Wang et al. 2014a) for KGs.

2.2 Parameter-Efficient Models

With the increase of existing deep learning model sizes, reducing model parameters and making them more efficient has attracted much research, including network pruning (Molchanov et al. 2017), quantification (Lin, Talathi, and Annapureddy 2016; Sachan 2020), parameter sharing (Dehghani et al. 2019; Lan et al. 2020), and knowledge distillation (Hinton, Vinyals, and Dean 2015).

For knowledge graph embedding, methods based on quantization and knowledge distillation are studied more. TS-CL (Sachan 2020) proposes a method based on quantization technology to reduce the size of KGEs by representing entities as vectors of discrete codes. LightKG (Wang et al. 2021a) is a lightweight end-to-end KGE framework

based on quantization, which contains a residual module to induce diversity among codebooks and proposes a novel dynamic negative sampling method based on quantization to further improve the performance of KGE. MulDE (Wang et al. 2021b) applies the knowledge distillation technology to transfer the knowledge from multiple low-dimensional hyperbolic KGE teacher models to a student model. DualDE (Zhu et al. 2022) considers the dual influence between the teacher and the student in the distillation process to make the teacher more suitable for the student and obtain better distillation results.

We do not directly compare against the methods above since they need to train standard KGE models in advance and then apply various compression methods. The most relevant work for our paper is NodePiece (Galkin et al. 2022), a recently proposed compositional method for representing entities in KGs. It uses anchors and relations to encode entities with a fixed-size vocabulary.

3 Methodology

In the context of our work, a knowledge graph consists of an entity set $\mathcal{E}$, a relation set $\mathcal{R}$, and a triple set $\mathcal{T}$. More precisely, a knowledge graph is represented as $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$, where $\mathcal{T} = \{(h, r, t)\} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$. Conventional knowledge graph embedding methods often learn embeddings to represent every entity and relation to predict missing triples (namely link prediction) based on specific score functions (Bordes et al. 2013; Sun et al. 2019).

Rather than storing embeddings for all entities and relations, we aim to design a model with fewer parameters to encode entity embeddings and obtain competitive performance compared with conventional KGE methods to make parameters more efficient. In our proposed entity-agnostic representation learning, EARL, we only learn specific embeddings for a small set of entities and refer to them as reserved entities $\mathcal{E}^{res}$. In practice, entities in $\mathcal{E}^{res}$ are randomly selected in advance. We encode three kinds of distinguishable information to obtain embeddings for all entities. The encoding procedure can decrease the parameter space costs since the number of parameters in encoders is independent of the number of entities. We show an intuition of this method in Figure 2 and explain the details as follows.

Next, we first introduce the components and the overview of three distinguishable information in Section 3.1, then we describe the details of encoding such information to obtain entity embeddings in Section 3.2, and finally, we illustrate the training process of our model in Section 3.3.

3.1 Distinguishable Information

The goal of designing distinguishable information is to represent each entity as uniquely as possible. As shown in Figure 2, we give an example for a clear description of different distinguishable information.

ConRel. For a specific entity, we first use its connected relations, including their directions, as the connected relation information (ConRel) since different entities connect different relations, and connected relations can reflect entities' semantics (Wang et al. 2014b). For example, head entities of the relation `occupation` are mainly people, and tail entities are mainly positions.

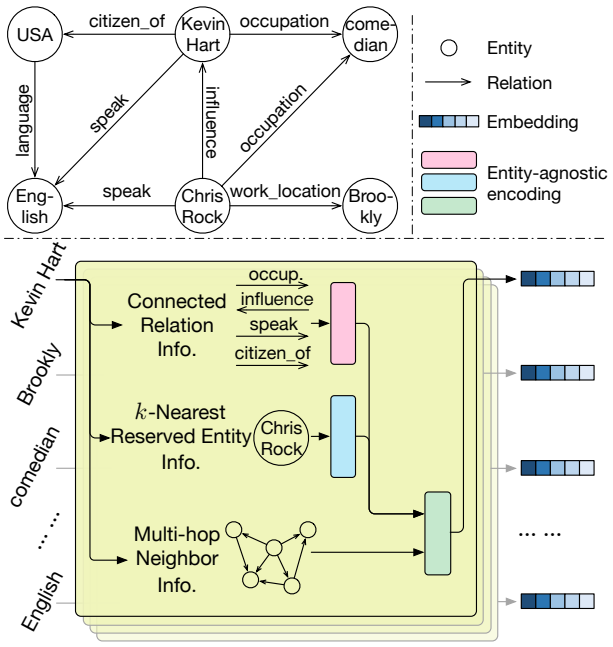

Figure 2: Illustration of distinguishable information.

kNResEnt. Nevertheless, the information provided by relations may be unclear since the granularities of relation definitions may be varied across KGs. More precisely, a relation `film_country` in Freebase (Bollacker et al. 2008) reflects that it connects a film and a country. However, from a relation `hypernym` in WordNet (Miller 1995), we only know that it connects two words, while all the entities in WordNet are words. Thus, connected relations (ConRel) cannot provide enough information to distinguish entities. Besides using relevant relations, we also use some relevant entities to represent a specific entity. We use k-nearest reserved entities (kNResEnt) as a kind of distinguishable information, and the similarity between entities is based on their connected relations. For example, in Figure 2, if the entity `Chris_Rock` is a reserved entity and it is the nearest entity for `Kevin_Hart`, then the embedding of `Chris_Rock` will be used to encode the embedding of `Kevin_Hart`.

MulHop. Even though ConRel and kNResEnt can capture distinguishable information, they fail when two entities have the same relation connection. To enhance the distinguishing capability, we consider using multi-hop neighbor (MulHop) information since it is almost impossible for two entities to have both the same relation connection and multi-hop neighbor. Specifically, we use a GNN to update the above two distinguishable information for incorporating neighbor information from KG structures.

3.2 Entity-Agnostic Encoding

In EARL, we only learn specific embeddings for a small set of entities (i.e., reserved entities) $\mathcal{E}^{res}$, and the embedding

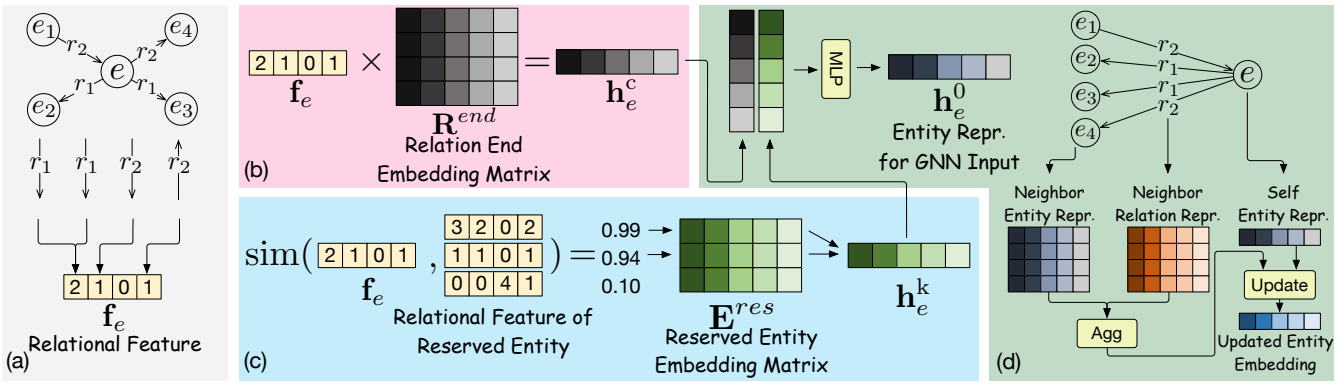

Figure 3: Overview of (a) constructing relational features, (b) ConRel encoding, (c) kNResEnt encoding and (d) MulHop encoding.

matrix of them is denoted as $\mathbf{E}^{res} \in \mathbb{R}^{|\mathcal{E}^{res}| \times d}$. These entities are randomly selected in advance, and their embeddings are trainable parameters. For the full set of entities, we obtain their embeddings via the following entity-agnostic encoding based on distinguishable information.

ConRel Encoding. To formally represent the connected relations of an entity, we propose the *relational feature*. Specifically, for each entity e, each dimension of its relational feature $\mathbf{f}_e \in \mathbb{Z}^{2|\mathcal{R}|}$ represents the frequency of being the head or the tail entity of a relation in $\mathcal{R} = \{r_i\}_{i=1}^{|\mathcal{R}|}$. Formally, we define each dimension of $\mathbf{f}_e$ as follows:

$$\mathbf{f}_{e,i} = \begin{cases} \mathrm{H}(e, r_i), & i \leq |\mathcal{R}| \\ \mathrm{T}(e, r_{i-|\mathcal{R}|}), & |\mathcal{R}| < i \leq 2|\mathcal{R}|, \end{cases} \tag{1}$$

where $\mathrm{H}(e, r) = |\{(e, r, x) | \exists x, (e, r, x) \in \mathcal{T}\}|$ denotes the frequency of the entity e as the head entity of the relation r; $\mathrm{T}(e, r) = |\{(x, r, e) | \exists x, (x, r, e) \in \mathcal{T}\}|$ denotes the frequency of the entity e as the tail entity of the relation r. The visual illustration is shown in Figure 3(a).

To maintain the semantics of being head or tail entities of relations, we propose *relation end embeddings* $\mathbf{R}^{end} \in \mathbb{R}^{2|\mathcal{R}| \times d}$ to encode relational feature:

$$\mathbf{h}_e^{\mathrm{c}} = f_r(\mathbf{f}_e^\top \mathbf{R}^{end}), \tag{2}$$

where $f_r : \mathbb{R}^d \to \mathbb{R}^d$ is a 2-layer MLP; $\mathbf{h}_e^{\mathrm{c}}$ denotes the encoded ConRel information for entity e.

kNResEnt Encoding. For encoding the information from k-nearest reserved entities for the entity e, we first calculate the cosine similarity between e and each entity e_i in $\mathcal{E}^{res}$ based on relational features:

$$\mathrm{sim}(e, e_i) = \frac{\mathbf{f}_e^\top \mathbf{f}_{e_i}}{\|\mathbf{f}_e\| \|\mathbf{f}_{e_i}\|}. \tag{3}$$

Next, we retrieve the top-k reserved entities based on similarity values:

$$\mathcal{P}_e^k = \mathrm{Top}^k(\{\mathrm{sim}(e, e_i) | e_i \in \mathcal{E}^{res}\}), \tag{4}$$

where $\mathcal{P}_e^k$ is a top-k reserved entity set for the entity e, and k is a hyper-parameter.

For utilizing the retrieved reserved entities, we use a weighted sum to encode k-nearest reserved entity information as follows:

$$\begin{aligned} \mathcal{V}_e^k &= \mathrm{Softmax}(\{\mathrm{sim}(e, e_i) | e_i \in \mathcal{P}_e^k\}), \\ \mathbf{h}_e^{\mathrm{k}} &= \sum_{e_i \in \mathcal{P}_e^k, v_i \in \mathcal{V}_e^k,} v_i \mathbf{E}_{e_i}^{res}, \end{aligned} \tag{5}$$

where $\mathbf{E}_{e_i}^{res}$ denotes the embedding for the reserved entity e_i, and $\mathbf{h}_e^{\mathrm{k}}$ denotes the encoded kNResEnt information for entity e.

MulHop Encoding. For incorporating multi-hop neighbor information, we use a GNN to update $\mathbf{h}_e^{\mathrm{c}}$ and $\mathbf{h}_e^{\mathrm{k}}$ for each entity e. We use the above two kinds of encoded information as the input representations of GNN, and we combine them as follows:

$$\mathbf{h}_e^0 = f_m([\mathbf{h}_e^{\mathrm{c}}; \mathbf{h}_e^{\mathrm{k}}]), \tag{6}$$

where the operation $[\cdot;\cdot]$ denotes the vector concatenation, and $f_m : \mathbb{R}^{2d} \to \mathbb{R}^d$ is a 2-layer MLP. Furthermore, $\mathbf{h}_e^0$ is the input representation of the GNN for e. Note that for the non-reserved entities, we use Equation (6) to obtain the input representation; for the reserved entities in $\mathcal{E}^{res}$, their input representations are directly looked up from the embedding matrix $\mathbf{E}^{res}$.

In our GNN framework, similar to previous works (Vashishth et al. 2020; Chen et al. 2022a), we use a linear transformation on the concatenation of entity and relation representations to aggregate the neighbor information. Specifically, the message aggregation for the entity e is:

$$\mathbf{m}_e^l = \sum_{(r,t) \in \mathcal{O}(e)} \mathbf{W}_{\text{out}}^l [\mathbf{h}_r^l; \mathbf{h}_t^l] + \sum_{(r,h) \in \mathcal{I}(e)} \mathbf{W}_{\text{in}}^l [\mathbf{h}_r^l; \mathbf{h}_h^l], \tag{7}$$

where $\mathcal{O}(e)$ denotes the out-going relation-entity pair set of e and $\mathcal{I}(e)$ denotes the in-going relation-entity pair set. $\mathbf{W}_{\text{out}}^l$ and $\mathbf{W}_{\text{in}}^l$ are transformation matrices for out-going and in-going pairs. $l \in [0, \ldots, L]$ denotes the layer of GNN and L is the total number of GNN layers. The input entity representations are calculated in Equation (6), and the input relation representations (e.g., $\mathbf{h}_r^0$) are looked up in a trainable relation embedding matrix $\mathbf{R} \in \mathbb{R}^{|\mathcal{R}| \times d}$.

The entity representation of e in the GNN is updated as follows:

$$\mathbf{h}_e^{l+1} = \sigma\left(\frac{1}{c}\mathbf{m}_e^l + \mathbf{W}_{\text{self}}^l \mathbf{h}_e^l\right), \tag{8}$$

where $c = |\mathcal{I}(e) + \mathcal{O}(e)|$ is a normalization constant. $\mathbf{W}_{\text{self}}^l$ is a matrix for self representation update, and σ is an activation function. Furthermore, relation representations will also be updated in each layer: $\mathbf{h}_r^{l+1} = \sigma\left(\mathbf{W}_{\text{rel}}^l \mathbf{h}_r^l\right)$. We use the output representations in the L-th layer for entities and relations as their embeddings to calculate scores next.

3.3 Model Training

Following the conventional KGE training regime, we optimize EARL to score true triples in the training set higher than sampled negative triples. Many score functions can be used in EARL. To show its versatility, we use RotatE (Sun et al. 2019), a representative method and one of the state-of-the-art KGE models, as the score function in EARL: $f(h,r,t) = -\|\mathbf{h} \circ \mathbf{r} - \mathbf{t}\|$. Here entities and relations are mapped into complex vector spaces, $\mathbf{h}, \mathbf{r}, \mathbf{t} \in \mathbb{C}^d$.

As for the loss function, we apply a widely used self-adversarial negative sampling loss:

$$\begin{aligned}\mathcal{L}(h,r,t) = &-\log\sigma\left(\gamma + f(h,r,t)\right) \\ &- \sum_{i=1}^{n} p\left(h_i', r, t_i'\right) \log\sigma\left(-\gamma - f(h_i', r, t_i')\right),\end{aligned} \tag{9}$$

where γ is a fixed margin and σ is the sigmoid function. (h_i', r, t_i') is a sampled negative triple for (h,r,t) and n is the number of negative triples. $p\left(h_i', r, t_i'\right)$ is the self-adversarial weight for this negative triple, and the calculation of this weight is as follows:

$$p\left(h_j', r, t_j'\right) = \frac{\exp \alpha f(h_j', r, t_j')}{\sum_i \exp \alpha f(h_i', r, t_i')}, \tag{10}$$

where α is the temperature factor.

4 Experiments

In this section, we conduct extensive experiments and analyses on various datasets to show the effectiveness of our proposed EARL. Note that the focus of our model is not outperforming the state-of-the-art KGE methods but showing that we are more parameter-efficient. Thus, this section is motivated by the following research questions: (**RQ1**) Is EARL parameter-efficient and capable of obtaining competitive performance? (**RQ2**) How does the effectiveness of components in EARL on different datasets? (**RQ3**) What is the impact of different settings on EARL? The source code is available at https://github.com/zjukg/EARL.

4.1 Experimental Setting

Datasets and Baselines. Our model is evaluated on several KG benchmarks with various sizes and characteristics, and the dataset statistics are shown in Table 1. Specifically, FB15k-237 (Toutanova et al. 2015) is derived from Freebase (Bollacker et al. 2008) with 237 relations, and the inverse relations are deleted. WN18RR (Dettmers et al. 2018) is a subset of WordNet (Miller 1995) with inverse relations deleted.

Dataset	#Ent	#Rel	#Train	#Valid	#Test
FB15k-237	14,505	237	272,115	17,526	20,438
WN18RR	40,559	11	86,835	2,824	2,924
CoDEx-L	77,951	69	551,193	30,622	30,622
YAGO3-10	123,143	37	1,079,040	4,978	4,982

Table 1: Dataset statistics. The number of entities, relations, training triples, validation triples, and test triples.

CoDEx (Safavi and Koutra 2020) is a recently proposed KG benchmark that contains more diverse and interpretable content and is more difficult than previous datasets. CoDEx-L is the large-size version. YAGO3-10 (Mahdisoltani, Biega, and Suchanek 2015) is a subset of YAGO3, which consists of entities that have a minimum of 10 relations each.

For comparison, we use RotatE (Sun et al. 2019), a representative and one of the state-of-the-art KGE methods, as a baseline, and the number of its model parameters can be controlled by the embedding dimension. Moreover, NodePiece (Galkin et al. 2022) which uses anchor nodes and node tokenization is the most proper baseline for EARL.

Evaluation Metrics. We evaluate models by the performance of link prediction on KGs, namely predicting missing triples in test sets. We report Mean Reciprocal Rank (MRR) and Hits@10 in the filtered setting (Bordes et al. 2013). For quantifying the efficiency of models, we propose a metric calculated by *MRR/#P* (#P denotes the number of parameters), and we denote it as *Effi*. For a fair comparison with baselines, we don't test the triples which involve entities that do not appear in the corresponding training sets.

Implementation Details. We conduct our experiments on NVIDIA RTX 3090 GPUs with 24GB RAM, and we use PyTorch (Paszke et al. 2019) and DGL (Wang et al. 2019) for handling automatic differentiation and graph structure modeling. For entity-agnostic encoding, we use 2-layer GNNs, and the default number of k for kNResEnt encoding is 10. We set the number of reserved entities as 10% of the number of all entities for each dataset, namely 1450, 4055, 7795, and 12314 for FB15k-237, WN18RR, CoDEx-L, and YAGO3-10. For model training, the learning rate is set to 0.001; the batch size is set to 1024; the number of negative samples (i.e., n) is set to 256; the margin is set to 15 for YAGO3-10 and 10 for other datasets.

4.2 Main Results

We summarize the results on four datasets in Table 2 and Table 3. We report the results of RotatE with large parameter counts to show the approximate upper-bound performance on datasets. Moreover, results from RotatE with similar parameter counts as NodePiece and EARL are used for comparison. Note that we don't try to outperform conventional KGE methods on a large parameter budget since parameter efficiency is not an important factor in that scenario but performance.

From these results, comparing the performance of EARL with NodePiece and RotatE using a similar parameter budget, we find that EARL outperforms them on MRR and

	FB15k-237					WN18RR				
	Dim	#P(M)	MRR	Hits@10	Effi	Dim	#P(M)	MRR	Hits@10	Effi
RotatE	1000	29.3	0.336	0.532	0.011	500	40.6	0.508	0.612	0.013
RotatE	100	2.9	0.296	0.473	0.102	50	4.1	0.411	0.429	0.100
NodePiece + RotatE*	100	3.2	0.256	0.420	0.080	100	4.4	0.403	0.515	0.092
EARL + RotatE	150	1.8	0.310	0.501	0.172	200	3.8	0.440	0.527	0.116
w/o Reserved Entity	150	1.1	0.306	0.492	0.278	200	1.7	0.347	0.461	0.204
w/o ConRel	150	1.2	0.309	0.501	0.257	200	3.0	0.432	0.520	0.144
w/o kNResEnt	150	1.6	0.301	0.488	0.188	200	3.3	0.409	0.498	0.124
w/o ConRel + kNResEnt	150	1.2	0.302	0.486	0.251	200	3.0	0.350	0.479	0.117
w/o MulHop	150	1.1	0.250	0.414	0.227	200	2.4	0.048	0.084	0.020

Table 2: Link prediction results on FB15k-237 and WN18RR. Results of * are taken from Galkin et al. (2022)

	CoDEx-L					YAGO3-10				
	Dim	#P(M)	MRR	Hits@10	Effi	Dim	#P(M)	MRR	Hits@10	Effi
RotatE*	500	78.0	0.258	0.387	0.003	500	123.2	0.495	0.670	0.004
RotatE*	25	3.8	0.196	0.322	0.052	20	4.8	0.121	0.262	0.025
NodePiece + RotatE*	100	3.6	0.190	0.313	0.053	100	4.1	0.247	0.488	0.060
EARL + RotatE	100	2.1	0.238	0.390	0.113	100	3.0	0.302	0.498	0.101
w/o Reserved Entity	100	0.5	0.203	0.337	0.406	100	0.4	0.119	0.226	0.296
w/o ConRel	100	1.9	0.237	0.384	0.124	100	2.8	0.322	0.522	0.115
w/o kNResEnt	100	2.0	0.232	0.374	0.116	100	2.9	0.249	0.429	0.086
w/o ConRel + kNResEnt	100	1.9	0.234	0.375	0.123	100	2.8	0.286	0.487	0.102
w/o MulHop	100	1.8	0.095	0.174	0.053	100	2.7	0.033	0.048	0.012

Table 3: Link prediction results on CoDEx-L and YAGO3-10. Results of * are taken from Galkin et al. (2022)

Hits@10 while using fewer parameters. Specifically, on FB15k-237, EARL uses only 62% parameters and obtains a relative increase of 4.7% on MRR in comparison with RotatE. On WN18RR, EARL achieves a 7% MRR improvement with 93% parameters compared with RotatE. In these two datasets, another baseline NodePiece uses even more parameters but does not outperform RotatE on MRR. On CoDEx-L, EARL uses 58% parameters and increases 25.4% relatively on MRR compared with NodePiece. EARL improves MRR with 22.2% on YAGO3-10 with only 73% parameters in contrast with NodePiece.

Furthermore, apart from analyzing the effectiveness of EARL from static parameter budgets in tables, we also explore the performance of different models with dynamic parameter counts in Figure 4. We adjust the dimensions of models to make their number of parameters from 1 to 5 million. We find that EARL obtains stable performance no matter how much the parameter is and improves significantly compared with RotatE in relatively low parameter budgets. Even though the results from the parameter-efficient baseline NodePiece are also stable, EARL achieves steady improvements in contrast with it. The dramatic drop with parameter decrease for RotatE's performance shows the inefficiency of entity-related KGE on small parameter budgets.

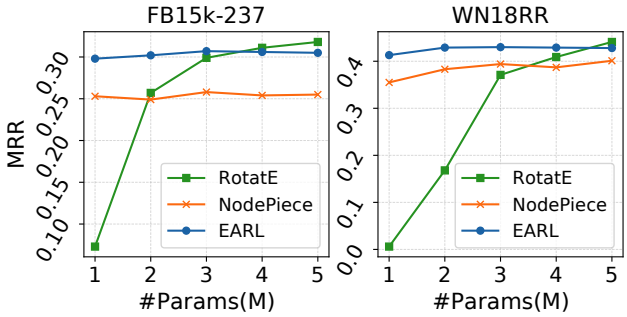

Figure 4: Performance with different parameter budgets.

Finally, from the values of metric Effi in Table 2 and 3, it's intuitive that EARL is more parameter-efficient than baselines. Above results indicate that our proposed EARL is parameter-efficient and answer the **RQ1**.

4.3 Ablation Study

In this part, we further probe the roles of different components in EARL by removing them separately. There are five types of ablation studies, as shown in Table 2 and 3.

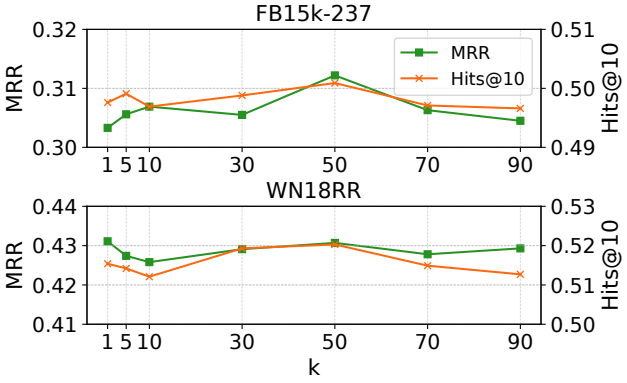

Figure 5: Performance with various k.

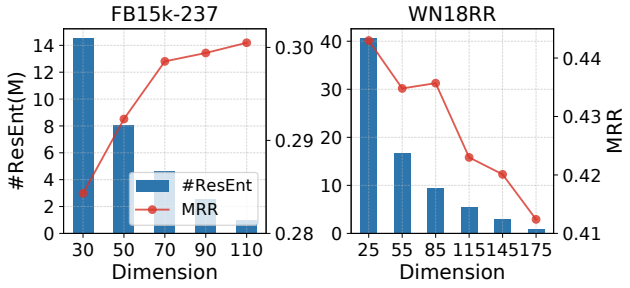

Figure 6: Performance of different model settings for a fixed parameter budget.

More precisely, for "w/o Reserved Entity", there are no reserved entities, and naturally, the kNREnt information is also disabled. For "w/o ConRel + kNResEnt", we use random entity representations for the GNN to encode MulHop information and output entity embeddings. Moreover, "w/o ConRel", "w/o kNResEnt" and "w/o MulHop" remove corresponding distinguishable information, respectively. Since the different characteristics of different datasets, the trends of ablation study results are distinct. We analyze them as follows respectively.

For *FB15k-237*, except for removing MulHop, other ablation settings affect the performance slightly but not significantly. For *WN18RR*, "w/o Reserved Entity" and "w/o kNResEnt" impairs the performance. Replacing ConRel and kNResEnt with random representations ("w/o ConRel + kNResEnt") also affect the results. Moreover, the performance is affected dramatically by removing MulHop information. For *CoDEx-L*, the trend is similar to that of FB15k-237, and "w/o MulHop" has a remarkable influence on performance. For *YAGO3-10*, the trend is similar to that of WN18RR.

From the above analysis, we find that in ablation studies, FB15k-237 and CoDEx-L have a similar trend, and WN18RR and YAGO3-10 have a similar trend. We explain this from their data statistics. That is, FB15k-237 and CoDEx-L have more relations than WN18RR and YAGO3-10, and diverse relations provide enough distinguishable information for entity embeddings. Thus, even in the "w/o Reserved Entity" and "w/o kNResEnt", performance is not affected dramatically since ConRel information still exists.

Overall, the ablation study shows the effectiveness of components of EARL and the different ablation behaviors on datasets with different statistics, which answers the **RQ2**.

4.4 Further Analysis

Analysis of k. We also investigate the effect of the value of k for retrieving k-nearest reserved entities to encode kNResEnt information. Based on FB15k-237 and WN18RR, we plot the MRR and Hits@10 results of training EARL with $k = [1, 5, 10, 30, 50, 70, 90]$ with 14,000 and 20,000 steps respectively in Figure 5. We find that our proposed EARL is robust to the value of k since the variance in MRR and Hits@10 is about 0.01, and there is no significant performance change. Specifically, we observe that k in a middle value (e.g., 50) leads to better performance, indicating that the results don't improve as k increases. It is possible that a higher k brings more noise into training. Moreover, a higher k costs more training computation in practice.

Fixed Parameter Budget. Two main factors controlling the parameter count in EARL are the embedding dimension and the number of reserved entities (#ResEnt), which contribute to the flexibility of EARL since we can adjust different values on the dimension and the #ResEnt for a fixed parameter budget. Given the 1M and 2M parameter budgets on FB15k-237 and WN18RR respectively, we slide the dimensions and adjust #ResEnt to satisfy the parameter budgets. From Figure 6, we find that on FB15k-237, dimension is a more critical factor in influencing the performance since MRR improves as the increase of dimension. On WN18RR, #ResEnt is more important. As the increase of #ResEnt, even though the dimension is very small (i.e., 25), EARL obtains better performance on WN18RR. These results are consistent with our analyses in ablation studies that FB15k-237 has more diverse relations for encoding entities discriminatively, while WN18RR depends more on reserved entities for distinguishable information. We suggest that when using EARL with a fixed parameter budget, the dimension and #ResEnt can be adjusted for better performance based on datasets' characteristics. Above analyses on various settings of EARL finally answer **RQ3**.

5 Conclusion

In this paper, we propose an entity-agnostic representation learning framework, EARL, for achieving parameter-efficient knowledge graph embedding. Unlike conventional entity-related KGE methods, EARL is entity-agnostic and does not map the model components to entities, preventing the number of parameters from scaling up linearly as the number of entities increases. Specifically, we design three kinds of distinguishable information to represent entities and then use an entity-agnostic encoding process to encode entity embeddings. Extensive empirical results show the effectiveness of our embedding encoding process and the parameter efficiency of EARL.

Acknowledgements

This work is partially supported by NSFC U19B2027 and 91846204, with Mingyang Chen supported by the China Scholarship Council (No. 202206320309) and Jeff Z. Pan supported by the Chang Jiang Scholars Program (J2019032).

References

Baek, J.; Lee, D. B.; and Hwang, S. J. 2020. Learning to Extrapolate Knowledge: Transductive Few-shot Out-of-Graph Link Prediction. In *NeurIPS*.

Bollacker, K. D.; Evans, C.; Paritosh, P. K.; Sturge, T.; and Taylor, J. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*.

Bordes, A.; Usunier, N.; Garcia-Duran, A.; Weston, J.; and Yakhnenko, O. 2013. Translating embeddings for modeling multi-relational data. In *NIPS*.

Carlson, A.; Betteridge, J.; Kisiel, B.; Settles, B.; Hruschka, E. R.; and Mitchell, T. M. 2010. Toward an architecture for never-ending language learning. In *AAAI*.

Chen, M.; Zhang, W.; Yao, Z.; Chen, X.; Ding, M.; Huang, F.; and Chen, H. 2022a. Meta-Learning Based Knowledge Extrapolation for Knowledge Graphs in the Federated Setting. In *IJCAI*, 1966–1972. ijcai.org.

Chen, M.; Zhang, W.; Yuan, Z.; Jia, Y.; and Chen, H. 2021a. FedE: Embedding Knowledge Graphs in Federated Setting. In *IJCKG*, 80–88. ACM.

Chen, M.; Zhang, W.; Zhu, Y.; Zhou, H.; Yuan, Z.; Xu, C.; and Chen, H. 2022b. Meta-Knowledge Transfer for Inductive Knowledge Graph Embedding. In *SIGIR*, 927–937. ACM.

Chen, X.; Zhang, N.; Xie, X.; Deng, S.; Yao, Y.; Tan, C.; Huang, F.; Si, L.; and Chen, H. 2022c. KnowPrompt: Knowledge-aware Prompt-tuning with Synergistic Optimization for Relation Extraction. In *WWW*, 2778–2788. ACM.

Chen, Z.; Chen, J.; Geng, Y.; Pan, J. Z.; Yuan, Z.; and Chen, H. 2021b. Zero-Shot Visual Question Answering Using Knowledge Graph. In *ISWC*, volume 12922 of *Lecture Notes in Computer Science*, 146–162. Springer.

Dehghani, M.; Gouws, S.; Vinyals, O.; Uszkoreit, J.; and Kaiser, L. 2019. Universal Transformers. In *ICLR (Poster)*. OpenReview.net.

Dettmers, T.; Minervini, P.; Stenetorp, P.; and Riedel, S. 2018. Convolutional 2d knowledge graph embeddings. In *AAAI*.

Galkin, M.; Wu, J.; Denis, E. G.; and Hamilton, W. L. 2022. NodePiece: Compositional and Parameter-Efficient Representations of Large Knowledge Graphs. In *ICLR*.

Geng, Y.; Chen, J.; Zhang, W.; Pan, J. Z.; Chen, M.; Chen, H.; and Jiang, S. 2022. Relational Message Passing for Fully Inductive Knowledge Graph Completion. *CoRR*, abs/2210.03994.

Hinton, G. E.; Vinyals, O.; and Dean, J. 2015. Distilling the Knowledge in a Neural Network. *CoRR*, abs/1503.02531.

Howard, A. G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; and Adam, H. 2017. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *CoRR*, abs/1704.04861.

Hu, Z.; Gutiérrez-Basulto, V.; Zhiliang Xiang, X. L.; Li, R.; and Pan, J. Z. 2022. Type-aware Embeddings for Multi-Hop Reasoning over Knowledge Graphs. In *IJCAI-ECAI 22*, 3078–3084.

Huang, F.; Li, Z.; Chen, S.; Zhang, C.; and Ma, H. 2020. Image Captioning with Internal and External Knowledge. In *CIKM*, 535–544. ACM.

Ji, S.; Pan, S.; Cambria, E.; Marttinen, P.; and Yu, P. S. 2022. A Survey on Knowledge Graphs: Representation, Acquisition, and Applications. *IEEE Trans. Neural Networks Learn. Syst.*, 33(2): 494–514.

Lan, Z.; Chen, M.; Goodman, S.; Gimpel, K.; Sharma, P.; and Soricut, R. 2020. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. In *ICLR*. OpenReview.net.

Lin, D. D.; Talathi, S. S.; and Annapureddy, V. S. 2016. Fixed Point Quantization of Deep Convolutional Networks. In *ICML*, volume 48 of *JMLR Workshop and Conference Proceedings*, 2849–2858. JMLR.org.

Mahdisoltani, F.; Biega, J.; and Suchanek, F. M. 2015. YAGO3: A Knowledge Base from Multilingual Wikipedias. In *CIDR*. www.cidrdb.org.

McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; and y Arcas, B. A. 2017. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*.

Miller, G. A. 1995. WordNet: A Lexical Database for English. *Commun. ACM*, 38(11): 39–41.

Molchanov, P.; Tyree, S.; Karras, T.; Aila, T.; and Kautz, J. 2017. Pruning Convolutional Neural Networks for Resource Efficient Inference. In *ICLR (Poster)*. OpenReview.net.

Pan, J. Z.; Perez, J. M. G.; Ren, Y.; Wu, H.; Wang, H.; and Zhu, M. 2014. Graph Pattern based RDF Data Compression. In *Proc. of the 4th Joint International Conference on Semantic Technologies (JIST 2014)*.

Pan, J. Z.; Taylor, S.; and Thomas, E. 2009. Reducing Ambiguity in Tagging Systems with Folksonomy Search Expansion. In *the Proc. of the 6th European Semantic Web Conference (ESWC2009)*.

Pan, J. Z.; Vetere, G.; Gomez-Perez, J. M.; and Wu, H. 2017. *Exploiting Linked Data and Knowledge Graphs in Large Organisations*. Springer. ISBN 978-3-319-45652-2.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. 2019. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 8026–8037.

Peng, H.; Li, H.; Song, Y.; Zheng, V. W.; and Li, J. 2021. Differentially Private Federated Knowledge Graphs Embedding. In *CIKM*.

Sachan, M. 2020. Knowledge Graph Embedding Compression. In *ACL*, 2681–2691. Association for Computational Linguistics.

Safavi, T.; and Koutra, D. 2020. CoDEx: A Comprehensive Knowledge Graph Completion Benchmark. In *EMNLP (1)*, 8328–8350. Association for Computational Linguistics.

Schlichtkrull, M.; Kipf, T. N.; Bloem, P.; Van Den Berg, R.; Titov, I.; and Welling, M. 2018. Modeling relational data with graph convolutional networks. In *ESWC*.

Sun, Z.; Deng, Z.; Nie, J.; and Tang, J. 2019. RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space. In *ICLR*.

Tanon, T. P.; Weikum, G.; and Suchanek, F. 2020. YAGO 4: A Reason-able Knowledge Base. In *European Semantic Web Conference*, 583–596. Springer.

Toutanova, K.; Chen, D.; Pantel, P.; Poon, H.; Choudhury, P.; and Gamon, M. 2015. Representing text for joint embedding of text and knowledge bases. In *EMNLP*.

Trouillon, T.; Welbl, J.; Riedel, S.; Gaussier, É.; and Bouchard, G. 2016. Complex embeddings for simple link prediction. In *ICML*.

Vashishth, S.; Sanyal, S.; Nitin, V.; and Talukdar, P. P. 2020. Composition-based Multi-Relational Graph Convolutional Networks. In *ICLR*.

Vrandečić, D.; and Krötzsch, M. 2014. Wikidata: A Free Collaborative Knowledge Base. *Communications of the ACM*.

Wang, H.; Wang, Y.; Lian, D.; and Gao, J. 2021a. A Lightweight Knowledge Graph Embedding Framework for Efficient Inference and Storage. In *CIKM*, 1909–1918. ACM.

Wang, K.; Liu, Y.; Ma, Q.; and Sheng, Q. Z. 2021b. MulDE: Multi-teacher Knowledge Distillation for Low-dimensional Knowledge Graph Embeddings. In *WWW*, 1716–1726. ACM / IW3C2.

Wang, K.; Wang, Z.; Topor, R. W.; Pan, J. Z.; and Antoniou, G. 2014a. Eliminating Concepts and Roles from Ontologies in Expressive Descriptive Logics. *Comput. Intell.*, 30(2): 205–232.

Wang, M.; Yu, L.; Zheng, D.; Gan, Q.; Gai, Y.; Ye, Z.; Li, M.; Zhou, J.; Huang, Q.; Ma, C.; et al. 2019. Deep graph library: Towards efficient and scalable deep learning on graphs. *arXiv preprint arXiv:1909.01315*.

Wang, Q.; Mao, Z.; Wang, B.; and Guo, L. 2017. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12): 2724–2743.

Wang, X.; Gong Cheng, T. L.; Xu, J.; Pan, J. Z.; Kharlamov, E.; and Qu, Y. 2014b. PCSG: Pattern-Coverage Snippet Generation for RDF Datasets. In *the 20th International Semantic Web Conference (ISWC2021)*, 3–20.

Wiharja, K.; Pan, J. Z.; Kollingbaum, M. J.; and Deng, Y. 2020. Schema Aware Iterative Knowledge Graph Completion. *Journal of Web Semantics*.

Xiong, W.; Du, J.; Wang, W. Y.; and Stoyanov, V. 2020. Pretrained Encyclopedia: Weakly Supervised Knowledge-Pretrained Language Model. In *ICLR*. OpenReview.net.

Xu, Z.; Zhang, W.; Ye, P.; Chen, H.; and Chen, H. 2022. Neural-Symbolic Entangled Framework for Complex Query Answering. *CoRR*, abs/2209.08779.

Yang, B.; Yih, W.; He, X.; Gao, J.; and Deng, L. 2015. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. In *ICLR*.

Yasunaga, M.; Ren, H.; Bosselut, A.; Liang, P.; and Leskovec, J. 2021. QA-GNN: Reasoning with Language Models and Knowledge Graphs for Question Answering. In *NAACL-HLT*.

Yu, D.; Zhu, C.; Yang, Y.; and Zeng, M. 2022. JAKET: Joint Pre-training of Knowledge Graph and Language Understanding. In *AAAI*, 11630–11638. AAAI Press.

Zhang, F.; Yuan, N. J.; Lian, D.; Xie, X.; and Ma, W. 2016. Collaborative Knowledge Base Embedding for Recommender Systems. In *KDD*.

Zhang, W.; Chen, X.; Yao, Z.; Chen, M.; Zhu, Y.; Yu, H.; Huang, Y.; Xu, Y.; Zhang, N.; Xu, Z.; Yuan, Z.; Xiong, F.; and Chen, H. 2022. NeuralKG: An Open Source Library for Diverse Representation Learning of Knowledge Graphs. In *SIGIR*, 3323–3328. ACM.

Zhang, W.; Paudel, B.; Wang, L.; Chen, J.; Zhu, H.; Zhang, W.; Bernstein, A.; and Chen, H. 2019. Iteratively Learning Embeddings and Rules for Knowledge Graph Reasoning. In *The World Wide Web Conference*, 2366–2377. ACM.

Zhu, M.; Wu, W.; Pan, J. Z.; Han, J.; Huang, P.; and Liu, Q. 2018. Predicate Invention Based RDF Data Compression. In *Proc. of the Joint International Semantic Technology Conference (JIST2018)*, 153–161.

Zhu, Y.; Zhang, W.; Chen, M.; Chen, H.; Cheng, X.; Zhang, W.; and Chen, H. 2022. DualDE: Dually Distilling Knowledge Graph Embedding for Faster and Cheaper Reasoning. In *WSDM*, 1516–1524. ACM.