Meta-Learning Based Knowledge Extrapolation for Knowledge Graphs in the Federated Setting

Mingyang Chen[1*] , **Wen Zhang**[2*] , **Zhen Yao**[2] , **Xiangnan Chen**[2] , **Mengxiao Ding**[3] ,
Fei Huang[4] and **Huajun Chen**[1,5,6†]

[1]College of Computer Science and Technology, Zhejiang University
[2]School of Software Technology, Zhejiang University
[3]Huawei Technologies Co., Ltd.
[4]Alibaba Group
[5]ZJU-Hangzhou Global Scientific and Technological Innovation Center
[6]Alibaba-Zhejiang University Joint Institute of Frontier Technologies
{mingyangchen, zhang.wen, yz0204, xnchen2020, huajunsir}@zju.edu.cn,
dingmengxiao@huawei.com, f.huang@alibaba-inc.com

Abstract

We study the knowledge extrapolation problem to embed new components (i.e., entities and relations) that come with emerging knowledge graphs (KGs) in the federated setting. In this problem, a model trained on an existing KG needs to embed an emerging KG with unseen entities and relations. To solve this problem, we introduce the meta-learning setting, where a set of tasks are sampled on the existing KG to mimic the link prediction task on the emerging KG. Based on sampled tasks, we meta-train a graph neural network framework that can construct features for unseen components based on structural information and output embeddings for them. Experimental results show that our proposed method can effectively embed unseen components and outperforms models that consider inductive settings for KGs and baselines that directly use conventional KG embedding methods.

1 Introduction

Knowledge graphs (KGs) are expressive data structures that consist of a large number of triples with the form of *(head entity, relation, tail entity)*. Nowadays, many large-scale KGs have become essential data supports for an increasing number of applications [Zhang *et al.*, 2016; Yasunaga *et al.*, 2021]. With the development of KGs, they are no longer only applied in a centralized way where all the triples of a KG can be accessed on one device, but also in a decentralized manner. To our best knowledge, many mobile applications build personal KGs on users' devices, and naturally, new KGs on new devices are emerging at any given moment. However, conventional large-scale KGs are well known to be incomplete; thus, emerging KGs also suffer from this problem. For KG
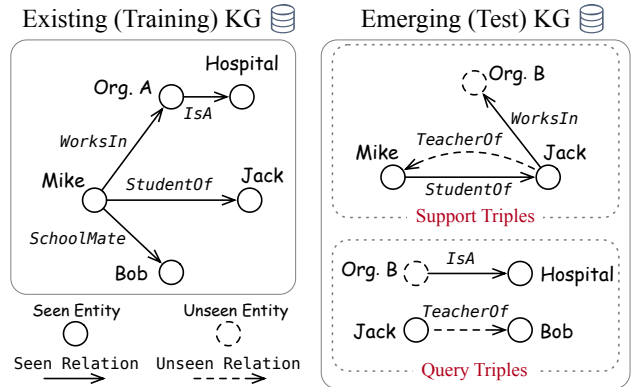

Figure 1: The challenges for embedding emerging KGs in the federated setting.

completion, extensive research has been devoted to predicting missing links by learning low-dimensional vector representations (a.k.a, knowledge graph embeddings) for entities and relations that proved effective.

Nevertheless, knowledge graph embedding (KGE) methods are not perfect in real-world applications. Conventional KGE methods cannot deal with new relations and entities since they learn embeddings for a fixed set of entities and relations. While as mentioned above, an emerging KG is often accompanied by new relations and entities. As shown in Fig. 1, the emerging KG contains an unseen entity `Org.B` and an unseen relation `TeacherOf`, so a KGE model trained on the existing KG cannot be applied to query missing triples about such unseen components based on support triples in the emerging KG.

Although existing methods [Teru *et al.*, 2020] focusing on inductive settings for KG completion can handle unseen entities in a new KG, they cannot consider both unseen relations and entities. Moreover, such inductive KG completion methods cannot take advantage of seen entities since no transferable information of seen entities, like trained vector repre-

---

*Equal Contribution.
†Corresponding author.

sentations, is considered; unless the existing KG is integrated with the emerging KG. However, such KG integration may not be allowed for multiple reasons in real applications, such as data privacy. We call this scenario KGs in the federated setting [McMahan *et al.*, 2017], namely, these KGs can use the same model while their data are not shared explicitly.

Motivated by the challenges faced with this real-world scenario, we raise a research question: *Can we have an embedding-based model trained on seen relations and entities and generalize to unseen relations and entities for KGs in the federated setting?*

Formally, the link prediction task for an emerging KG can be viewed as predicting the plausibility of a set of query triples based on another set of support triples, as shown in Fig. 1. To handle the unseen components in the emerging KG, inspired by the ability of "learning to learn" brought by meta-learning [Santoro *et al.*, 2016], we formulate a set of tasks consisting of support triples and query triples on the existing KG to mimic the link prediction task in the emerging KG, and learn to embed unseen components in each task. Furthermore, we propose a Graph Neural Network (GNN) framework that can embed both seen and unseen components based on support triples of a task. Our model mainly contains three modules: 1) a relation feature representation module, in which we construct a Relation Position Graph (RPG) from triples to reveal relative positions between relations and construct relation features based on it; 2) an entity feature representation module where we use the connections between entities and their neighbors to represent their features; 3) a GNN that encodes features and outputs embeddings for both seen and unseen components to achieve knowledge extrapolation.

This model is meta-trained on tasks sampled from the existing KG and learns the ability to embed unseen components for a task; thus, the learned model can generalize to an emerging KG with unseen components. To evaluate our method, we introduce datasets for knowledge extrapolation in the federated setting, which are sampled from KG benchmarks. The evaluation results show that our method outperforms models that only consider unseen entities under inductive settings for KGs and baselines that calculate embeddings for unseen components based on conventional KGE assumptions[1].

2 Related Work

Knowledge Graph Embedding. A lot of KGE models [Wang *et al.*, 2017; Yang *et al.*, 2015] have been proposed to embed KGs into low-dimensional vector spaces. Different methods map entities and relations into different vector spaces and design various score functions. TransE [Bordes *et al.*, 2013] is a representative method that maps entities and relations as vectors in the same real space, and relations are interpreted as translation vectors between entities for triples. ComplEx [Trouillon *et al.*, 2016] and RotatE [Sun *et al.*, 2019] embed KGs into complex spaces and propose more sophisticated score functions. However, conventional KGE models embed a fixed set of components and cannot generalize to unseen entities and relations after model training.

---

[1]Source code is available at https://github.com/zjukg/MaKEr.

Inductive Settings for Knowledge Graph. Some existing works have proposed models applicable in inductive settings where there are unseen entities during the test. [Hamaguchi *et al.*, 2017] and [Wang *et al.*, 2019] learn to embed unseen entities by neighborhood aggregation based on GNNs, while they can only embed unseen entities connected with the training KG. [Teru *et al.*, 2020], [Chen *et al.*, 2021a] and [Liu *et al.*, 2021] train KG completion methods and can generalize to a new KG with unseen entities. However, they cannot generalize to unseen relations and cannot take advantage of seen entities during the test.

Meta-Learning. Meta-learning, known as "learning to learn", focuses on learning to generalize over the distribution of tasks but not data points. Metric-based methods [Snell *et al.*, 2017; Vinyals *et al.*, 2016] learn generalizable parameterized metrics from a set of training tasks. Optimized-based methods [Finn *et al.*, 2017] learn the optimization of model parameters given the gradients on a task. Black-box methods [Santoro *et al.*, 2016] train a model that can represent another model's parameters by standard supervised learning. Some existing works [Xiong *et al.*, 2018; Chen *et al.*, 2019] apply meta-learning on KGs to solve the few-shot problem, but not the problem in this paper.

Knowledge Graph and GNN. Graph neural networks [Kipf and Welling, 2017] encode representations for nodes in graphs via their neighbor structures. Recently, R-GCN [Schlichtkrull *et al.*, 2018] considers relation-specific transformations for neighbor aggregation of a node. CompGCN [Vashishth *et al.*, 2020] leverages entity-relation composition operations to embed entities and relations jointly and generalizes several previous multi-relational GCNs. We extend CompGCN as the GNN for updating entity and relation features in our model by replacing composition operators with a linear transformation for entity-relation aggregations. Such linear transformations are more general and flexible when dimensions for entities and relations are diverse.

Federated Settings for Knowledge Graph. Traditional research for knowledge graphs focuses on centralized settings, while KGs are evolving and may be built by different controllers [Chen *et al.*, 2021b]. Some existing works [Zhang *et al.*, 2021] train a generalizable graph mining model without sharing graph data from multiple local systems. [Peng *et al.*, 2021] tries to embed components from different KGs while being privacy-preserving. However, previous works learn embeddings for KGs from different data sources based on federated learning [McMahan *et al.*, 2017], and all the entities and relations for test triples are seen during training. However, the problem in our work focuses on the emerging KG with unseen components in the federated setting.

3 Problem Formulation

A knowledge graph is defined as $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$, where $\mathcal{E}$ denotes a set of entities, $\mathcal{R}$ is a set of relations, and $\mathcal{T}$ is a set of triples. Specifically, $\mathcal{T} = \{(h, r, t)\} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$, where $h, t \in \mathcal{E}$ and $r \in \mathcal{R}$. The link prediction task for $\mathcal{G}$ refers to the problem of predicting $e \in \mathcal{E}$, given an entity and a relation, namely, $(h, r, ?)$ or $(?, r, t)$, to make (h, r, e) or (e, r, t) a new true triple for completing $\mathcal{G}$.

In the context of traditional knowledge graph completion [Bordes *et al.*, 2013], to evaluate the link prediction ability of a model, there are two part of triples, training (a.k.a, support) triples $\mathcal{T}_{sup}$ and testing (a.k.a, query) triples $\mathcal{T}_{que}$. Specifically, $\mathcal{T}_{sup}$ is used for training a model $\mathcal{M}$ which usually has the ability to score the plausibility of a triple. Furthermore, $\mathcal{T}_{que}$ are ground-truth triples for evaluating the learned model. For example, for tail link prediction, given an ground-truth triple $(h, r, t) \in \mathcal{T}_{que}$, we rank $\mathcal{M}(h, r, t)$ over its candidates triples $\{\mathcal{M}(h, r, e) | e \in \mathcal{E}, (h, r, e) \notin \mathcal{T}_{sup} \cup \mathcal{T}_{que}\}$. The higher the ranks for ground-truth triples in $\mathcal{T}_{que}$, the more effective the model is.

Next, based on the above definitions, we formally define the problem of knowledge extrapolation for knowledge graphs in the federated setting. Given a training KG $\mathcal{G}^{tr} = (\mathcal{E}^{tr}, \mathcal{R}^{tr}, \mathcal{T}^{tr})$, we aim at training a triple scoring model $\mathcal{M}$ that can generalize to a test KG $\mathcal{G}^{te} = (\mathcal{E}^{te}, \mathcal{R}^{te}, \mathcal{T}^{te}_{sup}, \mathcal{T}^{te}_{que})$, where $\mathcal{E}^{tr} \neq \mathcal{E}^{te}, \mathcal{E}^{tr} \cap \mathcal{E}^{te} \neq \emptyset$ and $\mathcal{R}^{tr} \neq \mathcal{R}^{te}, \mathcal{R}^{tr} \cap \mathcal{R}^{te} \neq \emptyset$. Moreover, this problem has two constrains: 1) *knowledge extrapolation*, compared with the traditional knowledge graph completion task, the model training is only conducted on the $\mathcal{T}^{tr}$ but not on the $\mathcal{T}^{te}_{sup}$, and triples in $\mathcal{T}^{te}_{sup}$ are only used for revealing the connections of entities and relations in $\mathcal{E}^{te}$ and $\mathcal{R}^{te}$; 2) *federated setting*, $\mathcal{G}^{tr}$ and $\mathcal{G}^{te}$ are not allowed to be integrated together.

4 Methodology

4.1 Meta-Learning Setting

The main problem here is how to embed unseen entities and relations on $\mathcal{G}^{te}$ with effective representations that can handle KG completion. Inspired by the concept of "learning to learn" from meta-learning, with the goal of "embedding unseen entities and relations" on the test KG, we can formulate a set of tasks on the training KG with simulated unseen entities and relations to mimic the test environment, and learn a model on such tasks to achieve "learning to embed unseen entities and relations".

Specifically, each task $\mathcal{S}^i = (\mathcal{E}^i, \mathcal{R}^i, \mathcal{T}^i_{sup}, \mathcal{T}^i_{que})$ over a task distribution $p(\mathcal{S})$ corresponds to a sub-KG sampled from $\mathcal{G}^{tr}$. Although $\mathcal{E}^i$ and $\mathcal{R}^i$ are sampled from $\mathcal{E}^{tr}$ and $\mathcal{R}^{tr}$, we re-label a part of entities and relations and treat them as unseen entities and relations. A task $\mathcal{S}^i$ is defined as follows:

$$\mathcal{S}^i = \left(\mathcal{E}^i = (\widehat{\mathcal{E}}^i, \widetilde{\mathcal{E}}^i), \mathcal{R}^i = (\widehat{\mathcal{R}}^i, \widetilde{\mathcal{R}}^i), \mathcal{T}^i_{sup}, \mathcal{T}^i_{que} \right), \quad (1)$$

where $\widehat{\mathcal{E}}^i \in \mathcal{E}^{tr}$ are seen entities and $\widetilde{\mathcal{E}}^i \notin \mathcal{E}^{tr}$ are unseen entities; $\widehat{\mathcal{R}}^i \in \mathcal{R}^{tr}$ and $\widetilde{\mathcal{R}}^i \notin \mathcal{R}^{tr}$. Based on the tasks sampled from $\mathcal{G}^{tr}$ for meta-learning, the meta-training objective is learning to embed both seen and unseen entities and relations based on support triples, to maximize the score of query triples as follows:

$$\max_\theta \mathbb{E}_{\mathcal{S}^i \sim p(\mathcal{S})} \left[\sum_{(h,r,t) \in \mathcal{T}^i_{que}} \frac{1}{|\mathcal{T}^i_{que}|} \mathcal{M}_\theta(h, r, t | \mathcal{T}^i_{sup}) \right], \quad (2)$$

where $\mathcal{M}$ is an arbitrary model that can calculate plausibility scores for query triples based on support triples.

However, we don't have an existing model $\mathcal{M}$ which fits this problem. Hence, we design a model that can embed both seen and unseen entities and relations based on support triples of each sampled task, and we describe the details of the proposed model in the following.

4.2 Proposed Model

The overall of our model is a Graph Neural Network (GNN) framework and follows the encoder-decoder paradigm as previous works that apply GNN on KGs [Schlichtkrull *et al.*, 2018]. Generally, the encoder is a GNN structure that takes feature representations as input and outputs the embeddings; the decoder is a KGE method that takes embeddings of the components of a triple as input and outputs the score. Note that we do not use any entity or relation attributes to get the feature representation since our model only considers structure information of KGs, which is a common scenario and makes our model can be generalized to more applications.

Naturally, for a task $\mathcal{S}^i$, the feature representations of seen entities $e \in \widehat{\mathcal{E}}^i$ and relations $r \in \widehat{\mathcal{R}}^i$ can be looked up from an entity feature matrix $\mathbf{E}^{tr} \in \mathbb{R}^{|\mathcal{E}^{tr}| \times d_e}$ and a relation feature matrix $\mathbf{R}^{tr} \in \mathbb{R}^{|\mathcal{R}^{tr}| \times d_r}$, which can be randomly initialized based on the training KG $\mathcal{G}^{tr}$ and are learned during training. However, the difficulty is how to represent unseen entities and relations effectively. To solve this problem, we design the following three modules. The overview of our model can be found in Fig. 2, and we describe it based on a specific sampled task $\mathcal{S}^i$ as illustrated in Equ. (1).

Relation Feature Representation

In the literature about transfer learning of GNNs [Zhu *et al.*, 2021], node degrees can be used as node features since they are sensitive to changes in the graph structures. In our work, we also try to find some structure-respecting features to represent relations. To represent a relation as intuitional as a node in graphs, we construct a Relation Position Graph from the original knowledge graph (i.e., support triples $\mathcal{T}^i_{sup}$ for a task $\mathcal{S}^i$), where each node in the RPG represents one type of relation in the KG, as shown in Fig. 2(b).

We define four relationships between nodes in RPG, based on four relation relative positions (Fig. 2(a)). To avoid confusion, we refer to such "relationships between relations" as *meta-relations* and refer to nodes that represent relations as *rel-nodes*. For instance, $(r_1, \texttt{t-h}, r_2)$ denotes that a tail entity linked to r_1 is the head entity linked to r_2. For two specific relations, if one type of relative position for them appears in the support triples, their corresponding rel-nodes in RPG will be connected by the corresponding meta-relation. After building the RPG, as shown in Fig. 2(c), we represent the feature of an unseen relation r by its neighbor meta-relations in RPG:

$$\mathbf{h}_r = \frac{1}{|\mathcal{N}(r)|} \sum_{m \in \mathcal{N}(r)} \mathbf{h}_m, \quad (3)$$

where $\mathcal{N}(r)$ denotes the set of in-going meta-relations of the rel-node representing r. Note that we only consider in-going meta-relations here, since $\texttt{t-h}$ is the inverse of $\texttt{h-t}$, and $\texttt{h-h}$ and $\texttt{t-t}$ are double-direction, thus out-going meta-relations can be indicated by in-going meta-relations. Moreover, $\mathbf{h}_m \in \mathcal{H}_M$ is the vector representation of a specific
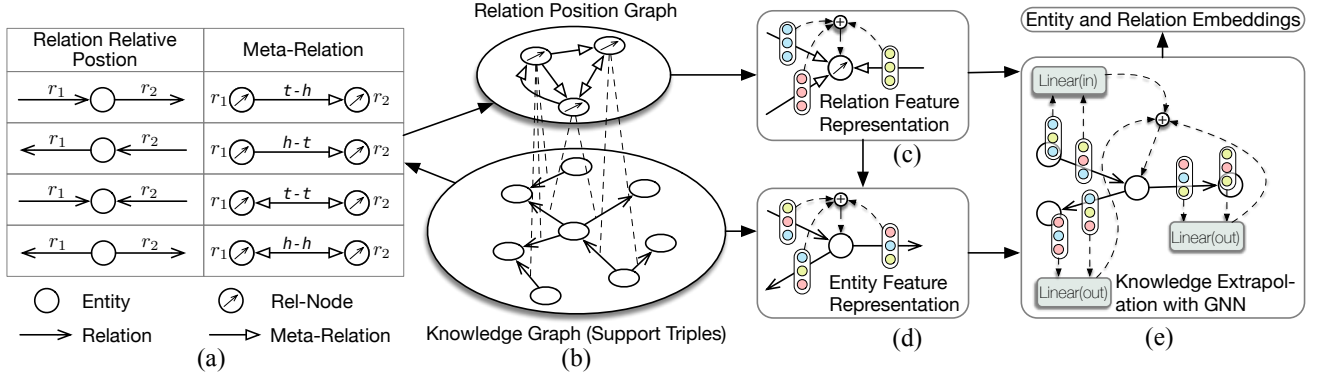
Figure 2: Overview of our proposed model. (a)(b) Construct RPG from the support triples of a task; (c)(d)(e) Three modules of the proposed model that represent features for entities and relations and output embeddings for them.

meta-relation, and $\mathcal{H}_M = (\mathbf{h}_{\text{t-h}}, \mathbf{h}_{\text{h-t}}, \mathbf{h}_{\text{t-t}}, \mathbf{h}_{\text{h-h}})$ is a set of learnable parameters for four meta-relations.

Entity Feature Representation
Similar to relation feature representation, we use entities' connected relation features to represent unseen entities. Considering the direction of relations connected to a specific unseen entity e, we calculate the entity feature representation as:

$$\mathbf{h}_e = \frac{1}{|\mathcal{N}(e)|} \sum_{r \in \mathcal{N}(e)} \mathbf{W}^{\text{ent}}_{\text{dir}(r)} \mathbf{h}_r, \qquad (4)$$

where $\mathcal{N}(e)$ is a set of relations connected to e; $\mathbf{W}^{\text{ent}}_{\text{dir}(r)}$ is a direction-specific parameter for transforming relation features to entity features, and $\mathbf{W}^{\text{ent}}_{\text{dir}(r)} = \mathbf{W}^{\text{ent}}_{\text{in}}$ if r is an in-going relation of e, or $\mathbf{W}^{\text{ent}}_{\text{dir}(r)} = \mathbf{W}^{\text{ent}}_{\text{out}}$.

Knowledge Extrapolation with GNN
We propose a GNN model to encode the entity and relation features. The previous two modules produce the features for unseen components, and seen components are looked up from learnable feature matrices. Our GNN model is based on CompGCN [Vashishth *et al.*, 2020], which has proven effective in encoding KGs. We extend its entity-relation composition operators to a liner transformation since: 1) this can be viewed as a more general operator between entities and relations; 2) it can use more sophisticated KGE methods as the decoder where the dimension of relations are not equal to that of entities (e.g., RotatE [Sun *et al.*, 2019]).

For an entity e, its message aggregation is conducted as:

$$\mathbf{m}^{l+1}_e = \sum_{(r,t) \in \mathcal{O}(e)} \mathbf{W}^l_{\text{out}}[\mathbf{h}^l_r; \mathbf{h}^l_t] + \sum_{(r,h) \in \mathcal{I}(e)} \mathbf{W}^l_{\text{in}}[\mathbf{h}^l_r; \mathbf{h}^l_h], \qquad (5)$$

where $\mathcal{O}(e)$ denotes a set of out-going relations and connected entities, $\mathcal{I}(e)$ denotes a set of in-going relations and connected entities; $\mathbf{W}^l_{\text{out}}$ and $\mathbf{W}^l_{\text{in}}$ are learnable parameters for out- and in-going relation-entity pairs at the l-th layer of GNN, $[\cdot ; \cdot]$ denotes the concatenation of two vectors, and $\mathbf{h}^0_r = \mathbf{h}_r$ and $\mathbf{h}^0_t = \mathbf{h}_t$ are input relation and entity features. After message aggregation from neighbors for each entity, the

entity representation for each e is updated by:

$$\mathbf{h}^{l+1}_e = \sigma \left(\frac{\mathbf{m}^{l+1}_e}{|\mathcal{O}(e)| + |\mathcal{I}(e)|} + \mathbf{W}^l_{\text{self}} \mathbf{h}^l_e \right), \qquad (6)$$

where $\mathbf{W}^l_{\text{self}}$ is a learnable parameter for self-loop updating for entities, and σ is an activation function. Besides entities, relations are also updated at each layer:

$$\mathbf{h}^{l+1}_r = \sigma \left(\mathbf{W}^l_{\text{rel}} \mathbf{h}^l_r \right). \qquad (7)$$

After updating entity and relation representations, the GNN outputs embeddings for both seen and unseen components in current task, and achieving knowledge extrapolation.

4.3 Model Learning
For the task $\mathcal{S}_i$, after obtaining embeddings via previous modules on support triples $\mathcal{T}^i_{sup}$, we train our model to score ground-truth query triples $\mathcal{T}^i_{que}$ higher than sampled negative triples. The score function can be chosen from various KGE methods, and score functions for representative KGE methods can be found in Appendix A. Furthermore, following previous practice, we use self-adversarial negative sampling loss proposed by [Sun *et al.*, 2019] to optimize our model:

$$\mathcal{L}(\mathcal{S}^i) = \frac{1}{|\mathcal{T}^i_{que}|} \sum_{(h,r,t) \in \mathcal{T}^i_{que}} - \log \sigma \left(\gamma + s(h,r,t) \right)$$
$$- \sum_{i=1}^n p\left(h'_i, r, t'_i \right) \log \sigma \left(-\gamma - s(h'_i, r, t'_i) \right), \qquad (8)$$

where $s(h,r,t)$ is the score for (h,r,t) using embeddings from our model based on $\mathcal{T}^i_{sup}$, γ is a fixed margin, n is the number of negative samples, (h'_i, r, t'_i) is a negative samples by corrupting a head or tail entity. $p\left(h'_i, r, t'_i \right)$ is the weight for a negative sample, and we put its calculation in Appendix B. Finally, we meta-train the model with the overall loss $\sum_i \mathcal{L}(\mathcal{S}^i)$ among all tasks sampled from $\mathcal{G}^{tr}$.

5   Experiments
In this section, we evaluate our proposed method MaKEr (for <u>Ma</u>ta-Learning Based <u>K</u>nowledge <u>E</u>xtrapolation) on datasets derived from KG benchmarks, and compared it with baselines to show the effectiveness of this model.

	FB-Ext						NELL-Ext					
	u_ent		u_rel		u_both		u_ent		u_rel		u_both	
	MRR	Hits@10	MRR	Hits@10	MRR	Hits@10	MRR	Hits@10	MRR	Hits@10	MRR	Hits@10
GraIL	<u>56.07</u>	<u>83.34</u>	—	—	—	—	<u>71.62</u>	<u>92.92</u>	—	—	—	—
INDIGO	42.98	60.25	—	—	—	—	50.31	67.68	—	—	—	—
Asmp-KGE (TransE)	<u>63.91</u>	<u>82.22</u>	33.79	35.50	<u>13.29</u>	23.88	<u>68.64</u>	<u>78.35</u>	5.26	2.50	9.25	9.65
Asmp-KGE (DistMult)	48.29	72.34	17.94	<u>36.50</u>	12.09	<u>25.24</u>	54.00	72.18	12.77	21.67	8.05	14.35
Asmp-KGE (ComplEx)	46.96	67.33	11.92	22.00	12.30	24.31	51.97	67.12	<u>14.32</u>	<u>22.50</u>	<u>9.94</u>	<u>17.48</u>
Asmp-KGE (RotatE)	55.12	71.65	**34.61**	35.50	12.96	23.42	58.70	61.42	6.34	3.34	6.73	4.00
MaKEr (TransE)	73.40	95.17	29.92	43.50	22.39	41.95	70.82	92.00	24.56	54.17	21.53	51.74
MaKEr (DistMult)	67.81	92.82	22.21	37.50	22.17	44.88	70.63	91.33	27.02	**60.00**	<u>41.39</u>	57.65
MaKEr (ComplEx)	70.09	93.67	24.39	43.00	24.56	**52.09**	72.24	91.91	18.27	34.17	29.39	59.65
MaKEr (RotatE)	**74.64**	**95.28**	<u>32.00</u>	**50.00**	**27.26**	49.51	**77.09**	**94.64**	**31.53**	55.00	31.45	**62.35**

Table 1: Link prediction results (%) on two datasets. We show results for query triples only containing unseen entities (u_ent), only containing unseen relations (u_rel), and containing both unseen entities and relations (u_both). **Bold** numbers denote the best results and <u>underline</u> numbers denote the best results in different kinds of methods. KGE methods after MaKEr denote the score functions used in MaKEr.

5.1 Experimental Setting

	Training KG $\mathcal{G}^{tr}$			Test KG $\mathcal{G}^{te}$			
	$|\mathcal{E}^{tr}|$	$|\mathcal{R}^{tr}|$	$|\mathcal{T}^{tr}|$	$|\mathcal{E}^{te}|$	$|\mathcal{R}^{te}|$	$|\mathcal{T}^{te}_{sup}|$	$|\mathcal{T}^{te}_{que}|$
FB-Ext	952	154	7,105	913 (806)	196 (56)	6,103	3,524
NELL-Ext	1,583	153	5,269	851 (753)	140 (30)	2,160	692

Table 2: Statistics of the datasets. The number in the bracket denotes the number of entities or relations that doesn't appear in corresponding training KG (i.e., unseen entities or relations).

Datasets. In conventional KG datasets, all entities and relations in test triples are seen during training. In order to evaluate the ability of a model for knowledge extrapolation in the federated setting, we create two datasets from two standard KG benchmarks, FB15k-237 [Toutanova *et al.*, 2015] and NELL-995 [Xiong *et al.*, 2017], named FB-Ext and NELL-Ext. For each dataset, we create a training KG $\mathcal{G}^{tr}$ and a test KG $\mathcal{G}^{ts}$ sampled separately from the original benchmark, and a part of entities and relations in the test KG are unseen in the training KG. We only put triples that contain at least one unseen component into query triples of the test KG. We also divide query triples into triples only containing unseen entities (u_ent), only containing unseen relations (u_rel), and containing both unseen entities and unseen relations (u_both). The statistics of two datasets are given in Table 2. The numbers of query triples for u_ent, u_rel and u_both are 1926, 20, 1578 in FB-Ext, and 565, 12, 115 in NELL-Ext. The details of generating datasets can be found in Appendix C.

Baselines. We compare our model with two state-of-the-art knowledge graph completion methods considering inductive settings for KGs, GraIL [Teru *et al.*, 2020], and INDIGO [Liu *et al.*, 2021], which can handle unseen entities in the test KG. Even though they don't consider unseen relations, they are representative baselines fitting our problems to the best of our knowledge. We also compare against baselines that use KGE methods directly on the test KG. Specifically, we first train a KGE model on the training KG. For unseen components in the test KG, we use the corresponding assumption from the score function in the KGE method to calculate their embed-

dings based on seen components' embeddings. For example, based on TransE, the embedding for an unseen entity t can be calculated by $\mathbf{h} + \mathbf{r} = \mathbf{t}$ if h and r are seen components and (h, r, t) exists in support triples. We refer to this baseline as *Asmp-KGE*, and the details are given in Appendix D.

Evaluation Metrics. We report Mean Reciprocal Rank (MRR) and Hits at N (Hits@N) to evaluate the link prediction performance of query triples in the test KG for each dataset. The evaluations consider both head and tail prediction. For a fair comparison with baselines, following their settings [Teru *et al.*, 2020; Liu *et al.*, 2021], all results are approximated five times by ranking each query triple among 50 other randomly sampled candidate negative triples.

Implementation Details. Our model is implemented in PyTorch and DGL. For GraIL and INDIGO, we use the implementations publicly provided by the authors with their best configurations. For Asmp-KGE, the dimension is 32. For MaKEr, the dimensions for embeddings and feature representations are 32; we employ the GNN with 2 layers, and the dimension for GNN's hidden representation is 32. The batch size for meta-training is 64, and we use the Adam optimizer with a learning rate of 0.001. Before meta-training our model, we sample 10,000 tasks on the training KG for each dataset, and the details of task sampling can be found in Appendix E. During training, we randomly treat entities and relations as unseen with the ratio of $30\% \sim 80\%$ for each task.

5.2 Main Results

We report the link prediction results in Table 1, and show the detail results for different kinds of query triples (i.e., u_ent, u_rel and u_both) respectively. For GraIL and INDIGO, they can only handle unseen entities, so we leave results for u_rel and u_both blank. The results show that our proposed MaKEr achieves improvements over various baselines and has stable performance using different KGE methods. Most best results are given by MaKEr with RotatE, a sophisticated KGE model proposed in recent years, showing that our proposed MaKEr can output reasonable embeddings and make full use of various KGE methods. Specifically,

	MaKEr	-Meta	-RelFeat	-EntFeat	-GNN
MRR	**50.31**	41.29	49.00	49.62	38.83
Hits@1	**39.00**	29.89	37.06	37.88	27.36

Table 3: Ablation study of using MaKEr (TransE) on FB-Ext.

for u_ent test triples, compared with methods for KG inductive settings (i.e., GraIL and INDIGO), MaKEr averagely increases by 20.4% and 8.1% on MRR and Hits@10 among two datasets, and the numbers are 14.5% and 18.3% compared with Asmp-KGE. Moreover, by comparing the overall results for different kinds of test triples, we find that the performance for u_rel and u_both triples are worse than the performance for u_ent, which indicates that it is challenging to handle unseen relations in the test KG. Despite the difficulties, our proposed MaKEr obtains significant improvements on u_rel and u_both triples compared to baselines. More precisely, MaKEr averagely increases by 0.56 and 1.02 times for MRR and Hits@10 on u_rel triples and 2.11 and 1.82 times on u_both triples. Overall, the results show that our model is meta-trained to obtain the ability to extrapolate knowledge for unseen entities and relations and conduct link predictions for those unseen components.

5.3 Further Analysis

Ablation Study. We conduct several ablation studies to show the importance of different parts of our proposed model. Specifically, we train our model based on following four ablation settings: removing 1) the meta-learning setting (-*Meta*); 2) the relation feature representation (-*RelFeat*); 3) the entity feature representation (-*EntFeat*); 4) the GNN for knowledge extrapolation (-*GNN*). The details of conducting ablation studies above can be found in Appendix F. The results of ablation studies using MaKEr (TransE) on FB-Ext are shown in Table 3. The results show that all ablation settings cause performance to decrease, indicating the importance of these designs. Moreover, we observe that the meta-learning setting is essential for model performance, indicating the effectiveness of meta-training our model on sampled tasks that mimic the task on the test KG. We also find that the performance drops significantly after removing the GNN, which is reasonable since the information provided for unseen components just by relation or entity feature representation is limited.

Case Study for Unseen Entities. We visualize the entity embeddings for NELL-Ext produced by our proposed MaKEr and Asmp-KGE in Fig. 3. In this figure, we show different types of entities with different colors. The distributions of embeddings from MaKEr are more consistent with their corresponding types than the embeddings produced by Asmp-KGE. For Asmp-KGE, the embeddings of different entity types are mixed, while for MaKEr, the embeddings are mapped into different clusters. Furthermore, we also find that in our model, the embeddings for unseen entities can be clustered with seen entities in the same type. The clustering for entities from different types indicates that MaKEr can represent unseen entities with embeddings containing reasonable semantics and informative knowledge.
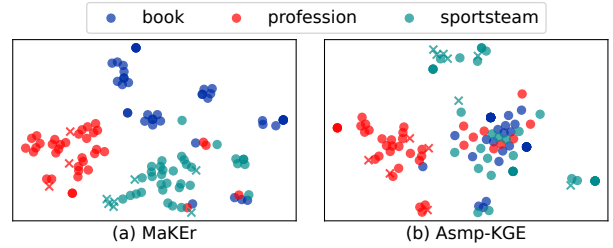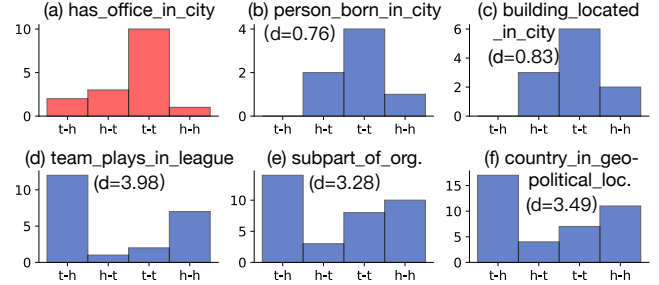

Figure 3: Visualization for t-SNE embeddings of MaKEr (TransE) and Asmp-KGE (TransE). The points using ○ are unseen entities and using × are seen entities. Their types are marked by colors.

Figure 4: The number of connected meta-relations in RPG for different relations. (a) is an unseen relation, and we show the euclidean distances between (a) and other relations in brackets based on embeddings produced by MaKEr (TransE).

Case Study for Unseen Relations. From Fig. 4, we find that for an unseen relation has_office_in_city in (a), its distribution of meta-relations is more similar to relations in (b)(c) which has similar semantics compared to relations in (d)(e)(f). Furthermore, three relations in the bottom half all represent the relationships about subordination, and they also have similar meta-relation distributions. These observations show that the connected meta-relations for rel-nodes in RPG are sensitive to the semantics of corresponding relations. We think this is also why such feature representations are effective for relations. Moreover, from the distances between the unseen relation in (a) and other relations, we find that the embeddings produced by MaKEr are reasonable where similar relations are close in the vector space, showing the effectiveness for embedding unseen relations in our proposed MaKEr.

6 Conclusion

In this paper, we formulate the problem of embedding unseen entities and relations for an emerging KG in the federated setting. To solve this problem, we resort to meta-learning and sample tasks on the training KG to mimic the link prediction task in the test KG. We design a GNN framework that can represent the features for unseen components and output embeddings for them. This model can generalize to unseen components of the test KG after being meta-trained on a set of tasks with unseen components. Our future work might consider designing explainable feature representations modules that can induce rules to make our model more human-friendly.

Acknowledgments

References

[Bordes *et al.*, 2013] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *NIPS*, 2013.

[Chen *et al.*, 2019] Mingyang Chen, Wen Zhang, Wei Zhang, Qiang Chen, and Huajun Chen. Meta relational learning for few-shot link prediction in knowledge graphs. In *EMNLP*, 2019.

[Chen *et al.*, 2021a] Jiajun Chen, Huarui He, Feng Wu, and Jie Wang. Topology-aware correlations between relations for inductive link prediction in knowledge graphs. In *AAAI*, pages 6271–6278. AAAI Press, 2021.

[Chen *et al.*, 2021b] Mingyang Chen, Wen Zhang, Zonggang Yuan, Yantao Jia, and Huajun Chen. Fede: Embedding knowledge graphs in federated setting. In *IJCKG*, pages 80–88. ACM, 2021.

[Finn *et al.*, 2017] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017.

[Hamaguchi *et al.*, 2017] Takuo Hamaguchi, Hidekazu Oiwa, Masashi Shimbo, and Yuji Matsumoto. Knowledge transfer for out-of-knowledge-base entities : A graph neural network approach. In *IJCAI*, 2017.

[Kipf and Welling, 2017] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[Liu *et al.*, 2021] Shuwen Liu, Bernardo Cuenca Grau, Ian Horrocks, and Egor V. Kostylev. Indigo: Gnn-based inductive knowledge graph completion using pair-wise encoding. In *NeurIPS*, 2021.

[McMahan *et al.*, 2017] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*, 2017.

[Peng *et al.*, 2021] Hao Peng, Haoran Li, Yangqiu Song, Vincent W. Zheng, and Jianxin Li. Differentially private federated knowledge graphs embedding. In *CIKM*, 2021.

[Santoro *et al.*, 2016] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *ICML*, 2016.

[Schlichtkrull *et al.*, 2018] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *ESWC*, 2018.

[Snell *et al.*, 2017] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *NIPS*, 2017.

[Sun *et al.*, 2019] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. In *ICLR*, 2019.

[Teru *et al.*, 2020] Komal Teru, Etienne Denis, and Will Hamilton. Inductive relation prediction by subgraph reasoning. In *ICML*, 2020.

[Toutanova *et al.*, 2015] Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. Representing text for joint embedding of text and knowledge bases. In *EMNLP*, 2015.

[Trouillon *et al.*, 2016] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *ICML*, 2016.

[Vashishth *et al.*, 2020] Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha P. Talukdar. Composition-based multi-relational graph convolutional networks. In *ICLR*, 2020.

[Vinyals *et al.*, 2016] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *NIPS*, 2016.

[Wang *et al.*, 2017] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743, 2017.

[Wang *et al.*, 2019] Peifeng Wang, Jialong Han, Chenliang Li, and Rong Pan. Logic attention based neighborhood aggregation for inductive knowledge graph embedding. In *AAAI*, 2019.

[Xiong *et al.*, 2017] Wenhan Xiong, Thien Hoang, and William Yang Wang. Deeppath: A reinforcement learning method for knowledge graph reasoning. In *EMNLP*, 2017.

[Xiong *et al.*, 2018] Wenhan Xiong, Mo Yu, Shiyu Chang, Xiaoxiao Guo, and William Yang Wang. One-shot relational learning for knowledge graphs. In *EMNLP*, 2018.

[Yang *et al.*, 2015] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. In *ICLR*, 2015.

[Yasunaga *et al.*, 2021] Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. QA-GNN: reasoning with language models and knowledge graphs for question answering. In *NAACL-HLT*, 2021.

[Zhang *et al.*, 2016] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. Collaborative knowledge base embedding for recommender systems. In *KDD*, 2016.

[Zhang *et al.*, 2021] Ke Zhang, Carl Yang, Xiaoxiao Li, Lichao Sun, and Siu Ming Yiu. Subgraph federated learning with missing neighbor generation. In *NeurIPS*, 2021.

[Zhu *et al.*, 2021] Qi Zhu, Yidan Xu, Haonan Wang, Chao Zhang, Jiawei Han, and Carl Yang. Transfer learning of graph neural networks with ego-graph information maximization. In *NeurIPS*, 2021.