CogKR: Cognitive Graph for Multi-hop Knowledge Reasoning

Zhengxiao Du, Chang Zhou, Jiangchao Yao, Teng Tu, Letian Cheng, Hongxia Yang, Jingren Zhou, *Fellow, IEEE*, Jie Tang, *Fellow, IEEE*

Abstract—Inferring new facts from an existing knowledge graph with explainable reasoning processes is an important problem, known as knowledge graph (KG) reasoning. The problem is often formulated as finding the specific path that represents the query relation and connects the query entity and the correct answer. However, due to the limited expressiveness of individual paths, the majority of previous works failed to capture the complex subgraph structure in the graph. We propose CogKR that traverses the knowledge graph to conduct multi-hop reasoning. More specifically, motivated by the dual process theory from cognitive science, our framework is composed of an extension module and a reasoning module. By setting up a cognitive graph through iteratively coordinating the two modules, CogKR can cope with more complex reasoning scenarios in the form of subgraphs instead of individual paths. Experiments on three knowledge graph reasoning benchmarks demonstrate that CogKR achieves significant improvements in accuracy compared with previous methods while providing the explainable capacity. Moreover, we evaluate CogKR on the challenging one-shot link prediction task, exhibiting the superiority of the framework on accuracy and scalability compared to the state-of-the-arts.

Index Terms—cognitive graph, knowledge graph representation and reasoning, multi-hop reasoning.

1 Introduction

Knowledge graphs (KGs) such as Freebase [1], NELL [2], and YAGO [3] have been built in the last decade and nourished a wide range of downstream tasks, including relation extraction [4], question answering [5], dialogue systems [6] and recommender systems [7]. However, the incompleteness challenge that exists in most KGs seriously limits the accuracy of downstream tasks. This thus motivates a lot of works proposed for new fact inference in these years [8], [9], [10], [11].

Prior arts for KG reasoning can be roughly categorized into embedding-based methods [8], [9], [12], [13], [14] and path-based methods [10], [15], [16], [17], [18]. Embedding-based methods collaboratively learn the distributed representations of entities and relations according to existing links in KGs. However, they usually consider only direct links and lack the ability of multi-hop reasoning, which involves multiple entities and facts in the reasoning process [10]. Path-based methods instead leverage multi-hop path information of entity pairs to infer their underlying relations [15], [19], [20]. But due to their poor generalization to unseen paths, the performances have largely been surpassed by embedding-based methods.

More recent advancements in multi-hop reasoning combine path-based methods with distributed representations [21], [22] and reinforcement learning [10], [16], [17] to improve the generalization ability. These methods have gained remarkable performance in many benchmarks. However, the expressiveness of these path-based methods is still limited to paths, which could only represent a small subset of first-order logical formulas [23]. A more powerful form to encode multi-hop relations is the subgraph of the KG, which can handle more complex logical formulas. Unfortunately, it is still challenging to search and reason over exclusive subgraphs of the KGs efficiently. To bridge the gap, we may get some heuristics from the dual-process theory [24], [25], [26], [27], one of the dominant theories in cognitive science that studies thoughts and reasoning processes. Specifically, according to the dual process theory, the reasoning system of human beings consists of two distinct processes, one to retrieve relevant information intuitively (System 1) and the other to reason over the collected information via a controllable, sequential, and logical reasoning process (System 2). Such two systems are also related to the capacity-limited working memory [28]. System 1 updates the working memory with the retrieved content, while System 2 operates on the content of the working memory [24].

Inspired by the two-System structure of the dual process theory, we propose a novel framework CogKR to efficiently conduct multi-hop KG reasoning over subgraphs. The core of the proposed CogKR is the cognitive graph [29], a subgraph of the original KG iteratively expanded in the reasoning process, which resembles the working memory in human brains. Our framework combines two iterative processes, one to expand the subgraph with relevant entities and edges from the neighborhood, and the other to conduct relational reasoning based on the subgraph. The two processes, which resemble System 1 and System 2 in dual process theory, are iterated to search and reason over subgraphs of the KG. Moreover, the model is fully differentiable, such that the model can be trained efficiently via the

- *Zhengxiao Du, Teng Tu, and Letian Cheng are with the Department of Computer Science and Technology, Tsinghua University. E-mail: {zxdu20,tut19,chenglt19}@mails.tsinghua.edu.cn*
- *Chang Zhou, Jiangchao Yao, Hongxia Yang, and Jingren Zhou are with the DAMO Academy, Alibaba Group. E-mail: {ericzhou.zc,jiangchao.yjc,yang.yhx,jingren.zhou}@alibaba-inc.com*
- *Jie Tang is with the Department of Computer Science and Technology, Tsinghua University. E-mail: jietang@tsinghua.edu.cn. Jie Tang is the corresponding author.*

Manuscript received March 14, 2021

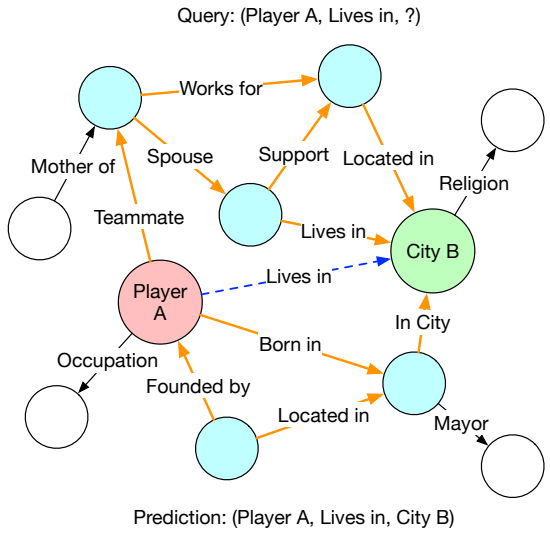


Fig. 1. An example of multi-hop KG reasoning. The aim is to find the entity that has the relation "Lives in" with the entity Player A. Blue circles and orange, bold lines represent the bridge entities and relations used in the reasoning process. By reasoning about the complex subgraph structure around Player A, we can find the correct answer City B.

stochastic gradient descent methods. Our contributions are summarized as follows:

- We propose CogKR, a novel framework for multi-hop KG reasoning based on the cognitive graph structure inspired by human cognition.
- We show that CogKR could model the complex structure in a subgraph, improving the multi-hop reasoning ability while preserving the explainability simultaneously in knowledge graph completion.
- We evaluate CogKR on the challenging task of one-shot link prediction in KGs. Our model achieves improvements on accuracy and scalability over the one-shot baseline.

2 Related Work

2.1 Knowledge Graph Embedding

Embedding methods for knowledge graphs have been extensively studied for KG completion. From a general perspective, entities and relations are represented as continuous vectors in latent space, and various scoring functions are defined for a fact (e_s, r, e_o). RESCAL [30] is one of the earlier works that model multi-relational data with vector representations. Following that, more advanced scoring functions have been proposed, such as vector difference [8], [12], vector product [9], [31], convolution [13], [32], and tensor operation [14], [33]. Although these embedding approaches have achieved impressive results on several KG completion benchmarks, they have been shown to suffer from cascading errors when modeling multi-hop relations [11], [34], which are indispensable for more complex reasoning tasks. Besides, since these methods all operate on latent space, their predictions are not interpretable.

Furthermore, these embedding methods usually assume enough training facts for all relations. One-shot link prediction on KGs has been proposed by [35] to handle queries about a new relation type with only one training instance. They propose a similarity metric based on graph convolutional network [36] and multi-step matching to compute scores for all the candidate facts. However, their model requires forward pass through neural networks for every candidate, which is computationally expensive or even intractable for large-scale KGs. Their method has been further extended to handle few-shot learning (more than one training instance is given) via aggregation network [37] or optimization-based meta learning [38]. Another line of research is to augment the KG embedding methods with the ability to handle unseen entities or relations by leveraging text descriptions [39], [40], [41].

2.2 Knowledge Graph Reasoning

Learning symbolic logic rules has been the mainstream to automatically infer new knowledge in its early days [42]. In statistical relational learning, machine learning is combined with symbolic rules to handle uncertainty [19], [20]. The Path-Ranking Algorithm [15] uses a random walk with restart mechanism to obtain paths between two entities and perform supervised classification of relations according to discrete path features. Various neural methods to learn first-order logical rules have also been proposed recently [23], [43], [44]. Although logical formulas are easy to explain, they have largely been superseded by distributed vector representations due to the poor generalization ability.

To overcome the limit, many works [10], [11], [16], [17] have proposed approaches that explicitly encode multi-step paths with deep learning, denoted as path-based methods. Such methods benefit from both the generalization of distributed representations and the explainability of logical rules. Both Chain-of-Reasoning [21] and Compositional Reasoning [22] infer underlying relations of two entities with neural networks taking multi-step paths found by random walk as input. Recently, DeepPath [10] uses RL-based agents combined with pre-trained KG embeddings to find better paths for reasoning than those found by random walks. MINERVA [16] reformulates the problem of predicting the missing entity in a new fact as a sequential decision problem of walking from one entity to reach the answer. MultihopKG [18] augments MINERVA with reward reshaping and action dropout to overcome false-negative supervision and spurious paths. M-Walk [17] uses Monte Carlo Tree Search to solve the reward sparsity problem. DIVA [11] unifies path-finding and path-reasoning with variational inference. DIVINE [45] uses generative adversarial imitation learning [46] to learn reasoning policies and reward functions self-adaptively through imitating the demonstrations sampled from KGs. Compared with previous KG reasoning methods, our proposed CogKR bases reasoning on subgraphs that can capture the interaction of multiple paths.

2.3 Graph Neural Networks for Relational Learning

Graph neural networks (GNN) [47], [48] are a class of neural networks that model graphs and structure their computations accordingly. Recently, graph convolutional network (GCN) [36] and its multiple variants [49], [50] have become

the dominant methods in representation learning and semi-supervised learning on graphs [51], [52]. GNNs pretrained on large-scale unlabeled data via self-supervised learning can further improve the performance on downstream tasks [53], [54].

Although GNNs have been successfully applied to link prediction on graphs [55], most GNNs operate on homogeneous graphs, in which only one type of edge exists. On the contrary, in a typical KG, there are various relation types, i.e., types of edges. Even in works of GNN for heterogeneous graphs [56], the number of edge types is quite limited, far fewer than that in a typical KG. Therefore, it is not easy to directly apply GNNs to KG reasoning. To overcome the limit, R-GCN [57] and its variants [58], [59] modify the GCN structure to use different parameters for different relation types. These methods only apply GNNs to the computation of entity embeddings, while the predictions are based on scoring functions of embedding-based methods. Therefore, their methods can be considered as more complicated ones of embedding-based methods, and cannot provide explanations for the predictions. GraIL [60] reasons over local subgraph structures with GNNs to make inductive relation prediction between two entities. However, directly applying the method to knowledge graph completion can lead to infeasible time cost, since it needs to extract subgraphs for every candidate entity of a query. Furthermore, [61] proposes an algorithm to synthesize benchmarks to evaluate the logical generalization of GNNs for relational learning. DPMPN [62] proposes a two-GNN framework to encode both global and local graph structures coordinated by an attention module. Their approach mainly focuses on pruning message passing in GNN to improve scalability. Our module, which introduces dual-process theory to improve multi-hop reasoning, is conceptually simpler and provides clearer explanations of prediction.

3 Problem Formulation

A knowledge graph $\mathcal{G}$ is represented as $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$, where $\mathcal{E}$ and $\mathcal{R}$ denote the entity set and the relation set. $\mathcal{T}$ is a set of triples $\{(e_s, r, e_o)\} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ which denote the facts about the specific relation $r \in \mathcal{R}$ from $e_s \in \mathcal{E}$ to $e_o \in \mathcal{E}$. e_s is called the *head entity* and e_o is called the *tail entity*. We can treat $\mathcal{G}$ as a directed graph composing of entities as nodes and relations as different edge types. Then, the triple (e_s, r, e_o) represents a directed edge of type r from e_s to e_o. There may be multiple edges of different types between two nodes. Following [16], we augment $\mathcal{G}$ with the inverse link (e_o, r^{-1}, e_s) for any $(e_s, r, e_o) \in \mathcal{T}$ to enhance the connectivity.

For multi-hop knowledge graph reasoning, we target to leverage a reasoning model to predict the tail entity from a background KG $\mathcal{G}$ for each query $(e_s, \hat{r}, ?)$. The reasoning model should traverse the knowledge graph, collect relevant evidence, and predict the correct answer based on the evidence. The evidence might include multiple entities and relations, which require multi-hop reasoning ability. Figure 1 illustrates an example of this task. Generally, in the training phase, we use a set of triples $\mathcal{T}_{\text{train}}$ to train a reasoning model, so that for each $(e_s, \hat{r}, e_o)$ in $\mathcal{T}_{\text{train}}$ the predicting probability of e_o given the query $(e_s, \hat{r}, ?)$ is maximized. Then, in the test phase, a different set of triples $\mathcal{T}_{\text{test}}$ is used to evaluate the performance of the trained model. That is, given a query $(e_s, \hat{r}, ?)$, we use the trained model to output prediction and compare it with the groudtruth.

4 Approach

In this section, we describe the proposed model for multi-hop KG reasoning and the training algorithm, discuss the connection with previous methods, and analyze the complexity of the algorithm.

Our whole framework for multi-hop KG reasoning problem is shown in Figure 2. It takes a head entity e_s and a relation $\hat{r}$ as input and predicts the correct tail entity e_o via an explicit reasoning process. The problem is challenging, as in most cases, inferring unseen relationships usually involves complicated reasoning processes. We connect our study to the dual-process theory from cognitive science [25]. Accordingly, the proposed model combines two iterative processes: retrieving information from the original KG (System 1) and reasoning over collected information (System 2). The retrieved information and reasoning results are stored in a unique structure called *cognitive graph*. In the following, we will introduce the cognitive graph, System 1 and System 2 respectively.

4.1 Cognitive Graph

The cognitive graph G is a subgraph of $\mathcal{G}$ that contains entities and edges selected from $\mathcal{G}$ as relevant evidence and latent representations for its entities as the reasoning results. Formally, $\mathrm{G} = (V, E, \mathbf{X})$, where $V \subseteq \mathcal{E}$, $E \subseteq \mathcal{T}$ and $\mathbf{X} \in \mathbb{R}^{|V| \times d}$. Here $\mathbf{X}$ is the matrix of latent representations, whose each element $\mathbf{X}[e]$ for the entity e represents the semantic information of e in the reasoning process.

In the beginning, V only contains the initially given head entity e_s. At each step t, the V and E in G are expanded based on currently involved nodes by System 1, and then the representations $\mathbf{X}$ for the expanded V are updated by System 2. To trace the currently involved entities, an attention flow on the graph [63] is constructed. We define the attention distribution $\boldsymbol{a}_t$ at step t, a probability distribution over entities in $\mathcal{G}$, to represent the current focus. Given the query $(e_s, \hat{r}, ?)$, the initial attention $\boldsymbol{a}_0$ is focused on e_s, that is, $\boldsymbol{a}_0$ has 1 for e_s and 0 for other entities. Then after each expansion step, we compute $\boldsymbol{a}_{t+1}$ based on $\boldsymbol{a}_t$ to update the focus.

Compared with individual paths found by previous path-based reasoning methods, the cognitive graph is more expressive, since some complex reasoning processes may require the interaction among paths. For example, the path-finding method cannot distinguish two paths $e_s \xrightarrow{r_1} e \xrightarrow{r_2} e_o$ and $e_s \xrightarrow{r_1} e \xrightarrow{r_2} e_o$ that interact at the entity e, and two independent paths $e_s \xrightarrow{r_1} e_1 \xrightarrow{r_2} e_o$ and $e_s \xrightarrow{r_1} e_2 \xrightarrow{r_2} e_o$. On the contrary, in the cognitive graph, the information from multiple paths could interact at the "bridge nodes", such as e in the previous case, which improves the expressiveness of CogKR.

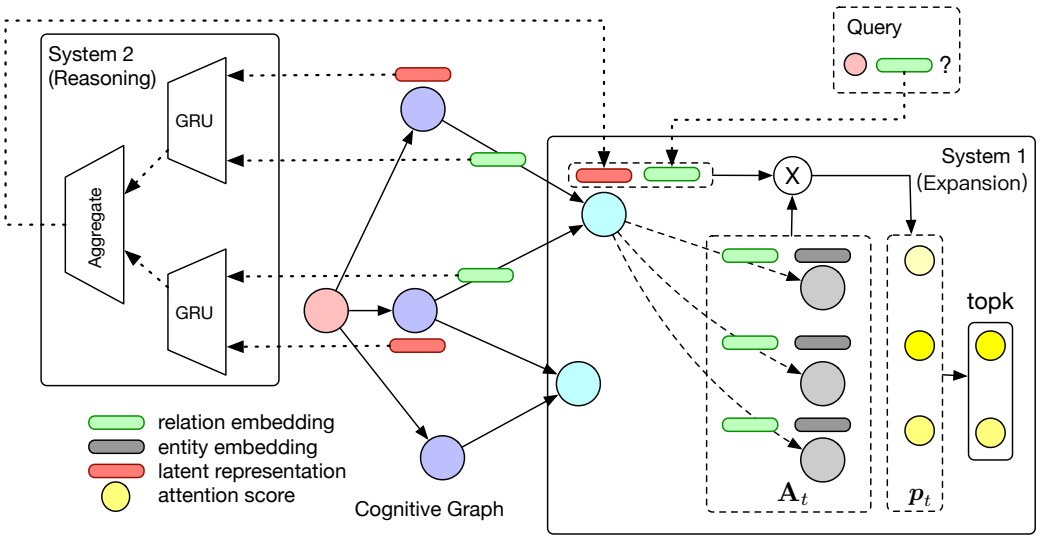


Fig. 2. Overview of CogKR. The core is the cognitive graph, which consists of nodes with latent representations and edges linking them. The expansion module (System 1) selects relevant edges from the neighborhood of current nodes (purple), according to their latent representations. Then, the reasoning module (System 2) updates the latent representations of newly attended nodes (green), via neighborhood aggregation and computes the attention values to measure their importance.

4.2 System 1 (Expansion)

The function of System 1 is to collect relevant evidence from $\mathcal{G}$ for the subsequent reasoning. It iteratively leverages a progressive subgraph expansion to search the possible evidence of entities and relations. Specifically, at each step t, System 1 selects the involved entities $\mathcal{F}_{t-1}$ at the last step and expand G with part of $\mathcal{F}_{t-1}$'s outgoing edges and the corresponding nodes.

Given the attention at last step $\boldsymbol{a}_{t-1}$, $\mathcal{F}_{t-1}$ is a collection of entities with probability mass larger than zero, i.e. $\mathcal{F}_{t-1} = \{e|\boldsymbol{a}_{t-1}(e) > 0\}$. For each entity e_k in $\mathcal{F}_{t-1}$, the candidate set to expand G consists of the outgoing edges of e_k in $\mathcal{G}$, termed as $A_t(e_k) = \{(r,e)|(e_k,r,e) \in \mathcal{G}\}$. To avoid out-of-memory error during computation, we rank the edges in $A_t(e_k)$ according to the PageRank values of connected entities and cut the maximum number of edges in $A_t(e_k)$ by a threshold η. To provide the agent with the option of staying at e_k, we add a self-loop link to $A_t(e_k)$. After obtaining $A_t(e_k)$, we can build the candidate matrix $\mathbf{A}_t(e_k) \in \mathbb{R}^{|A_t(e_k)|\times 3d}$ by stacking the embeddings of all edges in $A_t(e_k)$. The embedding of an edge (r,e) is the concatenation of the entity embedding $\boldsymbol{v}_e$, the relation embedding $\boldsymbol{v}_r$, and the entity's latent representation $\mathbf{X}[e]$ (filled with 0 if $e \notin V$). Based on the neighborhood features $\mathbf{A}_t$ of e_k and e_k's own representation $(\mathbf{X}[e_k], \boldsymbol{v}_{e_k})$, as well as the query relation embedding $\boldsymbol{v}_{\hat{r}}$, the probabilities of selecting edges are computed as

$$\begin{aligned} \boldsymbol{s}_t(e_k) &= \sigma\left(\mathbf{A}_t(e_k)\mathbf{W}_1\right) \cdot \sigma\left(\mathbf{W}_2[\mathbf{X}[e_k] \oplus \boldsymbol{v}_{e_k} \oplus \boldsymbol{v}_{\hat{r}}]\right) \\ \boldsymbol{p}_t(e_k) &= a_{t-1}(e_k)\,\mathrm{Softmax}(\boldsymbol{s}_t(e_k)) \end{aligned} \tag{1}$$

where $\mathbf{W}_1 \in \mathbb{R}^{3d\times d}, \mathbf{W}_2 \in \mathbb{R}^{d\times 3d}$ are parameters, and $\oplus$ denotes concatenation.

The concatenation of the probability vectors $\boldsymbol{p}_t(e)$ for all the entities e_k in $\mathcal{F}_{t-1}$, denoted as $\boldsymbol{p}_t$, represent a probability distribution over the outgoing edges of entities in $\mathcal{F}_{t-1}$. From all of the outgoing edges, we select n edges with the largest probability values, denoted as E_t. n is the action budget, i.e., the maximum number of selected edges at each step. After that, we add edges in E_t to E and add entities that are connected with selected edges but never visited before to V, which implements the cognitive graph expansion. Note that, there are two differences compared with path-finding methods. Firstly, when $n > 1$, we select multiple edges at each step, so that the search subgraph can form a directed acyclic graph (DAG) rather than a single path. Secondly, the attention flow mechanism is deterministic and differentiable, which means that we could train the whole module end-to-end without reinforcement learning.

4.3 System 2 (Reasoning)

After the expansion by System 1, the relational reasoning can be conducted by System 2 over the extended cognitive graph. To be more specific, we need System 2 to provide the following two operations: (1) update the latent representations of entities that are visited via the newly selected edges; (2) adjust the attention distribution over entities according to selected edges.

For the first one, out of consideration about the generalization ability [21], [22], we adopt the deep learning module instead of previous rule-based reasoning modules [43], [44]. Concretely, considering the flexible structure of the cognitive graph, we compute the node representations with graph neural networks [47], which capture the dependence of graphs via message passing [64] between the nodes of graphs:

$$\begin{aligned} \mathbf{X}[e] &= \boldsymbol{U}(e, \boldsymbol{m}_e) \\ \boldsymbol{m}_e &= \sum_{(e_k,r_k)\in E_e} \boldsymbol{M}(e_k, r_k, e), \end{aligned} \tag{2}$$

where $E_e = \{(e_k,r_k)|(e_k,r_k,e) \in E\}$ is the ingoing edges of e in G, $\boldsymbol{M}(e_k,r_k,e)$ is the message vector passed from e_k to e via relation r_k, and $\boldsymbol{U}(\cdot,\cdot)$ is the node update function. Unlike conventional GNNs where the current layer of representations are computed from the previous layer(s), all

the representations are computed in the same layer but in a sequential way. In this sense, the way we update node representations is similar to the homogeneous AC-GCN in [65], in which the GNN shares the same parameters across layers.

By extending the framework of [65] to directed graph with more than one edge type, we can study the logical expressiveness of System 2 in terms of first-order predicate logic. Given a query $(e_s, \hat{r}, ?)$, the rule to look for the correct answer can be expressed as a logical node classifier, which is given by a formula $\phi(x)$ with exactly one free variable x. For example,

$$\phi(x) := \exists x''\left(\exists x'\left(r_1(x', x'') \wedge x' = e_s\right) \wedge r_3(x'', x)\right), \quad (3)$$

where $\wedge$ is the logical AND operator and $r(e_1, e_2)$ represents that $(e_1, r, e_2) \in \mathcal{T}$. The free variable x is evaluated over the entity set $\mathcal{E}$ and any entity e that makes the body part evaluate to true is the correct answer. With the definition of logical node classifier in hand, we can give the theorem about the logical expressiveness of System 2:

Definition 1. The set Φ of logical node classifiers is defined as:

- $(x = e_s) \in \Phi$ for $r \in \mathcal{R}$.
- $\phi_1(x) \wedge \phi_2(x) \wedge \cdots \wedge \phi_n(x) \in \Phi$ for $\phi_1(x), \phi_2(x), \cdots, \phi_n(x) \in \Phi$
- $\exists x'(r(x', x) \wedge \phi(x')) \text{ for } r \in \mathcal{R}, \phi(x') \in \Phi$

Theorem 1. Every logical node classifier $\phi(x) \in \Phi$ can be captured by our System 2.

The proof of the theorem is in Appendix A.

Considering the success of RNN models in path-based methods [16], [22], we use GRU [66], a variant of RNN with a gating mechanism as the message function

$$\boldsymbol{M}(e_k, r_k, e) = \mathrm{GRU}(\mathbf{X}[e_k], \boldsymbol{v}_{r_k} \oplus \boldsymbol{v}_e), \quad (4)$$

where $\mathrm{GRU}(\mathbf{X}[e_k], [\boldsymbol{v}_{r_k}; \boldsymbol{v}_e])$ is the one-step update of GRU with the input of the previous hidden state $\mathbf{X}[e_k]$ and the relation-entity feature $\boldsymbol{v}_{r_k} \oplus \boldsymbol{v}_e$. We do not use more advanced RNN models such as LSTM [67] that improve long-term memory since reasoning paths are usually short.

The node update function is simply the average function

$$\boldsymbol{U}(e, \boldsymbol{m}_e) = \frac{1}{|E_e|}\boldsymbol{m}_e \quad (5)$$

It can be considered as an extension of the Path-RNN [22], with the ability to encode not only a single path but a complex subgraph.

For the second operation, we directly aggregate the probabilistic values of edges pointing to the entities to compute a new attention distribution over entities, which is formulated as follows

$$\begin{aligned} \tilde{a}_t(e) &= \sum_{(e',r,e)\in E_t} \boldsymbol{p}_t(e', r, e) \\ a_t(e) &= \frac{\tilde{a}_t(e)}{\sum_{e'\in\mathcal{E}} \tilde{a}_t(e')} \end{aligned} \quad (6)$$

Note that the normalization is necessary since the sum of the probabilities of E_t might be smaller than 1 due to previous top-n selection.

Algorithm 1 Graph-based KG Reasoning Algorithm

Require: Query $(e_s, \hat{r}, ?)$; A background KG $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$

$V \leftarrow \{e_s\}$, $E \leftarrow \varnothing$
$\boldsymbol{a}_0 \leftarrow \text{one\_hot}(e_s)$, $\mathbf{X}[e_s] \leftarrow \mathbf{0}$
for $t = 1$ **to** T **do**
  $\mathcal{F}_{t-1} \leftarrow \{e | \boldsymbol{a}_{t-1}(e) > 0\}$
  for entity e_k **in** $\mathcal{F}_{t-1}$ **do**
    $\boldsymbol{s}_t(e_k) \leftarrow \text{score}(A_t(e_k); \boldsymbol{v}_{e_k}, \mathbf{X}[e_k], \boldsymbol{v}_{\hat{r}})$
    $\boldsymbol{p}_t(e_k) \leftarrow \boldsymbol{a}_{t-1}(e_k)\text{Softmax}(\boldsymbol{s}_t(e_k))$
  end for
  $\boldsymbol{p}_t \leftarrow \cup_{e_k\in\mathcal{F}_{t-1}} \boldsymbol{p}_t(e_k)$, $E_t \leftarrow \cup_{e_k\in\mathcal{F}_{t-1}} A_t(e_k)$
  if $t \neq T$ **then**
    $E_t \leftarrow \text{Top-k}(E_t, \boldsymbol{p}_t, n)$
  end if
  $\boldsymbol{a}_t \leftarrow \mathbf{0}^{|\mathcal{E}|}$, $V_t \leftarrow \varnothing$
  for (e', r, e) **in** E_t **do**
    $\boldsymbol{a}_t(e) \leftarrow \boldsymbol{a}_t(e) + \boldsymbol{p}_t(e', r, e)$
    $V_t \leftarrow V_t \cup \{e'\}$
  end for
  if $t \neq T$ **then**
    $V \leftarrow V \cup V_t$,$E \leftarrow E \cup E_t$
    Update $\mathbf{X}[V_t]$ with Equations (2) and (4)
    $\boldsymbol{a}_t \leftarrow \boldsymbol{a}_t / \sum_{e'} \boldsymbol{a}_t(e')$
  end if
end for
return $\text{argmax}_{e\in V} \boldsymbol{a}_T(e)$

4.4 Prediction and Optimization

In the previous sections, we describe how CogKR builds the cognitive graph and conducts relational reasoning via deep learning. In this section, we will introduce how the final answer is predicted and what objective is used to optimize the model.

Generally, the prediction of the tail entity e_o in the query could be straightforward. At the last step T, we directly compute $\boldsymbol{a}_T$ based on $\boldsymbol{a}_{T-1}$ as described in Sections 4.2 and 4.3 but without selecting n edges since the expansion of the cognitive graph is no need any more. Then, the $\boldsymbol{a}_T$ is considered as the prediction distribution and we maximize the probability of e_o being predicted as the correct entity. A cross-entropy loss fashion $\ell(\boldsymbol{a}_T, e_o) = -\log a_T(e_o)$ can be applied. However, one critical issue is that the prediction distribution $\boldsymbol{a}_T$ does not have full support over the entity set $\mathcal{E}$. As we only compute the probabilities for entities in the T-hop neighborhood of e_s, the others are all $a_T(e) = 0$. Moreover, since only top n edges are selected in the iterative process and the local nature of the attention mechanism, $\boldsymbol{a}_T$ could be more sparse than the T-hop neighborhood. Therefore, it is quite possible that $a_T(e_o) = 0$ in the initial or the intermediate training procedure. To avoid such ill-posed cases to the log function, we assume the probability of e_o is a small number ϵ close to zero when $a(e_o) = 0$. So the final objective is defined as:

$$\ell(\boldsymbol{a}_T, e_o) = \begin{cases} -\log a_T(e_o) & a_T(e_o) > 0 \\ -\log(1 + \epsilon - \sum_e a_T(e)) & a_T(e_o) = 0 \end{cases} \quad (7)$$

We use the stochastic gradient descent to approximate the gradient descent on the full dataset in Equation (7). The complete algorithm is summarized in Algorithm 1.

4.5 Discussion

4.5.1 The difference with path-finding methods

As illustrated in Section 1, our framework has two main differences with path-finding methods. Firstly, our framework can search the knowledge graph in the form of a DAG instead of a single path. Secondly, our model can be trained end-to-end without reinforcement learning. There is also the third difference. According to [16], [18], beam search is often used in the evaluation stage of path-finding methods to improve sample efficiency. The model rolls out n paths for a query, in which n edges that have the largest accumulated probability values are selected greedily at each step. The way our model explores the KG following attention is similar to beam search, but the explored entities and edges are integrated into the cognitive graph, instead of individual paths. This enables the interaction of multiple paths and avoids inconsistency between training and evaluation.

4.5.2 Explainability of cognitive graph

After predicting the correct answer e_o for a query $(e_s, \hat{r}, ?)$, we can extract a graphical explanation for the prediction from the cognitive graph. The explanation is generated by extracting the enclosing subgraph between the head entity e_s and the predicted answer e_o in the cognitive graph G. The enclosing subgraph of G between entities e_s and e_o is defined as the subgraph of G that consists of entities on a (directed) path between e_s and e_o and edges between the entities. From Section 4.3 we know that the subgraph between e_s and e_o can decide e_o's latent representation. Therefore it can provide graphical explanation for the prediction. In Section 5.4 we show examples of graphical explanations generated by CogKR.

4.5.3 Complexity Analysis

To complete a query $(e_s, \hat{r}, ?)$, embedding-based methods need to enumerate the whole entity set, so it takes $O(|\mathcal{E}|)$ time for every query. For a large KG containing millions of entities, this is highly expensive especially combined with complex scoring functions.

CogKR, on the other hand, utilizes the local structure of the KG to reduce the time complexity. For System 1, at each step, at most n entities are visited and for each entity we compute scores of at most η outgoing edges. Given the iteration times T, it thus takes $O(Tn\eta)$ time to complete graph expansion. Similarly for System 2, at each step, at most n nodes' latent representations are updated. To update each entity, we need to aggregate the messages from at most $|E|$ edges. Therefore, it takes $O(Tn|E|)$ time for latent representation computation. Since at each step at most n edges are added to the cognitive graph, we have $|E| \leq Tn$. Overall, CogKR takes at most $O(Tn\eta + T^2n^2)$ time. Given predefined T, n and η, the maximum time is a constant that does not depend on the entity number, and thus is scalable to handle the large KGs.

5 Experiment

In this section, we provide empirical results to validate the effectiveness of CogKR on multi-hop KG reasoning. Firstly, we evaluate CogKR on three knowledge graph completion benchmarks, to show that CogKR could outperform both embedding-based and path-based baselines. Secondly, we evaluate CogKR on the one-shot link prediction task proposed recently, to show CogKR's superiority in accuracy and scalability on this challenging setting. Thirdly, we analyze the performance of CogKR from different perspectives, including the ablation study, the reasoning ability for different hops, the convergence speed, and the influence of hyperparameters. Finally, we conduct a case study over the graphical explanations extracted from cognitive graphs to show that CogKR could utilize the subgraph structure to conduct relational reasoning and provide explanations.[1]

5.1 Knowledge Graph Completion

5.1.1 Datasets

TABLE 1
Statistics of datasets in knowledge graph completion.

Dataset	#Ent.	#Rel	#Train	#Valid	#Test
FB15K-237	14,541	237	272,115	17,535	20,466
WN18RR	40,943	11	86,835	3,034	3,134
YAGO3-10	123,182	37	1,079,040	5,000	5,000

We use two public KG datasets, FB15K-237 [68] and WN18RR [13] for the KG completion task. These two datasets are sampled from FB15K [8] and WN18 [8] with inverse relations causing test set leakage removed. Therefore, they are more challenging and realistic. The original datasets, FB15K and WN18, have been shown to suffer from test set leakage due to inverse relations from the training set being present in the test set [13]. To further validate that CogKR can tackle large-scale KGs, we also evaluate CogKR on YAGO3-10, a subset of of YAGO3 [69] which consists of entities that have a minimum of 10 relations each. For all the datasets, we use the original data split, which is widely used in most papers. The dataset statistics are shown in Table 1.

5.1.2 Experimental Setting

We add inverse relations into the training set as data augmentation. But these inverse relations are not added into the validation or test set on FB15K-237 and WN18RR, to avoid inconsistency in evaluation data with other papers [16], [18], [45]. During training, given a training triple $(e_s, \hat{r}, e_o)$, the corresponding edge and its inverse are masked for the model to avoid leakage. On FB15K-237, following [62], all the edges from e_s to e_o are also masked, forcing the model to learn a composite reasoning pattern rather than a single-hop pattern.

During the evaluation, for a test triple $(e_s, \hat{r}, e_o)$, all the correct answers for the query $(e_s, \hat{r}, ?)$ except e_o are removed from the prediction. This is called the filtered setting [8] and has been used by most papers, since it can provide a more reliable performance metric in the presence of multiple correct triples. We use Hits@1,3,10 and mean reciprocal rank (MRR), which are standard metrics for KB completion, as evaluation metrics.

1. The code can be downloaded from https://github.com/THUDM/CogKR

TABLE 2
KG reasoning results for FB15K-237 and WN18RR. Results of [♠] are taken from [16]. Results of [♡]are taken from the corresponding papers. Other results are obtained by running the official implementations.

Model	FB15K-237				WN18RR			
	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR
DistMult [♠]	27.5	41.7	56.8	37.0	41.0	44.1	47.5	43.3
ComplEx [♠]	30.3	43.4	57.2	39.4	38.2	43.3	48.0	41.5
R-GCN	19.9	32.0	47.5	29.1	7.6	11.8	18.4	11.1
ConvE	31.3	45.7	60.0	41.0	40.3	45.2	51.9	43.8
RotatE	33.1	48.2	**63.7**	43.4	44.6	52.1	60.3	49.9
TuckER	33.6	46.7	60.7	42.7	45.5	50.2	53.5	48.5
NeuralLP [♠]	16.6	24.8	34.8	22.7	37.6	46.8	**65.7**	46.3
MINERVA[♡]	21.7	32.9	45.6	29.3	41.3	45.6	51.3	44.8
MultihopKG[♡]	32.9	-	54.4	39.3	43.7	-	54.2	47.2
M-Walk[♡]	16.5	24.3	-	23.2	41.4	44.5	-	43.7
DIVINE[♡]	22.3	33.1	-	29.6	-	-	-	-
CogKR	**36.9**	**49.2**	61.4	**44.9**	**48.4**	**54.3**	60.6	**52.3**

TABLE 3
KG reasoning results for YAGO3-10. Results of ♦ are taken from [14]. Other results are obtained by running the official implementation.

Model	H@1	H@3	H@10	MRR
DistMult[♦]	24	38	54	34
ComplEx[♦]	26	40	55	36
ConvE[♦]	35	49	62	44
RotatE[♦]	40.2	55.0	67.0	49.5
TuckER	41.6	55.3	**67.1**	50.5
MultihopKG	40.5	53.7	63.4	48.5
CogKR	**47.7**	**57.3**	64.4	**53.6**

We compare CogKR with various state-of-the-arts. For embedding-based methods, we compare with TransE [8], DistMult [31], ComplEx [9], R-GCN [57], ConvE [13], RotatE [14], and TuckER [33]. Among the embedding-based methods, R-GCN also applies GNN to conduct relational reasoning over knowledge graphs. For path-based methods, we compare with NeuralLP [44], MINERVA [16], MultihopKG [18], M-Walk [17] and DIVINE [45].

5.1.3 Hyperparameters

The dimensions of embeddings and node representations are set to 100 and 200 respectively. The entity and relation embeddings are randomly initialized. As two key hyperparameters, maximum step T is set to 4 on FB15K-237 and YAGO3-10 and 5 on WN18RR and action budget n is set to 64 on FB15K-237 and YAGO3-10 and 32 on WN18RR. The maximum neighbor number η is set to 256. The activation function is LeakyReLU in all the layers.

We use the ADAM optimization algorithm for model training with a learning rate of 1e-3. We also add L2 regularization with a weight decay of 0.0001. The batch size is 28 on FB15K-237 and YAGO3-10 and 64 on WN18RR. The maximum training step is 30,000 on WN18RR and YAGO3-10 and 80,000 on FB15K-237. We use the MRR on the validation set as the criteria to pick the best model checkpoint.

5.1.4 Result Analysis

Table 2 reports the KG completion performance on FB15K-237 and WN18RR datasets. As shown on both datasets, CogKR produces consistent improvements over previous methods in terms of Hits@1, 3 and MRR. Path-based methods, including NeuralLP, MINERVA, MultihopKG, M-Walk, and DIVINE, generally perform worse than embedding-based methods. Our model could outperform all the embedding-based and path-based methods. This demonstrates the effectiveness of CogKR with the structure of System 1 and System 2. Besides, the improvements of our model on FB15K-237 are particularly substantial in all the evaluated metrics, with relative improvements of 9.8% on Hits@1. We speculate that it can be related to the fact that on FB15K-237 there are no edges in the training set directly linking any pair of head and tail in the validation (or test) set. Therefore, the multi-hop reasoning ability is particularly important on this dataset, which is exactly what our model is good at. On the contrary, for a large portion (35.2%) of facts in the validation and test set of WN18RR, the head and the tail are linked by some edge (of a different relation) in the train set. Therefore, the multi-hop reasoning ability may not be so important, which limits the improvements we can achieve. We will further analyze the multi-hop reasoning performance on WN18RR in Section 5.3.2.

Table 3 shows the experimental results on YAGO3-10. The dataset is not used in previous path-based methods. We select MultihopKG, which has the best performance among path-based methods in the previous experiment, as the path-based baseline. CogKR can also outperform baselines on YAGO3-10, in terms of all the metrics except Hits@10.

5.2 One-shot Link Prediction on KGs

To further validate the effectiveness of CogKR, we evaluate the model on a more challenging task proposed recently: one-shot link prediction on KGs [35]. In this task, we need to perform link prediction for relation types with only one training fact per relation. Since the information about each relation type is limited and vague, it poses more challenges to the model about its reasoning ability.

5.2.1 Datasets

Most benchmarks for knowledge graph completion, like FB15K-237 and WN18RR used in our previous experiment, are subsets of real-world KGs but do not contain sufficient

TABLE 4
Statistics of datasets in one-shot link prediction.

Dataset	#Ent.	#Rel	# Triples	# Tasks
NELL-One	68,545	358	181,109	67
Wiki-One	4,838,244	822	5,859,240	183

relation types to train and evaluate one-shot learning algorithms [35]. Therefore, we use the newly proposed NELL-One and Wiki-One datasets in [35], both of which are created from real-world KGs (NELL [2] and Wikidata [70] respectively) for one-shot relational learning. The datasets are created with a similar process: relations with less than 500 but more than 50 triples are selected as one-shot tasks and randomly divided into training, validation, and testing relations. The dataset statistics are shown in Table 4. Following [35], we use Hits@1,5,10 and MRR as evaluation metrics. Note that the Wiki-One dataset is an order of magnitude larger than any other benchmark datasets in terms of the number of entities and relations. In practice, we found that the Wiki-One dataset suffers from sparsity and non-connectivity in the backend KG. In the test set, 15.8% of the entity pairs are not connected at all, and the distances of the other 25.5% pairs are no less than 5. For these 41.3% pairs, we do not have any reasonable paths to infer their relations. To better evaluate the reasoning ability, we remove evaluation facts whose entity pairs' distances are no less than 5 in Wiki-One.

5.2.2 Experimental Setting

For each dataset, we have a set of training relations $\mathcal{R}_{\text{train}}$. For each relation $r \in \mathcal{R}_{\text{train}}$, or $\mathcal{R}_{\text{valid}}$, $\mathcal{R}_{\text{test}}$, we have the corresponding fact set $D_r = \{(e_s, e_o)|(e_s, r, e_o) \in \mathcal{G}\}$. To generate a training instance, we first sample a relation $\hat{r}$ from $\mathcal{R}_{\text{train}}$ and then sample the training triple $(e_{s\hat{r}}, \hat{r}, e_{o\hat{r}})$ and the query triple $(e_s, \hat{r}, e_o)$ from D_r.

For evaluation, another two sets of relations $\mathcal{R}_{\text{valid}}$ and $\mathcal{R}_{\text{test}}$ are given as the validation and test relations. For each relation $\hat{r}$ in $\mathcal{R}_{\text{valid}}$ or $\mathcal{R}_{\text{test}}$, we select one triple $(e_s, \hat{r}, e_o)$ as the support fact and use all the other pairs in $D_{\hat{r}}$ for evaluation. For cross-validation, it is guaranteed that relations in $\mathcal{R}_{\text{test}}$ will not appear in $\mathcal{R}_{\text{train}}$ or $\mathcal{R}_{\text{valid}}$. Note that in [35], for a query $(e_s, \hat{r}, e_o)$, they only rank entities in a filtered candidate set $\mathcal{C}_{e_s,\hat{r}}$. We follow this setting and filter entities $e \notin \mathcal{C}_{e_s,\hat{r}}$ from our prediction.

We compare with various baselines. For embedding-based methods, we compare with TransE, DistMult, and ComplEx. More advanced embedding methods such as ConvE failed to scale to Wiki-One, as reported in [35]. We also include MINERVA as the path-based baseline and GMatching [35] as the one-shot baseline.

5.2.3 Hyperparameters

To deploy CogKR on this task, we combine it with the one-shot model GMatching proposed in [35]. GMatching learns to match the training triple $(e_{s\hat{r}}, \hat{r}, e_{o\hat{r}})$ and a candidate triple $(e_s, \hat{r}, e_o)$ with GCN [36]. Instead, we use the same architecture to map the training triple to a vector representation of $\hat{r}$:

$$\begin{aligned}\boldsymbol{\omega}_{\hat{r}} &= \sigma(\mathbf{W}_r(\boldsymbol{\omega}_{e_{s\hat{r}}} \oplus \boldsymbol{\omega}_{e_{o\hat{r}}}) + \boldsymbol{b}_r)\\ \boldsymbol{\omega}_e &= \sigma(\mathbf{W}_s\boldsymbol{v}_e + \boldsymbol{b}_s + \mathbf{W}_c \cdot \frac{1}{|\mathcal{N}_e|}\sum_{(r_k,e_k)\in\mathcal{N}_e} \boldsymbol{v}_{r_k} \oplus \boldsymbol{v}_{e_k}),\end{aligned} \quad (8)$$

which is then passed to CogKR as $\boldsymbol{v}_{\hat{r}}$ to answer the query $(e_s, \hat{r}, ?)$, as described in Section 4.

On NELL-One, the dimensions of embeddings and node representations are set to 100 and 200 respectively. On Wiki-One, the dimensions are set to 50 and 100 due to the extremely large scale of the dataset. On both datasets, we use $T = 3$ and $n = 16$.

We use the ADAM optimization algorithm for model training with learning rates of 1e-3 for parameters of CogKR and 1e-4 for parameters of the entity encoder and the relation layer. We also add L2 regularization with a weight decay of 0.0001. The batch size is 160 on NELL-One and 24 on Wiki-One. The maximum training step is 10000 on NELL-One and 30000 on Wiki-One. We use the MRR on the validation set as the criteria to pick the best model checkpoint.

5.2.4 Result Analysis

Table 5 reports the one-shot link prediction performance on NELL-One and Wiki-One datasets. On NELL-One, CogKR produces consistent improvements over previous works in all evaluation metrics. The absolute improvement is 7.4% for Hits@1 and 7.8% for MRR. On Wiki-One, our model also achieves improvements on Hits@1 and MRR, but does not show advantages against the embedding-based method GMatching on Hits@5, 10. We can also observe similar patterns in the performance of MINERVA. We speculate that when the information about the target is insufficient, multi-hop reasoning methods, including CogKR and MINERVA, cannot conduct effective reasoning and lose their advantages. Nevertheless, CogKR can always perform better than MINERVA, confirming its improvements over path-based methods. We also find that the traditional embedding methods do not perform well on Wiki-One, possibly due to the contrast between its extremely large scale and the scarcity of training data in the one-shot setting. Another observation is that the path-based method MINERVA can generally outperform embedding-based methods in the one-shot setting, contrary to its disadvantage on the fully-supervised setting in Section 5.1. We speculate that knowledge graph reasoning methods, including MINERVA and CogKR, are more suitable for one-shot link prediction on KGs.

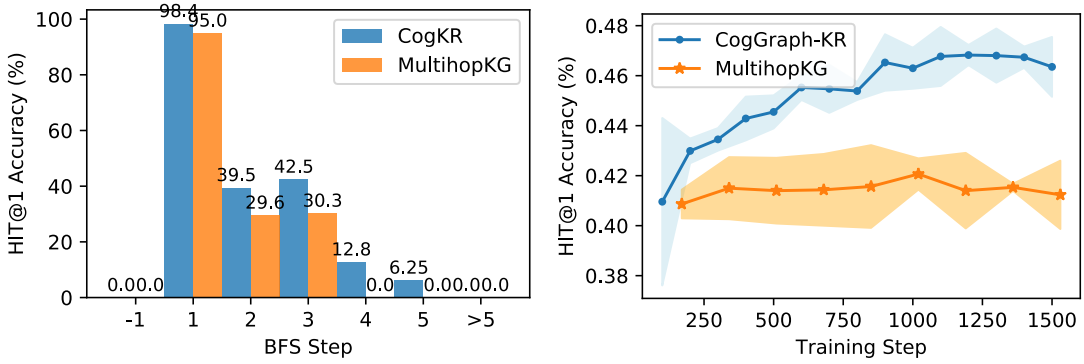
5.2.5 Inference Time

A disadvantage of embedding-based methods is that each candidate entity e for the query $(e_s, \hat{r}, ?)$ needs to be evaluated with a scoring function. This is especially slow for GMatching [35], whose scoring function is a complex neural network that compares the candidate triple $(e_s, \hat{r}, e)$ with the training triple $(e_{s\hat{r}}, \hat{r}, e_{o\hat{r}})$. This is almost infeasible for large-scale KGs that contain millions of entities.

To validate CogKR's advantage on time complexity, we compare the inference time of CogKR against DistMult

TABLE 5
One-shot KG reasoning results for NELL and Wikidata.

Model	NELL-One H@1	NELL-One H@5	NELL-One H@10	NELL-One MRR	Wiki-One H@1	Wiki-One H@5	Wiki-One H@10	Wiki-One MRR
TransE	4.4	14.9	29.6	11.1	2.5	4.3	5.2	3.5
ComplEx	9.4	19.4	23.9	14.1	4.0	9.2	12.1	6.9
DistMult	12.3	23.1	26.9	16.3	1.9	7.0	10.1	4.8
MINERVA	16.0	26.1	29.0	20.6	21.4	25.6	26.6	23.3
GMatching	13.3	22.6	29.6	18.3	17.0	**27.3**	**33.5**	22.6
CogKR	**23.4**	**34.1**	**36.9**	**28.4**	**23.4**	26.5	27.0	**24.7**



(a) Multi-hop Analysis (b) Convergence Analysis

Fig. 3. Performance analysis of CogKR.

TABLE 6
Inference time of different methods on Wiki-One test set (sec / samples).

#Candidates	CogKR	GMatching	DistMult
5,000	3.0×10^{-3}	4.6×10^{-2}	1.6×10^{-3}
4,838,243	3.0×10^{-3}	4.2×10	5.1×10^{-1}

TABLE 7
Ablation study on FB15k-237 benchmark.

Model	H@1	H@3	H@10	MRR
CogKR	**36.9**	**49.2**	**61.4**	**44.9**
PG-KR	3.3	5.7	9.2	5.3
Path-KR	10.4	13.6	16.1	12.3
– System 2	27.5	40.3	53.6	35.9

as the traditional baseline and GMatching as the neural baseline. All the models are implemented in PyTorch and evaluated on a single RTX 2080 GPU. The results with both the truncated candidate set (no more than 5000) and the full entity set (4,838,244) are reported in Table 6. As we can see, when the candidate number is limited to 5000, the running time of GMatching and DistMult is comparable to that of CogKR. DistMult is even faster than CogKR due to its simple scoring function. However, with the full entity set as candidates, the running time of GMatching and DistMult increases proportional to the number of candidates, whereas the running time of CogKR remains the same. This shows the efficiency of CogKR when the entity number of KGs becomes very large.

5.3 Quantitative Analysis

5.3.1 Ablation Analysis

We conduct the ablation study to analyze the contributions of different components in CogKR. The results are shown in Table 7. PG-KR has the same architecture with CogKR but uses the stochastic policy to select edges and policy gradient for training. We observed that basically the policy gradient does not work for CogKR, possibly because the search space of graphs is even larger than that of paths. Path-KR is the variation of CogKR with $n = 1$, which means that the cognitive graph is reduced to a single path. According to the results, this hurts the performance greatly, leading to a 71.8% relative drop in Hits@1. Besides, we also created a variation of CogKR without System 2, which means no latent representations for nodes. As can be seen, removing System 2 has a smaller effect, but still causes a 25.5% relative loss of Hits@1. In summary, different components of CogKR all contribute to the final improvement in the KG reasoning.

5.3.2 Multi-hop Analysis

To evaluate the multi-hop reasoning ability, in Figure 3a, we show the Hits@1 of CogKR on WN18RR, categorized by the shortest path lengths from the query entity to the correct answer. The baseline is MultihopKG [18], which has shown the best performance among path-based methods on WN18RR. We observe that CogKR outperforms such a strong baseline by 9.9-12.8% for facts that require 2, 3 or 4 reasoning steps. For facts that require only one-step reasoning, both methods can achieve accuracy close to 100%, and

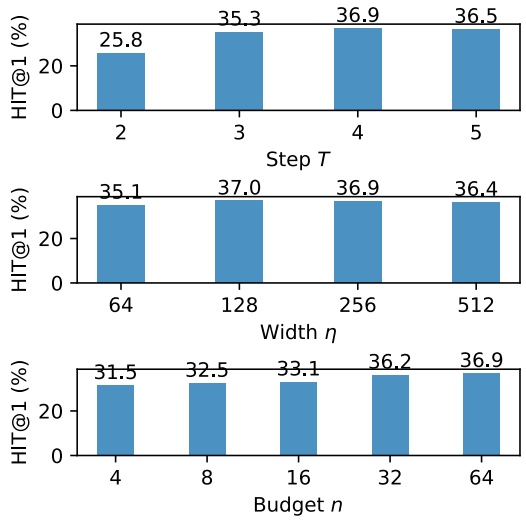


Fig. 4. Hyperparameter sensitivity of CogKR on FB15K-237.

the improvement of our method is marginal. For facts with longer distances, both methods can hardly make effective predictions. It indicates the long reasoning paths are highly uncertain and hard to distill useful information. Above all, CogKR can improve the multi-hop reasoning ability for the appropriate length.

5.3.3 Convergence Analysis

To evaluate the convergence speed of CogKR, we plot the model's performance on the validation set of WN18RR at different training points in Figure 3b. The baseline is MultihopKG, which uses reinforcement learning with reward reshaping and action dropout to train path-finding agents [18] and has the official implementation to trace the training process. We show the average of 3 runs and the 95% confidence interval. From Figure 3b, compared with the path-finding method, the performance of our model can improve faster while the final performance is also better. This confirms our motivation of using differentiable training to efficiently search over the subgraph space.

5.3.4 Hyperparameter Sensitivity

Finally, we analyze the influence of hyperparameters on the performance of CogKR. We mainly focus on three hyperparameters, the maximum step T, the maximum number of edges η, and the action budget n. The performance of CogKR under different choices of T, η, and n on FB15K-237 is shown in Figure 4. T is the maximum length of reasoning. When T is small, e.g., $T = 2$, the expressiveness of CogKR is limited to the short reasoning paths, leading to a significant drop of accuracy. However, overlong reasoning paths, e.g., $T = 5$, are highly uncertain, also leading to a slight drop of accuracy. η is the maximum number of edges for entities in $\mathcal{G}$. CogKR is not sensitive to the choice of η, achieving high accuracy even when $\eta = 64$. n is the maximum number of selected edges at each step. According to Figure 4, larger n, which indicates the topological structure of cognitive graphs could be more complex, generally improves the performance. However, when n is too large, the noise is also introduced into the cognitive graph, which may affect the convergence and lead to the improvement constrained.

TABLE 8
Statistics of the cognitive graphs on the test set. Avg. Size indicates the average number of nodes in the cognitive graphs. Avg. Recall indicates the fraction of cognitive graphs that contain the correct answers.

Dataset	Avg. Size	Avg. Recall
FB15K-237	88.3	67.6
YAGO3-10	76.6	68.2

5.4 Qualitative Analysis

In Figure 5, we give three examples of graphical explanations built by CogKR. According to Section 4.5.2, each graphical explanation is built by extracting all the paths from the head entity $\hat{r}$ in the query to the predicted tail entity in the cognitive graph. From the examples, we can conclude: 1) the expressiveness of cognitive graphs is more powerful than individual paths. Graphs in three cases contain complex interaction of multiple paths that cannot be captured by path-finding methods. 2) the cognitive graph can provide graphical explanations for the prediction of CogKR. For example, in case (a), to predict the missing genre of the movie Wonder Boys, CogKR looks at the movie About Schmidt that shares two common genres with Wonder Boys. In case (c), CogKR predicts that Michael Biehn acted in the movie Terminator 2, because Michael also acted in the movie The Terminator, which is its prequel and shares the same genre with Terminator 2. The three cases are all consistent with human cognition. We also show the statistics of the complete cognitive graphs on the test set in Table 8. We can observe that in a large portion of incorrect predictions by CogKR the correct answers are missing in the cognitive graphs.

6 Conclusion

This paper presents CogKR, a novel framework to tackle the multi-hop KG reasoning problem. Under the inspiration of the dual process theory in cognitive science, we organize the reasoning process with a cognitive graph, achieving more powerful reasoning ability than previous path-based methods and end-to-end training following gradient methods. Experimental results on both knowledge graph completion and one-shot link prediction benchmarks demonstrate the superiority of our framework. For further improvements, we find that reasoning-based methods often get stuck to the non-connectivity of KGs. Therefore, we will explore to improve System 1 by allowing the non-connected node expansion.

Appendix
Proof of Theorem 1

Let $\phi(x)$ be a logical formula in Φ. $\mathrm{sub}(\phi) = \phi_1, \phi_2, \cdots, \phi_L$ is an enumeration of the sub-formulas of ϕ such that if ϕ_k is a sub-formula of ϕ_l then $k \leq l$. Note that the formula $x = e_s$ has no sub-formula and must exist in the enumeration. We always represent it with ϕ_1.

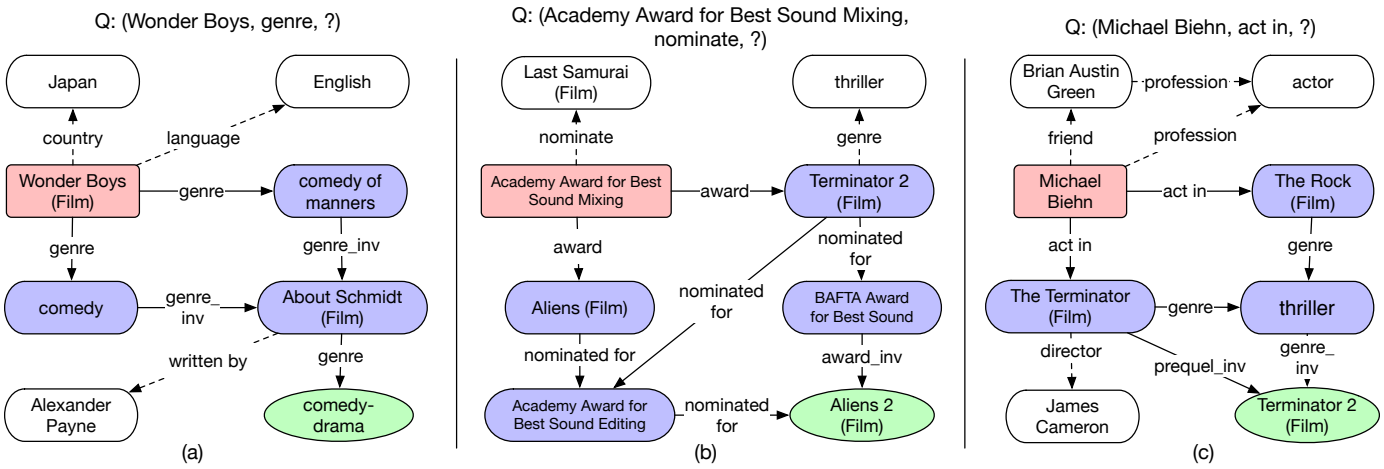


Fig. 5. Case Study: different forms of graphical explanations in the experiments. Red rectangles denote query entities, green eclipses denote final answers, and purple capsules denote intermediate nodes. White eclipses and dashed lines denote irrelevant entities and edges.

We will build the System 2 such that component l of entity e's latent representation $\mathbf{X}[e]$ gets a value 1 if and only if the sub-formula ϕ_l is satisfied for e. In this way, the last component of $\mathbf{X}[e]$ after the reasoning process will get a value 1 if and only if e satisfies ϕ since $\phi_l = \phi$. Therefore our System 2 captures the formula ϕ.

We choose the message and update functions in our System 2 as:

$$\boldsymbol{M}(e_k, r_k, e) = \sigma(\mathbf{A}\mathbf{X}[e_k] + \mathbf{B}\boldsymbol{v}_{r_k} + \boldsymbol{b}) \tag{9}$$

$$\boldsymbol{U}(e, \boldsymbol{m}_e) = \sigma(\mathbf{C}\sigma(\boldsymbol{m}_e) + \boldsymbol{c}) \tag{10}$$

where $\mathbf{A}, \mathbf{B}, \mathbf{C} \in \mathbb{R}^{L\times L}$ and $\boldsymbol{b}, \boldsymbol{c} \in \mathbb{R}^L$ are defined next. σ is the activation function, which is the truncated ReLU activation, i.e., $\sigma(x) = \min(\max(x, 0), 1)$. $\boldsymbol{v}_{r_k}$ is the one-hot encoding of relation type r_k. The entries of the l-th rows of $\mathbf{A}, \mathbf{B}, \mathbf{C}$ and $\boldsymbol{b}, \boldsymbol{c}$ are defined according to ϕ_l as:

- If $\phi_l = \phi_{j_1} \wedge \phi_{j_2} \wedge \cdots \wedge \phi_{j_n}$, then $C_{lj_1} = C_{lj_2} = \cdots = C_{lj_n} = 1$ and $c_l = -n + 1$.
- If $\phi_l(x) = \exists x'(r(x', x) \wedge \phi_j(x'))$, then $B_{lr_k} = A_{lj} = 1$, $b_l = -1$, and $C_{ll} = 1$, $c_l = 0$.

and all other values in the l-th entries of $\mathbf{A}, \mathbf{B}, \mathbf{C}$ and $\boldsymbol{b}, \boldsymbol{c}$ are 0. We skip the case when $\phi_l = (x = e_s)$, which is only possible for $l = 1$ and $x = e_s$. We initialize the latent representations $\mathbf{X}[e_s]$ of e_s with the first component as 1 and the other components as 0.

Next we prove that for every $\phi_l \in \mathrm{sub}(\phi)$ and every entity e in G, $(\mathbf{X}[e])_l = 1$ if and only if e satisfies ϕ. We prove it by induction on the entities in G in topological ordering. The first entity of topological sort can only be e_s. Given that $(\mathbf{X}[e_s])_1 = 1$ and other components of $\mathbf{X}[e_s]$ are 0, we know that the property holds for e_s.

Now we assume that for every entity e_k before e in the topological ordering, the property holds. Then we need to prove that for e the property also holds. This is quite straightforward given our definition of $\mathbf{A}, \mathbf{B}, \mathbf{C}$ and $\boldsymbol{b}, \boldsymbol{c}$. If $\phi_l(x) = \exists x'(r(x', x) \wedge \phi_j(x'))$, the l-th component of $\mathbf{X}[e]$ is computed as:

$$\begin{aligned}(\boldsymbol{M}(e_k, r_k, e))_l &= \sigma((\mathbf{X}[e_k])_j + \mathbb{I}(r_k = r) - 1)\\ (\mathbf{X}[e])_l &= \sigma(\sigma(\sum_{(e_k, r_k)\in E_e} (\boldsymbol{M}(e_k, r_k, e))_l))\end{aligned} \tag{11}$$

By induction hypothesis we know that $(\mathbf{X}[e_k])_j = 1$ if and only if e_k satisfies ϕ_j. $(\mathbf{X}[e])_l = \sigma(|\{e_k|(e_k, r_k) \in E_e$ and $r_k = r$ and e_k satisfies $\phi_k\}|)$. Therefore $(\mathbf{X}[e])_l = 1$ if and only if e satisfies ϕ_l. We can also observe that $(\sigma(\boldsymbol{m}_e))_l = 1$ if and only if e satisfies ϕ_l.

If $\phi_l = \phi_{j_1} \wedge \phi_{j_2} \wedge \cdots, \wedge \phi_{j_n}$, without loss of generality, we assume that each sub-formula is in the form of $\exists x'(r(x', x) \wedge \phi(x'))$. The l-th component of $\mathbf{X}[e]$ is computed as:

$$(\mathbf{X}[e])_l = \sigma\left(\sum_{i=1}^{n} (\sigma(\boldsymbol{m}_e))_{j_i} - n + 1\right) \tag{12}$$

Since $(\sigma(\boldsymbol{m}_e))_{j_i} = 1$ if and only if e satisfies ϕ_{j_i}, we have that $(\mathbf{X}[e])_l = 1$ if and only if e satisfies $\phi_{j1}, \phi_{j2}, \cdots, \phi_{jn}$.

Acknowledgments

The work is supported by the NSFC for Distinguished Young Scholar (61825602), and a research fund supported by Alibaba.

References

[1] K. D. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: a collaboratively created graph database for structuring human knowledge," in *SIGMOD*, 2008, pp. 1247–1250.

[2] T. M. Mitchell, W. W. Cohen, E. R. H. Jr., P. P. Talukdar, J. Betteridge, A. Carlson, B. D. Mishra, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. A. Platanios, A. Ritter, M. Samadi, B. Settles, R. C. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling, "Never-ending learning," in *AAAI*, 2015, pp. 2302–2310.

[3] F. M. Suchanek, G. Kasneci, and G. Weikum, "Yago: a core of semantic knowledge," in *WWW*, 2007, pp. 697–706.

[4] G. Wang, W. Zhang, R. Wang, Y. Zhou, X. Chen, W. Zhang, H. Zhu, and H. Chen, "Label-free distant supervision for relation extraction via knowledge graph embedding," in *EMNLP*, 2018, pp. 2246–2255.

[5] S. Mohammed, P. Shi, and J. Lin, "Strong baselines for simple question answering over knowledge graphs with and without neural networks," in *NAACL-HLT*, 2018, pp. 291–296.

[6] T. Young, E. Cambria, I. Chaturvedi, H. Zhou, S. Biswas, and M. Huang, "Augmenting end-to-end dialogue systems with commonsense knowledge," in *AAAI*, 2018, pp. 4970–4977.

[7] X. Wang, D. Wang, C. Xu, X. He, Y. Cao, and T. Chua, "Explainable reasoning over knowledge graphs for recommendation," in *AAAI*, 2019, pp. 5329–5336.

[8] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *NIPS*, 2013, pp. 2787–2795.
[9] T. Trouillon, C. R. Dance, É. Gaussier, J. Welbl, S. Riedel, and G. Bouchard, "Knowledge graph completion via complex tensor factorization," *JMLR*, vol. 18, pp. 130:1–130:38, 2017.
[10] W. Xiong, T. Hoang, and W. Y. Wang, "Deeppath: A reinforcement learning method for knowledge graph reasoning," in *EMNLP*, 2017, pp. 564–573.
[11] W. Chen, W. Xiong, X. Yan, and W. Y. Wang, "Variational knowledge graph reasoning," in *NAACL-HLT*, 2018, pp. 1823–1832.
[12] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion," in *AAAI*, 2015, pp. 2181–2187.
[13] T. Dettmers, P. Minervini, P. Stenetorp, and S. Riedel, "Convolutional 2d knowledge graph embeddings," in *AAAI*, 2018, pp. 1811–1818.
[14] Z. Sun, Z. Deng, J. Nie, and J. Tang, "Rotate: Knowledge graph embedding by relational rotation in complex space," in *ICLR*, 2019.
[15] N. Lao, T. M. Mitchell, and W. W. Cohen, "Random walk inference and learning in A large scale knowledge base," in *EMNLP 2011*, 2011, pp. 529–539.
[16] R. Das, S. Dhuliawala, M. Zaheer, L. Vilnis, I. Durugkar, A. Krishnamurthy, A. Smola, and A. McCallum, "Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning," in *ICLR*, 2018.
[17] Y. Shen, J. Chen, P.-S. Huang, Y. Guo, and J. Gao, "M-walk: Learning to walk over graphs using monte carlo tree search," in *NIPS*, 2018, pp. 6786–6797.
[18] X. V. Lin, R. Socher, and C. Xiong, "Multi-hop knowledge graph reasoning with reward shaping," in *EMNLP*, 2018, pp. 3243–3253.
[19] S. Muggleton, "Inductive logic programming," *New Generation Comput.*, vol. 8, no. 4, pp. 295–318, 1991.
[20] S. Kok and P. M. Domingos, "Statistical predicate invention," in *ICML*, 2007, pp. 433–440.
[21] A. McCallum, A. Neelakantan, R. Das, and D. Belanger, "Chains of reasoning over entities, relations, and text using recurrent neural networks," in *EACL*, 2017, pp. 132–141.
[22] A. Neelakantan, B. Roth, and A. McCallum, "Compositional vector space models for knowledge base inference," in *AAAI Spring Symposia*, 2015.
[23] Y. Yang and L. Song, "Learn to explain efficiently via neural logic inductive learning," in *ICLR*, 2020.
[24] J. Evans, "Dual-processing accounts of reasoning, judgment, and social cognition," *Annual review of psychology*, vol. 59, pp. 255–78, 02 2008.
[25] S. A. Sloman, "The empirical case for two systems of reasoning," *Psychological Bulletin*, vol. 119, pp. 3–22, 1996.
[26] J. Evans, "Heuristic and analytic processes in reasoning," *British Journal of Psychology*, vol. 75, pp. 451 – 468, 04 1984.
[27] J. S. Evans, "In two minds: dual-process accounts of reasoning," *Trends in Cognitive Sciences*, vol. 7, no. 10, pp. 454 – 459, 2003.
[28] A. Baddeley, "Working memory," *Science*, vol. 255, no. 5044, pp. 556–559, 1992.
[29] M. Ding, C. Zhou, Q. Chen, H. Yang, and J. Tang, "Cognitive graph for multi-hop reading comprehension at scale," in *ACL 2019*, 2019, pp. 2694–2703.
[30] M. Nickel, V. Tresp, and H. Kriegel, "A three-way model for collective learning on multi-relational data," in *ICML*, 2011, pp. 809–816.
[31] B. Yang, W. Yih, X. He, J. Gao, and L. Deng, "Embedding entities and relations for learning and inference in knowledge bases," in *ICLR 2015*, 2015.
[32] I. Balazevic, C. Allen, and T. M. Hospedales, "Hypernetwork knowledge graph embeddings," in *ICANN 2019*, 2019, pp. 553–565.
[33] ——, "Tucker: Tensor factorization for knowledge graph completion," *arXiv preprint arXiv:1901.09590*, 2019.
[34] K. Guu, J. Miller, and P. Liang, "Traversing knowledge graphs in vector space," in *EMNLP*, 2015, pp. 318–327.
[35] W. Xiong, M. Yu, S. Chang, X. Guo, and W. Y. Wang, "One-shot relational learning for knowledge graphs," in *EMNLP*, 2018, pp. 1980–1990.
[36] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2017.
[37] C. Zhang, H. Yao, C. Huang, M. Jiang, Z. Li, and N. V. Chawla, "Few-shot knowledge graph completion," *CoRR*, vol. abs/1911.11298, 2019.
[38] M. Chen, W. Zhang, W. Zhang, Q. Chen, and H. Chen, "Meta relational learning for few-shot link prediction in knowledge graphs," in *EMNLP-IJCNLP 2019*, 2019, pp. 4216–4225.
[39] H. Shah, J. Villmow, A. Ulges, U. Schwanecke, and F. Shafait, "An open-world extension to knowledge graph completion models," in *AAAI*, 2019, pp. 3044–3051.
[40] B. Shi and T. Weninger, "Open-world knowledge graph completion," in *AAAI*, 2018, pp. 1957–1964.
[41] R. Xie, Z. Liu, J. Jia, H. Luan, and M. Sun, "Representation learning of knowledge graphs with entity descriptions," in *AAAI*, 2016, pp. 2659–2665.
[42] J. McCarthy, "Programs with common sense," in *Semantic Information Processing*. MIT Press, 1968, pp. 403–418.
[43] Y. Shen, P. Huang, M. Chang, and J. Gao, "Modeling large-scale structured relationships with shared memory for knowledge base completion," in *Rep4NLP@ACL*, 2017, pp. 57–68.
[44] F. Yang, Z. Yang, and W. W. Cohen, "Differentiable learning of logical rules for knowledge base reasoning," in *NIPS*, 2017, pp. 2316–2325.
[45] R. Li and X. Cheng, "DIVINE: A generative adversarial imitation learning framework for knowledge graph reasoning," in *EMNLP-IJCNLP*, 2019, pp. 2642–2651.
[46] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *NIPS*, 2016, pp. 4565–4573.
[47] F. Scarselli, S. L. Yong, M. Gori, M. Hagenbuchner, A. C. Tsoi, and M. Maggini, "Graph neural networks for ranking web pages," in *2005 IEEE / WIC / ACM International Conference on Web Intelligence (WI 2005), 19-22 September 2005, Compiegne, France*, 2005, pp. 666–672.
[48] Y. Li, D. Tarlow, M. Brockschmidt, and R. S. Zemel, "Gated graph sequence neural networks," in *ICLR*, 2016.
[49] W. L. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *NIPS*, 2017, pp. 1024–1034.
[50] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *ICLR*, 2018.
[51] W. L. Hamilton, R. Ying, and J. Leskovec, "Representation learning on graphs: Methods and applications," *IEEE Data Eng. Bull.*, vol. 40, no. 3, pp. 52–74, 2017.
[52] Z. Yang, W. W. Cohen, and R. Salakhutdinov, "Revisiting semi-supervised learning with graph embeddings," in *ICML*, 2016, pp. 40–48.
[53] Q. Jiezhong, C. Qibin, D. Yuxiao, Z. Jing, Y. Hongxia, D. Ming, W. Kuansan, and T. Jie, "Gcc: Graph contrastive coding for graph neural network pre-training," in *KDD*, 2020.
[54] W. Hu, B. Liu, J. Gomes, M. Zitnik, P. Liang, V. S. Pande, and J. Leskovec, "Strategies for pre-training graph neural networks," in *ICLR*, 2020.
[55] M. Zhang and Y. Chen, "Link prediction based on graph neural networks," in *NeurIPS*, 2018, pp. 5171–5181.
[56] Y. Cen, X. Zou, J. Zhang, H. Yang, J. Zhou, and J. Tang, "Representation learning for attributed multiplex heterogeneous network," in *KDD*, 2019, pp. 1358–1368.
[57] M. S. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *ESWC*, 2018, pp. 593–607.
[58] S. Vashishth, S. Sanyal, V. Nitin, and P. P. Talukdar, "Composition-based multi-relational graph convolutional networks," in *ICLR*, 2020.
[59] R. Ye, X. Li, Y. Fang, H. Zang, and M. Wang, "A vectorized relational graph convolutional network for multi-relational network alignment," in *IJCAI*, 2019, pp. 4135–4141.
[60] K. K. Teru and W. L. Hamilton, "Inductive relation prediction on knowledge graphs," *CoRR*, vol. abs/1911.06962, 2019.
[61] K. Sinha, S. Sodhani, J. Pineau, and W. L. Hamilton, "Evaluating logical generalization in graph neural networks," *CoRR*, vol. abs/2003.06560, 2020.
[62] X. Xu, W. Feng, Y. Jiang, X. Xie, Z. Sun, and Z.-H. Deng, "Dynamically pruned message passing networks for large-scale knowledge graph reasoning," in *ICLR 2020*, 2020.
[63] X. Xu, S. Zu, C. Gao, Y. Zhang, and W. Feng, "Modeling attention flow on graphs," *CoRR*, vol. abs/1811.00497, 2018.
[64] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *ICML*, 2017, pp. 1263–1272.

[65] P. Barceló, E. V. Kostylev, M. Monet, J. Pérez, J. L. Reutter, and J. P. Silva, "The logical expressiveness of graph neural networks," in *ICLR*, 2020.
[66] K. Cho, B. van Merrienboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *EMNLP*, 2014, pp. 1724–1734.
[67] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, 12 1997.
[68] K. Toutanova, D. Chen, P. Pantel, H. Poon, P. Choudhury, and M. Gamon, "Representing text for joint embedding of text and knowledge bases," in *EMNLP*, 2015, pp. 1499–1509.
[69] F. Mahdisoltani, J. Biega, and F. M. Suchanek, "YAGO3: A knowledge base from multilingual wikipedias," in *CIDR 2015*, 2015.
[70] D. Vrandecic and M. Krötzsch, "Wikidata: a free collaborative knowledgebase," *Commun. ACM*, vol. 57, no. 10, pp. 78–85, 2014.

Zhengxiao Du is a PhD student with the Department of Computer Science and Technology of Tsinghua University. His main research interests include deep learning, reinforcement learning and their applications in recommender systems. He has published papers on KDD, TKDE, and PKDD/ECML.

Chang Zhou is working as the Senior Algorithm Engineer in Alibaba Group. He received his Ph.D. degree from Peking University in 2017. His research interests include representation learning, graph mining and recommender systems. He has published over 20 papers on NeurIPS, KDD, WWW, AAAI, VLDB.

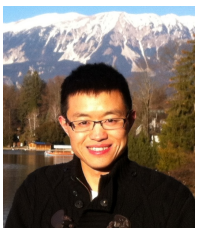
Jiangchao Yao is working as the Senior Algorithm Engineer in Alibaba Group. He got his dual Ph.D. degree under the supervision of Ya Zhang in Shanghai Jiao Tong University and Ivor W. Tsang in University of Technology Sydney in 2019. His research interests include deep representation learning and robust machine learning, and he has publised more than 10 papers on the Conferences like NeurIPS, ICML and AAAI *etc.*.

Teng Tu is a sophomore undergraduate student with the Department of Computer Science and Technology of Tsinghua University.

Letian Cheng is a sophomore undergraduate student with the Department of Computer Science and Technology of Tsinghua University.

Hongxia Yang is working as the Senior Staff Data Scientist and Director in Alibaba Group. She got her PhD degree in Statistics from Duke University in 2010. She has published over 30 papers and held 9 filed/to be filed US patents and is serving as the associate editor for Applied Stochastic Models in Business and Industry.

Jingren Zhou is working as the Senior Vice President in Alibaba Group. He holds a Ph.D. in Computer Science from Columbia University. Dr. Zhou has published dozens of papers in top conferences and journals in the fields of large-scale distributed systems, query processing and the optimization of distributed databases, and holds several patented inventions of key technologies in the industry. His current research directions are data processing methods based on large-scale distributed systems and machine learning algorithm platforms.



Jie Tang is a full professor with the Department of Computer Science and Technology of Tsinghua University. His main research interests include data mining algorithms and social network theories. He has been a visiting scholar with Cornell University, Chinese University of Hong Kong, Hong Kong University of Science and Technology, and Leuven University. He has published more than 200 research papers in major international journals and conferences including: KDD, IJCAI, AAAI, ICML, WWW, SIGIR, SIGMOD, ACL, Machine Learning Journal, TKDD, and TKDE.