Variational Knowledge Graph Reasoning

Wenhu Chen, Wenhan Xiong, Xifeng Yan, William Yang Wang
Department of Computer Science
University of California, Santa Barbara
Santa Barbara, CA 93106
{wenhuchen,xwhan,xyan,william}@cs.ucsb.edu

Abstract

Inferring missing links in knowledge graphs (KG) has attracted a lot of attention from the research community. In this paper, we tackle a practical query answering task involving predicting the relation of a given entity pair. We frame this prediction problem as an inference problem in a probabilistic graphical model and aim at resolving it from a variational inference perspective. In order to model the relation between the query entity pair, we assume that there exist underlying latent variables (assemble of all paths connecting these two nodes) in the KG, which carries the equivalent semantics of their relation. However, due to the intractability of connections in large KGs, we propose to use variation inference to maximize the evidence lower bound. More specifically, our framework (DIVA) is composed of three modules, i.e. a posterior approximator, a prior (path finder), and a likelihood (path reasoner). By using variational inference, we are able to incorporate them closely into a unified architecture and jointly optimize them to perform KG reasoning. With active interactions among these sub-modules, DIVA is better at handling noise and cope with more complex reasoning scenarios. In order to evaluate our method, we conduct the experiment of the link prediction task on NELL-995 and FB15K datasets and achieve state-of-the-art performances on both datasets.

1 Introduction

Automated reasoning, the ability for computing systems to make new inferences from the observed evidence, has attracted lots of attention from the research community. In recent years, there are surging interests in designing machine learning algorithms for complex reasoning tasks, especially in large knowledge graphs (KGs) where the countless entities and links have posed great challenges to traditional logic-based algorithms. Specifically, we situate our study in this large KG multi-hop reasoning scenario, where the goal is to design an automated inference model to complete the missing links between existing entities in large KGs. For examples, if the KG contains a fact like *president*(*BarackObama*, *USA*) and *spouse*(*Michelle, BarackObama*), then we would like the machines to complete the missing link *livesIn*(*Michelle*, *USA*) automatically. Systems for this task are essential to complex question answering applications.

To tackle the multi-hop link prediction problem, various approaches have been proposed. Some earlier works like PRA (Lao et al., 2011; Gardner et al., 2014, 2013) use bounded-depth random walk with restarts to obtain paths. More recently, DeepPath (Xiong et al., 2017) and MINERVA (Das et al., 2017), frame the path-finding problem as a Markov Decision Process (MDP) and utilize reinforcement learning (RL) to maximize the expected return. Another line of work along with ours are Chain-of-Reasoning (Das et al., 2016) and Compositional Reasoning (Neelakantan et al., 2015), which take multi-hop chains learned by PRA as input and aim to infer its relation.

Here we frame the KG reasoning task as a two sub-steps, i.e. "Path-Finding" and "Path-Reasoning". We found that most of the related research is only focused on one step, which contributes to major drawbacks—lack of interactions between these two steps. More specifically, DeepPath (Xiong et al., 2017) and MINERVA (Das et al., 2017) can be interpreted as enhancing the "Path-Finding" step while compositional reasoning (Neelakantan et al., 2015) and chains of reasoning (Das et al., 2016) can be interpreted as enhancing the "Path-Reasoning" step. DeepPath is trained to paths more efficiently between two given entities while being agnostic to whether the entity pairs are positive or negative, whereas MIN-

ERVA learns to reach target nodes given an entity-query pair while being agnostic to the quality of the searched path[1]. In contrast, chains of reasoning and compositional reasoning only learn to predict relation given paths while being agnostic to the path-finding procedure. The lack of interaction prevents the model from understanding more diverse inputs and make the model very sensitive to noise and adversarial samples.

In order to increase the robustness of existing KG reasoning model and handle more noisy environments, we propose to combine these two steps together as a whole from a perspective of the latent variable graphic model. This graphic model views the paths as discrete latent variables and relation as the observed variables with given entity pair as condition, thus the path-finding module can be viewed as a prior distribution to infer the underlying links in the KG. In contrast, the path-reasoning module can be viewed as the likelihood distribution, which classifies underlying links into multiple classes. With this assumption, we introduce an approximate posterior and design a variational auto-encoder (Kingma and Welling, 2013) algorithm to maximize the evidence lower-bound. This variational framework closely incorporates two modules into a unified framework and jointly train them together. By active cooperations and interactions, the path finder can take into account the value of searched path and resort to the more meaningful paths. Meanwhile, the path reasoner can receive more diverse paths from the path finder and generalizes better to unseen scenarios. Our contributions are three-fold:

- We introduce a variational inference framework for KG reasoning, which tightly integrates the path-finding and path-reasoning processes to perform joint reasoning.
- We have successfully leveraged negative samples into training and increase the robustness of existing KG reasoning model.
- We show that our method can scale up to large KG and achieve state-of-the-art results in two tasks.

The rest of the paper is organized as follow. In Section 2 we will outline related work on KG embedding, multi-hop reasoning, and variational auto-encoder. We describe our variational knowledge reasoner DIVA in Section 3. Experimental results are presented in Section 4, and we conclude in Section 5.

[1]MINERVA assigns constant rewards to all paths reaching the destination.

2 Related Work

2.1 Knowledge Graph Embeddings

Embedding methods to model multi-relation data from KGs have been extensively studied in recent years (Nickel et al., 2011; Bordes et al., 2013; Socher et al., 2013; Lin et al., 2015; Trouillon et al., 2017). From a representation learning perspective, all these methods are trying to learn a projection from symbolic space to vector space. For each triple (e_s, r, e_d) in the KG, various score functions can be defined using either vector or matrix operations. Although these embedding approaches have been successful capturing the semantics of KG symbols (entities and relations) and achieving impressive results on knowledge base completion tasks, most of them fail to model multi-hop relation paths, which are indispensable for more complex reasoning tasks. Besides, since all these models operate solely on latent space, their predictions are barely interpretable.

2.2 Multi-Hop Reasoning

The Path-Ranking Algorithm (PRA) method is the first approach to use a random walk with restating mechanism to perform multi-hop reasoning. Later on, some research studies (Gardner et al., 2014, 2013) have revised the PRA algorithm to compute feature similarity in the vector space. These formula-based algorithms can create a large fan-out area, which potentially undermines the inference accuracy. To mitigate this problem, a CNN-based model (Toutanova et al., 2015) has been proposed to perform multi-hop reasoning. Recently, DeepPath (Xiong et al., 2017) and MINERVA (Das et al., 2017) view the multi-hop reasoning problem as a Markov Decision Process, and leverages REINFORCE (Williams, 1992) to efficiently search for paths in large knowledge graph. These two methods are reported to achieve state-of-the-art results, however, these two models both use heuristic rewards to drive the policy search, which could make their models sensitive to noises and adversarial examples.

2.3 Variational Auto-encoder

Variational auto-encoder (Kingma and Welling, 2013) is a very popular algorithm to perform approximate posterior inference in large-scale scenarios, especially in neural networks. Recently, VAE has been used with success for various complex machine learning tasks like image generation (Mansimov et al., 2015), machine translation (Zhang et al., 2016), sentence generation (Guu et al., 2017a) and question answering (Zhang et al., 2017). Zhang et al. (2017) is closest to ours, this paper proposes a variational framework to understand the variability of human language about entity referencing. In contrast, our model uses a variational framework to cope with the complex link connections in large KG. Unlike the previous research in VAE, both Zhang et al. (2017) and our model uses discrete variables as the latent representation to infer the semantics of given entity pairs. More specifically, we view the generation of relation as a stochastic process controlled by a latent representation, i.e. the connected multi-hop link existed in the KG. Though the potential link paths are discrete and countable, its amount is still very large and poses challenges to direct optimization. Therefore, we resort to variational auto-encoder as our approximation strategy.

3 Our Approach

3.1 Background

Here we formally define the background of our task. Let $\mathcal{E}$ be the set of entities and R be the set of relations. Then a KG is defined as a collection of triple facts (e_s, r, e_d), where $e_s, e_d \in \mathcal{E}$ and $r \in \mathcal{R}$. We are particularly interested in the problem of relation inferring, which seeks to answer the question in the format of $(e_s, ?, e_d)$, the problem setting is slightly different from standard link prediction to answer the question of $(e_s, r, ?)$. Next, in order to tackle this classification problem, we assume that there is a latent representation for given entity pair in the KG, i.e. the collection of linked paths, these hidden variables can reveal the underlying semantics between these two entities. Therefore, the link classification problem can be decomposed into two modules – acquire underlying paths (Path Finder) and infer relation from latent representation (Path Reasoner).

Path Finder The state-of-the-art approach (Lao et al., 2011; Xiong et al., 2017; Das et al., 2017) is to view this process as a Markov Decision Process (MDP). A tuple $< S, A, P >$ is defined to represent the MDP, where S denotes the current state, e.g. the current node in the knowledge graph, A is the set of available actions, e.g. all the outgoing edges from the state, while P is the transition probability describing the state transition mechanism. In the knowledge graph, the transition of the state is deterministic, so we do not need to model the state transition P.

Path Reasoner The common approach (Lao et al., 2011; Neelakantan et al., 2015; Das et al., 2016) is to encode the path as a feature vector and use a multi-class discriminator to predict the unknown relation. PRA (Lao et al., 2011) proposes to encode paths as binary features to learn a log-linear classifier, while (Das et al., 2016) applies recurrent neural network to recursively encode the paths into hidden features and uses vector similarity for classification.

3.2 Variational KG Reasoner (DIVA)

Here we draw a schematic diagram of our model in Figure 1. Formally, we define the objective function for the general relation classification problem as follows:

$$\begin{aligned} Obj &= \sum_{(e_s,r,e_d)\in D} \log p(r|(e_s,e_d)) \\ &= \sum_{(e_s,r,e_d)\in D} \log \sum_{L} p_\theta(L|(e_s,e_d))p(r|L) \end{aligned} \tag{1}$$

where D is the dataset, (e_s, r, e_d) is the triple contained in the dataset, and L is the latent connecting paths. The evidence probability $p(r|(e_s, e_d))$ can be written as the marginalization of the product of two terms over the latent space. However, this evidence probability is intractable since it requires summing over the whole latent link space. Therefore, we propose to maximize its variational lower bound as follows:

$$\begin{aligned} ELBO = &\underset{L\sim q_\varphi(L|r,(e_s,e_d))}{E}[\log p_\theta(r|L)] - \\ &KL(q_\varphi(L|r,(e_s,e_d))||p_\beta(L|(e_s,e_d))) \end{aligned} \tag{2}$$

Specifically, ELBO is composed of three different terms – likelihood $p_\theta(r|L)$, prior $p_\beta(L|(e_s, e_t))$,

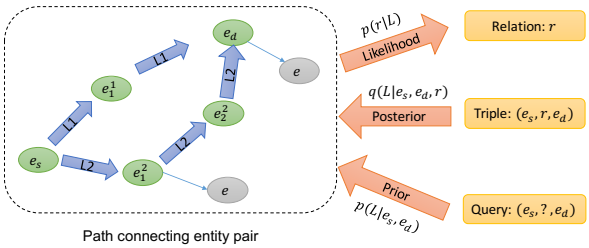

Figure 1: The probabilistic graphical model of our proposed approach. Arrows with dotted border represent the approximate posterior, which is modeled as a multinomial distribution over the whole link space. Arrows with solid border represent the prior and likelihood distributions.

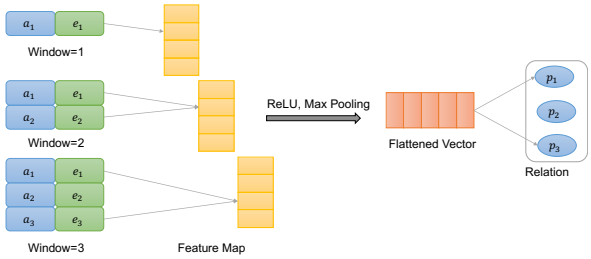

Figure 2: Overview of the CNN Path Reasoner.

and posterior $q_\varphi(L|(e_s, e_d), r)$. In this paper, we use three neural network models to describe these terms and then follow (Kingma and Welling, 2013) to apply variational auto-encoder to maximize the approximate lower bound. We describe these three models in details below:

Path Reasoner (Likelihood). Here we propose a path reasoner using Convolutional Neural Networks (CNN) (LeCun et al., 1995) and a feed-forward neural network. This model takes path sequence $L = \{a_1, e_1, \cdots, a_i, e_i, \cdots a_n, e_n\}$ to output a softmax probability over the relations set R, where a_i denotes the i-th intermediate relation and e_i denotes the i-th intermediate entity between the given entity pair. Here we first project them into embedding space and concatenate i_{th} relation embedding with i-th entity embedding as a combined vector, which we denote as $\{f_1, f_2, \cdots, f_n\}$ and $f_i \in \mathcal{R}^{2E}$. As shown in Figure 2, we pad the embedding sequence to a length of N. Then we design three convolution layers with window size of $(1 \times 2E), (2 \times 2E), (3 \times 2E)$, input channel size 1 and filter size D. After the convolution layer, we use $(N \times 1), (N - 1 \times 1), (N - 2 \times 1)$ to max pool the convolution feature map. Finally, we concatenate the three vectors as a combined vector $F \in \mathcal{R}^{3D}$. Finally, we use two-layered MLP with intermediate hidden size of M to output a softmax distribution over all the relations set R.

$$\begin{aligned} F &= f(f_1, f_2, \cdots, f_N) \\ p(r|L;\theta) &= softmax(W_r F + b_r) \end{aligned} \tag{3}$$

where f denotes the convolution and max-pooling function applied to extract reasoning path feature F, and W_r, b_r denote the weights and bias for the output feed-forward neural network.

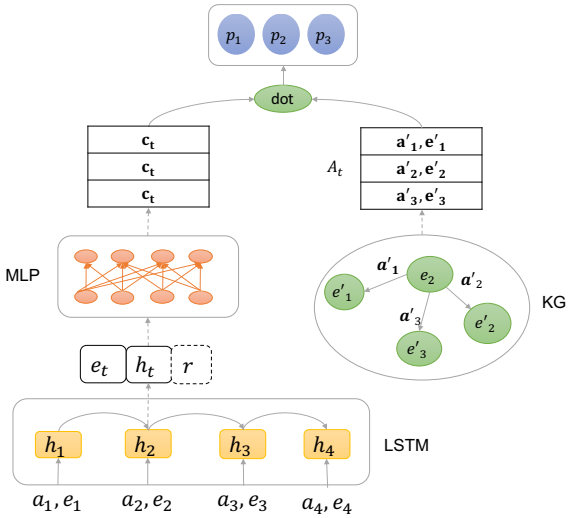

Figure 3: An overview of the path finder model. Note that r_q exists in the approximate posterior while disappears in the path finder model.

Path Finder (Prior). Here we formulate the path finder $p(L|(e_s, e_d))$ as an MDP problem, and recursively predict actions (an outgoing relation-entity edge (a, e)) in every time step based on the previous history h_{t-1} as follows:

$$\begin{aligned} c_t &= ReLU(W_h[h_t; e_d] + b_h) \\ p((a_{t+1}, e_{t+1})|h_t, \beta) &= softmax(A_t c_t) \end{aligned} \tag{4}$$

where the $h_t \in \mathcal{R}^H$ denotes the history embedding, $e_d \in \mathcal{R}^E$ denotes the entity embedding, $A_t \in \mathcal{R}^{|A| \times 2E}$ is outgoing matrix which stacks the concatenated embeddings of all outgoing edges and $|A|$ denotes the number of outgoing edge, we use W_h and b_h to represent the weight and bias of the feed-forward neural network outputting feature vector $c_t \in \mathcal{R}^{2E}$. The history embedding h_t is obtained using an LSTM network (Hochreiter and Schmidhuber, 1997) to encode all the previous decisions as follows:

$$h_t = LSTM(h_{t-1}, (a_t, e_t)) \tag{5}$$

As shown in Figure 3, the LSTM-based path finder interacts with the KG in every time step and decides which outgoing edge (a_{t+1}, e_{t+1}) to follow, search procedure will terminate either the target node is reached or the maximum step is reached.

Approximate Posterior. We formulate the posterior distribution $q(L|(e_s, e_d), r)$ following the similar architecture as the prior. The main difference lies in the fact that posterior approximator is aware of the relation r, therefore making more relevant decisions. The posterior borrows the history vector from finder as h_t, while the feed-forward neural network is distinctive in that it takes the relation embedding also into account. Formally, we write its outgoing distribution as follows:

$$\begin{aligned} u_t &= ReLU(W_{hp}[h_t; e_d; r] + b_{hp}) \\ q((a_{t+1}, e_{t+1})|h_t; \varphi) &= softmax(A_t u_t) \end{aligned} \tag{6}$$

where W_{hp} and b_{hp} denote the weight and bias for the feed-forward neural network.

3.3 Optimization

In order to maximize the ELBO with respect to the neural network models described above, we follow VAE (Kingma and Welling, 2013) to interpret the negative ELBO as two separate losses and minimize these them jointly using a gradient descent:

Reconstruction Loss. Here we name the first term of negative ELBO as reconstruction loss:

$$J_R = \underset{L \sim q_\varphi(L|r,(e_s,e_d))}{E} [-\log p_\theta(r|L)] \tag{7}$$

this loss function is motivated to reconstruct the relation R from the latent variable L sampled from approximate posterior, optimizing this loss function jointly can not only help the approximate posterior to obtain paths unique to particular relation r, but also teaches the path reasoner to reason over multiple hops and predict the correct relation.

KL-divergence Loss. We name the second term as KL-divergence loss:

$$J_{KL} = KL(q_\varphi(L|r, (e_s, e_d))|p_\beta(L|(e_s, e_d))) \tag{8}$$

this loss function is motivated to push the prior distribution towards the posterior distribution. The intuition of this loss lies in the fact that an entity pair already implies their relation, therefore, we can teach the path finder to approach the approximate posterior as much as possible. During test-time when we have no knowledge about relation, we use path finder to replace posterior approximator to search for high-quality paths.

Derivatives. We show the derivatives of the loss function with respect to three different models. For the approximate posterior, we re-weight the KL-diverge loss and design a joint loss function as follows:

$$J = J_R + w_{KL} J_{KL} \tag{9}$$

where w_{KL} is the re-weight factor to combine these two losses functions together. Formally, we write the derivative of posterior as follows:

$$\frac{\partial J}{\partial \varphi} = \underset{L \sim q_\varphi(L))}{E} [-f_{re}(L) \frac{\partial \log q_\varphi(L|(e_s, e_d), r)}{\partial \varphi} \tag{10}$$

where $f_{re}(L) = \log p_\theta + w_{KL} \log \frac{p_\beta}{q_\varphi}$ denotes the probability assigned by path reasoner. In practice, we found that the KL-reward term $\log \frac{p_\beta}{q_\varphi}$ causes severe instability during training, so we finally leave this term out by setting w_{KL} as 0. For the path reasoner, we also optimize its parameters with regard to the reconstruction as follows:

$$\frac{\partial J_R}{\partial \theta} = \underset{L \sim q_\varphi(L)}{E} - \frac{\partial \log p_\theta(r|L)}{\partial \theta} \tag{11}$$

For the path finder, we optimize its parameters with regard to the KL-divergence to teach it to infuse the relation information into the found links.

$$\frac{\partial J_{KL}}{\partial \beta} = \underset{L \sim q_\varphi(L)}{E} - \frac{\partial \log p_\beta(L|(e_s, e_d))}{\partial \beta} \tag{12}$$

Train & Test During training time, in contrast to the preceding methods like Das et al. (2017); Xiong et al. (2017), we also exploit negative samples by introducing an pseudo "n/a" relation, which indicates "no-relation" between two entities. Therefore, we manage to decompose the data sample $(e_q, r_q, [e_1^-, e_2^-, \cdots, e_n^+])$ into a series of tuples (e_q, r'_q, e_i), where $r'_q = r_q$ for positive samples and $r'_q = n/a$ for negative samples. During training, we alternatively update three submodules with SGD. During test, we apply the path-finder to beam-search the top paths for all tuples and rank them based on the scores assign by path-reasoner. More specifically, we demonstrate the pseudo code in Algorithm 1.

Algorithm 1 The DIVA Algorithm.

procedure TRAINING & TESTING
Train:
  for episode $\leftarrow$ 1 to N **do**
    Rollout 20 paths from posterior p_φ
    if Train-Posterior **then**
      $\varphi \leftarrow \varphi - \eta \times \frac{\partial L_r}{\partial \varphi}$
    else if Train-Likelihood **then**
      $\theta \leftarrow \theta - \eta \times \frac{\partial L_r}{\partial \theta}$
    else if Train-Prior **then**
      $\beta \leftarrow \beta - \eta \times \frac{\partial L_{KL}}{\partial \beta}$
    end if
  end for
Test MAP:
  Restore initial parameters θ, β
  Given sample $(e_s, r_q, (e_1, e_2, \cdots, e_n))$
  $L_i \leftarrow BeamSearch(p_\beta(L|e_s, e_i))$
  $S_i \leftarrow \frac{1}{|L_i|} \sum_{l \in L_i} p_\theta(r_q|l)$
  Sort S_i and find positive rank ra^+
  $MAP \leftarrow \frac{1}{1+ra^+}$
end procedure

3.4 Discussion

We here interpret the update of the posterior approximator in equation Equation 10 as a special case of REINFORCE (Williams, 1992), where we use Monte-Carlo sampling to estimate the expected return $\log p_\theta(r|L)$ for current posterior policy. This formula is very similar to DeepPath and MINERVA (Xiong et al., 2017; Das et al., 2017) in the sense that path-finding process is described as an exploration process to maximize the policy's long-term reward. Unlike these two models assigning heuristic rewards to the policy, our model assigns model-based reward $\log p_\theta(r|L)$, which is known to be more sophisticated and considers more implicit factors to distinguish between good and bad paths. Besides, our update formula for path reasoner Equation 11 is also similar to chain-of-reasoning (Das et al., 2016), both models are aimed at maximizing the likelihood of relation given the multi-hop chain. However, our model is distinctive from theirs in a sense that the obtained paths are sampled from a dynamic policy, by exposing more diverse paths to the path reasoner, it can generalize to more conditions. By the active interactions and collaborations of two models, DIVA is able to comprehend more complex inference scenarios and handle more noisy environments.

Dataset	#Ent	#R	#Triples	#Tasks
FB15k-237	14,505	237	310,116	20
NELL-995	75,492	200	154,213	12

Table 1: Dataset statistics.

4 Experiments

To evaluate the performance of DIVA, we explore the standard link prediction task on two different-sized KG datasets and compare with the state-of-the-art algorithms. Link prediction is to rank a list of target entities $(e_1^-, e_2^-, \cdots, e_n^+)$ given a query entity e_q and query relation r_q. The dataset is arranged in the format of $(e_q, r_q, [e_1^-, e_2^-, \cdots, e_n^+])$, and the evaluation score (Mean Averaged Precision, MAP) is based on the ranked position of the positive sample.

4.1 Dataset and Setting

We perform experiments on two datasets, and the details of the statistics are described in Table 1. The samples of FB15k-237 (Toutanova et al., 2015) are sampled from FB15k (Bordes et al., 2013), here we follow DeepPath (Xiong et al., 2017) to select 20 relations including Sports, Locations, Film, etc. Our NELL dataset is downloaded from the released dataset[2], which contains 12 relations for evaluation. Besides, both datasets contain negative samples obtained by using the PRA code released by Lao et al. (2011). For each query r_q, we remove all the triples with r_q and r_q^{-1} during reasoning. During training, we set number of rollouts to 20 for each training sample and update the posterior distribution using Monte-Carlo REINFORCE (Williams, 1992) algorithm. During testing, we use a beam of 5 to approximate the whole search space for path finder. We follow MINERVA (Das et al., 2017) to set the maximum reasoning length to 3, which lowers the burden for the path-reasoner model. For both datasets, we set the embedding size E to 200, the history embedding size H to 200, the convolution kernel feature size D to 128, we set the hidden size of MLP for both path finder and path reasoner to 400.

4.2 Quantitative Results

We mainly compare with the embedding-based algorithms (Bordes et al., 2013; Lin et al., 2015; Ji et al., 2015; Wang et al., 2014), PRA (Lao et al., 2011), MINERVA (Das et al., 2017),

[2]https://github.com/xwhan/DeepPath

Model	12-rel MAP	9-rel MAP
RPA (Lao et al., 2011)	67.5	-
TransE (Bordes et al., 2013)	75.0	-
TransR (Lin et al., 2015)	74.0	-
TransD (Ji et al., 2015)	77.3	-
TransH (Wang et al., 2014)	75.1	-
MINERVA (Das et al., 2017)	-	**88.2**
DeepPath (Xiong et al., 2017)	79.6	80.2
RNN-Chain (Das et al., 2016)	79.0	80.2
CNN Path-Reasoner	82.0	82.2
DIVA	**88.6**	87.9

Table 2: MAP results on the NELL dataset. Since MINERVA (Das et al., 2017) only takes 9 relations out of the original 12 relations, we report the known results for both version of NELL-995 dataset.

Model	20-rel MAP
PRA (Lao et al., 2011)	54.1
TransE (Bordes et al., 2013)	53.2
TransR (Lin et al., 2015)	54.0
MINERVA (Das et al., 2017)	55.2
DeepPath (Xiong et al., 2017)	57.2
RNN-Chain (Das et al., 2016)	51.2
CNN Path-Reasoner	54.2
MML (Guu et al., 2017b)	58.7
DIVA	**59.8**

Table 3: Results on the FB15k dataset, please note that MINERVA's result is obtained based on our own implementation.

DeepPath (Xiong et al., 2017) and Chain-of-Reasoning (Das et al., 2016), besides, we also take our standalone CNN path-reasoner from DIVA. Besides, we also try to directly maximize the marginal likelihood $p(r|e_s, e_d) = \sum_L p(L|e_s, e_d)p(r|L)$ using only the prior and likelihood model following MML (Guu et al., 2017b), which enables us to understand the superiority of introducing an approximate posterior. Here we first report our results for NELL-995 in Table 2, which is known to be a simple dataset and many existing algorithms already approach very significant accuracy. Then we test our methods in FB15k (Toutanova et al., 2015) and report our results in Table 3, which is much harder than NELL and arguably more relevant for real-world scenarios.

Besides, we also evaluate our model on FB-15k 20-relation subset with HITS@N score. Since our model only deals with the relation classification problem $(e_s, ?, e_d)$ with e_d as input, so it's hard for us to directly compare with MINERVA (Das et al., 2017). However, here we compare with chain-RNN (Das et al., 2016) and CNN Path-Reasoner model, the results are demonstrated as Table 4. Please note that the HITS@N score is computed against relation rather than entity.

Model	HITS@3	HITS@5
RNN-Chain (Das et al., 2016)	0.80	0.82
CNN Path-Reasoner	0.82	0.83
DIVA	**0.84**	**0.86**

Table 4: HITS@N results on the FB15k dataset

Result Analysis We can observe from the above tables Table 3 and Table 2 that our algorithm has significantly outperformed most of the existing algorithms and achieves a very similar result as MINERVA (Das et al., 2017) on NELL dataset and achieves state-of-the-art results on FB15k. We conclude that our method is able to deal with more complex reasoning scenarios and is more robust to the adversarial examples. Besides, we also observe that our CNN Path-Reasoner can outperform the RNN-Chain (Das et al., 2016) on both datasets, we speculate that it is due to the short lengths of reasoning chains, which can extract more useful information from the reasoning chain.

From these two pie charts in Figure 5, we can observe that in NELL-995, very few errors are coming from the path reasoner since the path length is very small. A large proportion only contains a single hop. In contrast, most of the failures in the FB15k dataset are coming from the path reasoner, which fails to classify the multi-hop chain into correct relation. This analysis demonstrates that FB15k is much harder dataset and may be closer to real-life scenarios.

4.3 Beam Size Trade-offs

Here we are especially interested in studying the impact of different beam sizes in the link prediction tasks. With larger beam size, the path finder can obtain more linking paths, meanwhile, more noises are introduced to pose greater challenges for the path reasoner to infer the relation. With smaller beam size, the path finder will struggle to find connecting paths between positive entity pairs, meanwhile eliminating many noisy links. Therefore, we first mainly summarize three different types and investigate their changing curve under different beam size conditions:

1. No paths are found for positive samples, while paths are found for negative samples, which we denote as Neg>Pos=0.

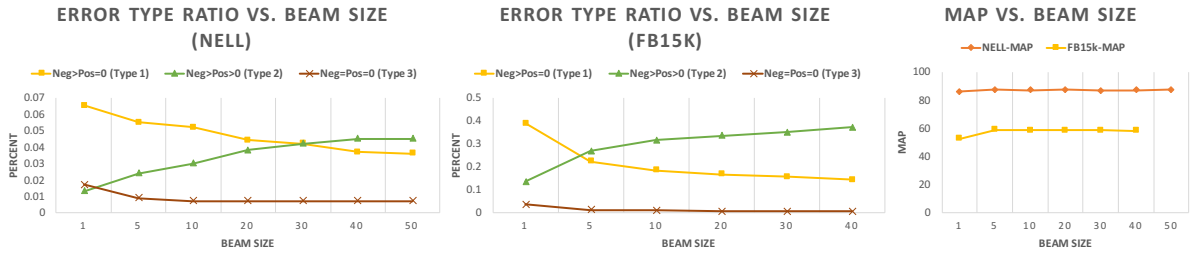

Figure 4: MAP results varying beam size and the error type's occurrence w.r.t to beam size. A beam size that is too large or too small would cause performance to drop.

Type	Reasoning Path	Score
Negative	athleteDirkNowitzki → athleteLedSportsteam → sportsteamMavericks	0.98
Positive	athleteDirkNowitzki → athleteLedSportsteam → sportsteamDallasMavericks	0.96
Explanation	*"maverick" is equivalent to "dallas-maverick", but treated as negative sample*	-
Negative	athleteRichHill → personBelongsToOrganization → sportsteamChicagoCubs	0.88
Positive	athleteRichHill → personBelongsToOrganization → sportsteamBlackhawks	0.74
Explanation	*Rich Hill plays in both sportsteam but the knowledge graph only include one*	-
Negative	coachNikolaiZherdev → athleteHomeStadium → stadiumOreventvenueGiantsStadium → teamhomestadium $^{-1}$ → sportsteamNewyorkGiants	0.98
Positive	coachNikolaiZherdev - athleteHomeStadium - stadiumOreventvenueGiantsStadium → teamHomestadium^{-1} → sportsteam-rangers	0.72
Explanation	*The home stadium accommodates multiple teams, therefore the logic chain is not valid*	-

Table 5: The three samples separately indicates three frequent error types, the first one belongs to "duplicate entity", the second one belongs to "missing entity", while the last one is due to "wrong reasoning".

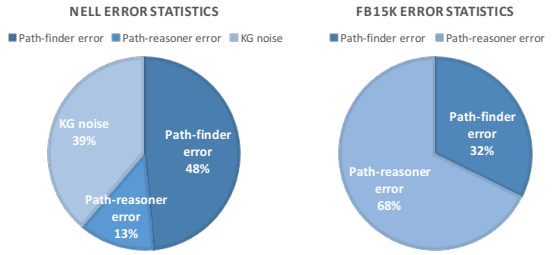

Figure 5: Error analysis for the NELL and FB15k link prediction task. Since FB15k dataset uses placeholders for entities, we are not able to analyze whether the error comes from KG noise.

2. Both positive samples and negative samples found paths, but the reasoner assigns higher scores to negative samples, which we denote as Neg>Pos>0.

3. Both negative and positive samples are not able to find paths in the knowledge graph, which we denote as Neg=Pos=0.

We draw the curves for MAP and error ratios in Figure 4 and we can easily observe the trade-offs, we found that using beam size of 5 can balance the burden of path-finder and path-reasoner optimally, therefore we keep to this beam size for the all the experiments.

4.4 Error Analysis

In order to investigate the bottleneck of DIVA, we take a subset from validation dataset to summarize the causes of different kinds of errors. Roughly, we classify errors into three categories, 1) KG noise: This error is caused by the KG itself, e.g some important relations are missing; some entities are duplicate; some nodes do not have valid outgoing edges. 2) Path-Finder error: This error is caused by the path finder, which fails to arrive destination. 3) Path-Reasoner error: This error is caused by the path reasoner to assign a higher score to negative paths. Here we draw two pie charts to demonstrate the sources of reasoning errors in two reasoning tasks.

4.5 Failure Examples

We also show some failure samples in Table 5 to help understand where the errors are coming from. We can conclude that the "duplicate entity" and "missing entity" problems are mainly caused by

the knowledge graph or the dataset, and the link prediction model has limited capability to resolve that. In contrast, the "wrong reasoning" problem is mainly caused by the reasoning model itself and can be improved with better algorithms.

5 Conclusion

In this paper, we propose a novel variational inference framework for knowledge graph reasoning. In contrast to prior studies that use a random walk with restarts (Lao et al., 2011) and explicit reinforcement learning path finding (Xiong et al., 2017), we situate our study in the context of variational inference in latent variable probabilistic graphical models. Our framework seamlessly integrates the path-finding and path-reasoning processes in a unified probabilistic framework, leveraging the strength of neural network based representation learning methods. Empirically, we show that our method has achieved the state-of-the-art performances on FB15K and the 12-relation tasks on the NELL-995 dataset.

References

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*. pages 2787–2795.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. 2017. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. *arXiv preprint arXiv:1711.05851* .

Rajarshi Das, Arvind Neelakantan, David Belanger, and Andrew McCallum. 2016. Chains of reasoning over entities, relations, and text using recurrent neural networks. *arXiv preprint arXiv:1607.01426* .

Matt Gardner, Partha Pratim Talukdar, Bryan Kisiel, and Tom Mitchell. 2013. Improving learning and inference in a large knowledge-base using latent syntactic cues .

Matt Gardner, Partha Pratim Talukdar, Jayant Krishnamurthy, and Tom Mitchell. 2014. Incorporating vector space similarity in random walk inference over knowledge bases .

Kelvin Guu, Tatsunori B Hashimoto, Yonatan Oren, and Percy Liang. 2017a. Generating sentences by editing prototypes. *arXiv preprint arXiv:1709.08878* .

Kelvin Guu, Panupong Pasupat, Evan Zheran Liu, and Percy Liang. 2017b. From language to programs: Bridging reinforcement learning and maximum marginal likelihood. *arXiv preprint arXiv:1704.07926* .

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

Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Knowledge graph embedding via dynamic mapping matrix. In *ACL (1)*. pages 687–696.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* .

Ni Lao, Tom Mitchell, and William W Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 529–539.

Yann LeCun, Yoshua Bengio, et al. 1995. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks* 3361(10):1995.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*. pages 2181–2187.

Elman Mansimov, Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. 2015. Generating images from captions with attention. *arXiv preprint arXiv:1511.02793* .

Arvind Neelakantan, Benjamin Roth, and Andrew McCallum. 2015. Compositional vector space models for knowledge base inference. In *2015 aaai spring symposium series*.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *ICML*. volume 11, pages 809–816.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in neural information processing systems*. pages 926–934.

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 *EMNLP*. volume 15, pages 1499–1509.

Théo Trouillon, Christopher R Dance, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2017. Knowledge graph completion via complex tensor factorization. *arXiv preprint arXiv:1702.06879* .

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *AAAI*. pages 1112–1119.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8(3-4):229–256.

Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. Deeppath: A reinforcement learning method for knowledge graph reasoning. *arXiv preprint arXiv:1707.06690* .

Biao Zhang, Deyi Xiong, Jinsong Su, Hong Duan, and Min Zhang. 2016. Variational neural machine translation. *arXiv preprint arXiv:1605.07869* .

Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander J Smola, and Le Song. 2017. Variational reasoning for question answering with knowledge graph. *arXiv preprint arXiv:1709.04071* .