Inductive Relation Prediction with Logical Reasoning Using Contrastive Representations

Yudai Pan[1,2] **Jun Liu**[1,2*] **Lingling Zhang**[1,3] **Tianzhe Zhao**[1]
Qika Lin[1] **Xin Hu**[1] and **Qianying Wang**[4]
[1]School of Computer Science and Technology, Xi'an Jiaotong University
[2]National Engineering Lab for Big Data Analytics
[3]Shaanxi Provincial Key Laboratory of Big Data Knowledge Engineering
[4]Lenovo Research, Beijing, China
{pyd418,qikalin,dr.huxin}@foxmail.com, {liukeen,zhanglling}@xjtu.edu.cn
ztz9712265887@stu.xjtu.edu.cn, wangqya@lenovo.com

Abstract

Relation prediction in knowledge graphs (KGs) aims at predicting missing relations in incomplete triples, whereas the dominant embedding paradigm has a restriction on handling unseen entities during testing. In the real-world scenario, the inductive setting is more common because entities in the training process are finite. Previous methods capture an inductive ability by implicit logic in KGs. However, it would be challenging to preciously acquire entity-independent relational semantics of compositional logic rules and to deal with the deficient supervision of logic caused by the scarcity of relational semantics. To this end, we propose a novel graph convolutional network (GCN)-based model LogCo with logical reasoning by contrastive representations. LogCo firstly extracts enclosing subgraphs and relational paths between two entities to supply the entity-independence. Then a contrastive strategy for relational path instances and the subgraph is proposed for the issue of deficient supervision. The contrastive representations are learned for a joint training regime. Finally, prediction results and logic rules for reasoning are attained. Comprehensive experiments on twelve inductive datasets show that LogCo achieves outstanding performance comparing with SOTA inductive baselines.

1 Introduction

Knowledge graphs (KGs) store plenty of facts by triples consisting of entities and relations. They have been widely used in different application scenarios, such as relation extraction (Hu et al., 2021), question answering (Abdelaziz et al., 2021) and information retrieval (Verlinden et al., 2021). Extracting triples from contexts consumes plenty of resources. Therefore, some dominant methods aim at the KG completion or relation prediction by learning representations of relations and entities, such as TransE(Bordes et al., 2013), RESCAL (Nickel et al., 2011), R-GCN (Schlichtkrull et al., 2018) and CompGCN (Vashishth et al., 2020).

However, the above methods predict relations assuming for a *transductive* setting, which means the entities are fixed during training and testing. In the application scenario, there will be new entities during testing. For the scenario in Figure 1(a), training and testing entities have no intersection, so the previous transductive methods will not accurately predict the relation `liveIn` between entities *Bill Gates* and *W.A.* in the test set without retraining the whole model. Thus, some studies like GraiL (Teru et al., 2020) focus more on models owning *inductive* ability, which can handle unseen entities by implicit first-order logic rules (Horn, 1951) fitting the cognition of human beings. For example, by the following logic rule $\boldsymbol{r_1}$:

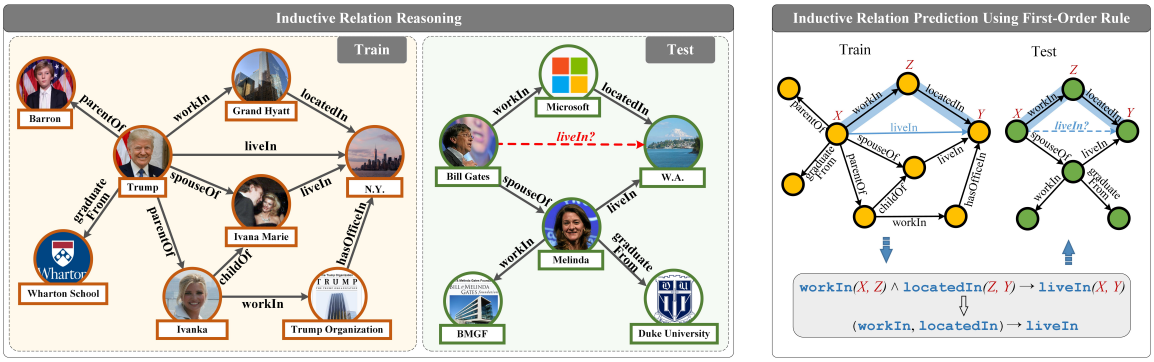
$$\texttt{workIn}(X, Z) \wedge \texttt{locatedIn}(Z, Y) \rightarrow \texttt{liveIn}(X, Y),$$

the relation `liveIn` in the test subgraph can be inferred. However, existing models majorly take advantage of the topological structure of entities and triples in KGs. There still remain two issues of mining logic in inductive relation prediction.

Firstly, inductive relation prediction has unseen entities in the test set, which requires the model to own **entity-independence** during reasoning. Although implicit logic rules in KGs provide inductive ability, previous methods (Mai et al., 2021; Teru et al., 2020) merely focus on the entity information, and have difficulties in modeling relational semantics of rules which are more critical for entity-independence. For example, Figure 1(b) indicates the inductive prediction process by $\boldsymbol{r_1}$. The training entities *Trump*, *Grand Hyatt* and *N.Y.* are generalized to variables X, Y, Z marked in red in Figure 1(b). It indicates that instantiated entities are less important than relation sequence (`workIn`, `locatedIn`) for predicting the relation `liveIn`.

* Corresponding author

Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 4261 - 4274
December 7-11, 2022



(a) A scenario of inductive relation prediction

(b) Illustrations of first-order rules and relational paths in logical reasoning

Figure 1: An example for inductive relation prediction and its inference process using a first-order logic rule.

Dataset	v1	v2	v3	v4
WN18RR	1.46	1.47	1.43	1.48
FB15K-237	3.13	9.31	16.76	25.27
NELL-995	4.68	15.40	20.95	24.22

Table 1: Statistics of average rules from all subgraphs with the max rule length as 3. There are four versions in each inductive relation prediction dataset, whose details are indicated in Appendix B.1.

Secondly, although logical reasoning provides inductive ability, the scarcity of relational semantics leads to **deficient supervision** of logic rules. Dominant logical reasoning methods like (Meilicke et al., 2019) indicates that in a KG owning m kinds of relations, the number of candidate rules within length N is $O(m^N)$, whereas in the real KG, there only contain a few rules during logical reasoning. For example, in Figure 1(b), there are actually 4 rules whose length within 3 from *Trump* to *N.Y.*, but at least $8^3 = 512$ candidate rules. The detailed statistics of rules in KGs are shown in Table 1. As a note, we treat number of relational paths as number of rules, like thick blue paths in Figure 1(b). It is unlikely to obtain all the supervised candidate logic rules, which would limit the performance of inductive relation prediction.

To address the above issues, we propose a model LogCo for inductive relation prediction with **Log**ical reasoning using **Co**ntrastive representations. LogCo firstly extracts subgraphs among the target relation and relational paths within a preset length. Relational paths will introduce the relation sequence semantics which is more important for entity-independence in logical reasoning. Secondly, a contrastive strategy to address the deficient supervision of logic rules is introduced into LogCo by constructing positive and negative relational paths. Then, LogCo obtains structural representations using a GCN. Positive and negative samples containing both topological structure and relation sequence semantics are all sent to the model for contrastive representations. Finally, LogCo applies a joint training regime combining the supervised and self-supervised information. The logical reasoning is illustrated by the relational path and the target relation, which are considered as the body and head respectively of a logic rule.

Our main contributions are three-fold:

- By regarding the relational path and the target relation as the body and head respectively of a logic rule, we integrate the neural network model and the discrete logic in inductive relation prediction by LogCo for the first time.
- To satisfy the entity-independence of inductive prediction in KGs, we represent relational paths as logic rules. A contrastive strategy is also devised to solve deficient supervision in logical reasoning. LogCo is the first to supplement the entity-independent semantics by relational paths and introduces contrastive representations into inductive relation prediction.
- Experiments of the relation prediction on twelve inductive datasets verify the superiority of LogCo comparing with latest inductive methods. Meanwhile, LogCo could obtain logic rules for interpretability of reasoning.

2 Preliminary

This section briefly introduces the first-order logic rule and its connection with the relational path. A first-order logic rule (Muggleton, 1991) learned from KGs is a Horn clause (Poole, 1993), which consists of an atom as head and a series of atoms

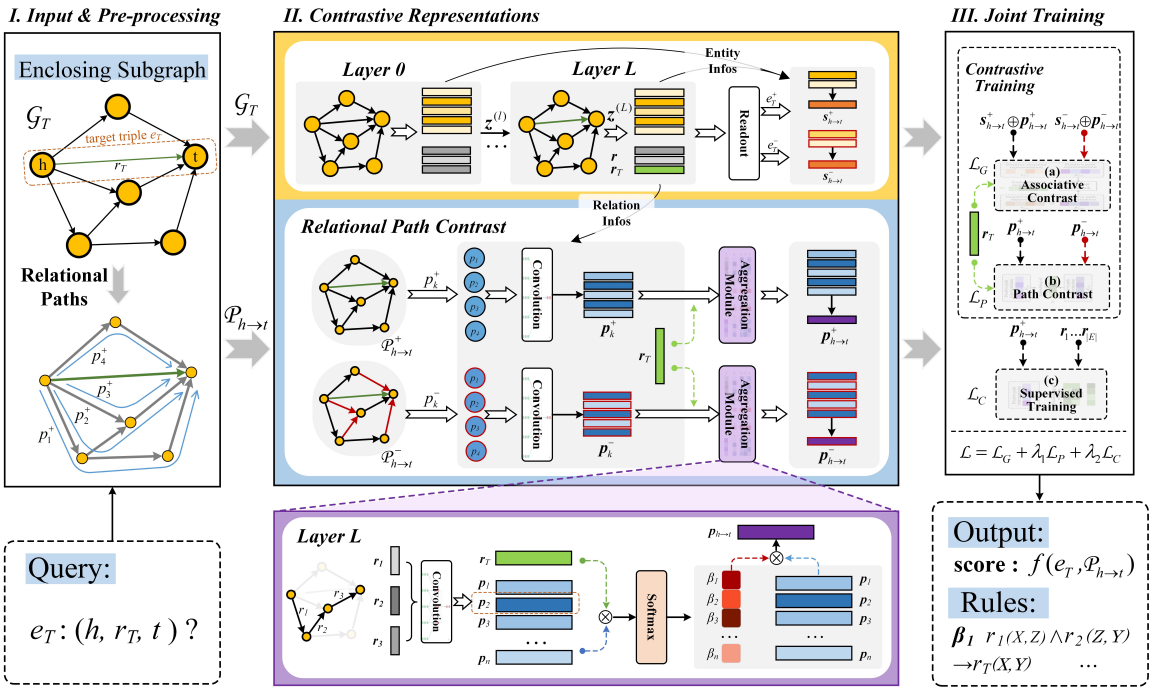


Figure 2: The overall framework of LogCo. It firstly extracts enclosing subgraphs and relational paths from the KG. Then it utilizes GCN to obtain contrastive representations of the subgraph and relational paths. The embeddings with red boxes are negative representations. Finally, the joint training regime is carried out to optimize the prediction model. LogCo outputs the prediction result and logic rules simultaneously.

as body. For example, here is a Horn clause with length N:

$$\beta \overbrace{\mathtt{r}_1(X,Z_1)\wedge \mathtt{r}_2(Z_1,Z_2)\wedge ...\mathtt{r}_N(Z_{N-1},Y)}^{body} \rightarrow \overbrace{\mathtt{r}_T(X,Y)}^{head}.$$

in which $\mathtt{r}_1, \mathtt{r}_2, ..., \mathtt{r}_N, \mathtt{r}_T$ are relations and $X, Z_1, Z_2, ..., Z_{N-1}, Y$ are variables generalized from entities in KGs. The body atoms are connected by a conjunction symbol $\wedge$ and point to the head by an implication symbol $\rightarrow$. In the rule, an atom contains two variables and adjacent atoms share the same variable. During inference, variables are instantiated to entities $x, z_1, z_2, \ldots, z_{N-1}, y$ and form a closed path, for example the blue thick paths in Figure 1(b). We use the relational path, target relation and $\beta \in [0, 1]$ to represent the body, head and *confidence* of the rule. As we discuss in Section 1, β can be obtained by the attention weight of a relational path.

3 Methodology

This section illustrates our proposed approach LogCo with the help of Figure 2.

3.1 Task Definition

Inductive relation reasoning in KGs is to make relation prediction on unseen entities. A target triple e_T is denoted as (h, r_T, t) in the train KG $G = \{R, E, T\}$, in which h and t are head and tail entities, and r_T is the target relation. R and E are sets of relations and entities in G, and $T \subseteq E \times R \times E$ is the set of triples. Relation prediction in a fully-inductive setting intends to quantify the score of every e_T in G and predict the relation between two unseen entities h' and t' in a testing KG $G' = \{R', E', T'\}$, where $R' \subseteq R$, $E' \cap E = \varnothing$. For clarify, we summarize important symbols in Table 6 of Appendix A.2

3.2 Initialization and Contrast Construction

Node Features. We extract enclosing subgraph $\mathcal{G}_T$ based on the target triple e_T from G, and implement the double radius vertex labeling scheme (Zhang and Chen, 2018) to entities in $\mathcal{G}_T$. The node i around target triple (h, r_T, t) is in the intersection of q-hop undirected neighborhoods of h and t. The node is labeled as $(d(i,h), d(i,t))$, in which d is the shortest topological distance between two entities. The label of node i is denoted as $[\text{one-hot}(d(i,h)) \oplus \text{one-hot}(d(i,t))] \in \mathbb{R}^{(2q+2)}$ to indicate the node feature, where $\oplus$ refers to the concatenation operation.

Contrastive Relational Paths Generation. Relational paths need extracting from $\mathcal{G}_T$. We use

breadth first search (BFS) algorithm (Cormen et al., 2001) for extracting every topological relational path whose length is no longer than L_{max} from h to t in $\mathcal{G}_T$. The set of extracted paths is denoted as $\mathcal{P}_{h\to t}$. For instance in Figure 2, if L_{max} is set as 3, then the algorithm would select 4 relational paths from the extracted subgraph $\mathcal{G}_T$. Moreover, for the contrastive samples, we consider r_T as the original instance and $\mathcal{P}_{h\to t}$ as the set of positive relational paths $\mathcal{P}^+_{h\to t}$. As for the set of negative samples $\mathcal{P}^-_{h\to t}$, they are constructed to distinguish semantics with the original instance and positive samples, so we randomly replace a part of every relational path in $\mathcal{G}_T$ and guarantee the negative samples not in $\mathcal{P}^+_{h\to t}$ for avoiding the false negative situation. The k-th positive and negative path samples are denoted as p^+_k and p^-_k respectively.

3.3 Contrastive Representations

The second step of LogCo is to get contrastive representations. The details are in the following.

Subgraph Embedding. We employ a GCN for obtaining embeddings of entities and relations in KGs. The propagation process for calculating the forward-pass update is defined as:

$$\boldsymbol{z}_i^{(l+1)} = \text{ReLU}(\sum_{r\in R}\sum_{j\in\mathcal{N}_i^r}\mu_{i,r}\mathbf{W}_r^{(l)}\boldsymbol{z}_j^{(l)} + \mathbf{W}_{self}^{(l)}\boldsymbol{z}_i^{(l)}), \quad (1)$$

where $\boldsymbol{z}_i^{(l+1)}$ denotes the embedding of node i in the $(l+1)$-th layer. $\mathcal{N}_i^r$ denotes the set of neighbors of i connected by relation r. $\mathbf{W}_r^{(l)}$ and $\mathbf{W}_{self}^{(l)}$ refer to the transformation matrices for propagating messages from layer l to $l+1$, where $\mathbf{W}_r^{(l)}$ is the matrix over relation r. $\mu_{i,r}$ is the edge attention weight corresponding to the edge connected via r:

$$\boldsymbol{o}_{i,r} = \sigma(\mathbf{W}_1[\boldsymbol{z}_i^{(l)} \oplus \boldsymbol{z}_j^{(l)} \oplus \boldsymbol{r}]), \quad (2)$$

$$\mu_{i,r} = \sigma(\mathbf{W}_2[\boldsymbol{o}_{i,r} \oplus \boldsymbol{r}_T]), \quad (3)$$

where $\boldsymbol{r}$ and $\boldsymbol{r}_T$ indicate the embeddings of relation r and r_T respectively. $\mathbf{W}_1$ and $\mathbf{W}_2$ are transformation matrices. σ is an activation function, such as ReLU$(\cdot)$ or Sigmoid$(\cdot)$.

Paths Representation. We design a strategy to obtain representations of relational paths in $\mathcal{G}_T$, which is shown in the purple block of Figure 2. In this phase, we use the information of relations in $\mathcal{G}_T$ as shown in Figure 2. Inspired by a rule mining work (Yang et al., 2015), we devise to calculate the semantic similarity between the target relation r_T and the relational path $p_k \in \mathcal{P}_{h\to t}$, for r_T and p_k connect the same h and t. Then, we utilize an aggregation function ψ to obtain the representation:

$$\boldsymbol{p}_{h\to t} = \psi(\{\boldsymbol{p}_k : p_k \in \mathcal{P}_{h\to t}\}). \quad (4)$$

The paths representation is given by:

$$\boldsymbol{p}_{h\to t} = \sum_{k=1}^{n}\beta_k\boldsymbol{p}_k \quad (5)$$

in which n is the number of paths in $\mathcal{G}_T$. β_k is the path attention weight between the path p_k and r_T. $\boldsymbol{p}_k$ is the representation of path p_k. We implement the continuous bag-of-words (CBOW) algorithm (Mikolov et al., 2013) on relation embeddings. For an alternate strategy, the path representation can be indicated by a convolution operation:

$$\boldsymbol{p}_k = \sum_{j=1}^{l_k-1}(\mathbf{W}\boldsymbol{r}_j^{in} + \boldsymbol{b}_j), \quad (6)$$

which utilizes a convolution neural network (CNN) to aggregate the relational path, considering the order of relations. l_k is the number of relations in p_k, and $\boldsymbol{r}_j^{in} = [\boldsymbol{r}_j \oplus \boldsymbol{r}_{j+1}]$ refers to the j-th window of the relation sequence. For the special condition when $l_k = 1$, we set $\boldsymbol{p}_k$ as the representation of the only relation. $\mathbf{W}$ is the convolution kernel and $\boldsymbol{b}_j$ is the optional bias. Because the rule length is not large, and we pay more attention to the sequence semantics of adjacent relations, we choose convolution rather than other operations. The attention weight β_k can be regarded as the confidence of corresponding rule for inference in $\mathcal{G}_T$, which is calculated as:

$$\beta_k = \text{softmax}(\boldsymbol{p}_k, \boldsymbol{r}_T) = \frac{\exp(\boldsymbol{p}_k{}^\top \boldsymbol{r}_T)}{\sum_{p_{k'}\in\mathcal{P}_{h\to t}}\exp(\boldsymbol{p}_{k'}{}^\top \boldsymbol{r}_T)}. \quad (7)$$

For further contrastive learning, representations of original sample, positive and negative path are $\boldsymbol{r}_T$, $\boldsymbol{p}^+_{h\to t}$ and $\boldsymbol{p}^-_{h\to t}$ respectively, in which $\boldsymbol{p}^+_{h\to t}$ and $\boldsymbol{p}^-_{h\to t}$ can be acquired with $\boldsymbol{p}^+_{\boldsymbol{k}}$ and $\boldsymbol{p}^-_{\boldsymbol{k}}$.

Logical Reasoning. LogCo extracts relational paths to capture the entity-independence during the reasoning process, which can be treated as first-order rules in KGs. After training, LogCo obtains relational paths p_k with attention weights β_k, which are actually rules' embeddings with confidences during logical reasoning. There would be several learned rules in a single subgraph that provide the interpretable process of reasoning.

3.4 Joint Training Regime

In this step, we propose a joint training regime combining the supervised and contrastive information, which consists of the associative contrast and path contrast. The detailed descriptions are as follows:

Associative Contrast. In order to associate the topological structure of $\mathcal{G}_T$ denoted as $\boldsymbol{s}_{h\to t}$ and semantics from the path representations denoted as $\boldsymbol{p}_{h\to t}$, we score the likelihood of target triple e_T as:

$$\boldsymbol{s}_{h\to t} = [\boldsymbol{z}^{(L)}_{\mathcal{G}_T} \oplus \boldsymbol{z}^{(L)}_{e_T}], \tag{8}$$

$$f(e_T, \mathcal{P}_{h\to t}) = \mathbf{W}_s[\boldsymbol{s}_{h\to t} \oplus \boldsymbol{r}_T \oplus \boldsymbol{p}_{h\to t}], \tag{9}$$

where $\mathbf{W}_s$ is the weight matrix. $\boldsymbol{z}^{(L)}_{e_T}$ is the entity embeddings' concatenation of e_T of all the L layers' messages, which can be indicated as $\big[\bigoplus_{l=1}^{L}(\boldsymbol{z}^{(l)}_h \oplus \boldsymbol{z}^{(l)}_t)\big]$. $\boldsymbol{z}^{(L)}_{\mathcal{G}_T}$ refers to the global representation of $\mathcal{G}_T$, given by the average readout:

$$\boldsymbol{z}^{(L)}_{\mathcal{G}_T} = \frac{1}{|\mathcal{V}_T|}\sum_{i\in\mathcal{V}_T} \boldsymbol{z}^{(L)}_i, \tag{10}$$

where $\mathcal{V}_T$ refers to the set of nodes in $\mathcal{G}_T$. We introduce margin-based loss to distance scores of positive and negative samples by an associative contrast:

$$\mathcal{L}_G = \sum_{e_T\in\mathcal{E}} \max(0, \eta + f(e^-_T, \mathcal{P}^-_{h\to t}) - f(e^+_T, \mathcal{P}^+_{h\to t})), \tag{11}$$

where e^+_T and e^-_T refer to the positive and negative triple samples, and e^-_T is the sample that replaces the head or tail of e^+_T. $f(\cdot)$ is the score function indicated as Eq.(9). $\mathcal{E}$ is the set of triples in G.

Path Contrast. If focusing on the semantic information given by relational paths, the contrastive learning should distinguish r_T with negative paths and make it close to positive paths. Therefore, inspiring by the InfoNCE loss in (van den Oord et al., 2018), for relational path samples pairs in KGs, the negative samples exist without distribution. The loss for path contrast is defined as:

$$\mathcal{L}_P = -\log\Big[\frac{\exp({\boldsymbol{p}^+_{h\to t}}^\top \boldsymbol{r}_T)}{\exp({\boldsymbol{p}^+_{h\to t}}^\top \boldsymbol{r}_T) + \exp({\boldsymbol{p}^-_{h\to t}}^\top \boldsymbol{r}_T)}\Big]. \tag{12}$$

Supervised Training. Except for the contrastive learning, we implement the supervised prediction. With the representation of positive paths $\boldsymbol{p}^+_{h\to t}$ in $\mathcal{G}_T$, the supervised learning intends to make it similar with the embedding of target relation $\boldsymbol{r}_T$. In our training regime, we apply the cross entropy loss on all relation labels in R to minimize the distance between $\boldsymbol{p}^+_{h\to t}$ and $\boldsymbol{r}_T$, and maximize the distances with other relations:

$$\mathcal{L}_C = -\log\Big[\frac{\exp({\boldsymbol{p}^+_{h\to t}}^\top \boldsymbol{r}_T)}{\sum_{r\in R}\exp({\boldsymbol{p}^+_{h\to t}}^\top \boldsymbol{r})}\Big]. \tag{13}$$

Eventually, the overall loss of LogCo is defined as the weighted summation of three losses, simultaneously optimizing them by a joint training process:

$$\mathcal{L} = \mathcal{L}_G + \lambda_1\mathcal{L}_P + \lambda_2\mathcal{L}_C, \tag{14}$$

where λ_1 and λ_2 are hyper-parameters representing weights of path contrast loss and supervised training loss respectively.

4 Experiments

In this section, we firstly introduce benchmarks, baselines, experiment settings and details. Secondly, to verify the effectiveness of LogCo, we implement comparison experiments on relation prediction task. Then, we use ablation studies, weight analysis and case studies to comprehensively demonstrate the performance.

4.1 Experimental Settings

Datasets. The inductive link prediction datasets are derived from WN18RR (Dettmers et al., 2018), FB15K-237 (Toutanova et al., 2015) and NELL-995 (Xiong et al., 2017), and each has been divided into four versions. The statistics of benchmark datasets are illustrated in (Teru et al., 2020). Each version of a dataset consists of train and test KGs, whose entities are totally different indicating the fully-inductive setting.

Baselines. The inductive baselines for comparison include rule-based RuleN (Meilicke et al., 2018), Neural-LP (Yang et al., 2017) and DRUM (Sadeghian et al., 2019), and graph-based GraIL (Teru et al., 2020), CoMPILE (Mai et al., 2021), TACT (Chen et al., 2021) and RED-GNN (Zhang and Yao, 2022). We also compare with two transductive methods TransE (Bordes et al., 2013) and CompGCN (Vashishth et al., 2020) to illustrate the effectiveness of them for inductive prediction.

Metrics. In the comparison, we implement both classification and ranking metrics to evaluate the model for multiple runs considering the random seeds and samples. AUC-PR is an indicator for classification task computing the area under prediction-recall curve and it evaluates if the triple is true. For the ranking metric Hits@10, we evaluate it in

Metric	Category	Method	WN18RR				FB15K-237				NELL-995			
			v1	v2	v3	v4	v1	v2	v3	v4	v1	v2	v3	v4
AUC-PR	Rule-based	RuleN	90.26	89.01	76.46	85.75	75.24	88.70	91.24	91.79	84.99	88.40	87.20	80.52
		Neural-LP	86.02	83.78	62.90	82.06	69.64	76.55	73.95	75.74	64.66	83.61	87.58	85.69
		DRUM	86.02	84.05	63.20	82.06	69.71	76.44	74.03	76.20	59.86	83.99	89.71	85.94
	Graph-based	GraIL	94.32	94.18	85.80	92.72	84.69	90.57	91.68	94.46	86.05	92.62	93.34	87.50
		TACT†	95.43	97.54	87.65	96.04	83.15	93.01	92.10	94.25	81.06	93.12	96.07	85.75
		CoMPILE†	98.29	99.36	93.60	**99.51**	83.06	90.21	93.12	93.24	82.39	93.30	95.71	52.98
	LogCo	L_{max}**=2**	98.62	99.10	91.24	98.36	86.62	91.99	94.58	94.54	88.72	**95.96**	96.10	**87.81**
		L_{max}**=3**	**99.43**	**99.45**	**93.99**	98.75	**89.74**	**93.65**	**94.91**	**95.26**	**91.24**	91.15	**96.28**	78.45
Hits@10	Transductive	TransE	18.09	18.03	20.25	20.68	17.07	19.77	21.97	20.25	24.00	21.64	21.94	19.49
		CompGCN	<0.001	<0.001	<0.001	0.035	<0.001	0.104	0.058	0.140	<0.001	<0.001	<0.001	<0.001
	Rule-based	RuleN	80.85	78.23	53.39	71.59	49.76	77.82	**87.69**	85.60	53.50	81.75	77.26	61.35
		Neural-LP	74.37	68.93	46.18	67.13	52.92	58.94	52.90	55.88	40.78	78.73	82.71	80.58
		DRUM	74.37	68.93	46.18	67.13	52.92	58.73	52.90	55.88	19.42	78.55	82.71	80.58
	Graph-based	GraIL	82.45	78.68	58.43	73.41	64.15	81.80	82.83	**89.29**	59.50	93.25	91.41	73.19
		RED-GNN	79.90	78.00	52.40	72.10	48.30	62.90	60.30	62.10	**86.60**	60.10	59.40	55.60
		TACT†	84.04	81.63	67.97	76.56	65.76	83.56	85.20	88.69	79.80	88.91	94.02	73.78
		CoMPILE†	81.91	76.64	57.35	71.80	62.20	82.01	84.67	87.44	58.33	88.86	93.63	60.81
	LogCo	L_{max}**=2**	**90.16**	84.69	**68.68**	**79.08**	**73.90**	81.91	80.64	84.20	61.75	**93.48**	94.19	**80.82**
		L_{max}**=3**	86.70	**86.73**	68.43	77.15	66.34	**84.21**	86.47	89.22	59.75	90.86	**94.44**	76.81

Table 2: Comparison of AUC-PR (%) and Hits@10 (%) results on inductive benchmarks from WN18RR, FB15K-237 and NELL-995. † means we reproduce the project by original codes. Other results are from (Teru et al., 2020) and (Zhang and Yao, 2022). The optimal and suboptimal values are marked in **bold** and underline respectively.

a general mode by ranking the test triples among 50 randomly negative samples, and see if the true triple can rank the top 10.

Experimental Details. For the subgraph extraction, we obtain 3-hop enclosing subgraphs by the double vertex labeling. In the graph embedding process, we employ a 3-layer GCN with the dimension as 32. We implement the experiments on one NVIDIA's Tesla V100 graphic card. Different parameters might influence the performance on different datasets, so the parameters are tuned separately. During the training process, the batch size is set as 16 and we use Adam (Kingma and Ba, 2015) as the optimizer with learning rate being 0.001. When extracting relational paths, we choose the max length as $L_{max} = 2$ and 3. For the hyper-parameters λ_1 and λ_2, we choose $\lambda_1, \lambda_2 \in [0.8, 1.2]$ and we will explain the choice in Subsection 4.4.[1]

More settings are detailed in Appendix B.3.

4.2 Comparison Results

Comparison of Prediction Results. The comparison results of relation prediction in Table 2 show that LogCo significantly outperforms them among the vast majority of datasets.

For the transductive methods, it is obvious in Table 2 that they are not suitable for solving inductive reasoning because of the lower Hits@10 results. For the rule-based inductive methods, the average boosts of LogCo on WN18RR, FB15K-237 and NELL-995 in AUC-PR are 12.54%, 6.65% and 7.55% respectively compared with the rule-based method RuleN. LogCo is also superior to differentiable methods Neural-LP and DRUM in terms of the classification and outperforms other rule-based methods on most datasets in terms of the ranking task, except for Hits@10 result on FB15K-237_v3. We attribute this phenomenon to the general performance of graph-based pattern.

After observation, graph-based inductive methods are generally more effective on most datasets than rule-based methods. Comparing with more competitive graph-based inductive methods, LogCo owns optimal results among these datasets in two metrics, which illustrate its superiority as well. LogCo results in as much as 6.15%, 3.04% and 2.95% average performance improvements in AUC-PR comparing to GraIL, which is the basic graph-based inductive method. As for the SOTA methods RED-GNN, CoMPILE and TACT, our method performs better on most datasets in terms of both metrics in the same experimental environment. The slight difference may be due to the fixed length of relational paths in LogCo.

Except for sampling one negative relational path during training, we implement LogCo with multiple negatives. However, the prediction results slightly increase with more time consumed. For example, the AUC-PR with one negative on WN18RR_v1 is 99.43% costing 23.66s per epoch, and 99.79% with 2 negatives costing 42.19s per epoch. More results are in Appendix C.1. In addi-

[1]Codes of LogCo will be available at https://github.com/pyd418/LogCo.

Method	WN18RR					FB15K-237					NELL-995				
	v1	v2	v3	v4	Avg	v1	v2	v3	v4	Avg	v1	v2	v3	v4	Avg
LogCo	**99.43**	**99.45**	**93.99**	**98.75**	**97.91**	**89.74**	**93.65**	**94.91**	**95.26**	**93.39**	**91.24**	**95.96**	**96.28**	**87.81**	**92.82**
LogCo w/o Paths	93.04	95.15	87.48	94.22	92.47	84.56	91.23	91.97	92.90	90.17	81.94	91.59	89.90	73.81	84.31
Δ	↓6.39	↓4.30	↓6.51	↓4.53	↓5.44	↓5.18	↓2.42	↓2.94	↓2.36	↓3.22	↓9.30	↓4.37	↓6.38	↓14.00	↓8.51
LogCo$^{\#1}$	95.64	96.05	87.57	95.25	93.63	85.01	92.03	92.48	92.42	90.49	88.12	89.70	93.33	81.18	88.08
Δ	↓3.79	↓3.40	↓6.42	↓3.50	↓4.28	↓4.73	↓1.62	↓2.43	↓2.84	↓2.90	↓3.12	↓6.26	↓2.95	↓6.63	↓4.74
LogCo$^{\#2}$	95.38	95.26	86.59	95.05	93.07	83.71	91.69	92.26	91.85	89.88	82.97	89.62	91.52	78.45	85.64
Δ	↓4.05	↓4.19	↓7.40	↓3.70	↓4.84	↓6.03	↓1.96	↓2.65	↓3.41	↓3.51	↓8.27	↓6.34	↓4.76	↓9.36	↓7.18

Table 3: Ablation results of AUC-PR (%) on inductive benchmarks derived from WN18RR, FB15K-237 and NELL-995. Especially, LogCo$^{\#1}$ removes the associative contrast. LogCo$^{\#2}$ removes path contrast compared with LogCo$^{\#1}$, where the performance comparison with LogCo$^{\#2}$ verifies the effectiveness of the path contrast.

Method	WN18RR					FB15K-237					NELL-995				
	v1	v2	v3	v4	Avg	v1	v2	v3	v4	Avg	v1	v2	v3	v4	Avg
LogCo	**86.70**	**86.73**	**68.43**	**77.15**	**79.75**	**70.24**	**84.21**	**86.47**	**89.22**	**82.54**	**59.75**	**93.48**	**94.19**	**80.82**	**82.06**
LogCo w/o Paths	83.51	78.68	63.05	76.28	75.38	65.36	81.17	80.59	81.25	77.09	48.50	81.93	89.25	68.05	71.93
Δ	↓3.19	↓8.05	↓5.38	↓0.87	↓4.37	↓4.88	↓3.04	↓5.88	↓7.97	↓5.45	↓11.25	↓11.55	↓4.94	↓12.77	↓10.13
LogCo$^{\#1}$	85.10	81.63	62.40	76.35	76.37	66.58	82.12	79.94	80.68	77.33	58.25	82.87	91.96	77.42	77.63
Δ	↓1.60	↓5.10	↓6.03	↓0.80	↓3.38	↓3.66	↓2.09	↓6.53	↓8.54	↓5.21	↓1.50	↓10.61	↓2.23	↓3.40	↓4.43
LogCo$^{\#2}$	84.04	81.63	62.31	76.34	76.08	63.65	81.38	75.61	54.92	68.89	45.50	76.79	91.03	66.21	69.88
Δ	↓2.66	↓5.10	↓6.12	↓0.81	↓3.67	↓6.59	↓2.83	↓10.86	↓34.30	↓13.65	↓14.25	↓16.69	↓3.16	↓14.61	↓12.18

Table 4: Ablation results of Hits@10 (%) on inductive benchmarks derived from WN18RR, FB15K-237 and NELL-995. The settings of LogCo$^{\#1}$ and LogCo$^{\#2}$ are the same as Table 3.

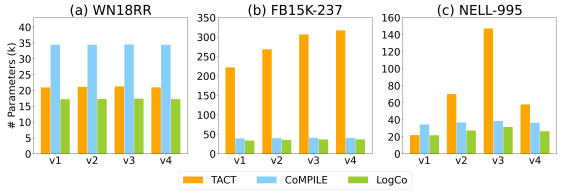


Figure 3: Comparing numbers of parameters (k) of LogCo and SOTA methods TACT and CoMPILE.

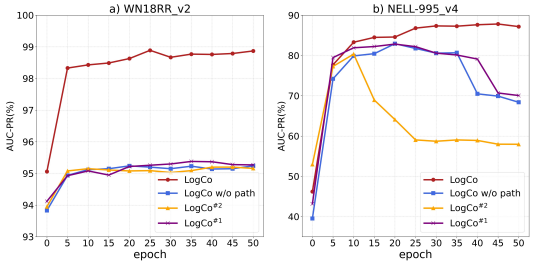


Figure 4: Training details of the proposed LogCo model and several kinds of ablation models.

tion, we implement LogCo when $L_{max} > 3$. However, there are circles and other noise when extracting relational paths, which will reduce the prediction results with more time. AUC-PR by LogCo on WN18RR_v1 is 98.86% if $L_{max} = 4$. Therefore, we record the results when $L_{max} = 2, 3$.

Comparison of Complexity. Moreover, LogCo needs less parameters than the SOTA methods, which means we achieve lower model complexity. The numbers of parameters on twelve datasets are shown in Figure 3(a), (b) and (c), where orange, blue and green bars refer to parameter quantities of TACT, CoMPILE and LogCo respectively. It is shown that CoMPILE needs much more parameters on WN18RR, and TACT consumes multiple parameters on FB15K-237 and NELL-995. LogCo performs better on complexity on all the twelve datasets. LogCo owns slightly lower results than CoMPILE on WN18RR_v4, but it has 17,288 parameters while CoMPILE has 34,465, reflecting the superiority of LogCo from an aspect.

4.3 Ablation Results

We investigate impacts of relational paths for logical reasoning and contrasts. Table 3 and 4 indicate the results when training the model by LogCo without the factors on all the datasets, and the methods are divided into two parts respectively. Especially, 1) "LogCo w/o Paths" removes the relational paths in LogCo. 2) LogCo$^{\#1}$ removes the associative contrast. 3) LogCo$^{\#2}$ removes the path contrast compared with LogCo$^{\#1}$. Δ indicates the reduction between the ablation model and LogCo. Other parameters remain the same during training and testing for a fair comparison.

From Table 3 and 4, the reduction of AUC-PR and Hits@10 values demonstrates that our contributions on LogCo all have positive impacts for inductive relation prediction. In addition, it shows that relational paths and contrasts are more effective on NELL-995 than other two datasets. Especially,

Dataset	β	body	$\rightarrow$head
WN18RR_v1	0.99	verb_group$(X,Z_1)\wedge$ hypernym$(Z_1,Z_2)\wedge$ hypernym(Z_2,Y)	$\rightarrow$hypernym(X,Y)
	0.50	derivationally_related_form$(X,Z_1)\wedge$ derivationally_related_form$(Z_1,Z_2)\wedge$ hypernym(Z_2,Y)	$\rightarrow$hypernym(X,Y)
	0.50	derivationally_related_form$(X,Z_1)\wedge$ hypernym$(Z_1,Z_2)\wedge$ derivationally_related_form(Z_2,Y)	$\rightarrow$hypernym(X,Y)
	<0.01	derivationally_related_form$(X,Z)\wedge$ derivationally_related_form(Z,Y)	$\rightarrow$hypernym(X,Y)
	0.41	derivationally_related_form$(X,Z_1)\wedge$ derivationally_related_form$(Z_1,Z_2)\wedge$ verb_group(Z_2,Y)	$\rightarrow$verb_group(X,Y)
	0.41	verb_group$(X,Z_1)\wedge$ derivationally_related_form$(Z_1,Z_2)\wedge$ derivationally_related_form(Z_2,Y)	$\rightarrow$verb_group(X,Y)
	0.18	verb_group$(X,Z)\wedge$ verb_group(Z,Y)	$\rightarrow$verb_group(X,Y)
	<0.01	hypernym(X,Y)	$\rightarrow$verb_group(X,Y)
FB15K-237_v1	1.00	location/contains$(X,Z_1)\wedge$ location/state$(Z_1,Z_2)\wedge$ location/contains(Z_2,Y)	$\rightarrow$location/contains(X,Y)
	1.00	location/contains$(X,Z)\wedge$ location/contains(Z,Y)	$\rightarrow$location/contains(X,Y)
	0.45	location/contains$(X,Z)\wedge$ location/adjoins(Z,Y)	$\rightarrow$location/contains(X,Y)
	<0.01	gardening_hint/split_to(X,Y)	$\rightarrow$location/contains(X,Y)
	0.97	film_release_region$(X,Z_1)\wedge$ netflix_genre/titles$(Z_1,Z_2)\wedge$ film/language(Z_2,Y)	$\rightarrow$film/language(X,Y)
	0.50	film/genre$(X,Z_1)\wedge$ netflix_genre/titles$(Z_1,Z_2)\wedge$ film/language(Z_2,Y)	$\rightarrow$film/language(X,Y)
	<0.01	film_release_region$(X,Z_1)\wedge$ location/contains$(Z_1,Z_2)\wedge$ major_field_of_study(Z_2,Y)	$\rightarrow$film/language(X,Y)
NELL-995_v1	1.00	agentControls$(X,Z_1)\wedge$ agentCollaboratesWithAgent$(Z_1,Z_2)\wedge$ worksFor(Z_2,Y)	$\rightarrow$worksFor(X,Y)
	0.53	worksFor$(X,Z)\wedge$ subpartOfOrganization(Z,Y)	$\rightarrow$worksFor(X,Y)
	<0.01	agentControls$(X,Z)\wedge$ agentControls(Z,Y)	$\rightarrow$worksFor(X,Y)
	1.00	subpartOf$(X,Z)\wedge$ subpartOf(Z,Y)	$\rightarrow$subpartOf(X,Y)
	0.38	agentBelongsToOrganization$(X,Z)\wedge$ subpartOf(Z,Y)	$\rightarrow$subpartOf(X,Y)
	<0.01	agentcollaborateswithagent(X,Y)	$\rightarrow$subpartOf(X,Y)

Table 5: Rules derived from three versions of datasets. The red rules indicate unreasonable ones.

the larger reduction of "LogCo w/o Paths" and "LogCo$^{\#2}$" in TABLE 3 verifies that the effectiveness is more obvious with the simultaneous action of paths and contrasts. It might because the more rules provide more relational semantics for logical reasoning. We also use Figure 4 to illustrate training details of LogCo and three ablation models on three versions of datasets. By comparing the AUC-PR values with "LogCo w/o path", "LogCo$^{\#1}$" and "LogCo$^{\#2}$", it is obvious that LogCo learns better on two datasets after 50 epochs, which representing the effectiveness of LogCo.

4.4 Weight Analysis

In our model, λ_1 and λ_2 are critical for adjusting functions of the supervised and self-supervised learning during training, so we rerun the training process in different values of λ_1 and $\lambda_2 \in [0.2, 1.2]$, and record the mean results after 5 tests on WN18RR_v1 and FB15K-237_v1 in Figure 5.

From the distribution of mean results, we observes that the inductive performance varies with different λ_1 or λ_2 values and better results gather at the lower right corner of the heat map. Especially, the best results occurs when $\lambda_1 = 1.0$ and $\lambda_2 = 1.2$ for WN18RR_v1. For FB15K-237_v1, when $\lambda_1 = 0.8$ and $\lambda_2 = 0.8$, LogCo obtains the best test result. The best results distribute near the diagonal of the lower right corner, which means the supervised and self-supervised learning are equally important for inductive reasoning. From the previous analysis, we choose λ_1 and λ_2 in [0.8, 1.2] for better performance. More weight analysis is in Appendix C.3.

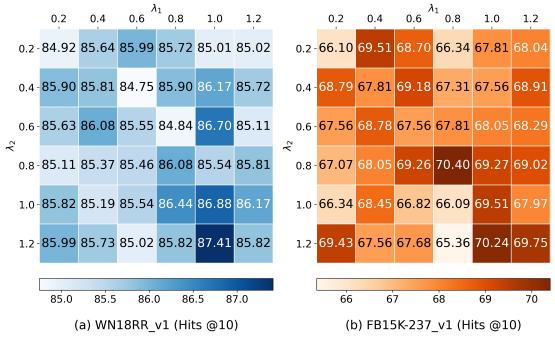


Figure 5: Effectiveness evaluation by Hits@10 (%) of parameters λ_1 and λ_2 over two datasets.

4.5 Case Studies

As stated in Section 3, a crucial advantage of LogCo is to represent first-order rules for logical reasoning explicitly. Table 5 shows examples of derived rules by LogCo on three datasets. The value in front of each rule is the confidence value in the corresponding subgraph. Rules in the same block are with the same head, which is generalized from an exact target triple, and the body is generalized from the reasoning path when predicting. The rules in red with the weights $\beta < 0.01$ indicate unreasonable rules when inference. They are less important for logical reasoning when predicting relations. Overall, LogCo implements the interpretability by these explicit rules.

5 Related Work

Inductive Learning in KGs can be divided into two aspects: rule-based and graph-based. Despite of the works with supplementary information (Xie

et al., 2016b,a), the statistical rule-based methods (Galárraga et al., 2013, 2015; Meilicke et al., 2018) are proposed to solve unseen entities without external knowledge. They induce inherent rules from KGs by enumerating all the candidates and select rules by preset thresholds. Some other differentiable models are proposed for scalability and less time. NeuralLP (Yang et al., 2017) and DRUM (Sadeghian et al., 2019) obtain rules by a neural controller system, especially the latter captures the backward and forward information about the order of atoms in rules. These methods are based on TensorLog (Cohen, 2016) to represent triples using matrices, which will cause high space complexity.

To solve the scalability and complexity issues, some graph-based inductive methods are proposed. GraIL (Teru et al., 2020) extracts subgraphs from KGs and implement the inductive capability by a graph neural network (GNN). CoMPILE (Mai et al., 2021) strengthens the message interactions between edges and entities. TACT (Chen et al., 2021) uses a relational correlation network to supply topological patterns between relations. RED-GNN (Zhang and Yao, 2022) proposes a relational directed graph to capture the KG's local evidence.

Distinguished from them, LogCo considers both aspects and utilizes the graph structure, and captures the entity-independent relational semantics in rules for logical reasoning.

Contrastive Learning aims to obtain representations by encoding similar or different samples to improve the effectiveness on downstream tasks. For the representation of text, CPC (van den Oord et al., 2018) gets context representations by predicting future information using a probabilistic contrastive loss. As for the images, MoCo (He et al., 2020) obtains visual representations by building large and consistent dictionaries with a momentum contrastive loss. SimCLR (Chen et al., 2020) declares that the composition of data augmentation, larger sizes and more training epochs are crucial for contrastive tasks. In the graph network, the deep graph infomax (Velickovic et al., 2019) is proposed to contrast the patch representations and corresponding high-level summary of graphs.

Contrastive learning is used for text, images and graphs, etc. For solving the deficient supervision of rules, we innovatively introduce the contrastive strategy into the inductive relation prediction.

6 Conclusion

We propose a novel inductive relation prediction model with logical reasoning using contrastive representations in KGs, named LogCo. We aim to solve two main issues in this task. To acquire the entity independence semantics from first-order logic rules, LogCo extracts relational paths in each subgraph. For the deficient supervision of logic rules caused by scarcity of relational semantics, LogCo introduces contrastive representations to obtain self-supervised information. The experiments on twelve fully-inductive datasets show the effectiveness of LogCo, and comprehensively demonstrate the impacts for relational paths and contrasts.

7 Limitations

LogCo still needs improving on performance and scalability. In LogCo, the path extracting method can be developed to own relational paths with flexible lengths. In addition, LogCo can be applied into more scenarios. For examples, LogCo solves the fully-inductive prediction, and it can be extended to both transductive and inductive learning. We also intend to implement LogCo on commonsense knowledge graphs (Speer et al., 2017) whose entities are free-form texts for high-order logical reasoning.

Acknowledgements

This work was supported by National Key Research and Development Program of China (2020AAA0108800), National Natural Science Foundation of China (62137002, 61937001, 62192781, 62176209, 62176207, 62106190, and 62250009), Innovative Research Group of the National Natural Science Foundation of China (61721002), Innovation Research Team of Ministry of Education (IRT_17R86), Consulting research project of Chinese academy of engineering "The Online and Offline Mixed Educational Service System for 'The Belt and Road' Training in MOOC China", "LENOVO-XJTU" Intelligent Industry Joint Laboratory Project, CCF-Lenovo Blue Ocean Research Fund, Project of China Knowledge Centre for Engineering Science and Technology, Foundation of Key National Defense Science and Technology Laboratory (6142101210201), the Fundamental Research Funds for the Central Universities (xhj032021013-02, xzy022021048, xpt012022033).

References

Ibrahim Abdelaziz, Srinivas Ravishankar, Pavan Kapanipathi, Salim Roukos, and Alexander G. Gray. 2021. A semantic parsing and reasoning-based approach to knowledge base question answering. In *Proceedings of AAAI*, pages 15985–15987.

Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2787–2795.

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

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. 2020. A simple framework for contrastive learning of visual representations. In *Proceedings of ICML*, pages 1597–1607.

William W. Cohen. 2016. Tensorlog: A differentiable deductive database. *CoRR*, abs/1605.06523.

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2001. *Introduction to Algorithms, Second Edition*. The MIT Press and McGraw-Hill Book Company.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Proceedings of AAAI*, pages 1811–1818.

Luis Galárraga, Christina Teflioudi, Katja Hose, and Fabian M. Suchanek. 2015. Fast rule mining in ontological knowledge bases with AMIE+. *The VLDB Journal*, pages 707–730.

Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian M. Suchanek. 2013. AMIE: association rule mining under incomplete evidence in ontological knowledge bases. In *Proceedings of WWW*, pages 413–422.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *Proceedings of CVPR*, pages 9726–9735.

Alfred Horn. 1951. On sentences which are true of direct unions of algebras. *J. Symb. Log.*, pages 14–21.

Zikun Hu, Yixin Cao, Lifu Huang, and Tat-Seng Chua. 2021. How knowledge graph and attention help? A qualitative analysis into bag-level relation extraction. In *Proceedings of ACL/IJCNLP*, pages 4662–4671.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR*.

Sijie Mai, Shuangjia Zheng, Yuedong Yang, and Haifeng Hu. 2021. Communicative message passing for inductive relation reasoning. In *Proceedings of AAAI*, pages 4294–4302.

Christian Meilicke, Melisachew Wudage Chekol, Daniel Ruffinelli, and Heiner Stuckenschmidt. 2019. Anytime bottom-up rule learning for knowledge graph completion. In *Proceedings of IJCAI*, pages 3137–3143.

Christian Meilicke, Manuel Fink, Yanjie Wang, Daniel Ruffinelli, Rainer Gemulla, and Heiner Stuckenschmidt. 2018. Fine-grained evaluation of rule- and embedding-based systems for knowledge graph completion. In *Proceedings of ISWC*, pages 3–20.

Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of ICLR. Workshop Track Proceedings*.

Stephen Muggleton. 1991. Inductive logic programming. *New generation computing*, pages 295–318.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *Proceedings of ICML*, pages 809–816.

David Poole. 1993. Probabilistic horn abduction and bayesian networks. *Artificial intelligence*, pages 81–129.

Ali Sadeghian, Mohammadreza Armandpour, Patrick Ding, and Daisy Zhe Wang. 2019. DRUM: end-to-end differentiable rule mining on knowledge graphs. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 15321–15331.

Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *Proceedings of ESWC*, pages 593–607.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proceedings of AAAI*, pages 4444–4451.

Komal K. Teru, Etienne Denis, and Will Hamilton. 2020. Inductive relation prediction by subgraph reasoning. In *Proceedings of ICML*, pages 9448–9457.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *Proceedings of EMNLP*, pages 1499–1509.

Aäron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748.

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

Petar Velickovic, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R. Devon Hjelm. 2019. Deep graph infomax. In *Proceedings of ICLR*.

Severine Verlinden, Klim Zaporojets, Johannes Deleu, Thomas Demeester, and Chris Develder. 2021. Injecting knowledge base information into end-to-end joint entity and relation extraction and coreference resolution. In *Findings of the Association for Computational Linguistics: ACL/IJCNLP*, pages 1952–1957.

Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. 2016a. Representation learning of knowledge graphs with entity descriptions. In *Proceedings of AAAI*, pages 2659–2665.

Ruobing Xie, Zhiyuan Liu, and Maosong Sun. 2016b. Representation learning of knowledge graphs with hierarchical types. In *Proceedings of IJCAI*, pages 2965–2971.

Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. Deeppath: A reinforcement learning method for knowledge graph reasoning. In *Proceedings of EMNLP*, pages 564–573.

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

Fan Yang, Zhilin Yang, and William W. Cohen. 2017. Differentiable learning of logical rules for knowledge base reasoning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2319–2328.

Muhan Zhang and Yixin Chen. 2018. Link prediction based on graph neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 5171–5181.

Yongqi Zhang and Quanming Yao. 2022. Knowledge graph reasoning with relational digraph. In *Proceedings of WWW*, pages 912–924.

A Model Details

A.1 Training Procedure of LogCo

LogCo can predict missing relations of incomplete triples in an inductive setting, and also mine first-order rules for interpretability at the same time. In Algorithm 1, we demonstrate the process of predicting relations by LogCo. LogCo uses a KG as the input and at last outputs the score of target triple and a set of first-order rules with confidences.

Algorithm 1 Process of learning first-order rules by LogCo

Input: KG $G\langle R, E, T\rangle$ and target triple e_T
Parameter: Max length of relational paths L_{max}, hyper-parameters $\lambda_1, \lambda_2, \eta$, etc.
Output: score of e_T and First-order logic rules F.

```
Extract subgraph G_T around each triple e_T, and initialize each node i by the double radius vertex labeling scheme.
for each training iteration do
    for each batch of triples in G do
        P^+_{h→t}, P^-_{h→t} ← generate contrastive relational paths within the length L_max.
        Obtain embeddings of entities and relations in G_T.
        p_{h→t}, β_i ← get representation of all the relational paths in G_T and the attention weight of each path by Eq. (4)(6)(5)(7).
        Calculate score of e_T by Eq. (9).
        L_G, L_N, L_C ← calculate associative contrast loss, path contrast loss and supervised loss by Eq. (11)(12)(13) respectively.
        L ← weighted summation of L_G, L_N and L_C by Eq. (14).
        Update the parameters by Adam optimizer.
    end for
end for
Score of target triples and first-order rules in F with the structure and confidence.
return Set of first-order rules F.
```

A.2 Notation Table

To clarify the process of LogCo, we use Table 6 to summarize the important symbols and their descriptions.

Symbol	Description
$e_T = (h, r_T, t)$	Target triple to be predicted
G, G'	Train and test KGs for inductive prediction
$\mathcal{G}_T$	The enclosing subgraph of e_T
$\boldsymbol{z}^{(l)}$	Embedding vector of node i at layer l
L_{max}	Max length of the relational paths
$\mathcal{P}^+_{h\to t}, \mathcal{P}^-_{h\to t}$	Positive and negative relational paths from h to t
$\boldsymbol{p}^+_{h\to t}, \boldsymbol{p}^-_{h\to t}$	Positive and negative paths' representations of $\mathcal{G}_T$
$\boldsymbol{p}_k, \beta_k$	Representation of path k and its attention weight
$\boldsymbol{p}_{h\to t}$	Overall path representation in $\mathcal{G}_T$
$\boldsymbol{s}_{h\to t}$	Structure representation of $\mathcal{G}_T$

Table 6: Important symbols and their descriptions.

B Experimental Details

B.1 Datasets

The inductive link prediction datasets are derived from three benchmarks, and have been generated into four versions respectively (Teru et al., 2020). The statistics of benchmark datasets are illustrated in Table 7. In each dataset, there is no intersection between entities in the train set and test set for the fully-inductive setting. In particular, each version of a dataset consists of a pair of KGs, *train-graph* and *ind-test-graph*, whose entities are totally different. Meanwhile, *train-graph* contains all the relations in *ind-test-graph*. Each *train-graph* has a corresponding valid KG during training.

B.2 State-of-the-Art Baselines

The baselines for comparison are previous methods for transductive and inductive relation prediction in KGs. The inductive methods are divided into two categories: rule-based and graph-based.

- TransE (Bordes et al., 2013) is the classical transductive reasoning method and CompGCN (Vashishth et al., 2020) is the recent and well-known state-of-the-art method. We choose the two methods to illustrate the effectiveness of transductive methods for inductive reasoning.

- RuleN (Meilicke et al., 2018) is the statistical rule-based inductive method which obtains rules for inductive relation prediction. Neural-LP (Yang et al., 2017) and DRUM (Sadeghian et al., 2019) are differentiable rule-based methods, and they generate explicit rules during the reasoning process.

	WN18RR			FB15K-237			NELL-995		
	#R	#E	#Tr	#R	#E	#Tr	#R	#E	#Tr
v1	9	2,746	6,678	183	2,000	5,226	14	10,915	5,540
v1-ind	9	922	1,991	146	1,500	2,404	14	225	1,034
v2	10	6,954	18,968	203	3,000	12,085	88	2,564	10,109
v2-ind	10	2,923	4,863	176	2,000	5,092	79	4,937	5,521
v3	11	12,078	32,150	218	4,000	22,394	142	4,647	20,117
v3-ind	11	5,084	7,470	187	3,000	9,137	122	4,921	9,668
v4	9	3,861	9,842	222	5,000	33,916	77	2,092	9,289
v4-ind	9	7,208	15,157	204	3,500	14,554	61	3,294	9,520

Table 7: Statistics of Datasets.

- Graph-based inductive methods GraIL (Teru et al., 2020), CoMPILE (Mai et al., 2021), TACT (Chen et al., 2021) and RED-GNN (Zhang and Yao, 2022) utilize the topological structure of subgraphs. However, they implement the prediction without interpretability by explicit rules. As a note, to reduce the influence of experimental environment and implement the comparison of complexity, we reproduce state-of-the-art methods CoMPILE and TACT with settings by original codes. Other results of baselines are from (Teru et al., 2020) and (Zhang and Yao, 2022).

B.3 Detailed Experimental Settings

We implement the experiments on one NVIDIA's Tesla V100 graphic card. During experiments, different parameters might influence the performance of LogCo on different datasets, so the parameters are tuned separately for each dataset. We conduct experiments in the following search space of parameters:

- Learning rate: {0.0005, 0.001, 0.005, 0.01}
- The number of subgraph hops: {2, 3, 4}
- The number of negative samples: {1, 2}
- Maximum length of relational paths L_{max}: {2, 3, 4}
- Margin hyper-parameter η: {8, 10, 16}

C Supplementary Results

C.1 Results of Multiple Negatives

LogCo uses contrastive representations in logical reasoning. It is reasonable to generate more negatives for better results, but it is limited by the space and time resource. We solve this problem by randomly constructing one negative during iterations. The contrastive losses are calculated by various negative samples. It can implement the diversity of negatives and avoid false negatives with too many negative samples. For clear illustration, Table 8 shows the results with one (Neg = 1) and multiple negatives (Neg = 2). It is indicated that multiple negatives can slightly increase the prediction results but cost more time. For example, the AUC-PR by LogCo on WN18RR_v1 is 99.43% with 23.66s per epoch, and 99.79% with 2 negatives, but costing 42.19s per epoch.

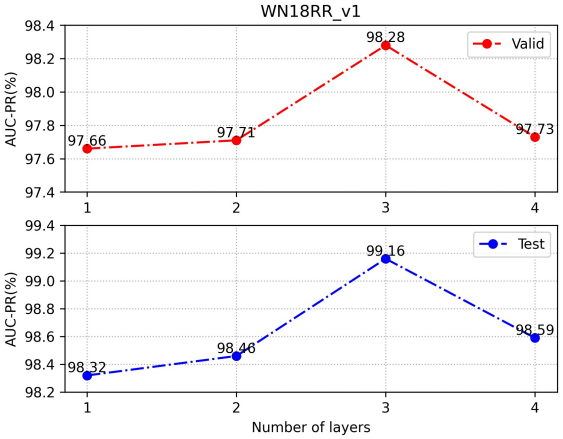


Figure 6: Analysis of the GCN layer number by AUC-PR on WN18RR_v1.

C.2 Analysis of Number of Layers

Figure 6 shows the result curves of different GCN layer numbers L during training. The optimal layer number is in $\{1, 2, 3, 4\}$. It is indicated that test and valid results are best when the layer number is 3. Thus we set the layer number of LogCo as 3

Dataset		Neg = 1		Neg = 2	
		AUC-PR	Time	AUC-PR	Time
WN18RR	v1	99.43	23.66s	99.79	42.19s
	v2	99.45	104.60s	99.48	118.04s
	v3	93.99	215.40s	91.79	232.62s
	v4	98.75	64.81s	98.82	79.51s

Table 8: Prediction results (%) and consumed time per epoch (s) for one and multiple negatives. The results of time for one epoch are recorded with the same environment and settings.

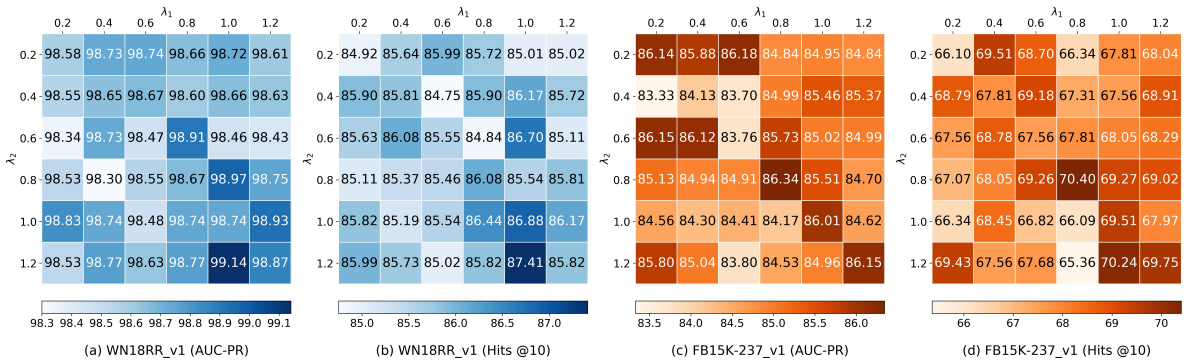


Figure 7: Effectiveness evaluation by AUC-PR and Hits@10 of parameters λ_1 and λ_2 over two datasets.

during experiments.

C.3 Hyper-parameter Sensitivity Analysis

In addition to the effectiveness evaluation by Hits@10 in Section 4, Figure 7 shows the distributions of results by two metrics AUC-PR and Hits@10 on WN18RR_v1 and FB15K-237_v1. From the distribution of mean results, we get several observations. Firstly, for WN18RR_v1, better results gather at the lower right corner of the heat map, especially when $\lambda_1 = 1.0$ and $\lambda_2 = 1.2$. Secondly, for FB15K-237_v1, apparently the best results distribute near the diagonal, which means the supervised and self-supervised learning are equally important for inductive reasoning. When $\lambda_1 = 0.8$ and $\lambda_2 = 0.8$, LogCo obtains the best test result. In addition, Fig. 5 also shows the distributions of results by the ranking metric Hits@10 on WN18RR_v1 and FB15K-237_v1. From the heat maps, we find that the results of AUC-PR and Hits@10 are in similar distributions. For WN18RR_v1, the best results gather at the lower right corner of Figure 7, and obtain the best result when $\lambda_1 = 1.0$ and $\lambda_2 = 1.2$. For FB15K-237_v1, despite slight differences, the best results distribute near the diagonal as well, especially when $\lambda_1 = 0.8, \lambda_2 = 0.8$ and $\lambda_1 = 1.0, \lambda_2 = 1.2$. The distributions are distinct on different datasets. It might be related to the number of paths in a subgraph, for the average paths in an enclosing subgraph of WN18RR are less than those of FB15K-237 and NELL-995 in TABLE 1. From the previous analysis, we choose λ_1 and λ_2 in [0.8, 1.2] for WN18RR and FB15K-237 and NELL-995.