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

Mingyang Chen¹*, Wen Zhang²*, Zhen Yao², Xiangnan Chen², Mengxiao Ding³, Fei Huang⁴ and Huajun Chen¹,⁵,⁶†

¹College of Computer Science and Technology, Zhejiang University
²School of Software Technology, Zhejiang University
³Huawei Technologies Co., Ltd.
⁴Alibaba Group
⁵ZJU-Hangzhou Global Scientific and Technological Innovation Center
⁶Alibaba-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

†Corresponding author.

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 represen-
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

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 $\mathcal{G}$.

1Source code is available at https://github.com/zjukg/MAKER.

1967
In the context of traditional knowledge graph completion [Bordes et al., 2013], to evaluate the link prediction ability of a model, there are two parts 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 \( \{ \mathcal{M}(h, r, e) \mid e \in \mathcal{E}, (h, r, e) \notin \mathcal{T}_{sup} \cup \mathcal{T}_{que}\} \). 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 \( \mathcal{G}^{te} = (\mathcal{E}^{te}, \mathcal{R}^{te}, \mathcal{T}^{te}, \mathcal{T}^{que}) \), 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 constraints: 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 \( \mathcal{T}^{sup} \), and triples in \( \mathcal{T}^{sup} \) 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 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 \( e \in \mathcal{E}^{i} \) and relations \( r \in \mathcal{R}^{i} \) 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 \( \mathcal{S} \) 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 \( \mathcal{T}^{sup} \) 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, \((r_1, t-h, r_2)\) 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:

\[
h_r = \frac{1}{|\mathcal{N}(r)|} \sum_{m \in \mathcal{N}(r)} h_m,
\]

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 \( t-h \) is the inverse of \( h-t \), and \( h-t \) and \( t-h \) are double-direction, thus out-going meta-relations can be indicated by in-going meta-relations. Moreover, \( h_m \in \mathcal{H}_M \) is the vector representation of a specific

1968
meta-relation, and \( H_{KL} = (h_{r_0, t_0}, h_{r_0, t_0}, h_{r_0, t_0}, h_{r_0, t_0}) \) 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:

\[
h_e = \frac{1}{|N(e)|} \sum_{r \in N(e)} W_{\text{dir}(r)}^{\text{ent}} h_r,
\]

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

**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:

\[
m^{t+1}_e = \sum_{(r, t) \in O(e)} W_{\text{out}}[h'_r, h'_t] + \sum_{(r, h) \in I(e)} W_{\text{in}}[h'_r, h'_h],
\]

where \( O(e) \) denotes a set of out-going relations and connected entities, \( I(e) \) denotes a set of in-going relations and connected entities; \( W_{\text{out}} \) and \( W_{\text{in}} \) are learnable parameters for out- and in-going relation-entity pairs at the \( l \)-th layer of GNN, \([;]\) denotes the concatenation of two vectors, and \( h'_r = h_r \) and \( h'_t = h_t \) are input relation and entity features. After message aggregation from neighbors for each entity, the entity representation for each \( e \) is updated by:

\[
h^{t+1}_e = \sigma \left( \frac{m^{t+1}_e}{|O(e)|} + W_{\text{self}} h_e \right),
\]

where \( W_{\text{self}} \) is a learnable parameter for self-loop updating for entities, and \( \sigma \) is an activation function. Besides entities, relations are also updated at each layer:

\[
h^{t+1}_r = \sigma \left( W_{\text{rel}} h_r \right).
\]

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 \( S_i \), after obtaining embeddings via previous modules on support triples \( T_{\text{sup}}^i \), we train our model to score ground-truth query triples \( T_{\text{que}}^i \) 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:

\[
\mathcal{L}(S^i) = \frac{1}{|T_{\text{que}}^i|} \sum_{(h, r, t) \in T_{\text{que}}^i} -\log \sigma \left( \gamma + s(h, r, t) \right) - \sum_{i=1}^n p(h_i', r, t_i') \log \sigma \left( -\gamma - s(h_i', r, t_i') \right),
\]

where \( s(h, r, t) \) is the score for \((h, r, t)\) using embeddings from our model based on \( T_{\text{sup}}^i \), \( \gamma \) is a fixed margin, \( n \) is the number of negative samples, \((h_i', r, t_i')\) is a negative sample by corrupting a head or tail entity. \( p(h_i', r, t_i') \) 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 \( \sum_i \mathcal{L}(S^i) \) among all tasks sampled from \( G^{\tau} \).

**5 Experiments**

In this section, we evaluate our proposed method MaKEr (for Meta-Learning Based Knowledge Extrapolation) on datasets derived from KG benchmarks, and compared it with baselines to show the effectiveness of this model.
5.1 Experimental Setting

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 respective KGE methods to calculate their embeddings based on seen components’ embeddings. For example, based on TransE, the embedding for an unseen entity \( t \) can be calculated by \( h + r = t \) 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,
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.

### 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
This work is funded by NSFC U19B2027/91846204, Zhejiang Provincial Natural Science Foundation of China (No. LGG22F030011), and Ningbo Natural Science Foundation (2021J190).

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 IJCAI, 2017.

[Hamaguchi et al., 2017] Taku 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 rotational relation 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, Sébastien Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In ICML, 2016.

[Vashisht et al., 2020] Shikhar Vashisht, 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, Thienn 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.