

# Beta Embeddings for Multi-Hop Logical Reasoning in Knowledge Graphs

Hongyu Ren  
Stanford University  
hyren@cs.stanford.edu

Jure Leskovec  
Stanford University  
jure@cs.stanford.edu

## 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 has been an outstanding challenge of how 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 as a consequence allows us to also faithfully model uncertainty. Logical operations are performed in the embedding space by neural operators over the probabilistic embeddings. We demonstrate the performance of BETAE on answering arbitrary FOL queries on three large, incomplete KGs. While being more general, BETAE also increases relative performance by up to 25.4% over the current state-of-the-art KG reasoning methods that can only handle conjunctive queries without negation.

## 1 Introduction

Reasoning is a process of deriving logical conclusion or making predictions from available knowledge/facts. Knowledge can be encoded in a knowledge graph (KG), where entities are expressed as nodes and relations as edges. Real-world KGs, such as Freebase [1], Yago [2], NELL [3], are large-scale as well as noisy and incomplete. Reasoning in KGs is a fundamental problem in Artificial Intelligence. In essence, it involves answering first-order logic (FOL) queries over KGs using operators existential quantification ( $\exists$ ), conjunction ( $\wedge$ ), disjunction ( $\vee$ ), and negation ( $\neg$ ).

To find answers, a given FOL query can be viewed as a computation graph which specifies the steps needed. A concrete example of the computation graph for the query “List the presidents of European countries that have never held the World Cup” is shown in Fig. 1. The query can be represented as a conjunction of three terms: “*Located(Europe, V)*”, which finds all European countries; “*¬Held(World Cup, V)*”, which finds all countries that never held the World Cup; and “*President(V, V<sub>?</sub>)*”, which finds presidents of given countries. In order to answer this query, one first locates the entity “*Europe*” and then traverses the KG by relation “*Located*” to identify a set of European countries. Similar operations are needed for the entity “*World Cup*” to obtain countries that hosted the World Cup. One then needs to complement the second set to identify countries that have never held the World Cup and intersect the complement with the set of European countries. The final step is to apply the relation

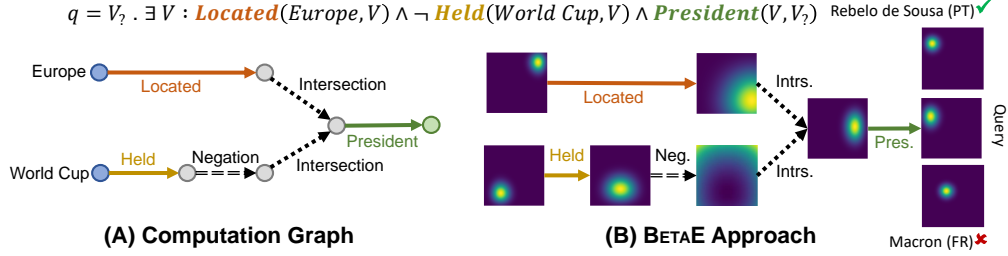


Figure 1: BETAE answers first-order logic queries that include  $\exists$ ,  $\wedge$ ,  $\vee$  and  $\neg$  logical operators. (A): A given query “List the presidents of European countries that have never held the World Cup” can be represented by its computation graph where each node represents a set of entities and each edge represents a logical operation. (B): BETAE models each node of the computation graph as a Beta distribution over the entity embedding space and each edge of the computation graph transforms the distribution via a projection, negation, or intersection operation. BETAE applies a series of logical operators that each transform and shape the Beta distribution. The answer to the query are then entities that are probabilistically close to the embedding of the query (e.g., embedding of “Macron” is closer to the query embedding and the embedding of “Rebello de Sousa”).

“President” to the resulting intersection set to find the list of country presidents, which gives the query answer.

KG reasoning presents a number of 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 [4, 5, 6, 7, 8] 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 [9, 10, 11, 12]. 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). An advantage of these approaches is that they **do not need to track all the intermediate entities, and that they can use the nearest neighbor search [13] in the embedding space to quickly discover answers**. However, these methods **only support existential positive first-order (EPFO) queries**, a subset of FOL queries with existential quantification ( $\exists$ ), conjunction ( $\wedge$ ) and disjunction ( $\vee$ ), but not negation ( $\neg$ ). Negation, however, 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 [9, 11, 12] or a box [10] 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 density and vice versa. (3) Our neural modeling of  $\wedge$  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, disjunction  $\vee$  can be approximated with  $\wedge$  and  $\neg$ , allowing BETAE to handle a complete set of FOL operators and thus supporting arbitrary FOL queries.

Our model is able to handle arbitrary first-order logic queries in an efficient and scalable manner. We perform experiments on standard KG datasets and compare BETAE to prior approaches [9, 10] that can only handle EPFO queries. Experiments show that our model BETAE is able to achieve state-of-the-art performance in handling arbitrary conjunctive queries (including  $\exists$ ,  $\wedge$ ) with a relative

increase of the accuracy by up to 25.4%. Furthermore, we also demonstrate that BETAE is more general and is able to accurately answer any FOL query that includes negation  $\neg$ . Project website with data and code can be found at <http://snap.stanford.edu/betae>.

## 2 Related Work

**Uncertainty in KG Embeddings.** Previous works on KG embeddings assign a learnable vector for each entity and relation with various geometric intuitions [4, 5, 6, 7, 8] and neural architectures [14, 15, 16]. Besides vector embeddings, KG2E [17] and TransG [18] both model the uncertainties of the entities and relations on KGs by using the Gaussian distributions and mixture models. However, their focus is link prediction and it is unclear how to generalize these approaches to multi-hop reasoning with logical operators. In contrast, our model aims at multi-hop reasoning and thus learns probabilistic embeddings for complex queries and also designs a set of neural logical operators over the probabilistic embeddings. Another line of work models the uncertainty using order embeddings [19, 20, 21, 22, 23], distributions [17, 24, 25] and Quantum logic [26]. The difference here is that our goal is to model the logical queries and their answers, which goes beyond modeling the inclusion and entailment between a pair of concepts in KGs.

**Multi-hop Reasoning on KGs.** Another line of related work is multi-hop reasoning on KGs. This includes (1) answering multi-hop logical queries on KGs, which is most relevant to our paper, and (2) using multi-hop rules or paths to improve the performance of link prediction. Previous methods that answer queries [9, 10, 11, 12] can only model a subset of FOL queries, while our method can handle arbitrary FOL queries with probabilistic embeddings. Rule and path-based methods [27, 28, 29, 30, 31, 32] pre-define or achieve these multi-hop rules in an online fashion that require a modeling of all the intermediate entities on the path, while our main focus is to directly embed and answer a complex FOL query without the need to model the intermediate entities, which leads to more scalable algorithms.

## 3 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  $r : \mathcal{V} \times \mathcal{V} \rightarrow \{\text{True}, \text{False}\}$  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 ( $\wedge$ ), disjunction ( $\vee$ ), existential quantification ( $\exists$ ) and negation ( $\neg$ )<sup>1</sup>. We define valid FOL queries in its disjunctive normal form (DNF), *i.e.*, disjunction of conjunctions.

**Definition 1** (First-order logic queries). *A first-order logic query  $q$  consists of a non-variable anchor entity set  $\mathcal{V}_a \subseteq \mathcal{V}$ , existentially quantified bound variables  $V_1, \dots, V_k$  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.*

$$q[V_?] = V_? \cdot \exists V_1, \dots, V_k : c_1 \vee c_2 \vee \dots \vee c_n$$

1. Each  $c$  represents a conjunctive query with one or more literals  $e$ .  $c_i = e_{i1} \wedge e_{i2} \wedge \dots \wedge e_{im}$ .
2. 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  $v_a \in \mathcal{V}_a$ ,  $V \in \{V_?, V_1, \dots, V_k\}$ ,  $V' \in \{V_1, \dots, V_k\}$ ,  $V \neq V'$ ,  $r \in \mathcal{R}$ .

**Computation Graph:** As shown in Fig. 1, we can derive, for a given query, its corresponding computation graph by representing each atomic formula with relation projection, merging by intersection and transforming negation by complement. 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. The computation graphs of FOL queries can be viewed as heterogeneous *trees*, where each leaf node corresponds to a set of cardinality 1 that contains a single anchor entity  $v_a \in \mathcal{V}_a$  (note that one anchor entity may

<sup>1</sup>Note that we do not consider FOL queries with universal quantification ( $\forall$ ) in this paper. Queries with universal quantification do not apply in real-world KGs since no entity connects with all the other entities.

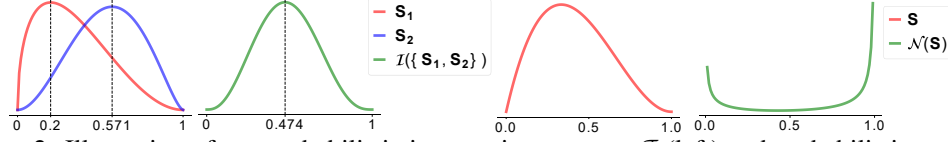


Figure 2: Illustration of our probabilistic intersection operator  $\mathcal{I}$  (left) and probabilistic negation operator  $\mathcal{N}$  (right).  $\mathcal{I}$  transforms the input distribution by taking the weighted product of the PDFs;  $\mathcal{N}$  transforms the input distribution by taking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

appear in multiple leaf nodes) and 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  $r$ :  $A_r(v) \equiv \{v' \in \mathcal{V} : r(v, v') = \text{True}\}$ .
2. **Intersection:** Given sets of entities  $\{S_1, S_2, \dots, S_n\}$ , compute their intersection  $\cap_{i=1}^n S_i$ .
3. **Complement/Negation:** Given a set of entities  $S \subseteq \mathcal{V}$ , compute its complement  $\bar{S} \equiv \mathcal{V} \setminus S$ .

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  $\{S_1, \dots, S_n\}$ ,  $\cup_{i=1}^n S_i$  is equivalent to  $\overline{\cap_{i=1}^n \bar{S}_i}$ .

In order to answer a given FOL query, we can follow the computation graph and execute logical operators. We can obtain the answers by looking at the entities in the root node. We denote the answer set as  $\llbracket q \rrbracket$ , which represents the set of entities on  $\mathcal{G}$  that satisfy  $q$ , i.e.,  $v \in \llbracket q \rrbracket \iff q[v] = \text{True}$ . Note that this symbolic traversal of the computation graph is equivalent to traversing the KG, however, it cannot handle noisy or missing edges in the KG.

## 4 Probabilistic Embeddings for Logical Reasoning

To answer queries in a large and incomplete KG, we first introduce our model BETAE, which embeds both entities and queries as Beta distributions. Then we define probabilistic logical operators for relation projection, intersection and negation. These operate on the Beta embeddings which allow us to support arbitrary FOL queries. Finally, we describe our training objective.

### 4.1 Beta Embeddings for Entities and Queries

In order to model any FOL query, the desirable properties of the embedding include: (1) the embedding can naturally model uncertainty; (2) we can design logical/set operators (conjunction/intersection and especially negation/complement) that are closed. The closure property is important for two reasons: (i) operators can be combined in arbitrary ways; (ii) the representation remains at a fixed space/time complexity and does not grow exponentially as additional operators are applied.

We propose to embed both the entities and queries into the same space using probabilistic embeddings with bounded support. With a bounded support, the negation/complement can be accordingly defined, where we follow the intuition to switch high-density regions to low density and vice versa (Fig. 2). Specifically, we look at the  $[0, 1]$  interval and adopt the Beta distribution. A Beta distribution  $\text{Beta}(\alpha, \beta)$  has two shape parameters, and our method relies on its probability density function (PDF):  $p(x) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{\mathbf{B}(\alpha, \beta)}$ , where  $x \in [0, 1]$  and  $\mathbf{B}(\cdot)$  denotes the beta function. The uncertainty of a Beta distribution can be measured by its differential entropy:  $H = \ln \mathbf{B}(\alpha, \beta) - (\alpha - 1)[\psi(\alpha) - \psi(\alpha + \beta)] - (\beta - 1)[\psi(\beta) - \psi(\alpha + \beta)]$ , where  $\psi(\cdot)$  represents the digamma function.

For each entity  $v \in \mathcal{V}$ , which can be viewed as a set with a single element, 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:  $\mathbf{S} = [(\alpha_1, \beta_1), \dots, (\alpha_n, \beta_n)]$ , where  $n$  is a hyperparameter. We denote the PDF of the  $i$ -th Beta distribution in  $\mathbf{S}$  as  $p_{S,i}$ . 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  $p_S$ .

## 4.2 Probabilistic Logical Operators

In order to answer a query using the computation graph, we need probabilistic logical operators for the Beta embedding. Next, we describe the design of these logical operators used in computation graphs, which include relation projection  $\mathcal{P}$ , intersection  $\mathcal{I}$  and negation  $\mathcal{N}$ . As discussed before, union can be implemented using intersection and complement. Each operator takes one or more Beta embeddings as input and then transforms them into a new 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):

$$\mathbf{S}' = \text{MLP}_r(\mathbf{S}) \quad (1)$$

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  $A_r(v) \equiv \{v' \in \mathcal{V} : r(v, v') = \text{True}\}$ . Importantly, projection operation represents a relation traversal from one (fuzzy) set of entities to another (fuzzy) set, and may yield a huge number of results, yet here we represent it with a single fixed-size Beta embedding, making BETAE scalable.

**Probabilistic Intersection Operator  $\mathcal{I}$ :** Given  $n$  input embeddings  $\{\mathbf{S}_1, \dots, \mathbf{S}_n\}$ , the goal of probabilistic intersection operator  $\mathcal{I}$  is to calculate the Beta embedding  $\mathbf{S}_{\text{Inter}}$  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:

$$p_{\mathbf{S}_{\text{Inter}}} = \frac{1}{Z} \prod p_{\mathbf{S}_1}^{w_1} \dots p_{\mathbf{S}_n}^{w_n}, \quad (2)$$

where  $Z$  is a normalization constant and  $w_1, \dots, w_n$  are the weights with their sum equal to 1.

To make the model more expressive, we use the attention mechanism and learn  $w_1, \dots, w_n$  through a  $\text{MLP}_{\text{Att}}$  that takes as input the parameters of  $\mathbf{S}_i$  and outputs a single attention scalar:

$$w_i = \frac{\exp(\text{MLP}_{\text{Att}}(\mathbf{S}_i))}{\sum_j \exp(\text{MLP}_{\text{Att}}(\mathbf{S}_j))} \quad (3)$$

Since  $\mathbf{S}_i$  is a Beta distribution  $[(\alpha_i, \beta_i)]$ , the weighted product  $p_{\mathbf{S}_{\text{Inter}}}$  is a linear interpolation of the parameters of the inputs. We derive the parameters of  $\mathbf{S}_{\text{Inter}}$  to be  $[(\sum w_i \alpha_i, \sum w_i \beta_i)]$ :

$$\begin{aligned} p_{\mathbf{S}_{\text{Inter}}}(x) &\propto x^{\sum w_i (\alpha_i - 1)} (1 - x)^{\sum w_i (\beta_i - 1)} \\ &= x^{\sum w_i \alpha_i - 1} (1 - x)^{\sum w_i \beta_i - 1} \end{aligned} \quad (4)$$

Our approach has three important advantages (Fig. 2): (1) Taking a weighted product of the PDFs demonstrates a zero-forcing behavior [33] where the effective support of the resulting Beta embedding  $\mathbf{S}_{\text{Inter}}$  approximates the intersection of the effective support of the input embeddings (effective support meaning the area with sufficiently large probability density [33]). This follows the intuition that regions of high density in  $p_{\mathbf{S}_{\text{Inter}}}$  should have high density in the PDF of all input embeddings  $\{p_{\mathbf{S}_1}, \dots, p_{\mathbf{S}_n}\}$ . (2) As shown in Eq. 4, the probabilistic intersection operator  $\mathcal{I}$  is closed, since the weighted product of PDFs of Beta distributions is proportional to a Beta distribution. (3) The probabilistic intersection operator  $\mathcal{I}$  is commutative w.r.t the input Beta embeddings following Eq. 2.

**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  $\alpha$  and  $\beta$ :  $\mathcal{N}([( \alpha, \beta )]) = [(\frac{1}{\alpha}, \frac{1}{\beta})]$ . As shown in Fig. 2, the embeddings switch from bell-shaped unimodal density function with  $1 < \alpha, \beta$  to bimodal density function with  $0 < \alpha, \beta < 1$ .

**Proposition 1.** *By defining the probabilistic logical operators  $\mathcal{I}$  and  $\mathcal{N}$ , BETAE has the following properties (with proof in Appendix A):*

1. *Given Beta embedding  $\mathbf{S}$ ,  $\mathbf{S}$  is a fixed point of  $\mathcal{N} \circ \mathcal{N}$ :  $\mathcal{N}(\mathcal{N}(\mathbf{S})) = \mathbf{S}$ .*

2. Given Beta embedding  $\mathbf{S}$ , we have  $\mathcal{I}(\{\mathbf{S}, \mathbf{S}, \dots, \mathbf{S}\}) = \mathbf{S}$ .

Proposition 1 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.

### 4.3 Learning Beta Embeddings

**Distance:** Assume we use a  $n$ -dimensional Beta embedding for entities and queries, which means that each embedding consists of  $n$  independent Beta distributions with  $2n$  number of parameters. Given an entity embedding  $\mathbf{v}$  with parameters  $[(\alpha_1^v, \beta_1^v), \dots, (\alpha_n^v, \beta_n^v)]$ , and a query embedding  $\mathbf{q}$  with parameters  $[(\alpha_1^q, \beta_1^q), \dots, (\alpha_n^q, \beta_n^q)]$ , 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:

$$\text{Dist}(v; q) = \sum_{i=1}^n \text{KL}(p_{\mathbf{v},i}; p_{\mathbf{q},i}), \quad (5)$$

where  $p_{\mathbf{v},i}$  ( $p_{\mathbf{q},i}$ ) represents the  $i$ -th Beta distribution with parameters  $\alpha_i^v$  and  $\beta_i^v$  ( $\alpha_i^q$  and  $\beta_i^q$ ). Note that we use  $\text{KL}(p_{\mathbf{v},i}; p_{\mathbf{q},i})$  rather than  $\text{KL}(p_{\mathbf{q},i}; p_{\mathbf{v},i})$  so that the query embeddings will “cover” the modes of all answer entity embeddings [34].

**Training Objective:** Our objective is to minimize the distance between the Beta embedding of a query and its answers while maximizing the distance between the Beta embedding of the query and other random entities via negative sampling [6, 10], which we define as follows:

$$L = -\log \sigma(\gamma - \text{Dist}(v; q)) - \sum_{j=1}^k \frac{1}{k} \log \sigma(\text{Dist}(v'_j; q) - \gamma), \quad (6)$$

where  $v \in \llbracket q \rrbracket$  belongs to the answer set of  $q$ ,  $v'_j \notin \llbracket q \rrbracket$  represents a random negative sample, and  $\gamma$  denotes the margin. In the loss function, we use  $k$  random negative samples and optimize the average.

**Discussion on Modeling Union:** With the De Morgan’s laws (abbreviated as DM), we can naturally model union operation  $S_1 \cup S_2$  with  $\overline{S_1} \cap \overline{S_2}$ , which we can derive as a Beta embedding. However, according to the Theorem 1 in [10], in order to model any queries with the union operation, we must have a parameter dimensionality of  $\Theta(M)$ , where  $M$  is of the same order as the number of entities [10]. The reason is that we need to model in the embedding space any subset of the entities. Q2B [10] overcomes this limitation by transforming queries into a disjunctive normal form (DNF) and only deals with union at the last step. Our DM modeling of union is also limited in this respect since the Beta embedding can be at most bi-modal and as a result, there are some union-based queries that BETAE cannot model in theory. However, in practice, union-based queries are constrained and we do not need to model all theoretically possible entity subsets. For example, a query “*List the union of European countries and tropical fruits.*” does not make sense; and we further learn high-dimensional Beta embeddings to alleviate the problem. Furthermore, our DM modeling is always linear w.r.t the number of union operations, while the DNF modeling is exponential in the worst case (with detailed discussion in Appendix B). Last but not least, BETAE can safely incorporate both the DNF modeling and DM modeling, and we show in the experiments that the two approaches work equally well in answering real-world queries.

**Inference:** Given a query  $q$ , BETAE directly embeds it as  $\mathbf{q}$  by following the computation graph without the need to model intermediate entities. To obtain the final answer entities, we rank all the entities based on the distance defined in Eq. 5 in constant time using Locality Sensitive Hashing [13].

## 5 Experiments

In this section, we evaluate BETAE on multi-hop reasoning over standard KG benchmark datasets. Our experiments demonstrate that: (1) BETAE effectively answers arbitrary FOL queries. (2) BETAE outperforms less general methods [9, 10] on EPFO queries (containing only  $\exists$ ,  $\wedge$  and  $\vee$ ) that these methods can handle. (3) The probabilistic embedding of a query corresponds well to its uncertainty.

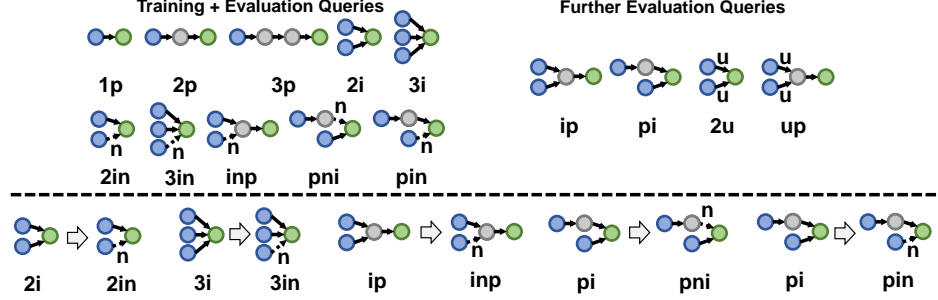


Figure 3: Top: Training and evaluation queries represented with their graphical structures, an abbreviation of the computation graph. Naming convention:  $p$  projection,  $i$  intersection,  $n$  negation,  $u$  union. Bottom: Query structures with negation used in our experiments.

| Dataset   | Model | 1p          | 2p          | 3p          | 2i          | 3i          | pi          | ip          | 2u          |      | up          |      | avg         |
|-----------|-------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|------|-------------|------|-------------|
|           |       |             |             |             |             |             |             |             | DNF         | DM   | DNF         | DM   |             |
| FB15k     | BETAE | 65.1        | 25.7        | 24.7        | 55.8        | 66.5        | 43.9        | 28.1        | 40.1        | 25.0 | 25.2        | 25.4 | 41.6        |
|           | Q2B   | <b>68.0</b> | 21.0        | 14.2        | 55.1        | <b>66.5</b> | 39.4        | 26.1        | 35.1        | -    | 16.7        | -    | 38.0        |
|           | GQE   | 54.6        | 15.3        | 10.8        | 39.7        | 51.4        | 27.6        | 19.1        | 22.1        | -    | 11.6        | -    | 28.0        |
| FB15k-237 | BETAE | 39.0        | <b>10.9</b> | <b>10.0</b> | 28.8        | <b>42.5</b> | <b>22.4</b> | <b>12.6</b> | <b>12.4</b> | 11.1 | <b>9.7</b>  | 9.9  | <b>20.9</b> |
|           | Q2B   | <b>40.6</b> | 9.4         | 6.8         | <b>29.5</b> | 42.3        | 21.2        | <b>12.6</b> | 11.3        | -    | 7.6         | -    | 20.1        |
|           | GQE   | 35.0        | 7.2         | 5.3         | 23.3        | 34.6        | 16.5        | 10.7        | 8.2         | -    | 5.7         | -    | 16.3        |
| NELL995   | BETAE | <b>53.0</b> | 13.0        | <b>11.4</b> | <b>37.6</b> | <b>47.5</b> | <b>24.1</b> | 14.3        | <b>12.2</b> | 11.0 | 8.5         | 8.6  | <b>24.6</b> |
|           | Q2B   | 42.2        | <b>14.0</b> | 11.2        | 33.3        | 44.5        | 22.4        | <b>16.8</b> | 11.3        | -    | <b>10.3</b> | -    | 22.9        |
|           | GQE   | 32.8        | 11.9        | 9.6         | 27.5        | 35.2        | 18.4        | 14.4        | 8.5         | -    | 8.8         | -    | 18.6        |

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

## 5.1 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. More precisely, given an incomplete KG, our goal is to obtain *non-trivial* answers to arbitrary FOL queries that *cannot* be discovered by directly traversing the KG. We use three standard KGs with official training/validation/test edge splits, FB15k [4], FB15k-237 [35] and NELL995 [27] and follow [10] for the preprocessing.

**Evaluation Protocol:** We follow the evaluation protocol in [10]. We first build three KGs: training KG  $\mathcal{G}_{\text{train}}$ , validation KG  $\mathcal{G}_{\text{valid}}$ , test KG  $\mathcal{G}_{\text{test}}$  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  $\llbracket q \rrbracket_{\text{test}} \setminus \llbracket q \rrbracket_{\text{val}}$  ( $\llbracket q \rrbracket_{\text{val}} \setminus \llbracket q \rrbracket_{\text{train}}$ ). That is, answer entities where at least one edge needs to be imputed in order to create an answer path to that entity. For each non-trivial answer  $v$  of a test query  $q$ , we rank it against non-answer entities  $\mathcal{V} \setminus \llbracket q \rrbracket_{\text{test}}$ . We denote the rank as  $r$  and calculate the Mean Reciprocal Rank (MRR):  $\frac{1}{r}$ ; and, Hits at  $K$  ( $H@K$ ):  $1[r \leq K]$  as evaluation metrics.

**Queries:** We base our queries on the 9 query structures proposed in Query2Box (Q2B) [10] and make two additional improvements. First, we notice that some test queries may have more than 5,000 answers. To make the task more challenging, we thus regenerate the same number of validation/test queries for each of the 9 structures, keeping only those with answers smaller than a threshold. We list the statistics of the new set of queries in Table 6 (in Appendix C). We evaluate BETAE on both the queries in Q2B and our new realistic queries, which are more challenging since they use the same training queries without any enforcement on the maximum number of answers for a fair comparison. Second, from the 9 structures we derive 5 new query structures with negation. As shown in Fig. 3, in order to create realistic structures with negation, 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. Additional information about query generation is given in Appendix C.

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 logical structures that the model has never seen during training. We further include  $ip/pi/2u/up$  for evaluation.



| Dataset   | Metrics | 2in  | 3in  | inp  | pin  | pni  | avg  |
|-----------|---------|------|------|------|------|------|------|
| FB15k     | MRR     | 14.3 | 14.7 | 11.5 | 6.5  | 12.4 | 11.8 |
|           | H@10    | 30.8 | 31.9 | 23.4 | 14.3 | 26.3 | 25.3 |
| FB15k-237 | MRR     | 5.1  | 7.9  | 7.4  | 3.6  | 3.4  | 5.4  |
|           | H@10    | 11.3 | 17.3 | 16.0 | 8.1  | 7.0  | 11.9 |
| NELL995   | MRR     | 5.1  | 7.8  | 10.0 | 3.1  | 3.5  | 5.9  |
|           | H@10    | 11.6 | 18.2 | 20.8 | 6.9  | 7.2  | 12.9 |

Table 2: MRR and H@10 results (%) of BETAE on answering queries with negation.

**Baselines:** We consider two state-of-the-art baselines for answering complex logical queries on KGs: Q2B [10] and GQE [9]. GQE embeds both queries and entities as point vectors in the Euclidean space; Q2B embeds the queries as hyper-rectangles (boxes) and entities as point vectors so that answers will be enclosed in the query box. 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. For fair comparison, we assign the same dimensionality to the embeddings of the three methods<sup>2</sup>. Note that since the baselines cannot model negation operation, the training set for the baselines only contain queries of the 5 conjunctive structures. We ran each method for 3 different random seeds after finetuning the hyperparameters. We list the hyperparameters, architectures and more details in Appendix D.

## 5.2 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 1 shows the MRR of the three methods. BETAE achieves on average 9.4%, 5.0% and 7.4% relative improvement MRR over previous state-of-the-art Q2B on FB15k, FB15k-237 and NELL995, respectively. We refer the reader to Tables 9 and 10 in Appendix E for the H@1 results. Again, on EPFO queries BETAE achieves better performance than the two baselines on all three datasets.

**DNF vs. DM:** As discussed in Sec. 4.3, we can model queries with disjunction in two ways: (1) transform them into disjunctive normal form (DNF); (2) represent disjunction with conjunction and negation using the De Morgan’s laws (DM). We evaluate both modeling schemes (Table 1(right)). DNF modeling achieves slightly better results than DM since it is able to better represent disjunction with multi-modal embeddings. However, it also demonstrates that our DM modeling provides a nice approximation to the disjunction operation, and generalizes really well since the model is not trained on  $2u$  and  $up$  queries. Note that BETAE is very flexible and can use and improve both modeling approaches while the baselines can only use DNF since they cannot model the negation operation.

**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 2. Note that answering queries with negation is challenging since only a small fraction of the training queries contain negation. As shown in Table 7 (Appendix), during training, the number of  $2in/3in/inp/pin/pni$  queries is 10 times smaller than the number of conjunctive queries. Overall, BETAE generalizes well and provides the first embedding-based method that can handle arbitrary FOL queries.

## 5.3 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  $\llbracket q \rrbracket$ , we aim to calculate the correlation between the differential entropy of the Beta embedding  $p_{\llbracket q \rrbracket}$  and the cardinality of the answer set  $|\llbracket q \rrbracket|$ . For comparison, Q2B embeds each query as a box, which can also model the uncertainty of the query by expanding/shrinking the box size. We consider two types of statistical correlations: Spearman’s rank correlation coefficient (SRCC), which measures the statistical dependence between the rankings of two variables; and Pearson’s correlation coefficient (PCC), which measures the linear correlation of the two variables. Table 3 and Table 11 (in Appendix E) 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.

<sup>2</sup>If GQE has embeddings of dimension  $2n$ , then Q2B has embeddings of  $n$  since it needs to model both the center and offset of a box, and BETAE also has  $n$  beta distributions since each has two parameters,  $\alpha$  and  $\beta$ .



| Dataset   | Model | 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 | <b>0.373</b> | <b>0.478</b> | <b>0.472</b> | <b>0.572</b> | <b>0.397</b> | <b>0.519</b> | <b>0.421</b> | 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 | <b>0.396</b> | <b>0.503</b> | <b>0.569</b> | <b>0.598</b> | <b>0.516</b> | <b>0.540</b> | <b>0.439</b> | 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 | <b>0.423</b> | <b>0.552</b> | <b>0.564</b> | <b>0.594</b> | <b>0.610</b> | <b>0.598</b> | <b>0.535</b> | 0.711 | 0.595 | 0.354 | 0.447 | 0.639 |

Table 3: Spearman’s rank correlation between learned embedding (differential entropy for BETAE, box size for Q2B) and the number of answers of queries. BETAE shows up to 77% relative improvement.

| 1p    | 2p    | 3p    | 2i    | 3i    | pi    | ip    | 2in   | 3in  | inp   | pin   | pni   |
|-------|-------|-------|-------|-------|-------|-------|-------|------|-------|-------|-------|
| 0.825 | 0.766 | 0.793 | 0.909 | 0.933 | 0.868 | 0.798 | 0.865 | 0.93 | 0.801 | 0.809 | 0.848 |

Table 4: ROC-AUC score of BETAE for all the 12 query structures on classification of queries with/without answers on the NELL dataset.

**Modeling Queries without Answers:** Since BETAE can effectively model the **uncertainty** of a given query, we can use the differential entropy of the query embedding as a measure to represent whether the query is an empty set (has no answers). For evaluation, we randomly generated 4k queries without answers and 4k queries with more than 5 answers for each of the 12 query structures on NELL. Then we calculate the differential entropy of the embeddings of each query with a trained BETAE and use this to classify whether a query has answers. As a result, we find an ROC-AUC score of 0.844 and list the ROC-AUC score of each query structure in Table 4. These results suggest that BETAE can naturally model queries without answers, since (1) we did not explicitly train BETAE to optimize for correlation between the differential entropy and the cardinality of the answer set; (2) we did not train BETAE on queries with empty answers.

## 6 Conclusion

We have presented BETAE, the first embedding-based method that could handle arbitrary FOL queries on KGs. Given a query, BETAE embeds it into Beta distributions using probabilistic logical operators by following the computation graph in a scalable manner. Extensive experimental results show that BETAE significantly outperforms previous state-of-the-art, which can only handle a subset of FOL, in answering arbitrary logical queries as well as modeling the uncertainty.

## Broader Impact

BETAE gives rise to the first method that handles all logical operators in large heterogeneous KGs. It will greatly increase the scalability and capability of multi-hop reasoning over real-world KGs and heterogeneous networks.

One potential risk is that the model may make undesirable predictions in a completely random KG, or a KG manipulated by adversarial and malicious attacks [36, 37]. Recent progress on adversarial attacks [36, 37] have shown that manipulation of the KG structure may effectively deteriorate the performance of embedding-based methods. And this may mislead the users and cause negative impact. We will continue to work on this direction to design more robust KG embeddings. Alternatively, this issue can also be alleviated through human regularization of real-world KGs.

## Acknowledgments and Disclosure of Funding

We thank Shengjia Zhao, Rex Ying, Jiaxuan You, Weihua Hu, Tailin Wu and Pan Li for discussions, and Rok Sosis for providing feedback on our manuscript. Hongyu Ren is supported by the Masason Foundation Fellowship. Jure Leskovec is a Chan Zuckerberg Biohub investigator. We also gratefully acknowledge the support of DARPA under Nos. FA865018C7880 (ASED), N660011924033 (MCS); ARO under Nos. W911NF-16-1-0342 (MURI), W911NF-16-1-0171 (DURIP); NSF under Nos. OAC-1835598 (CINES), OAC-1934578 (HDR), CCF-1918940 (Expeditions), IIS-2030477 (RAPID); Stanford Data Science Initiative, Wu Tsai Neurosciences Institute, Chan Zuckerberg Biohub, Amazon, Boeing, JPMorgan Chase, Docomo, Hitachi, JD.com, KDDI, NVIDIA, Dell.

## References

- [1] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, “Freebase: a collaboratively created graph database for structuring human knowledge,” in *ACM SIGMOD international conference on Management of data (SIGMOD)*, ACM, 2008.
- [2] F. M. Suchanek, G. Kasneci, and G. Weikum, “Yago: a core of semantic knowledge,” in *Proceedings of the International World Wide Web Conference (WWW)*, ACM, 2007.
- [3] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. Hruschka, and T. M. Mitchell, “Toward an architecture for never-ending language learning,” in *AAAI Conference on Artificial Intelligence (AAAI)*, 2010.
- [4] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, “Translating embeddings for modeling multi-relational data,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2013.
- [5] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard, “Complex embeddings for simple link prediction,” in *International Conference on Machine Learning (ICML)*, 2016.
- [6] Z. Sun, Z.-H. Deng, J.-Y. Nie, and J. Tang, “Rotate: Knowledge graph embedding by relational rotation in complex space,” in *International Conference on Learning Representations (ICLR)*, 2019.
- [7] S. Zhang, Y. Tay, L. Yao, and Q. Liu, “Quaternion knowledge graph embeddings,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.
- [8] B. Yang, W.-t. Yih, X. He, J. Gao, and L. Deng, “Embedding entities and relations for learning and inference in knowledge bases,” in *International Conference on Learning Representations (ICLR)*, 2015.
- [9] W. Hamilton, P. Bajaj, M. Zitnik, D. Jurafsky, and J. Leskovec, “Embedding logical queries on knowledge graphs,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.
- [10] H. Ren, W. Hu, and J. Leskovec, “Query2box: Reasoning over knowledge graphs in vector space using box embeddings,” in *International Conference on Learning Representations (ICLR)*, 2020.
- [11] K. Guu, J. Miller, and P. Liang, “Traversing knowledge graphs in vector space,” in *Empirical Methods in Natural Language Processing (EMNLP)*, 2015.
- [12] R. Das, A. Neelakantan, D. Belanger, and A. McCallum, “Chains of reasoning over entities, relations, and text using recurrent neural networks,” in *European Chapter of the Association for Computational Linguistics (EACL)*, 2017.
- [13] P. Indyk and R. Motwani, “Approximate nearest neighbors: towards removing the curse of dimensionality,” in *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, ACM, 1998.
- [14] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, “Modeling relational data with graph convolutional networks,” in *European Semantic Web Conference*, Springer, 2018.
- [15] T. Dettmers, P. Minervini, P. Stenetorp, and S. Riedel, “Convolutional 2d knowledge graph embeddings,” in *AAAI Conference on Artificial Intelligence (AAAI)*, 2018.
- [16] X. Jiang, Q. Wang, and B. Wang, “Adaptive convolution for multi-relational learning,” in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019.
- [17] S. He, K. Liu, G. Ji, and J. Zhao, “Learning to represent knowledge graphs with gaussian embedding,” in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, 2015.
- [18] H. Xiao, M. Huang, and X. Zhu, “Transg: A generative model for knowledge graph embedding,” in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2016.
- [19] I. Vendrov, R. Kiros, S. Fidler, and R. Urtasun, “Order-embeddings of images and language,” in *International Conference on Learning Representations (ICLR)*, 2016.

- [20] A. Lai and J. Hockenmaier, “Learning to predict denotational probabilities for modeling entailment,” in *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2017.
- [21] X. Li, L. Vilnis, and A. McCallum, “Improved representation learning for predicting common-sense ontologies,” *arXiv preprint arXiv:1708.00549*, 2017.
- [22] L. Vilnis, X. Li, S. Murty, and A. McCallum, “Probabilistic embedding of knowledge graphs with box lattice measures,” in *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2018.
- [23] X. Li, L. Vilnis, D. Zhang, M. Boratko, and A. McCallum, “Smoothing the geometry of probabilistic box embeddings,” in *International Conference on Learning Representations (ICLR)*, 2019.
- [24] L. Vilnis and A. McCallum, “Word representations via gaussian embedding,” in *International Conference on Learning Representations (ICLR)*, 2014.
- [25] B. Athiwaratkun and A. G. Wilson, “Hierarchical density order embeddings,” in *International Conference on Learning Representations (ICLR)*, 2018.
- [26] D. Garg, S. Ikbāl, S. K. Srivastava, H. Vishwakarma, H. Karanam, and L. V. Subramaniam, “Quantum embedding of knowledge for reasoning,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.
- [27] W. Xiong, T. Hoang, and W. Y. Wang, “Deeppath: A reinforcement learning method for knowledge graph reasoning,” in *Empirical Methods in Natural Language Processing (EMNLP)*, 2017.
- [28] X. V. Lin, R. Socher, and C. Xiong, “Multi-hop knowledge graph reasoning with reward shaping,” in *Empirical Methods in Natural Language Processing (EMNLP)*, 2018.
- [29] X. Chen, M. Chen, W. Shi, Y. Sun, and C. Zaniolo, “Embedding uncertain knowledge graphs,” in *AAAI Conference on Artificial Intelligence (AAAI)*, 2019.
- [30] S. Guo, Q. Wang, L. Wang, B. Wang, and L. Guo, “Knowledge graph embedding with iterative guidance from soft rules,” in *AAAI Conference on Artificial Intelligence (AAAI)*, 2018.
- [31] S. Guo, Q. Wang, L. Wang, B. Wang, and L. Guo, “Jointly embedding knowledge graphs and logical rules,” in *Empirical Methods in Natural Language Processing (EMNLP)*, 2016.
- [32] H. Wang, H. Ren, and J. Leskovec, “Entity context and relational paths for knowledge graph completion,” *arXiv preprint arXiv:2002.06757*, 2020.
- [33] K. Sun and F. Nielsen, “Information-geometric set embeddings (igse): From sets to probability distributions,” *arXiv preprint arXiv:1911.12463*, 2019.
- [34] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” *arXiv preprint arXiv:1312.6114*, 2013.
- [35] K. Toutanova and D. Chen, “Observed versus latent features for knowledge base and text inference,” in *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, 2015.
- [36] D. Zügner, A. Akbarnejad, and S. Günnemann, “Adversarial attacks on neural networks for graph data,” in *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2018.
- [37] H. Dai, H. Li, T. Tian, X. Huang, L. Wang, J. Zhu, and L. Song, “Adversarial attack on graph structured data,” in *International Conference on Machine Learning (ICML)*, 2018.

# Appendix

## A Proof for Proposition 1

We restate the proposition 1 and its proof here.

**Proposition 2.** *Given the probabilistic logical operators  $\mathcal{I}$  and  $\mathcal{N}$  defined in Sec. 4.2, BETAE has the following properties:*

1. *Given Beta embedding  $\mathbf{S}$ ,  $\mathbf{S}$  is a fixed point of  $\mathcal{N} \circ \mathcal{N}$ :  $\mathcal{N}(\mathcal{N}(\mathbf{S})) = \mathbf{S}$ .*
2. *Given Beta embedding  $\mathbf{S}$ , we have  $\mathcal{I}(\{\mathbf{S}, \mathbf{S}, \dots, \mathbf{S}\}) = \mathbf{S}$ .*

*Proof.* For the first property, the probabilistic negation operator  $\mathcal{N}$  takes the reciprocal of the parameters of the input Beta embeddings. If we apply  $\mathcal{N}$  twice, it naturally equals the input Beta embeddings. For the second property, the probabilistic intersection operator  $\mathcal{I}$  takes the weighted product of the PDFs of the input Beta embeddings, and according to Eq. 4, the parameters of the output Beta embeddings are linear interpolation of the parameters of the input Beta embeddings. Then we naturally have  $\mathbf{S} = \mathcal{I}(\{\mathbf{S}, \dots, \mathbf{S}\})$ .  $\square$

## B Computation Complexity of DM and DNF

Here we discuss the computation complexity of representing any given FOL query using the De Morgan’s laws (DM) and the disjunctive normal form (DNF). Given a FOL query  $q$ , representing  $q$  with DNF may in the worst case creates exponential number of atomic formulas. For example, transforming a valid FOL query  $(q_{11} \vee q_{12}) \wedge (q_{21} \vee q_{22}) \cdots \wedge (q_{n1} \vee q_{n2})$  leads to exponential explosion, resulting in a query with  $2^n$  number of formulas in the DNF. For DM, since we could always represent a disjunction operation with three negation operation and one conjunction operation:  $q_1 \vee q_2 = \neg(\neg q_1 \wedge \neg q_2)$ , which is a constant. Hence, the DM modeling only scales linearly.

## C Query Generation and Statistics

**Generation of EPFO (with  $\exists$ ,  $\vee$  and  $\wedge$ ) Queries:** Following [10], 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 5, we first create  $\mathcal{G}_{\text{train}}$ ,  $\mathcal{G}_{\text{valid}}$ ,  $\mathcal{G}_{\text{test}}$  as discussed in Sec. 5.1. 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  $\mathcal{G}_{\text{train}}/\mathcal{G}_{\text{valid}}$ . Different from the dataset in [10], 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 6 and the number of training/validation/test queries in Table 7.

| 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 5: 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 6: 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 7: 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.

## D Experimental Details

We implement our code using Pytorch. We use the implementation of the two baselines GQE [9] and Q2B [10] 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  $\gamma$  from  $\{20, 30, 40, 50, 60, 70\}$ . We list the hyperparameters of each model in the Table 8. Additionally, for our BETAE, we finetune the structure of the probabilistic projection operator  $MLP_r$  and the attention module  $MLP_{Att}$ . 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 8: Hyperparameters used for each method.

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

## E Additional Experimental Results

Here we list some additional experimental results.

We show in Table 1 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 10 the MRR results of the three methods on answering EPFO queries in the dataset proposed in [10], where the queries may have more than 5,000 answers. Our method is still better than the two baselines.

We show in Table 11 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   | Model | 1p          | 2p          | 3p          | 2i          | 3i          | pi          | ip          | 2u          |      | up          |      | avg         |
|-----------|-------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|------|-------------|------|-------------|
|           |       |             |             |             |             |             |             |             | DNF         | DM   | DNF         | DM   |             |
| FB15k     | BETAE | <b>52.0</b> | <b>17.0</b> | <b>16.9</b> | <b>43.5</b> | <b>55.3</b> | <b>32.3</b> | <b>19.3</b> | <b>28.1</b> | 17.0 | <b>16.9</b> | 17.4 | <b>31.3</b> |
|           | Q2B   | <b>52.0</b> | 12.7        | 7.8         | 40.5        | 53.4        | 26.7        | 16.7        | 22.0        | -    | 9.4         | -    | 26.8        |
|           | GQE   | 34.2        | 8.3         | 5.0         | 23.8        | 34.9        | 15.5        | 11.2        | 11.5        | -    | 5.6         | -    | 16.6        |
| FB15k-237 | BETAE | <b>28.9</b> | <b>5.5</b>  | <b>4.9</b>  | <b>18.3</b> | <b>31.7</b> | <b>14.0</b> | 6.7         | <b>6.3</b>  | 6.1  | <b>4.6</b>  | 4.8  | <b>13.4</b> |
|           | Q2B   | 28.3        | 4.1         | 3.0         | 17.5        | 29.5        | 12.3        | <b>7.1</b>  | 5.2         | -    | 3.3         | -    | 12.3        |
|           | GQE   | 22.4        | 2.8         | 2.1         | 11.7        | 20.9        | 8.4         | 5.7         | 3.3         | -    | 2.1         | -    | 8.8         |
| NELL995   | BETAE | <b>43.5</b> | 8.1         | <b>7.0</b>  | <b>27.2</b> | <b>36.5</b> | <b>17.4</b> | 9.3         | <b>6.9</b>  | 6.0  | 4.7         | 4.7  | <b>17.8</b> |
|           | Q2B   | 23.8        | <b>8.7</b>  | 6.9         | 20.3        | 31.5        | 14.3        | <b>10.7</b> | 5.0         | -    | <b>6.0</b>  | -    | 14.1        |
|           | GQE   | 15.4        | 6.7         | 5.0         | 14.3        | 20.4        | 10.6        | 9.0         | 2.9         | -    | 5.0         | -    | 9.9         |

Table 9: H@1 results (%) of BETAE, Q2B and GQE on answering EPFO ( $\exists$ ,  $\wedge$ ,  $\vee$ ) queries.

| Dataset   | Model | 1p          | 2p          | 3p          | 2i          | 3i          | pi          | ip          | 2u          | up          | avg         |
|-----------|-------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| FB15k     | BETAE | 65.0        | <b>42.1</b> | <b>37.8</b> | <b>52.9</b> | <b>64.0</b> | <b>41.5</b> | <b>22.9</b> | 48.8        | 26.9        | <b>44.6</b> |
|           | Q2B   | <b>67.1</b> | 38.0        | 27.5        | 49.2        | 62.8        | 36.2        | 19.2        | <b>49.0</b> | <b>28.9</b> | 42.0        |
|           | GQE   | 54.6        | 30.5        | 22.2        | 37.7        | 48.4        | 24.8        | 14.7        | 33.8        | 24.7        | 32.4        |
| FB15k-237 | BETAE | 39.1        | <b>24.2</b> | <b>20.4</b> | <b>28.1</b> | <b>39.2</b> | <b>19.4</b> | <b>10.6</b> | <b>22.0</b> | 17.0        | <b>24.4</b> |
|           | Q2B   | <b>40.3</b> | 22.8        | 17.5        | 27.5        | 37.9        | 18.5        | 10.5        | 20.5        | <b>17.4</b> | 23.6        |
|           | GQE   | 35.0        | 19.0        | 14.4        | 22.0        | 31.2        | 14.6        | 8.8         | 15.0        | 14.6        | 19.4        |
| NELL995   | BETAE | <b>53.0</b> | <b>27.5</b> | <b>28.1</b> | <b>32.9</b> | <b>45.1</b> | <b>21.8</b> | 10.4        | <b>38.6</b> | <b>19.6</b> | <b>30.7</b> |
|           | Q2B   | 41.8        | 22.9        | 20.8        | 28.6        | 41.2        | 19.9        | <b>12.3</b> | 26.9        | 15.5        | 25.5        |
|           | GQE   | 32.8        | 19.3        | 17.9        | 23.1        | 31.9        | 16.2        | 10.3        | 17.3        | 13.1        | 20.2        |

Table 10: MRR results (%) on queries from [10], where we show that we are also able to achieve higher performance than baselines Q2B and GQE on all three KGs.

| Dataset   | Model | 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 | <b>0.216</b> | <b>0.357</b> | <b>0.383</b> | <b>0.386</b> | <b>0.299</b> | <b>0.311</b> | <b>0.312</b> | 0.438 | 0.413 | 0.343 | 0.360 | 0.442 |
| FB15k-237 | Q2B   | 0.017        | 0.194        | 0.261        | <b>0.366</b> | <b>0.488</b> | <b>0.335</b> | 0.197        | -     | -     | -     | -     | -     |
|           | BETAE | <b>0.225</b> | <b>0.365</b> | <b>0.450</b> | 0.362        | 0.307        | 0.319        | <b>0.332</b> | 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 | <b>0.236</b> | <b>0.403</b> | <b>0.433</b> | <b>0.404</b> | <b>0.385</b> | <b>0.403</b> | <b>0.403</b> | 0.515 | 0.514 | 0.255 | 0.354 | 0.455 |

Table 11: 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.