LATENTLOGIC: Learning Logic Rules in Latent Space over Knowledge Graphs

Junnan Liu[1], **Qianren Mao**[2], **Chenghua Lin**[3], **Yangqiu Song**[4], **Jianxin Li**[1,2,*]
[1]School of Computer Science and Engineering, Beihang University, Beijing, P.R.China.
[2]Zhongguancun Laboratory, Beijing, P.R.China.
[3]Department of Computer Science, University of Manchester, U.K.
[4]Department of Computer Science and Engineering, HKUST, Hong Kong SAR, China.
liujn@act.buaa.edu.cn, {maoqr,lijx}@zgclab.edu.cn, chenghua.lin@manchester.ac.uk, yqsong@cse.ust.hk

Abstract

Learning logic rules for knowledge graph reasoning is essential as such rules provide interpretable explanations for reasoning and can be generalized to different domains. However, existing methods often face challenges such as searching in a vast search space (e.g., enumeration of relational paths or multiplication of high-dimensional matrices) and inefficient optimization (e.g., techniques based on reinforcement learning or EM algorithm). To address these limitations, this paper proposes a novel framework called LATENTLOGIC to efficiently mine logic rules by controllable generation in the latent space. Specifically, to map the discrete relational paths into the latent space, we leverage a pre-trained VAE and employ a discriminator to establish an energy-based distribution. Additionally, we incorporate a sampler based on ordinary differential equations, enabling the efficient generation of logic rules in our approach. Extensive experiments on benchmark datasets demonstrate the effectiveness and efficiency of our proposed method.

1 Introduction

Knowledge graphs usually contain collections of real-world facts encoded in triplets with entity and relation information, and find broad applications in across multiple domains (Lukovnikov et al., 2017; Xiong et al., 2017a; Wang et al., 2018; Zhang et al., 2019; Huang et al., 2019; Tang et al., 2023a,b). Despite some knowledge graphs holding hundreds of millions of triples, they still suffer from incompleteness, whereby many valid triples are missing since it is impractical to identify them all manually. Therefore, a fundamental and essential task in knowledge graphs is to utilize existing facts to predict the missing ones.

Recent studies have focused on learning logic rules from knowledge graphs and utilizing these learned rules to predict absent facts. An example of such a rule is $\forall X, Y, Z\ \texttt{nationality}(X, Y) \leftarrow \texttt{classmate}(X, Z) \wedge \texttt{nationality}(Z, Y)$, indicating that if Z is the classmate of X and has a nationality of Y, then X is likely to have a nationality of Y. This rule can be applied to deduce the nationalities of new individuals. Compared to other methods such as knowledge graph embedding approaches (Bordes et al., 2013; Sun et al., 2019; Li et al., 2022), the rule-based method (Zhang et al., 2020) is more interpretable and can be applied to inductive scenarios (Teru and Hamilton, 2019).

Most rule-based methods involve enumerating relational paths as candidate rules, followed by assigning weights to each rule to indicate their quality (Lao and Cohen, 2010; Richardson and Domingos, 2006; Yang et al., 2017; Sadeghian et al., 2019). When the scale of the knowledge graph expands, these methods face the challenge of exponentially growing search space. To overcome this problem, RNNLogic (Qu et al., 2021) introduces a rule generator and a reasoning predictor to separate rule generation from rule weight learning. However, the optimization process based on the Expectation-Maximization (EM) algorithm tends to have slow convergence, leading to extended training periods. Another line of research utilizes reinforcement learning (RL) to search for logic rules by making sequential decisions (Xiong et al., 2017b; Lin et al., 2018; Das et al., 2018; Lu et al., 2022). Nevertheless, RL-based methods often encounter challenges such as large action spaces and sparse rewards during training. As a result, efficiently mining high-quality logic rules for knowledge graph reasoning remains a challenging task.

In this paper, we propose a novel framework named **LATENTLOGIC**, which overcomes the aforementioned challenges. Our approach bypasses the enumeration of relational paths by employing controllable sampling in latent space. Furthermore, each component of LATENTLOGIC is trained in an end-to-end fashion, avoiding the indirect and

* Jianxin Li is the corresponding author.

inefficient optimization procedure. Concretely, we utilize a VAE-based autoencoder (Kingma and Welling, 2014; Li et al., 2020) to map discrete relational paths into a low-dimensional latent space. We employ a discriminator to measure the semantic coherence between the latent vector and the rule head, thereby creating an energy-based distribution in the latent space. To obtain latent vector samples that correspond to the desired rule head, we employ a sampler based on ordinary differential equations (Song et al., 2021; Nie et al., 2021). The latent vectors are used to generate rule bodies for the given rule head using the VAE generator.

Extensive experiments demonstrated the computational efficiency of LATENTLOGIC in comparison to previous rule learning methods (Yang et al., 2017; Sadeghian et al., 2019; Qu et al., 2021), indicating that LATENTLOGIC exhibits enhanced scalability for mining logic rules on larger-scale knowledge graphs. Furthermore, through experiments on two commonly used benchmark datasets, FB15k-237 (Toutanova and Chen, 2015) and WN18RR (Dettmers et al., 2018), we observed that LATENTLOGIC successfully generates high-quality logic rules for knowledge graph reasoning and evidently outperform the salient baseline methods.

2 Framework

Problem Definition First, we introduce some definitions and notations. A knowledge graph $\mathcal{G} = (\mathcal{V}, \mathcal{T}, \mathcal{R})$ is usually defined by a triple set $\mathcal{T} = \{(h, r, t)\} \subseteq \mathcal{V} \times \mathcal{R} \times \mathcal{V}$, where $\mathcal{V}$ denotes an entity set, and $\mathcal{R}$ represents a relation set. In this paper, we aim to learn logic rules in the conjunctive form $\forall\{X_i\}_{i=0}^{l} : \mathtt{r}(X_0, X_l) \leftarrow \mathtt{r}_1(X_0, X_1) \wedge \cdots \wedge \mathtt{r}_l(X_{l-1}, X_l)$ from the given knowledge graph. A logic rule, which can be abbreviated as $\mathtt{r} \leftarrow \mathtt{r}_1 \wedge \cdots \wedge \mathtt{r}_l$, consists of a rule head, denoted as $\mathtt{r}$, and a rule body (relational path), represented as $\mathtt{r}_1 \wedge \cdots \wedge \mathtt{r}_l$.

Framework Overview Our approach converts rule learning problems to controllable generation problems by developing a generative model, denoted as $p_\theta(\mathcal{P}|s)$, to generate rules given a specific rule head s. Here, $\mathcal{P}$ represents the rule body, i.e., relational path, $\mathcal{P} = (\mathtt{r}_1, \mathtt{r}_2, \ldots \mathtt{r}_l)$. The main innovation behind our approach is that we substitute the enumeration of the relational path with the sampling of latent vectors, significantly reducing training overhead. Additionally, to accomplish our aim, we incorporate a relational path autoencoder and a rule discriminator that operates in the latent space. As shown in Fig. 1, we firstly utilize a VAE-based *Relational Path Autoencoder*, which consists of an encoder $\mathcal{E}$ and a decoder $\mathcal{D}$, to compress relational paths into a low-dimensional latent space. To perform controllable generation of rule bodies, we need to construct a joint distribution of latent vectors and rule heads for sampling. Therefore, we introduce a *Rule Discriminator* on the latent space to form the joint distribution represented by the energy-based model. This joint distribution allows us to obtain latent vectors associated with the desired rule head by using a sampler based on an ordinary differential equation. Finally, we can decode the latent vectors into rule bodies using the VAE decoder, allowing us to generate rules given certain rule heads.

2.1 Relational Path Autoencoder

Note that each rule body $\mathtt{r}_1 \wedge \cdots \wedge \mathtt{r}_l$ can be considered a sequence of relations $[\mathtt{r}_1, \ldots, \mathtt{r}_l]$. Such sequences of relations can be effectively modeled by sequence neural networks (Das et al., 2017; Kotnis et al., 2021; Liu et al., 2022), and thus we introduce RNN (Hochreiter and Schmidhuber, 1997) to parameterize the relational path autoencoder. Specifically, we map relational path $\mathcal{P}$ into latent vector $\boldsymbol{z}$ using an RNN-based encoder $q(\boldsymbol{z}|\mathcal{P})$, and an RNN-based decoder $p(\mathcal{P}|\boldsymbol{z}, \boldsymbol{q})$ that maps $\boldsymbol{z}$ into the relational path $\boldsymbol{q}$ is a unified query embedding for all inputs. Our decoder does not use an autoregressive approach. Instead, it takes the unified query embedding $\boldsymbol{q}$ and positional embedding as input, simultaneously generating relational paths. We optimize the encoder and decoder parameters for each input relational path $\mathcal{P}$ with the objective:

$$\begin{aligned}\mathcal{L}_{\text{VAE}}(\mathcal{P}) = &- \mathbb{E}_{q(\boldsymbol{z}|\mathcal{P})}[\log p(\mathcal{P}|\boldsymbol{z}, \boldsymbol{q})] \\ &+ \text{KL}\left(q(\boldsymbol{z}|\mathcal{P})||\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})\right),\end{aligned} \tag{1}$$

where $\text{KL}(\cdot||\cdot)$ is the Kullack-Leibler divergence that pushes q to be close to the prior $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$.

2.2 Rule Discriminator

Now we aim to model the joint distribution $p(\boldsymbol{z}, s)$, where $\boldsymbol{z}$ denotes the latent vector of a relational path and s represents a desired rule head. This joint distribution can be represented as $p(\boldsymbol{z}, s) = p_{\text{prior}}(\boldsymbol{z})p(s|\boldsymbol{z})$, where $p_{\text{prior}}(\boldsymbol{z})$ is the prior distribution, i.e., standard Gaussian distribution, and $p(s|\boldsymbol{z})$ is conditional distribution on s given $\boldsymbol{z}$. Inspired by Nie et al. (2021), we define $p(s|\boldsymbol{z})$ as an

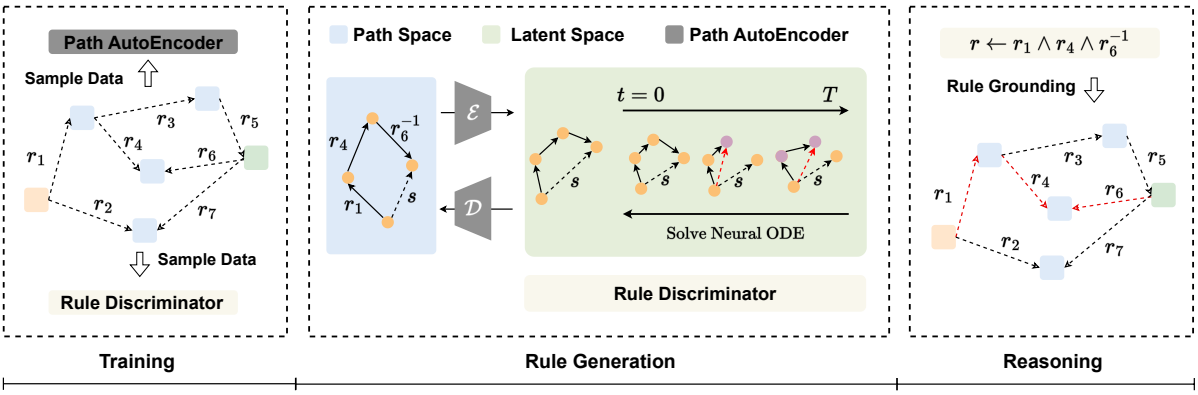


Figure 1: Framework overview of LATENTLOGIC. We first train the rule discriminator and relational path autoencoder. Then we generate logic rules by sampling on the latent space. The generated rules can be used to answer queries over knowledge graphs.

energy-based model (EBM) (LeCun et al., 2006) as follows:

$$p(s|\boldsymbol{z}) \propto e^{-E_\theta(s|\boldsymbol{z})}, \\ E_\theta(s|\boldsymbol{z}) = -f(g(\boldsymbol{z}))[s] + \text{const}, \tag{2}$$

where const $= \log \sum_{s'} \exp(f(g(\boldsymbol{z}))[s'])$ is a normalization term corresponding to all rule heads, $g(\cdot)$ is the fixed VAE decoder that decodes $\boldsymbol{z}$ into $\mathcal{P}$, and f indicates a neural network that takes $g(\boldsymbol{z})$ and s as input and produces a score that measures how well s is carried in $\boldsymbol{z}$. The function f can be any advanced model if it measures the consistency between relational paths and the rule heads (e.g., other rule mining models like neural logical programming). We adopt a simple network as suggested by Das et al. (2017), which encodes the relational path using the recurrent neural network, computing the score using the similarity between path representation and relation representation. For each training sample $(r_\text{h}, (r_0, \ldots, r_i))$. Suppose $\hat{\boldsymbol{y}}$ denotes the output logits of the rule discriminator; the objective can be expressed as:

$$\mathcal{L}_\text{Discriminator} = -\sum_{\mathcal{S}} \sum_{r}^{|\mathcal{R}|} \boldsymbol{y}_r^{r_\text{h}} \log \hat{\boldsymbol{y}}_r, \tag{3}$$

where $\mathcal{S}$ represents all the training samples and $\boldsymbol{y}^{r_\text{h}}$ is a one-hot vector that only the r_h-th position is 1.

2.3 Rule Generation

Model Training We train both the relational path autoencoder and the rule discriminator on the given knowledge graph $\mathcal{G}$. For the relational path autoencoder, we adopt a random walk (Spitzer, 1975)-based procedure to efficiently sample relational paths to obtain training data. Each subsequent node is generated using the following distribution:

$$p(x_i|x_{i-1}) = \begin{cases} \dfrac{1}{|\mathcal{N}(x_{i-1})|}, & (x_i, \cdot, x_{i-1}) \in \mathcal{T}, \\ 0, & \text{otherwise}, \end{cases} \tag{4}$$

where $\mathcal{N}(x_i)$ denotes all the neighborhoods of entity x_i. Then we optimize the auto-encoder by minimizing the Eq. 1. For the rule discriminator, we employ a similar sampling strategy. Each time we sample the next entity x_i, we include relation r_h, which directly connects x_o and x_i, to create a training sample $(r_\text{h}, (r_0, \ldots, r_i))$. Suppose $\hat{\boldsymbol{y}}$ denotes the output logits of the rule discriminator, the objective can be expressed as follows:

$$\mathcal{L}_\text{Discriminator} = -\sum_{\mathcal{S}} \sum_{r}^{|\mathcal{R}|} \boldsymbol{y}_r^{r_\text{h}} \log \hat{\boldsymbol{y}}_r, \tag{5}$$

where $\mathcal{S}$ represents all the training samples and $\boldsymbol{y}^{r_\text{h}}$ is a one-hot vector that only the r_h-th position is 1.

Latent Vector Sampling Given the joint distribution $p(\boldsymbol{z}, s)$, we would like to draw samples $\boldsymbol{z}$ conditioned on the target rule head s, which are then fed to the VAE decoder to obtain the desired rule bodies. According to Song et al. (2021), sampling from an EBM can be achieved by solving a specific ordinary differential equation (ODE). In our work, the ODE in the latent space can be expressed as: $d\boldsymbol{z} = \frac{1}{2}\beta(t)E_\theta(s|g(\boldsymbol{z}))dt$, with negative time increments from T to 0. To generate latent vectors based on the given rule head s, we first draw $\boldsymbol{z}(T)$ from $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$, and then apply a neural ODE solver[1] (Chen et al., 2018, 2021) to

[1] https://github.com/rtqichen/torchdiffeq

Methods	FB15k-237				WN18RR			
	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10
TransE	0.294	-	-	46.5	0.226	-	-	50.1
DistMult	0.241	15.5	26.3	41.9	0.430	39.0	44.0	49.0
TuckER	0.358	26.6	39.4	54.4	0.470	44.3	48.2	52.6
RotatE	0.338	24.1	37.5	53.3	0.476	42.8	49.2	57.1
PathRank†	0.087	7.4	0.2	11.2	0.189	17.1	20.0	22.5
NeuralLP†	0.237	17.3	25.9	36.1	0.381	36.8	38.6	40.8
DRUM†	0.238	17.4	26.1	36.4	0.382	36.9	38.8	41.0
RNNLogic†	0.288	20.8	31.5	44.5	0.455	41.4	47.5	53.1
RLogic‡	0.312	20.3	-	50.1	0.473	44.3	-	53.7
LATENTLOGIC	**0.320**	**21.2**	**32.9**	**51.4**	**0.481**	**45.2**	**49.7**	**55.3**

Table 1: Knowledge graph reasoning performance on FB15k-237 and WN18RR. The model with † means the results are from Qu et al. (2021), and ‡ means the results are copied from Cheng et al. (2022). For RNNLogic, we choose the variant *w/o emb.* for a fair comparison.

obtain $\boldsymbol{z} = \boldsymbol{z}(0)$.

Rule Generation We draw n samples $\{\boldsymbol{z}_i(T)\}_{i=0}^n$ from $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$ for each rule head s and obtain n latent vectors $\{\boldsymbol{z}_i(0)\}_{i=0}^n$. Then, we feed $\{\boldsymbol{z}_i(0)\}_{i=0}^n$ to pretrained VAE decoder to generate corresponding rule bodies. Finally, we calculate the confidence scores of the generated rules using the aforementioned rule discriminator.

3 Experimental Setup

Datasets We conduct experiments on two widely used knowledge graph reasoning benchmark datasets: FB15k-237 (Toutanova and Chen, 2015) and WN18RR (Dettmers et al., 2018). WN18RR comprises of 40,943 entities and 11 relations, with 86k/3k/3k instances set aside for training/validation/testing correspondingly. FB15k-237 comprises 14,541 entities and 237 relations and has 272k/17k/20k instances reserved for training/validation/testing, respectively.

4 Experimental Results

Baselines We compared LATENTLOGIC with two taxonomies of methods, including: Knowledge graph embedding methods: TransE (Bordes et al., 2013), DistMult (Yang et al., 2015), TuckER (Balazevic et al., 2019), and RotatE (Sun et al., 2019). Rule-based methods: PathRank (Lee et al., 2013), NeuralLP (Yang et al., 2017), DRUM (Sadeghian et al., 2019), RNNLogic (Qu et al., 2021), and RLogic (Cheng et al., 2022).

Evaluation Metrics We adopt *forward chaining* (Salvat and Mugnier, 1996) for inferring missing facts from logical rules. For every test triplet (h, r, t), two queries are created: $(h, r, ?)$ and $(?, r, t)$, using t and h as answers, respectively. To maintain consistency with previous studies, Mean Rank (MR), Mean Reciprocal Rank (MRR), and hit@k are selected as evaluation metrics under the *filtered* setting (Sun et al., 2019). Furthermore, to mitigate *the effects of random sampling*, we evaluate the model performance on five different random seeds, and report the average performance.

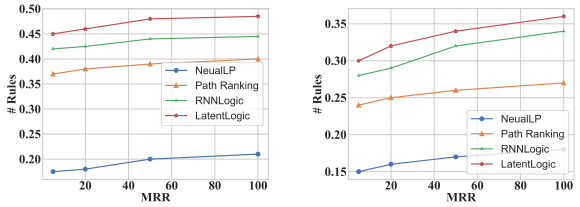
Overall Performance As shown in Tab. 1, we present the experimental results on the FB15k-237 and WN18RR datasets. Firstly, we compare LATENTLOGIC with rule-based methods and observe that LATENTLOGIC outperforms the statistical learning method PathRank, neural differentiable methods NeuralLP and DRUM, as well as recent methods RNNLogic and RLogic. In particular, we obtain 2.56% and 2.59% relative increase in MRR and Hits@10 on FB15k-237 against the state-of-the-art rule-based method RLogic (Cheng et al., 2022). Similarly, LATENTLOGIC achieves 1.69% and 2.98% increase in MRR and Hits@10 on the WN18RR dataset against RLogic. Then, we also compare LATENTLOGIC against salient knowledge graph embedding-based methods and find that LATENTLOGIC yields comparable performance to embedding-based methods, especially on WN18RR, where it outperforms selected embedding-based baselines.

Quality of Learned Rules We assess the quality of the logic rules learned from different models in this part. Following the settings of Qu et al. (2021),

$\texttt{film\_language}(x, y) \leftarrow \texttt{film\_actor}(x, z) \wedge \texttt{person\_languages}(z, y)$ $\texttt{film\_language}(x, y) \leftarrow \texttt{film\_prequel}(x, z) \wedge \texttt{film\_language}(z, y)$
$\texttt{film\_country}(x, y) \leftarrow \texttt{film\_produced\_by}(x, z) \wedge \texttt{person\_nationality}(z, y)$ $\texttt{film\_country}(x, y) \leftarrow \texttt{film\_produced\_by}(x, z_1) \wedge \texttt{people\_lived\_location}(z_1, z_2) \wedge \texttt{location\_country}(z_2, y)$
$\texttt{person\_nationality}(x, y) \leftarrow \texttt{place\_of\_birth}(x, z) \wedge \texttt{location\_country}(z, y)$ $\texttt{person\_nationalit}(x, y) \leftarrow \texttt{organization\_founder}(x, z) \wedge \texttt{organization\_country}(z, y)$

Table 2: Case study on the FB15k-237 dataset.

we generate n logic rules with the highest quality score for each query relation and then use them for training a predictor for knowledge graph reasoning. As mentioned before, the quality of each rule learned by LATENTLOGIC can be calculated using the rule discriminator. Results for different n values are reported in Fig. 2. Our observations suggest that LATENTLOGIC outperforms the compared methods remarkably. Moreover, even with a limited number of rules considered per relation, LATENTLOGIC still achieves competitive results.



(a) WN18RR (b) FB15k-237

Figure 2: MRR performance w.r.t. n rules on WN18RR and FB15k-237 datasets. LATENTLOGIC achieves prominent results even with only 10 rules.

Case Study To demonstrate that LATENTLOGIC is capable of generating helpful and varied rules for knowledge graph reasoning, we present some of the logic rules that LATENTLOGIC produced on the FB15k-237 dataset in Tab. 2. It can be observed that the logic rules generated by LATENTLOGIC are semantically meaningful and diverse.

5 Related Work

Over the past years, learning logical rules over knowledge graphs has been an active research area. Most traditional methods enumerate relational paths between query entities and answer entities as candidate logic rules, and further learn a scalar weight for each rule to assess the quality. Some representative works include Markov Logic Network (MLN) (Kok and Domingos, 2005; Richardson and Domingos, 2006), path ranking (Lao and Cohen, 2010; Lao et al., 2011) and probabilistic personalized page rank (ProPPR) algorithms (Wang et al., 2013, 2014a,b). Then some methods extend the idea by simultaneously learning logic rules and the weights in a differentiable way. For example, some works (Rocktäschel and Riedel, 2017; Yang et al., 2017; Sadeghian et al., 2019) try to use neural logic programming to model rule-based reasoning and to learn high-quality logical rules. Another kind of rule-learning method is based on reinforcement learning. The basic idea is to train a path-finding agent, which is used to search for reasoning paths over knowledge graphs to answer queries, and then extract logic rules from reasoning paths. Some representative works include DeepPath (Xiong et al., 2017b), MINERVA (Das et al., 2018), M-Walk (Shen et al., 2018) and R5 (Lu et al., 2022). Recently, RNNLogic (Qu et al., 2021) solves this problem by training a generator and a predictor alternately. RLgoic (Cheng et al., 2022) learns rules in a recursive manner.

6 Conclusion

Learning logic rules for knowledge graph reasoning is crucial, as they offer interpretable explanations for the process of reasoning. We propose a novel framework for learning logic rules in latent space. Concretely, we introduce a relational path autoencoder to map paths into latent space and a rule discriminator to access the consistency between rule bodies and rule heads. With the sampler based on ODE, LATENTLOGIC can generate logic rules efficiently. Experimental results showed that LATENTLOGIC outperforms strong baseline methods and is efficient for training.

Acknowledgements

We thank the anonymous reviewers for their insightful feedback. This work is supported by the National Natural Science Foundation of China (No.U20B2053). We thank the Beijing Advanced Innovation Center for Big Data and Brain Computing for its computational resources.

Limitations

In this paper, we introduce a generative framework for rule mining on knowledge graphs. We believe this approach still has much room for improvement: 1) We have not used the more complicated model architecture of the rule discriminator. 2) The proposed sampling strategy may lead to some same relational paths or out-of-domain relational paths. For future work, finding a way to incorporate a more advanced rule discriminator and prevent generating invalid relational paths is worth exploring.

References

Ivana Balazevic, Carl Allen, and Timothy M. Hospedales. 2019. Tucker: Tensor factorization for knowledge graph completion. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 5184–5193. Association for Computational Linguistics.

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 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 2787–2795.

Ricky T. Q. Chen, Brandon Amos, and Maximilian Nickel. 2021. Learning neural event functions for ordinary differential equations. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. 2018. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 6572–6583.

Kewei Cheng, Jiahao Liu, Wei Wang, and Yizhou Sun. 2022. Rlogic: Recursive logical rule learning from knowledge graphs. In *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*, pages 179–189. ACM.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. 2018. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Rajarshi Das, Arvind Neelakantan, David Belanger, and Andrew McCallum. 2017. Chains of reasoning over entities, relations, and text using recurrent neural networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 1: Long Papers*, pages 132–141. Association for Computational Linguistics.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 1811–1818. AAAI Press.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Xiao Huang, Jingyuan Zhang, Dingcheng Li, and Ping Li. 2019. Knowledge graph embedding based question answering. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, WSDM 2019, Melbourne, VIC, Australia, February 11-15, 2019*, pages 105–113. ACM.

Diederik P. Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.

Stanley Kok and Pedro M. Domingos. 2005. Learning the structure of markov logic networks. In *Machine Learning, Proceedings of the Twenty-Second International Conference (ICML 2005), Bonn, Germany, August 7-11, 2005*, volume 119 of *ACM International Conference Proceeding Series*, pages 441–448. ACM.

Bhushan Kotnis, Carolin Lawrence, and Mathias Niepert. 2021. Answering complex queries in knowledge graphs with bidirectional sequence encoders. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 4968–4977. AAAI Press.

Ni Lao and William W. Cohen. 2010. Relational retrieval using a combination of path-constrained random walks. *Mach. Learn.*, 81(1):53–67.

Ni Lao, Tom M. Mitchell, and William W. Cohen. 2011. Random walk inference and learning in A large scale knowledge base. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, 27-31 July 2011, John McIntyre Conference Centre, Edinburgh, UK, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 529–539. ACL.

Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and Fujie Huang. 2006. A tutorial on energy-based learning. *Predicting structured data*, 1(0).

Sangkeun Lee, Sungchan Park, Minsuk Kahng, and Sang-goo Lee. 2013. Pathrank: Ranking nodes on a heterogeneous graph for flexible hybrid recommender systems. *Expert Syst. Appl.*, 40(2):684–697.

Ruizhe Li, Xiao Li, Guanyi Chen, and Chenghua Lin. 2020. Improving variational autoencoder for text modelling with timestep-wise regularisation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2381–2397.

Yizhi Li, Wei Fan, Chao Liu, Chenghua Lin, and Jiang Qian. 2022. TranSHER: Translating knowledge graph embedding with hyper-ellipsoidal restriction. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8517–8528.

Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2018. Multi-hop knowledge graph reasoning with reward shaping. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 3243–3253. Association for Computational Linguistics.

Xiao Liu, Shiyu Zhao, Kai Su, Yukuo Cen, Jiezhong Qiu, Mengdi Zhang, Wei Wu, Yuxiao Dong, and Jie Tang. 2022. Mask and reason: Pre-training knowledge graph transformers for complex logical queries. In *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*, pages 1120–1130. ACM.

Shengyao Lu, Bang Liu, Keith G. Mills, Shangling Jui, and Di Niu. 2022. R5: rule discovery with reinforced and recurrent relational reasoning. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Denis Lukovnikov, Asja Fischer, Jens Lehmann, and Sören Auer. 2017. Neural network-based question answering over knowledge graphs on word and character level. In *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017*, pages 1211–1220. ACM.

Weili Nie, Arash Vahdat, and Anima Anandkumar. 2021. Controllable and compositional generation with latent-space energy-based models. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 13497–13510.

Meng Qu, Junkun Chen, Louis-Pascal A. C. Xhonneux, Yoshua Bengio, and Jian Tang. 2021. Rnnlogic: Learning logic rules for reasoning on knowledge graphs. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Matthew Richardson and Pedro M. Domingos. 2006. Markov logic networks. *Mach. Learn.*, 62(1-2):107–136.

Tim Rocktäschel and Sebastian Riedel. 2017. End-to-end differentiable proving. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 3788–3800.

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 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 15321–15331.

Eric Salvat and Marie-Laure Mugnier. 1996. Sound and complete forward and backward chainingd of graph rules. In *Conceptual Structures: Knowledge Representation as Interlingua, 4th International Conference on Conceptual Structures, ICCS '96, Sydney, Australia, August 19-22, 1996, Proceedings*, volume 1115 of *Lecture Notes in Computer Science*, pages 248–262. Springer.

Yelong Shen, Jianshu Chen, Po-Sen Huang, Yuqing Guo, and Jianfeng Gao. 2018. M-walk: Learning to walk over graphs using monte carlo tree search. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 6787–6798.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. 2021. Score-based generative modeling through stochastic differential equations. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Frank Ludwig Spitzer. 1975. *Principles of Random Walk*. Principles of Random Walk.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Chen Tang, Hongbo Zhang, Tyler Loakman, Chenghua Lin, and Frank Guerin. 2023a. Enhancing dialogue generation via dynamic graph knowledge aggregation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4604–4616.

Chen Tang, Hongbo Zhang, Tyler Loakman, Chenghua Lin, and Frank Guerin. 2023b. Terminology-aware medical dialogue generation. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE.

Komal K. Teru and William L. Hamilton. 2019. Inductive relation prediction on knowledge graphs. *CoRR*, abs/1911.06962.

Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality, CVSC 2015, Beijing, China, July 26-31, 2015*, pages 57–66. Association for Computational Linguistics.

Hongwei Wang, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2018. Ripplenet: Propagating user preferences on the knowledge graph for recommender systems. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018, Torino, Italy, October 22-26, 2018*, pages 417–426. ACM.

William Yang Wang, Kathryn Mazaitis, and William W. Cohen. 2013. Programming with personalized pagerank: a locally groundable first-order probabilistic logic. In *22nd ACM International Conference on Information and Knowledge Management, CIKM'13, San Francisco, CA, USA, October 27 - November 1, 2013*, pages 2129–2138. ACM.

William Yang Wang, Kathryn Mazaitis, and William W. Cohen. 2014a. Proppr: Efficient first-order probabilistic logic programming for structure discovery, parameter learning, and scalable inference. In *Statistical Relational Artificial Intelligence, Papers from the 2014 AAAI Workshop, Québec City, Québec, Canada, July 27, 2014*, volume WS-14-13 of *AAAI Technical Report*. AAAI.

William Yang Wang, Kathryn Mazaitis, and William W. Cohen. 2014b. Structure learning via parameter learning. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM 2014, Shanghai, China, November 3-7, 2014*, pages 1199–1208. ACM.

Chenyan Xiong, Russell Power, and Jamie Callan. 2017a. Explicit semantic ranking for academic search via knowledge graph embedding. In *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017*, pages 1271–1279. ACM.

Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017b. Deeppath: A reinforcement learning method for knowledge graph reasoning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 564–573. Association for Computational Linguistics.

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 *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

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 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 2319–2328.

Yuyu Zhang, Xinshi Chen, Yuan Yang, Arun Ramamurthy, Bo Li, Yuan Qi, and Le Song. 2020. Efficient probabilistic logic reasoning with graph neural networks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. ERNIE: enhanced language representation with informative entities. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 1441–1451. Association for Computational Linguistics.

7 Appendix

7.1 Dataset Statistics

In this part, we provide a brief introduction and statistics of FB15k-237 and WN18RR, which are used in this paper.

- FB15k-237 is one of the most commonly used benchmark knowledge graph datasets, which is an online collection of structured data harvested from many sources, including individual, user-submitted wiki contributions.

- WN18RR is recognized as a widely adopted benchmark knowledge graph dataset. Its purpose is to generate a user-friendly dictionary and thesaurus and facilitate automatic text analysis. The entities in the dataset represent word senses, while relationships define the lexical relations between these senses.

Dataset	# Data	# Relation	# Entity
FB15k-237	310,116	237	14,541
WN18RR	93,003	11	40,943

Table 3: Dataset statistics.

7.2 Implementation Details

We implement LATENTLOGIC over Pytorch[2]. We use the Adam optimizer to train both the relational path autoencoder and the rule discriminator. A grid search is performed to determine the best hyperparameters based on the performance on the validation sets. We employ the *dopri5* neural ODE solver, with $(10^{-3}, 10^{-3})$ tolerances, and set $T = 1$, $\beta_{\text{min}} = 0.1$, and $\beta_{\text{max}} = 20$. The maximum length of relational paths is set to 3 in FB15k-237 and 2 in WN18RR. All experiments are executed on a single Nvidia Tesla V100 GPU.

7.3 Details of Forward Chain

We provide the details of how to use *forward chain* to infer missing facts given learned logic rules. Specifically, Given a query (h, r, t), let $\mathcal{A}$ be the set of candidate answers that can be discovered by any learned rule using forward chaining. For each candidate answer $a \in \mathcal{A}$, the score of triple (h, r, a) can be calculated as $\sum_{rule} \sum_{path} \phi(rule)$, where ϕ is the confidence score of logic rule. Then we can rank candidate answers by the scores.

[2] https://pytorch.org/

Methods	FB15k-237↓ (mins)	WN18RR↓ (mins)
NerualLP	395	122.3
DRUM	373.8	118.7
RNNLogic	331.6	100.3
LATENTLOGIC	24.6	12.4

Table 4: Training time running on a single NVIDIA Tesla V100 GPU.

7.4 Supplementary Experiment

Evaluation of Training Efficiency To showcase the efficiency of LATENTLOGIC, we compare the training time of different models in Tab. 4. For a fair comparison, we utilize a single GPU (Tesla V100) to train each model and implement the hyperparameters recommended by the original papers. Our observations are as follows: (i) LATENTLOGIC outperforms the baselines in efficiency while still achieving competitive performance. (ii) NeuralLP and DRUM do not perform well due to their involvement in large matrix multiplication. (iii) RNNLogic is also less efficient because of the EM-based optimization procedure.

Sensitivity of Randomness Since our approach relies on random sampling techniques, we would like to conduct experiments to examine the sensitivity of LATENTLOGIC to randomness. In Tab.5, we report the mean and standard deviations of the results under different random seeds. We can notice that the randomness sampling has minimal impact on the performance of LATENTLOGIC.

	FB15k-237			WN18RR		
	MRR	H@1	H@10	MRR	H@1	H@10
μ	0.320	21.2	51.4	0.481	45.2	55.3
σ	0.013	0.118	0.247	0.009	0.211	0.187

Table 5: result w.r.t. randomness.