Beta Embeddings for Multi-Hop Logical Reasoning in Knowledge Graphs

Anonymous Authors[1]

Abstract

One of the fundamental problems in Artificial Intelligence is to perform complex multi-hop logical reasoning over the facts captured by a knowledge graph (KG). This problem is challenging, because KGs can be massive and incomplete. Recent approaches embed KG entities in a low dimensional space and then use these embeddings to find the answer entities. However, it remains a challenge to handle arbitrary first-order logic (FOL) queries as present methods are limited to only a subset of FOL operators. In particular, the negation operator is not supported. An additional limitation of present methods is also that they cannot naturally model uncertainty. Here, we present BETAE, a probabilistic embedding framework for answering arbitrary FOL queries over KGs. BETAE is the first method that can handle a complete set of first-order logical operations: conjunction ($\wedge$), disjunction ($\vee$), and negation ($\neg$). A key insight of BETAE is to use probabilistic distributions with bounded support, specifically the Beta distribution, and embed queries/entities as distributions, which allows us to also faithfully model uncertainty. Logical operations are performed in the embedding space by neural operators over the probabilistic embeddings. BETAE achieves state-of-the-art performance on answering arbitrary FOL queries on three large, incomplete KGs.

1. Introduction

Reasoning in KGs is a fundamental problem in AI. In essence, it involves answering first-order logic (FOL) queries over KGs using operators existential quantification ($\exists$), conjunction ($\wedge$), disjunction ($\vee$), and negation ($\neg$). It presents a number of significant challenges. One challenge is the scale of KGs. Although queries could be in principle answered by directly traversing the KG, this is problematic in practice since multi-hop reasoning involves an exponential growth in computational time/space. Another challenge is incompleteness, where some edges between entities are missing. Most real-world KGs are incomplete and even a single missing edge may make the query unanswerable.

Previous methods (Bordes et al., 2013; Trouillon et al., 2016; Sun et al., 2019; Zhang et al., 2019; Yang et al., 2015) aim to address the above challenges by using embeddings and this way implicitly impute the missing edges. Methods also embed logical queries into various geometric shapes in the vector space (Hamilton et al., 2018; Ren et al., 2020; Guu et al., 2015; Das et al., 2017). The idea here is to design neural logical operators and embed queries iteratively by executing logical operations according to the query computation graph (Fig. 1). However, these methods only support existential positive first-order queries, a subset of FOL queries with existential quantification ($\exists$), conjunction ($\wedge$) and disjunction ($\vee$), but not negation ($\neg$). Negation is a fundamental operation and required for the complete set of FOL operators. Modeling negation so far has been a major challenge. The reason is that these methods embed queries as closed regions, e.g., a point (Hamilton et al., 2018; Guu et al., 2015; Das et al., 2017) or a box (Ren et al., 2020) in the Euclidean space, but the complement (negation) of a closed region does not result in a closed region. Furthermore, current methods embed queries as static geometric shapes and are thus unable to faithfully model uncertainty.

Here we propose *Beta Embedding (*BETAE*)*, a method for multi-hop reasoning over KGs using full first-order logic (FOL). We model both the entities and queries by probabilistic distributions with bounded support. Specifically, we embed entities and queries as Beta distributions defined on the $[0, 1]$ interval. Our approach has the following important advantages: (1) Probabilistic modeling can effectively capture the uncertainty of the queries. BETAE adaptively learns the parameters of the distributions so that the uncertainty of a given query correlates well with the differential entropy of the probabilistic embedding. (2) We design neural logical operators that operate over these Beta distributions and support full first-order logic: $\exists$, $\wedge$, $\vee$ and most importantly $\neg$. The intuition behind negation is that we can transform the parameters of the Beta distribution so that the regions of high probability density become regions of low probability

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

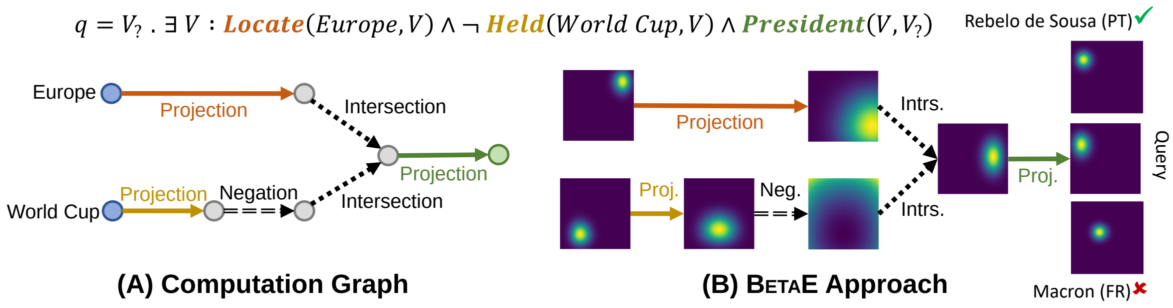

Figure 1. BETAE **answers first-order logic queries that include $\exists$, $\land$, $\lor$ and $\neg$ logical operators.** (A): A given query *"List the presidents of European countries that have never held the World Cup"* can be represented by its computation graph where each node represents a set of entities and each edge represents a logical operation. (B): BETAE models each node as Beta distributions and each edge transforms the distribution via a projection, negation, or intersection operation. Portuguese president Rebelo de Sousa is an answer entity, since its embedding is "close" to the query embedding, while the French president Macron is not.

density and vice versa. (3) Our neural modeling of $\land$ and $\neg$ naturally corresponds to the real operations and captures several properties of first-order logic. For example, applying the negation operator twice will return the same input. (4) Using the De Morgan's laws, $\lor$ can be approximated with $\land$ and $\neg$, allowing BETAE to handle a complete set of FOL operators and thus supporting arbitrary FOL queries.

We perform experiments on three standard KG datasets and show that BETAE is able to achieve state-of-the-art performance in handling conjunctive queries and the first to handle arbitrary FOL logic queries in a scalable manner.

2. Preliminaries

Knowledge Graph (KG) $\mathcal{G}$ is heterogeneous graph structure that consists of a set of entities $\mathcal{V}$ and a set of relation types $\mathcal{R}$, $\mathcal{G} = (\mathcal{V}, \mathcal{R})$. Each relation type $r \in \mathcal{R}$ is a binary function $r : \mathcal{V} \times \mathcal{V} \to \{\texttt{True}, \texttt{False}\}$ that indicates (directed) edges of relation type r between pairs of entities.

We are interested in answering first-order logic (FOL) queries with logical operations including conjunction ($\land$), disjunction ($\lor$), existential quantification ($\exists$) and negation ($\neg$) We define valid FOL queries in its disjunctive normal form (DNF), i.e., disjunction of conjunctions.

First-order logic queries: A FOL query q consists of a non-variable anchor entity set $\mathcal{V}_a \subseteq \mathcal{V}$, existentially quantified bound variables $V_1, \ldots, V_k$ and a single target variable $V_?$, which provides the query answer. The disjunctive normal form of a logical query q is a disjunction of one or more conjunctions: $q[V_?] = V_? \,.\, \exists V_1, \ldots, V_k : c_1 \lor c_2 \lor \ldots \lor c_n$, where each c represents a conjunctive query with one or more literals e. $c_i = e_{i1} \land e_{i2} \land \cdots \land e_{im}$; each literal e represents an atomic formula or its negation. $e_{ij} = r(v_a, V)$ or $\neg\, r(v_a, V)$ or $r(V', V)$ or $\neg\, r(V', V)$, where $v_a \in \mathcal{V}_a$, $V \in \{V_?, V_1, \ldots, V_k\}$, $V' \in \{V_1, \ldots, V_k\}$, $r \in \mathcal{R}$.

Computation Graph: As shown in Fig. 1, we can derive, for a given query, its corresponding computation graph (tree). This directed graph demonstrates the computation process to answer the query. Each node of the computation graph represents a distribution over a set of entities in the KG and each edge represents a logical transformation of this distribution. Specifically, the root node represents the unique target variable, which is the set of answer entities. The mapping along each edge applies a certain logical operator: (1) **Relation Projection:** Given a set of entities $S \subseteq \mathcal{V}$ and relation type $r \in \mathcal{R}$, compute adjacent entities $\cup_{v \in S} A_r(v)$ related to S via r: $A_r(v) \equiv \{v' \in \mathcal{V} : r(v, v') = \texttt{True}\}$; (2) **Intersection:** Given sets of entities $\{S_1, S_2, \ldots, S_n\}$, compute their intersection $\cap_{i=1}^n S_i$; (3) **Complement/Negation:** Given a set of entities $S \subseteq \mathcal{V}$, compute its complement $\overline{S} \equiv \mathcal{V} \setminus S$. We do not define a union operator for the computation graph, which corresponds to disjunction. However, this operator is not needed, since according to the De Morgan's laws, given sets of entities, $\cup_{i=1}^n S_i$ is equivalent to $\overline{\cap_{i=1}^n \overline{S}}$. In order to answer a given FOL query, we can follow the computation graph and execute logical operators.

3. Probabilistic Embeddings for Reasoning

3.1. Beta Embeddings for Entities and Queries

For each entity $v \in \mathcal{V}$, we assign an initial Beta embedding with learnable parameters. We also embed each query q with a Beta embedding, which is calculated by a set of probabilistic logical operators (introduced in the next section) following the computation graph. Note that BETAE learns high-dimensional embeddings where each embedding consists of multiple independent Beta distributions, capturing a different aspect of a given entity or a query: $\mathbf{S} = [(\alpha_1, \beta_1), \ldots, (\alpha_n, \beta_n)]$, where n is a hyperparameter. We denote the PDF of the i-th Beta distribution in $\mathbf{S}$ as $p_{\mathbf{S},\mathbf{i}}$. Without loss of generality and to ease explanation, we shall assume that each embedding only contains one Beta distribution: $\mathbf{S} = [(\alpha, \beta)]$, and we denote its PDF as $p_{\mathbf{S}}$.

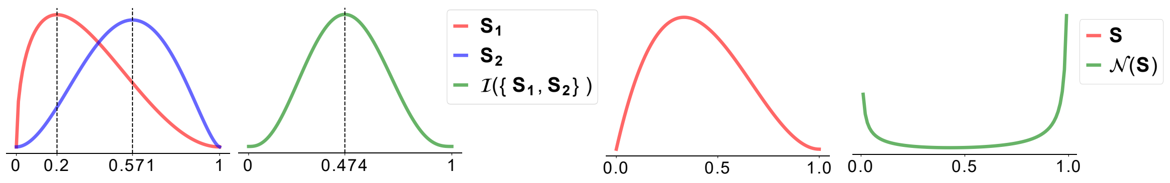

Figure 2. Illustration of our probabilistic intersection operator $\mathcal{I}$ (left) and probabilistic negation operator $\mathcal{N}$ (right). $\mathcal{I}$ transforms the input distribution by taking the weighted product of the PDFs; $\mathcal{N}$ transforms the input distribution by taking the reciprocal of its parameters.

3.2. Probabilistic Logical Operators

In order to answer a query using the computation graph, we need probabilistic logical operators for the Beta embedding.

Probabilistic Projection Operator $\mathcal{P}$: In order to model the relation projection from one distribution to another, we design a probabilistic projection operator $\mathcal{P}$ that maps from one Beta embedding $\mathbf{S}$ to another Beta embedding $\mathbf{S}'$ given the relation type r. We then learn a transformation neural network for each relation type r, which we implement as a multi-layer perceptron (MLP): $\mathbf{S}' = \texttt{MLP}_r(\mathbf{S})$. The goal here is that for all entities S covered by the input distribution, we can achieve the embedding distribution that covers entities $S' = \cup_{v \in S} A_r(v)$, where $A_r(v) \equiv \{v' \in \mathcal{V} : r(v, v') = \texttt{True}\}$.

Probabilistic Intersection Operator $\mathcal{I}$: Given n input embeddings $\{\mathbf{S_1}, \ldots, \mathbf{S_n}\}$, the goal of probabilistic intersection operator $\mathcal{I}$ is to calculate the Beta embedding $\mathbf{S}_{\texttt{Inter}}$ that represents the intersection of the distributions (i.e, the intersection of the distributions defining fuzzy input sets of entities). We model $\mathcal{I}$ by taking the weighted product of the PDFs of the input Beta embeddings $p_{\mathbf{S}_{\texttt{Inter}}} = \frac{1}{Z} \prod p_{\mathbf{S_1}}^{w_1} \ldots p_{\mathbf{S_n}}^{w_n}$, where Z is a normalization constant and $w_1, \ldots, w_n$ are the weights with their sum equal to 1.

We use the attention mechanism and learn $w_1, \ldots, w_n$ through a $\texttt{MLP}_{\texttt{Att}}$ that takes as input the parameters of $\mathbf{S_i}$ and outputs a scalar. Since $\mathbf{S_i}$ is a Beta distribution $[(\alpha_i, \beta_i)]$, the weighted product $p_{\mathbf{S}_{\texttt{Inter}}}$ is a linear interpolation of the parameters of the inputs. We derive the parameters of $\mathbf{S}_{\texttt{Inter}}$ to be $[(\sum w_i \alpha_i, \sum w_i \beta_i)]$:

$$\begin{aligned} p_{\mathbf{S}_{\texttt{Inter}}}(x) &\propto x^{\sum w_i(\alpha_i - 1)}(1-x)^{\sum w_i(\beta_i - 1)} \\ &= x^{\sum w_i \alpha_i - 1}(1-x)^{\sum w_i \beta_i - 1} \end{aligned} \tag{1}$$

Our approach has three advantages (Fig. 2): (1) $\mathcal{I}$ demonstrates a zero-forcing behavior (Sun & Nielsen, 2019) where the effective support of $\mathbf{S}_{\texttt{Inter}}$ approximates the intersection of the effective support of the input embeddings. (2) $\mathcal{I}$ is closed, since the weighted product of PDFs of Beta distributions is proportional to a Beta distribution. (3) $\mathcal{I}$ is commutative w.r.t the input Beta embeddings.

Probabilistic Negation Operator $\mathcal{N}$: We require a probabilistic negation operator $\mathcal{N}$ that takes Beta embedding $\mathbf{S}$ as input and produces an embedding of the complement $\mathcal{N}(\mathbf{S})$ as a result. A desired property of $\mathcal{N}$ is that the density function should reverse in the sense that regions of high density in $p_{\mathbf{S}}$ should have low probability density in $p_{\mathcal{N}(\mathbf{S})}$ and vice versa (Fig. 2). For the Beta embeddings, this property can be achieved by taking the reciprocal of the shape parameters α and β: $\mathcal{N}([(\alpha, \beta)]) = [(\frac{1}{\alpha}, \frac{1}{\beta})]$. As shown in Fig. 2, the embeddings switch from bell-shaped unimodal density function with $1 < \alpha, \beta$ to bimodal density function with $0 < \alpha, \beta < 1$.

By defining the probabilistic logical operators $\mathcal{I}$ and $\mathcal{N}$, BetaE has the following properties: Given Beta embedding $\mathbf{S}$, $\mathbf{S}$ is a fixed point of $\mathcal{N} \circ \mathcal{N}$: $\mathcal{N}(\mathcal{N}(\mathbf{S})) = \mathbf{S}$; we have $\mathcal{I}(\{\mathbf{S}, \mathbf{S}, \ldots, \mathbf{S}\}) = \mathbf{S}$. This shows that our design of the probabilistic intersection operator and the probabilistic negation operator achieves two important properties that obey the rules of real logical operations.

3.3. Learning Beta Embeddings

Distance: Assume each embedding consists of n independent Beta distributions. Given an entity embedding $\mathbf{v}$ with parameters $[(\alpha_1^v, \beta_1^v), \ldots, (\alpha_n^v, \beta_n^v)]$, and a query embedding $\mathbf{q}$ with parameters $[(\alpha_1^q, \beta_1^q), \ldots, (\alpha_n^q, \beta_n^q)]$, we define the distance between this entity v and the query q as the sum of KL divergence between the two Beta embeddings along each dimension: $\texttt{Dist}(v; q) = \sum_{i=1}^{n} \texttt{KL}(p_{\mathbf{v,i}}; p_{\mathbf{q,i}})$, where $p_{*,\mathbf{i}}$ represents the i-th Beta distribution.

Training Objective: Our objective is to minimize the distance between a query and its answers while maximizing the distance between the query and other random entities via negative sampling (Sun et al., 2019; Ren et al., 2020): $L = -\log \sigma (\gamma - \texttt{Dist}(v; q)) - \sum_{j=1}^{k} \frac{1}{k} \log \sigma \left(\texttt{Dist}(v_j'; q) - \gamma\right)$, where $v \in [\![q]\!]$ belongs to the answer set of q, $v_j' \notin [\![q]\!]$ represents a random negative sample, and γ denotes the margin. In the loss function, we use k random negative samples and optimize the average.

4. Experiments

We evaluate BetaE on multi-hop reasoning over standard KGs. Our experiments demonstrate that: (1) BetaE effectively answers arbitrary FOL queries. (2) BetaE outperforms less general methods (Hamilton et al., 2018; Ren et al., 2020) on EPFO queries (with only $\exists$, $\wedge$ and $\vee$). (3) The probabilistic embeddings correspond well to the uncertainty.

Dataset	Method	1p	2p	3p	2i	3i	pi	ip	2in	3in	inp	pin	pni
FB15k	Q2B	0.301	0.219	0.262	0.331	0.270	0.297	0.139	-	-	-	-	-
	BETAE	**0.373**	**0.478**	**0.472**	**0.572**	**0.397**	**0.519**	**0.421**	0.622	0.548	0.459	0.465	0.608
FB15k-237	Q2B	0.184	0.226	0.269	0.347	0.436	0.361	0.199	-	-	-	-	-
	BETAE	**0.396**	**0.503**	**0.569**	**0.598**	**0.516**	**0.540**	**0.439**	0.685	0.579	0.511	0.468	0.671
NELL995	Q2B	0.154	0.288	0.305	0.380	0.410	0.361	0.345	-	-	-	-	-
	BETAE	**0.423**	**0.552**	**0.564**	**0.594**	**0.610**	**0.598**	**0.535**	0.711	0.595	0.354	0.447	0.639

Table 1. Spearman's rank correlation between learned embedding and the number of answers of queries.

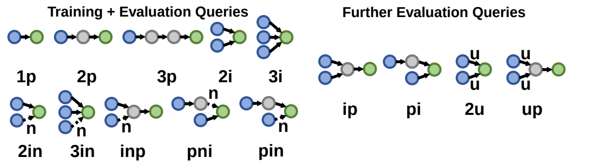

Figure 3. Training and evaluation queries with query structures.

Dataset	Model	1p	2p	3p	2i	3i	pi	ip	2u	up	avg
FB15k	BETAE	48.9	**16.0**	**16.1**	**40.2**	**51.6**	**29.6**	**18.3**	**25.2**	**16.0**	**29.1**
	Q2B	**50.9**	11.7	7.2	37.9	50.5	25.8	15.7	20.4	8.4	25.4
	GQE	34.2	8.3	5.0	23.8	34.9	15.5	11.2	11.5	5.6	16.6
FB15k-237	BETAE	**28.0**	**4.8**	**4.6**	**16.9**	**30.1**	**13.2**	5.6	**5.9**	**4.0**	**12.6**
	Q2B	**28.0**	4.0	2.8	16.7	28.5	12.0	**6.8**	4.6	3.2	11.8
	GQE	22.4	2.8	2.1	11.7	20.9	8.4	5.7	3.3	2.1	8.8
NELL995	BETAE	**42.8**	7.9	**6.6**	**25.7**	**36.3**	**16.8**	8.8	**6.2**	4.5	**17.3**
	Q2B	23.4	**8.5**	**6.6**	19.9	30.6	14.1	**10.9**	4.7	**5.6**	13.8
	GQE	15.4	6.7	5.0	14.3	20.4	10.6	9.0	2.9	5.0	9.9

Table 2. H@1 results (%) of BETAE, Q2B and GQE on answering EPFO ($\exists$, $\wedge$, $\vee$) queries.

4.1. Experiment Setup

Our experimental setup is focused on incomplete KGs and thus we measure performance only over answer entities that require (implicitly) imputing at least one edge. We use three standard KGs with official training/validation/test edge splits, FB15k (Bordes et al., 2013), FB15k-237 (Toutanova & Chen, 2015) and NELL995 (Xiong et al., 2017) and follow (Ren et al., 2020) for the preprocessing.

Evaluation Protocol: We first build three KGs: training KG, validation KG, test KG using training edges, training+validation edges, training+validation+test edges, respectively. Our evaluation focuses on incomplete KGs, so given a test (validation) query q, we are interested in discovering *non-trivial* answers $[\![q]\!]_{\texttt{test}} \backslash [\![q]\!]_{\texttt{val}}$ ($[\![q]\!]_{\texttt{val}} \backslash [\![q]\!]_{\texttt{train}}$). For each non-trivial answer v of a test query q, we rank it against non-answer entities $\mathcal{V} \backslash [\![q]\!]_{\texttt{test}}$. We denote the rank as r and calculate the Mean Reciprocal Rank (MRR) and Hits at K (H@K) as evaluation metrics.

Queries: We base our queries on the 9 query structures proposed in Query2Box (Q2B) (Ren et al., 2020) and further create queries with negation. As shown in Fig. 3, we look at the 4 query structures with intersection ($2i/3i/ip/pi$) and perturb one edge to perform set complement before taking the intersection, resulting in $2in/3in/inp/pni/pin$ structures. See Appen. A for query generation details.

As summarized in Fig. 3, our training and evaluation queries consist of the 5 conjunctive structures ($1p/2p/3p/2i/3i$) and also 5 novel structures with negation ($2in/3in/inp/pni/pin$). Furthermore, we also evaluate model's generalization ability which means answering queries with structures that the model has never seen during training. We further include $ip/pi/2u/up$ for evaluation.

Baselines: We consider two state-of-the-art baselines for answering complex logical queries on KGs: Q2B (Ren et al., 2020) and GQE (Hamilton et al., 2018). Both methods design their corresponding projection and intersection operators, however, neither can handle the negation operation since the complement of a point/box in the Euclidean space is no longer a point/box.

4.2. Modeling Arbitrary FOL Queries

Modeling EPFO (containing only $\exists$, $\wedge$ and $\vee$) Queries: First we compare BETAE with baselines that can only model queries with conjunction and disjunction (but no negation). Table 2 shows the H@1 of the three methods. BETAE achieves on average 14.5%, 6.7% and 25.4% relative improvement H@1 over previous state-of-the-art Q2B on FB15k, FB15k-237 and NELL995, respectively. We refer the reader to Tables 8 in Appen. C for the MRR results.

Dataset	Metrics	2in	3in	inp	pin	pni	avg
FB15k	MRR	13.5	13.8	11.4	6.1	11.7	11.3
	H@10	29.2	30.4	23.2	13.4	24.9	24.2
FB15k-237	MRR	4.9	7.5	7.4	3.6	3.2	5.3
	H@10	10.6	16.5	16.0	7.7	6.7	11.5
NELL995	MRR	5.2	7.7	9.7	3.2	3.2	5.8
	H@10	11.5	17.8	20.8	7.0	6.5	12.7

Table 3. Results (%) of BETAE on queries with negation.

Modeling Queries with Negation: Next, we evaluate our model's ability to model queries with negation. We report both the MRR and H@10 results in Table 3. Overall, BETAE generalizes well and provides the first embedding-based method that can handle arbitrary FOL queries.

4.3. Modeling the Uncertainty of Queries

We also investigate whether our Beta embeddings are able to capture uncertainty. The uncertainty of a (fuzzy) set can be characterized by its cardinality. Given a query with answer set $[\![q]\!]$, we aim to calculate the correlation between the differential entropy of the Beta embedding $p_{[\![\mathbf{q}]\!]}$ and the cardinality of the answer set $|[\![q]\!]|$. For comparison, Q2B embeds each query as a box, which can also model the uncertainty of the query by expanding/shrinking the box size. Table 1 and Table 9 (in Appen. C) show that BETAE achieves up to 77% better correlation than Q2B. We conclude that BETAE with Beta embeddings is able to capture query uncertainty. Furthermore, note that BETAE naturally learns this property without any regularization to impose the correlation during training.

References

Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., and Yakhnenko, O. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 2787–2795, 2013.

Das, R., Neelakantan, A., Belanger, D., and McCallum, A. Chains of reasoning over entities, relations, and text using recurrent neural networks. In *European Chapter of the Association for Computational Linguistics (EACL)*, pp. 132–141, 2017.

Guu, K., Miller, J., and Liang, P. Traversing knowledge graphs in vector space. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2015.

Hamilton, W., Bajaj, P., Zitnik, M., Jurafsky, D., and Leskovec, J. Embedding logical queries on knowledge graphs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Ren, H., Hu, W., and Leskovec, J. Query2box: Reasoning over knowledge graphs in vector space using box embeddings. In *International Conference on Learning Representations (ICLR)*, 2020.

Sun, K. and Nielsen, F. Information-geometric set embeddings (igse): From sets to probability distributions. *arXiv preprint arXiv:1911.12463*, 2019.

Sun, Z., Deng, Z.-H., Nie, J.-Y., and Tang, J. Rotate: Knowledge graph embedding by relational rotation in complex space. In *International Conference on Learning Representations (ICLR)*, 2019.

Toutanova, K. and Chen, D. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pp. 57–66, 2015.

Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., and Bouchard, G. Complex embeddings for simple link prediction. In *International Conference on Machine Learning (ICML)*, pp. 2071–2080, 2016.

Xiong, W., Hoang, T., and Wang, W. Y. Deeppath: A reinforcement learning method for knowledge graph reasoning. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2017.

Yang, B., Yih, W.-t., He, X., Gao, J., and Deng, L. Embedding entities and relations for learning and inference in knowledge bases. In *International Conference on Learning Representations (ICLR)*, 2015.

Zhang, S., Tay, Y., Yao, L., and Liu, Q. Quaternion knowledge graph embeddings. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Appendix

A. Query Generation and Statistics

Generation of EPFO (with $\exists$, $\vee$ and $\wedge$) Queries: Following (Ren et al., 2020), we generate the 9 EPFO query structures in a similar manner. Given the three KGs, and its training/validation/test edge splits, which is shown in Table 4, we first create $\mathcal{G}_{\texttt{train}}$, $\mathcal{G}_{\texttt{valid}}$, $\mathcal{G}_{\texttt{test}}$ as discussed in Sec. 4.1. Then for each query structure, we use pre-order traversal starting from the target node/answer to assign an entity/relation to each node/edge iteratively until we instantiate every anchor nodes (the root of the query structure). After the instantiation of a query, we could perform post-order traversal to achieve the answers of this query. And for validation/test queries, we explicitly filter out ones that do not exist non-trivial answers, i.e., they can be fully answered in $\mathcal{G}_{\texttt{train}}/\mathcal{G}_{\texttt{valid}}$. Different from the dataset in (Ren et al., 2020), where the maximum number of test queries may exceed 5,000, we set a bar for the number of answers one query has, and additionally filter out unrealistic queries with more than 100 answers. We list the average number of answers the new test queries have in Table 5 and the number of training/validation/test queries in Table 6.

Dataset	Entities	Relations	Training Edges	Validation Edges	Test Edges	Total Edges
FB15k	14,951	1,345	483,142	50,000	59,071	592,213
FB15k-237	14,505	237	272,115	17,526	20,438	310,079
NELL995	63,361	200	114,213	14,324	14,267	142,804

Table 4. Knowledge graph dataset statistics as well as training, validation and test edge splits.

Dataset	1p	2p	3p	2i	3i	ip	pi	2u	up	2in	3in	inp	pin	pni
FB15k	1.7	19.6	24.4	8.0	5.2	18.3	12.5	18.9	23.8	15.9	14.6	19.8	21.6	16.9
FB15k-237	1.7	17.3	24.3	6.9	4.5	17.7	10.4	19.6	24.3	16.3	13.4	19.5	21.7	18.2
NELL995	1.6	14.9	17.5	5.7	6.0	17.4	11.9	14.9	19.0	12.9	11.1	12.9	16.0	13.0

Table 5. Average number of answers of test queries in our new dataset.

Queries	Training		Validation		Test	
Dataset	1p/2p/3p/2i/3i	2in/3in/inp/pin/pni	1p	others	1p	others
FB15k	273,710	27,371	59,097	8,000	67,016	8,000
FB15k-237	149,689	14,968	20,101	5,000	22,812	5,000
NELL995	107,982	10,798	16,927	4,000	17,034	4,000

Table 6. Number of training, validation, and test queries generated for different query structures.

Generation of Queries with Negation: For the additional queries with negation, we derive 5 new query structures from the 9 EPFO structures. Specifically, as shown in Fig. 3, we only consider query structures with intersection for the derivation of queries with negation. The reason is that queries with negation are only realistic if we take negation with an intersection together. Consider the following example, where negation is not taken with intersection, "*List all the entities on KG that is not European countries.*", then both "*apple*" and "*computer*" will be the answers. However, realistic queries will be like "*List all the countries on KG that is not European countries.*", which requires an intersection operation. In this regard, We modify one edge of the intersection to further incorporate negation, thus we derive $2in$ from $2i$, $3in$ from $3i$, inp from ip, pin and pni from pi. Note that following the 9 EPFO structures, we also enforce that all queries with negation have at most 100 answers.

B. Experimental Details

We implement our code using Pytorch. We use the implementation of the two baselines GQE (Hamilton et al., 2018) and Q2B (Ren et al., 2020) in `https://github.com/hyren/query2box`. We finetune the hyperparameters for the three methods including number of embedding dimensions from $\{200, 400, 800\}$ and the learning rate from $\{1e^{-4}, 5e^{-3}, 1e^{-3}\}$, batch size from $\{128, 256, 512\}$, and the negative sample size from $\{32, 64, 128\}$, the margin γ from $\{20, 30, 40, 50, 60, 70\}$. We list the hyperparameters of each model in the Table 7. Additionally, for our BETAE, we finetune the structure of the probabilistic projection operator $\texttt{MLP}_r$ and the attention module $\texttt{MLP}_{\texttt{Att}}$. For both modules, we implement a three-layer MLP with 512 latent dimension and ReLU activation.

	embedding dim	learning rate	batch size	negative sample size	margin
GQE	800	0.0005	512	128	30
Q2B	400	0.0005	512	128	30
BETAE	400	0.0005	512	128	60

Table 7. Hyperparameters used for each method.

Dataset	**Model**	**1p**	**2p**	**3p**	**2i**	**3i**	**pi**	**ip**	**2u**	**up**	**avg**
FB15k	BETAE	62.2	**24.6**	**23.9**	52.5	**64.4**	**41.3**	**27.1**	**36.8**	**24.3**	**39.7**
	Q2B	**67.1**	19.9	13.5	**53.1**	64.4	38.8	24.9	33.4	15.7	36.7
	GQE	54.6	15.3	10.8	39.7	51.4	27.6	19.1	22.1	11.6	28.0
FB15k-237	BETAE	38.2	**10.3**	**9.7**	27.3	40.9	**21.1**	11.2	**11.7**	**9.2**	**20.0**
	Q2B	**40.3**	9.1	6.6	**28.8**	**41.4**	21.0	**12.4**	10.6	7.3	19.7
	GQE	35.0	7.2	5.3	23.3	34.6	16.5	10.7	8.2	5.7	16.3
NELL995	BETAE	**52.5**	12.7	**11.0**	**36.1**	**46.8**	**23.8**	13.9	**11.7**	8.6	**24.1**
	Q2B	41.8	**13.7**	10.9	33.1	43.9	22.4	**16.7**	11.1	**9.7**	22.6
	GQE	32.8	11.9	9.6	27.5	35.2	18.4	14.4	8.5	8.8	18.6

Table 8. MRR results (%) of BETAE, Q2B and GQE on answering EPFO ($\exists$, $\wedge$, $\vee$) queries.

We run our method and two baselines for three times and report the average result with standard deviation.

Each single experiment is run on a single NVIDIA GeForce RTX 2080 TI GPU, and we run each method for 300k iterations.

C. Additional Experimental Results

We show in Table 8 the MRR results of the three methods on answering EPFO queries. Our methods show a significant improvement over the two baselines in all three datasets.

We show in Table 9 the Pearson correlation coefficient between the learned embedding and the number of answers of queries. Our method is better than the baseline Q2B in measuring the uncertainty of the queries.

Dataset	Method	**1p**	**2p**	**3p**	**2i**	**3i**	**pi**	**ip**	**2in**	**3in**	**inp**	**pin**	**pni**
FB15k	Q2B	0.075	0.217	0.258	0.285	0.226	0.245	0.133					
	BETAE	**0.216**	**0.357**	**0.383**	**0.386**	**0.299**	**0.311**	**0.312**	0.438	0.413	0.343	0.360	0.442
FB15k-237	Q2B	0.017	0.194	0.261	**0.366**	**0.488**	**0.335**	0.197					
	BETAE	**0.225**	**0.365**	**0.450**	0.362	0.307	0.319	**0.332**	0.464	0.409	0.390	0.361	0.484
NELL995	Q2B	0.068	0.211	0.306	0.362	0.287	0.240	0.338					
	BETAE	**0.236**	**0.403**	**0.433**	**0.404**	**0.385**	**0.403**	**0.403**	0.515	0.514	0.255	0.354	0.455

Table 9. Pearson correlation coefficient between learned embedding (differential entropy for BETAE, box size for Q2B) and the number of answers of queries (grouped by different query type). Ours achieve higher correlation coefficient.