ECAI 2023
K. Gal et al. (Eds.)

doi:10.3233/FAIA230432

POINE^2: Improving Poincaré Embeddings for Hierarchy-Aware Complex Query Reasoning over Knowledge Graphs

Junnan Liu[a,b], Qianren Mao[c], Jianxin Li[a,b,c;*], Xingcheng Fu[a,b] and Zheng Wang[d]

[a]School of Computer Science and Engineering, Beihang University, Beijing, P.R.China
[b]Beijing Advanced Innovation Center for Big Data and Brain Computing, Beijing, P.R.China
[c]Zhongguancun Laboratory, Beijing, P.R.China
[d]The School of Computing, University of Leeds, U.K.

Abstract. Reasoning complex logical queries on incomplete and massive knowledge graphs (KGs) remains a significant challenge. The prevailing method for this problem is query embedding, which embeds KG units (i.e., entities and relations) and complex queries into low-dimensional space. Recent developments in the field show that embedding queries as geometric shapes is a viable means for modeling entity set and logical relationships between them. Despite being promising, current geometric-based methods face challenges in capturing hierarchical structures of complex queries, which leaves considerable room for improvement. This paper presents **POINE^2**, a geometric-based query embedding framework based on hyperbolic geometry to handle complex queries on knowledge graphs. POINE^2 maps entities and queries as geometric shapes on a Cartesian product space of Poincaré ball spaces. To capture the hierarchical structures of complex queries, we use the Poincaré radius to represent the different levels of the hierarchy, and we use the aperture of the shape to indicate semantic differences at the same level of the hierarchy. Additionally, POINE^2 offers a flexible and expressive definition of logical operations. Experimental results show that POINE^2 outperforms existing salient geometric-based embedding methods and significantly improves these methods on evaluation datasets.

1 Introduction

Many real-life applications like information retrieval, dialogue, and recommendation systems require reasoning about knowledge graphs (KGs) [16, 24, 17]. One of the fundamental tasks in KG reasoning is to answer complex logical queries with logic operators, including existential quantification ($\exists$), conjunction ($\wedge$), disjunction ($\vee$), and negation ($\neg$). Early methods attempted to find the answers by traversing the KG. However, such a traverse-based strategy is prohibitively expensive [22] since many real-life KGs are large and have a high degree of nodes (i.e., entities). Moreover, because many KGs are incomplete with missing relations, traverse-based methods often cannot locate all answers [22].

Recent approaches attempt to address the drawbacks of traversed-based approaches by adopting an embedding-based approach. This is achieved by representing entities and queries in a low-dimensional embedding space [6, 27, 11, 26]. Since generating the query embedding is a one-off cost and the subsequent query-answering is performed on the embedding space using low-overhead methods like nearest neighbors, embedding-based methods can scale well to massive KGs [11, 22]. As such, embedding-based methods have emerged as the dominant approach for KG reasoning.

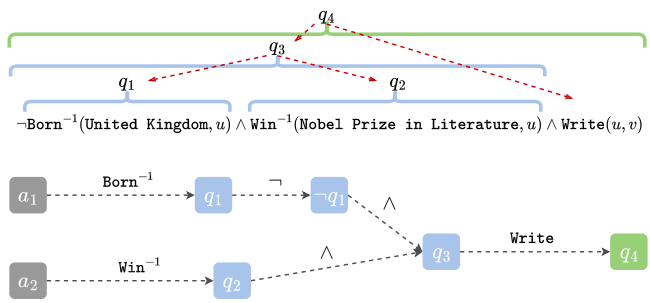


Figure 1: The hierarchical structure of a complex query "*List all the works of the winners of Nobel Prize in Literature, who were not born in the UK.*".

Geometric-based approaches among existing embedding-based methods have shown promising results by representing the query as a specific geometric shape on the embedding space. Query2Box [22], for instance, embeds queries as boxes, where points inside the box are considered as answer entities to the query. Compared to those shallow embedding-based methods [11, 26], such geometric-based embedding methods can naturally represent answer sets of queries and model logical operations among those sets.

However, most existing geometric-based embedding methods have difficulty leveraging hierarchical information of complex query structures. Due to the deductive nature of complex logical queries, a complex query can be progressively decomposed into sub-queries up to one-hop queries. As shown in Figure 1, the complex query $v = ?v : \exists u.\neg\texttt{Born}^{-1}(a_1, u) \wedge \texttt{Win}^{-1}(a_2, u) \wedge \texttt{Write}(u, v)$ can be decomposed into two one-hop queries: $u = ?u : \texttt{Born}^{-1}(a_1, u)$ and $u = ?u : \texttt{Win}^{-1}(a_2, u)$. As a result, the computation graph resembles a *tree* structure, and the associated entities are organized hierarchically. Preserving this hierarchical organization is crucial for accurate

* Corresponding Author. Email lijx@act.buaa.edu.cn.

reasoning, as the superior distribution of embeddings in the embedding space can aid in searching for answer entities. Although some work takes the semantic hierarchies into account [8, 12]. These methods usually cannot adequately model the hierarchical structures of complex queries and only support a subset of logical operations (i.e., Existential Positive First-Order logic operators). As a result, there is still a need to develop a geometric-based embedding approach that can effectively model the hierarchical structures of complex queries.

This paper presents **POINE^2**, a novel geometric-based embedding approach for complex query reasoning[1]. Our proposed POINE^2 model embeds queries and entities as geometric shapes on the Poincaré ball space and provides interpretable parameterizations. The Poincaré radius measures the *inter-level semantic* for different hierarchy levels, while the aperture signifies the *intra-level semantic* at the same level. POINE^2 leverages hierarchical information of complex logical query structures to answer complex logical queries and is equipped with flexible and expressive logical operations including relational projection, intersection, and union. Furthermore, to the best of our knowledge, POINE^2 is the first hyperbolic query embedding method to handle negation operations. During inference, candidate entities are ranked by the distance value between the query embedding and the candidate entity embeddings. We conduct experiments on standard benchmark datasets. Our extensive evaluation shows that POINE^2 can capture the hierarchical structures and outperform the salient geometric-based embedding approaches by a large margin. This paper makes the following contributions:

- We propose a novel complex query reasoning framework designed to capture hierarchical structures of complex queries (Section 4);
- Our framework provides flexible and expressive definitions of logical operations on the Poincaré ball space (Section 4.2);
- Our method outperforms salient baselines and achieves improved performance in experimental results (Section 6.1).

2 Related Work

2.1 Complex Query Reasoning over Knowledge Graphs

Previous works on complex query reasoning over KGs focus on path-based methods [31, 18]. These methods traverse the entire knowledge graph, trying to find all intermediate entities on the path, which leads to heavy computational overhead. Hence, GQE [11] embeds complex queries and entities into low-dimensional embedding spaces and designs neural logical operations to efficiently reason in the embedding space. Then, Query2Box [22] and ConE [33] model the complex queries using box embeddings and cone embeddings, respectively, improving reasoning performance by representing logic queries as geometric shapes (boxes and cones) on the embedding space. Other recent methods [3, 23, 8] try to embed queries into advanced embedding space. For example, BetaE [23] represents queries and entities using probabilistic distribution, i.e., beta distribution. HypE [8] embeds queries and entities as hyperboloids.

Some other works combine neural methods with symbolic algorithms [1, 30] to solve the complex query reasoning problem. CQD [1] converts complex logical query reasoning to a link predictor optimization problem using *t-norms* and *t-conorms*, with continuous optimization (CQD-CO) and beam search (CQD-Beam).

[1] POINE^2 =**Poin**car**é** Ball based Query **E**mbedding. Code and data available at: *https://github.com/spankeran/CQA-PoinE*

Another line of research focuses on solving the Existential Positive First-Order (EPFO) query reasoning task using transformer-based models. BiQE [15] and kgTransformer [19] achieve this by converting the task to a masked entity prediction problem. However, transformer-based models can struggle with negation operations as they divide the computation graph into multiple paths for sequence modeling. Additionally, these models take longer to train and are harder to converge. On the contrary, embedding-based methods utilize neural logical operators to perform logical operations, which is more intuitive and interpretable than implicit learning through deep models such as transformers.

2.2 Hyperbolic Embedding

Hyperbolic embeddings have gained significant attention due to their ability to model data with latent hierarchies. Nickel and Kiela [21] were pioneers in embedding the transitive closure of the WordNet noun hierarchy on the Poincaré ball space. Their research shows that low-dimensional hyperbolic embeddings can outperform higher-dimensional Euclidean embeddings in terms of representation capacity and generalization ability. Based on these findings, Ganea et al. [10] and Shimizu et al. [25] have designed Hyperbolic Neural Networks that extend deep learning techniques to the hyperbolic space.

In the field of knowledge graphs, Balazevic et al. [4], Suzuki et al. [28], and Chami et al. [7] have explored hyperbolic embeddings to solve the knowledge graph completion problem. Their research has shown that hyperbolic-based embedding methods can outperform Euclidean-based methods by considering the semantic hierarchies of the knowledge graph. Our approach is related to the hyperbolic embedding for complex logical reasoning on KGs. We embed entities and queries on the Poincaré ball and use the tangent space for better optimization.

3 Background and Preliminaries

3.1 Complex Queries on Knowledge Graphs

Given a set of entities $\mathcal{V}$ and a set of relations $\mathcal{R}$, a knowledge graph (KG) $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{R})$ is composed of a series of triplets, $\mathcal{E} = \{(h_i, r_i, t_i)\} \subseteq \mathcal{V} \times \mathcal{R} \times \mathcal{V}$, where h_i is the head entity, t_i is the tail entity and r_i means a relation.

Our work addresses the problem of answering complex queries on KGs, specifically First-Order Logic (FOL) queries in the Disjunctive Normal Form (DNF) on KGs. FOL queries in this paper involve logical operators including conjunction ($\wedge$), disjunction ($\vee$), existential quantification ($\exists$), and negation ($\neg$).

Let $\mathcal{V}_a \subseteq \mathcal{V}$ denote a non-variable anchor entity set, and $V_1, \ldots, V_k$ denote existentially quantified bound variables while $V_?$ indicates a single target variable. A valid FOL query in DNF takes the form:

$$q[V_?] = V_?.\exists V_1, \ldots, V_k : c_1 \vee c_2 \vee \ldots \vee c_n, \tag{1}$$

where $c_{i;i=1,\ldots,n}$ indicates a conjunctive formula (conjunctive query) composed of one or more literals e, i.e., $c_i = e_{i,1} \wedge e_{i,2} \wedge \ldots \wedge e_{i,n}$. Each literal indicates an atomic formula. The objective of FOL query reasoning is to identify all *answer entities* denoted by $[\![q]\!] \subset \mathcal{V}$. Each $v \in [\![q]\!]$ must make the above formula true, i.e., $q[v]$ is true. Throughout this paper, we will use $[\![q]\!]$ to represent the answer entity set of conjunctive query q.

A FOL query can be transformed into a computation graph that outlines the reasoning process. Each node in the computation graph

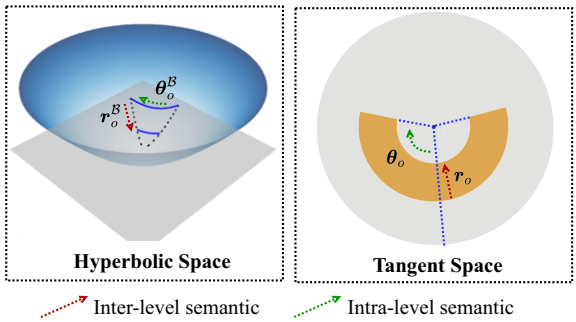


Figure 2: Illustration of POINE^2's query embedding.

represents an entity set (i.e., the answer entity set of sub-query), and each edge indicates an operation, such as relational projection, intersection, union, and negation.

3.2 Hyperbolic Geometry

Hyperbolic geometry is a non-Euclidean geometry with constant negative curvature. In this paper, we utilize the Poincaré ball model, which exhibits negative curvature $-c$. An n-dimensional Poincaré ball model is defined by $(\mathcal{B}^{n,c}, g^{\mathcal{B},c})$, where

$$\mathcal{B}^{n,c} = \{\boldsymbol{x} \in \mathbb{R}^n : \|\boldsymbol{x}\|^2 < \frac{1}{c}\}, \quad g^{\mathcal{B},c}_{\boldsymbol{x}} = (\lambda^c_{\boldsymbol{x}})^2 g^E. \tag{2}$$

Here $\|\cdot\|$ means L_2 norm function, $\lambda^c_{\boldsymbol{x}} = \frac{2}{1-\sqrt{c}\|\boldsymbol{x}\|^2}$ is a conformal factor, and g^E is Euclidean metric tensor. Formally, $\mathcal{B}^{n,c}$ is an open n-dimensional ball with radius $\frac{1}{\sqrt{c}}$. The inner product on the Poincaré ball is induced by $\lambda^c_{\boldsymbol{x}}$: $\langle \boldsymbol{u}, \boldsymbol{v}\rangle^c_{\boldsymbol{x}} = (\lambda^c_{\boldsymbol{x}})^2 \langle \boldsymbol{u}, \boldsymbol{v}\rangle$. The tangent space $\mathcal{T}_{\boldsymbol{x}}\mathcal{B}$ at point $\boldsymbol{x} \in \mathcal{B}^{n,c}$ is a vector space that contains all directions of paths in $\mathcal{B}^{n,c}$ leaving from $\boldsymbol{x}$, which has similar properties to Euclidean space. Moreover, the Poincaré norm (Poincaré radius) is defined as the distance of any point $\boldsymbol{x} \in \mathcal{B}^{n,c} \setminus \{\mathbf{0}\}$ from the origin ($\mathbf{0}$) of the Poincaré ball:

$$\|\boldsymbol{x}\|_{\mathcal{B}} = \frac{2}{\sqrt{c}} \tanh^{-1}(\sqrt{c}\,\|\boldsymbol{x}\|). \tag{3}$$

4 Proposed Framework of POINE^2

Figure 3 depicts the POINE^2 framework. In this section, we first introduce how to embed both entities and queries into the Poincaré ball space in Section 4.1. Then, we describe how neural logical operators can be implemented in our framework in Section 4.2. Finally, we describe the training details of our model and how to predict answer entities using learned embeddings in Section 4.3.

4.1 Embeddings for Entities and Queries

The query embedding is designed to model the answer entity set $[\![q]\!]$ of conjunctive query q. We expect query embedding to form a *geometric shape* on the embedding space that can capture the hierarchical structure of a complex query. To do this, we propose utilizing the *Poincaré radius* to characterize the different levels of the hierarchy, referred to as the *inter-level* semantic, and the *aperture* to capture the semantic differences within the same level of the hierarchy, referred to as the *intra-level* semantic. Further, we define the *center* and *offset* on inter-level and intra-level directions to form a shape.

Specifically, we embed queries into a Cartesian product space $\mathcal{S}$ of d Poincaré planes each of which is a 2-dimensional Poincaré ball space. $\mathcal{S}$ can be denoted by $\mathcal{S} = \mathcal{B}^2_1 \times \ldots \times \mathcal{B}^2_d$, where d denotes the number of Poincaré planes. We use $\boldsymbol{r}^{\mathcal{B},i}_{\mathrm{c}}$ and $\boldsymbol{\theta}^{\mathcal{B},i}_{\mathrm{c}}$ to represent the inter-level center and intra-level center on the i-th Poincaré ball space, respectively. Additionally, we use $\boldsymbol{r}^{\mathcal{B},i}_{\mathrm{o}}$ and $\boldsymbol{\theta}^{\mathcal{B},i}_{\mathrm{o}}$ to represent the inter-level offset and intra-level offset on the i-th Poincaré ball space, respectively. This allows us to summarize the query embedding as:

$$\boldsymbol{V}^{\mathcal{B}}_q = (\boldsymbol{r}^{\mathcal{B}}_{\mathrm{c}}, \boldsymbol{\theta}^{\mathcal{B}}_{\mathrm{c}}, \boldsymbol{r}^{\mathcal{B}}_{\mathrm{o}}, \boldsymbol{\theta}^{\mathcal{B}}_{\mathrm{o}}), \tag{4}$$

where $(\boldsymbol{r}^{\mathcal{B}}_{\mathrm{c}}, \boldsymbol{\theta}^{\mathcal{B}}_{\mathrm{c}})$ represent the semantics of entity set $[\![q]\!]$, and $(\boldsymbol{r}^{\mathcal{B}}_{\mathrm{o}}, \boldsymbol{\theta}^{\mathcal{B}}_{\mathrm{o}})$ indicates the size of $[\![q]\!]$. For efficient optimization, we define all embeddings on the tangent space at the origin of Poincaré ball space, i.e., $\boldsymbol{V}_q = (\boldsymbol{r}_{\mathrm{c}}, \boldsymbol{\theta}_{\mathrm{c}}, \boldsymbol{r}_{\mathrm{o}}, \boldsymbol{\theta}_{\mathrm{o}})$. The tangent space is a vector space that allows us to use advanced optimization methods, such as Adam [14]. $\boldsymbol{V}_q$ can be mapped to $\boldsymbol{V}^{\mathcal{B}}_q$ using the exponential map:

$$\boldsymbol{V}^{\mathcal{B},i}_q = \exp_{\mathbf{0}}(\boldsymbol{V}^i_q). \tag{5}$$

An entity $v \in \mathcal{V}$ can be viewed as an entity set with only one element (i.e., $\{v\}$), which allows us to embed such an entity as a shape with zero offset. The embedding of entity v is represented as $\boldsymbol{v} = (\boldsymbol{r}^v_{\mathrm{c}}, \boldsymbol{\theta}^v_{\mathrm{c}}, \mathbf{0}, \mathbf{0})$, where $(\boldsymbol{r}^v_{\mathrm{c}}, \boldsymbol{\theta}^v_{\mathrm{c}})$ denotes the semantics of the entity v and $\mathbf{0}$ represents the zero vector.

4.2 Logical Operators

In this section, we present the logical operators in POINE^2, which operate on the tangent space. Integration with neural networks, such as a multi-layer perceptron network, is readily possible in the tangent space, facilitating operation design.

Projection Operator $\mathcal{P}$. The projection operator performs relational projection from one query embedding $\boldsymbol{V}_q$ to another query embedding $\boldsymbol{V}'_q$. For each relation $r \in \mathcal{R}$, we assign an embedding $\boldsymbol{R} = (\boldsymbol{r}^r_{\mathrm{c}}, \boldsymbol{\theta}^r_{\mathrm{c}}, \boldsymbol{r}^r_{\mathrm{o}}, \boldsymbol{\theta}^r_{\mathrm{o}})$. Then, we implement the $\mathcal{P}$ operator as:

$$\boldsymbol{V}'_q = f\left(\mathbf{MLP}([\boldsymbol{r}_{\mathrm{c}} \circ \boldsymbol{r}^r_{\mathrm{c}}; \boldsymbol{\theta}_{\mathrm{c}} + \boldsymbol{\theta}^r_{\mathrm{c}}; \boldsymbol{r}_{\mathrm{o}} + \boldsymbol{r}^r_{\mathrm{o}}; \boldsymbol{\theta}_{\mathrm{o}} + \boldsymbol{\theta}^r_{\mathrm{o}}])\right), \tag{6}$$

where $\mathbf{MLP} : \mathbb{R}^{4d} \to \mathbb{R}^{4d}$ is a multi-layer perceptron network, $[;]$ is the concatenation function, $\circ$ is the element-wise multiplication function, and f is a mapping function that ensures the output values from the $\mathbf{MLP}$ are valid. Specifically, we define f as:

$$f(\boldsymbol{x}) = \begin{cases} L \cdot \sigma(\boldsymbol{x}_i), & i \in [0, d), \\ 2\pi \cdot \sigma(\boldsymbol{x}_i) - \pi, & i \in [d, 2d), \\ L \cdot \sigma(\boldsymbol{x}_i), & i \in [2d, 3d), \\ \pi \cdot \sigma(\boldsymbol{x}_i), & i \in [3d, 4d), \end{cases} \tag{7}$$

where $\sigma(\cdot)$ is the sigmoid function. This projection operation is flexible, owing to its capacity to generate query embeddings with adaptive offset sizes. Furthermore, the rotation translation of phase embedding $(\boldsymbol{\theta}_{\mathrm{c}} + \boldsymbol{\theta}^r_{\mathrm{c}})$ means the projection operation can model logical patterns of relations, such as *symmetric* and *anti-symmetric* [27].

Intersection Operator $\mathcal{I}$. The objective of $\mathcal{I}$ is to produce an intersection embedding $\boldsymbol{V}_{q,\mathcal{I}}$ that represents the intersection $\cap^N_{i=1}[\![q_i]\!]$ of N input conjunctive queries $\{q_i\}^N_{i=1}$, where $\boldsymbol{V}_{q,i}$ denotes the embedding of the i-th query. Inspired by the Venn diagram [13], we implement the $\mathcal{I}$ by performing attention-based weighted average over the centers [2] and rescaling the minimum offsets by the permutation-invariant deep neural network [32]. Specifically, we use an attention-based weighted average to generate the inter-level center of $\boldsymbol{V}_{q,\mathcal{I}}$.

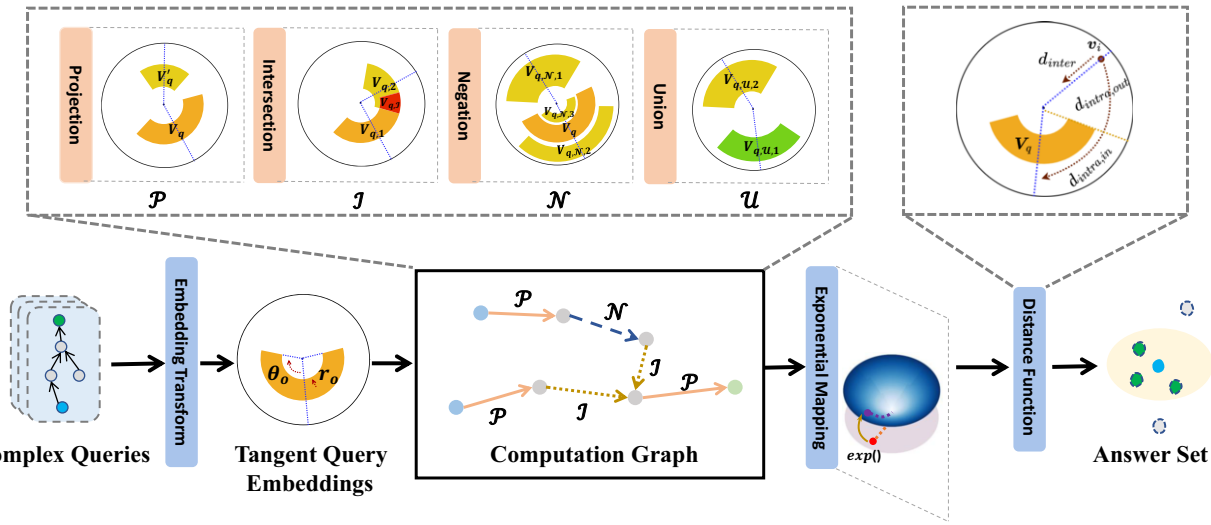


Figure 3: Illustration of the POINE^2 framework. Given a complex query, we begin by transforming its anchor entities to query embeddings on the tangent space. Then, we perform neural logical operations over these embeddings to generate the query embedding. Finally, we utilize the exponential map function to map the query embedding to Poincaré ball space. During the inference phase, candidate entities are ranked by measuring the distance between their embeddings and the query embedding.

Formally, the computation process is:

$$\boldsymbol{r}_{\mathrm{c},\mathcal{I}} = \sum_{i=1}^{N} [\boldsymbol{\alpha}_i \circ \boldsymbol{r}_{\mathrm{c},\mathrm{i}}], \quad \boldsymbol{\alpha}_i = \frac{\exp\left(\mathbf{MLP}(\boldsymbol{V}_{q,i})\right)}{\sum_{j=1}^{N} \exp\left(\mathbf{MLP}(\boldsymbol{V}_{q,j})\right)}, \tag{8}$$

where $\boldsymbol{\alpha}_i$ is attention score vector, $\circ$ is the element-wise multiplication function, and $\mathbf{MLP} : \mathbb{R}^{4d} \to \mathbb{R}^{d}$ is a multi-layer perceptron network for calculating attention score. Similarly, we calculate the intra-level center of $\boldsymbol{V}_{q,\mathcal{I}}$ using attention-based weighted average on $\{\boldsymbol{\theta}_{\mathrm{c,i}}\}_{i=1}^{N}$. Besides, we implement the operation of $\mathcal{I}$ over the offset by adjusting the minimum offset of $\{\boldsymbol{V}_{q,i}\}_{i=1}^{N}$ using neural constraints:

$$\begin{aligned}[\boldsymbol{r}_{\mathrm{o},\mathcal{I}} / \boldsymbol{\theta}_{\mathrm{o},\mathcal{I}}] = \min\left(\{[\boldsymbol{r}_{\mathrm{o},\mathcal{I}} / \boldsymbol{\theta}_{\mathrm{o},\mathcal{I}}]\}_{i=1}^{N}\right) \cdot \\ \sigma\left(\text{DeepSetsNet}\left(\{[\boldsymbol{r}_{\mathrm{o},\mathcal{I}} / \boldsymbol{\theta}_{\mathrm{o},\mathcal{I}}]\}_{i=1}^{N}\right)\right),\end{aligned} \tag{9}$$

where $\min(\cdot)$ is a dimension-wise minimum funciton, $\sigma(\cdot)$ is the sigmoid function, and DeepSetsNet$(\cdot)$ is a permutation-invariant network.

Union Operator $\mathcal{U}$. The objective of the union operator $\mathcal{U}$ is to generate a representation, denoted by $\boldsymbol{V}_{q,\mathcal{U}}$, for a disjunctive query that is represented as the disjunction $\cup_{i=1}^{N} [\![q_i]\!]$ of N input conjunctive queries $\{q_i\}_{i=1}^{N}$. However, directly modeling the disjunction leads to an unscalable situation [22]. Referring to Ren et al. [22], the union operation only appears in the last step when generating embeddings of complex queries in DNF. Therefore, we can use the generated embeddings of the conjunctive queries to represent the disjunction embedding, which is denoted by $\boldsymbol{V}_{q,\mathcal{U}} = \{\boldsymbol{V}'_{q,i}\}_{i=1}^{N}$.

Negation Operator $\mathcal{N}$. The negation operator aims to obtain the complement $[\![\neg q]\!]$ of the input entity set $[\![q]\!]$ over the universe of entities $\mathcal{V}$. We first perform the *difference* operation between the entire space and the region indicated by input query embedding $\boldsymbol{V}_q$. After obtaining the complementary region, we divide it into three sub-complementary regions standing for inter-level complement and intra-level complement as shown in Figure 3. These regions are well-equipped with our definition of query embedding. Hence, we represent the $\boldsymbol{V}_{q,\mathcal{N}}$ via the sub-complements, $\boldsymbol{V}_{q,\mathcal{N}} = \{\boldsymbol{V}_{q,\mathcal{N},i}\}_{i=1}^{3}$.

4.3 Learning Strategy

In this section, we provide the details of the distance function to measure the similarity between entity embeddings and query embeddings. Further, we show how to train POINE^2 using the distance-based loss function.

4.3.1 Distance Function

In our framework, we divide the overall distance into *inter-distance* and *intra-distance*, which respectively denote inter-level distance and intra-level distance. Following Ren et al. [22], we define them as:

$$d(\boldsymbol{v}; \boldsymbol{V}_q) = d_{\mathrm{out}}(\boldsymbol{v}; \boldsymbol{V}_q) + \tau \cdot d_{\mathrm{in}}(\boldsymbol{v}; \boldsymbol{V}_q), \tag{10}$$

where d_{in} means the inside distance and d_{out} means the outside distance. We use the scaling factor $\tau \in (0, 1)$ to rescale the inside distance, so that if $v \in [\![q]\!]$, the distance value between $\boldsymbol{v}$ and $\boldsymbol{V}_q$ would be significantly smaller than the distance between $\boldsymbol{v}'$ and $\boldsymbol{V}_q$ where $v' \notin [\![q]\!]$.

We use inter-distance to distinguish embeddings at the different levels of the hierarchy. Let $\boldsymbol{V}_{q,\min} = (\boldsymbol{r}_{\mathrm{c}} - \boldsymbol{r}_{\mathrm{o}}, \boldsymbol{\theta}_{\mathrm{c}}, \boldsymbol{r}_{\mathrm{o}}, \boldsymbol{\theta}_{\mathrm{o}})$ and $\boldsymbol{V}_{q,\max} = (\boldsymbol{r}_{\mathrm{c}} + \boldsymbol{r}_{\mathrm{o}}, \boldsymbol{\theta}_{\mathrm{c}}, \boldsymbol{r}_{\mathrm{o}}, \boldsymbol{\theta}_{\mathrm{o}})$. We define inter-level distance as:

$$\begin{aligned} d_{\mathrm{inter,in}} &= \left\| \min\left(g\left(\boldsymbol{V}_q^{\mathcal{B}}; \boldsymbol{V}_{q,\min}^{\mathcal{B}}\right), \left|g\left(\boldsymbol{v}^{\mathcal{B}}; \boldsymbol{V}_q^{\mathcal{B}}\right)\right|\right) \right\|_1, \\ d_{\mathrm{inter,out}} &= \left\| \max\left(g\left(\boldsymbol{V}_{q,\min}^{\mathcal{B}}; \boldsymbol{v}^{\mathcal{B}}\right), g\left(\boldsymbol{v}^{\mathcal{B}}; \boldsymbol{V}_{q,\max}^{\mathcal{B}}\right), \mathbf{0}\right) \right\|_1, \end{aligned} \tag{11}$$

where $\|\cdot\|_1$ is L_1 norm function, $\min(\cdot)$ is an element-wise minimum function, $\max(\cdot)$ is an element-wise maximization function and $g(;)$ is a function to calculate the subtraction of the value of Poincaré radius between two Poincaré embeddings.

We use intra-distance to distinguish embeddings at the same hierarchy level. Suppose that $\boldsymbol{\theta}_{\min}$ and $\boldsymbol{\theta}_{\max}$ is the lower bound and upper bound of the $\boldsymbol{\theta}_{\mathrm{c}}$ (e.g., $\boldsymbol{\theta}_{\min} = \boldsymbol{\theta}_{\mathrm{c}} - \boldsymbol{\theta}_{\mathrm{o}}$), the intra-level dis-

tance is defined as:

$$d_{\text{intra,in}} = \left\| \min\left(\left| \sin\left(\frac{\boldsymbol{\theta}_{\text{c}}^{v} - \boldsymbol{\theta}_{\text{c}}}{2} \right) \right|, \left| \sin\left(\frac{\boldsymbol{\theta}_{\text{o}}}{2} \right) \right| \right) \right\|_1,$$
$$d_{\text{intra,out}} = \left\| \min\left(\left| \sin\left(\frac{\boldsymbol{\theta}_{\text{c}}^{v} - \boldsymbol{\theta}_{\min}}{2} \right) \right|, \left| \sin\left(\frac{\boldsymbol{\theta}_{\text{c}}^{v} - \boldsymbol{\theta}_{\max}}{2} \right) \right| \right) \right\|_1, \tag{12}$$

where $\|\cdot\|_1$ is L_1 norm function, $\sin(\cdot)$ is element-wise sine function, and $\min(\cdot)$ is element-wise minimization function.

Finally, the overall distance, with fixed weight parameters of α and β, is defined as:

$$d(\boldsymbol{v}; \boldsymbol{V}_q) = \alpha \cdot d_{\text{inter}}(\boldsymbol{v}; \boldsymbol{V}_q) + \beta \cdot d_{\text{intra}}(\boldsymbol{v}; \boldsymbol{V}_q). \tag{13}$$

As mentioned before, $\mathcal{U}$ and $\mathcal{N}$ will lead to a set of conjunctive query embeddings $\{\boldsymbol{V}'_{q,i}\}_{i=1}^{N}$. We can obtain several distance values $\{d(\boldsymbol{v}, \boldsymbol{V}'_{q,i})\}_{i=1}^{N}$ after performing these operators. However, we only take the smallest one as the final distance value. This is based on the fact that the distance from an entity embedding to the union of several query embeddings equals the minimum distance between the entity embedding and each query embedding separately.

4.3.2 *Training Objective.*

The POINE^2 model is trained using a negative sampling loss function [27]. Specifically, given a query in the training set, the loss function is defined as:

$$\mathcal{L} = -\log \sigma(\gamma - d(\boldsymbol{v}; \boldsymbol{V}_q)) - \frac{1}{k} \sum_{i=1}^{K} \log \sigma(d(\boldsymbol{v}'_i; \boldsymbol{V}_q) - \gamma). \tag{14}$$

Here, $\boldsymbol{v}$ represents positive entities, which are answers to the query. On the other hand, $\boldsymbol{v}'$ represents negative entities, and K represents the number of sampled negative entities. The fixed margin used to distinguish positive and negative entities is denoted by γ.

During inference, candidate entities are ranked based on their distances, and the top n entities are selected as the answer entities for the query.

5 Experimental Setup

Datasets and Queries We conduct experiments on three commonly benchmark datasets: FB15k [5], FB15k-237 [29], and NELL [31]. We also involve another dataset, WN18RR [9], which is a subset from WordNet [20]. Following BetaE [23], we consider 14 query structures as shown in Figure 4, which cover the current comprehensive FOL queries.

Specifically, the training set consists of five Existential Positive First-Order (EPFO) query structures ($1p/2p/3p/2i/3i$), and the other five structures involved negation operation ($2in/3in/inp/pni/pin$). For evaluation and testing, we included four additional structures ($ip/pi/2u/up$) to measure the model's generalization ability in learning the missing facts of KGs. The 'p', 'i', 'u', and 'n' correspond to relational projection, intersection, union, and negation, respectively. And the number before the symbols indicates the number of anchor entity set. For a fair comparison, we use the same queries as those in BetaE [23] on FB15k, FB15k-237, and NELL. For WN18RR, we follow the setup in Query2Box [22] to generate queries. Table 1 shows the number of different queries in the training set, validation set, and test set on benchmark datasets.

Table 1: Statistics of query structures. For the training set, 'EPFO' query structures contain $1p, 2p, 3p, 2i, 3i$, and 'Other' query structures contain $2in, 3in, inp, pni, pin$.

Dataset	Train		Valid		Test	
	EPFO	Other	$1p$	Other	$1p$	Other
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
NELL	107,982	10,798	16,927	4,000	17,034	4,000
WN18RR	103,479	10,347	5,202	1,000	5,356	1,000

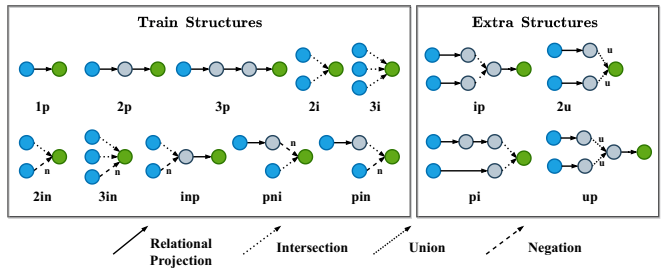


Figure 4: Illustration of query structures used in the experiments. ● denotes the node of the anchor entity set. ● denotes the node of the intermediate entity set. ● denotes the target entity set. The 'p', 'i', 'u', and 'n' in the query structure name represent relational projection, intersection, union, and negation, respectively.

Evaluation Methodology We follow the evaluation protocol in Query2Box [22]. For each answer entity v of a query q, we rank it against non-answer entities. Then we use the *Mean Reciprocal Rank* (MRR) and *Hits at K* (Hits@k) to measure the performance of an approach. Specifically, we compute MRR as the average of the reciprocal ranks for a query q, $MRR = \frac{1}{|\mathcal{A}|} \sum_{i=1}^{|\mathcal{A}|} \frac{1}{rank_i}$, where $\mathcal{A}$ is the answer set. We compute Hits@k as the number of entities whose ranks are below k $Hits@k = \frac{1}{|\mathcal{A}|} \sum_{i=1}^{|\mathcal{A}|} \mathbb{1}[rank_i \leq k]$, where $\mathbb{1}[\cdot]$ is an indication function.

Baseline Methods We compare POINE^2 against several salient geometric-based methods: GQE [11], Query2Box [22] and ConE [33]. GQE embeds queries as points on the embedding space, while Query2Box and ConE embed queries as boxes and cones on the embedding space, respectively. Since GQE and Query2Box cannot handle negation operations, they are trained with only EPFO query structures ($1p/2p/3p/2i/3i$). In addition, we also involve an optimization-based method CQD [1]. In contrast to the former, GQD settles the logical query reasoning problem by decomposing the logical queries to atomic queries and optimizing pre-train link predictors using t-norm to solve atomic queries.

6 Experimental Results

Highlights of our evaluation are:

- POINE^2 outperforms all salient baselines for answering complex queries on KGs (Section 6.1);
- POINE^2 also outperforms methods that are tailored to leverage hierarchies (Section 6.1);
- POINE^2 is efficient at modeling fuzzy entity set (Section 6.2).
- POINE^2 can capture the hierarchical structure of complex logical queries (Section 6.3).

Table 2: MRR results (%) on FB15k, FB15k-237, NELL, and WN18RR. **Avg** $_{\text{p}}$ means the average MRR performance of EPFO queries, which only contains $\exists$, $\wedge$, and $\vee$. **Avg** $_{\text{n}}$ means the average MRR performance of logical queries with negation ($\neg$).

Dataset	Model	Avg_p	Avg_n	1p	2p	3p	2i	3i	ip	pi	2u	up	2in	3in	inp	pin	pni
FB15k	GQE	28.2	N/A	53.9	15.5	11.1	40.2	52.4	27.5	19.4	22.3	11.7	N/A	N/A	N/A	N/A	N/A
	Q2B	38.4	N/A	70.6	22.5	14.1	55.0	66.7	26.0	39.4	35.0	16.7	N/A	N/A	N/A	N/A	N/A
	ConE	49.3	14.7	73.8	32.0	28.5	64.7	74.0	34.2	49.8	56.4	29.9	18.0	17.9	12.5	9.8	15.1
	POINE^2	**53.9**	**15.4**	**79.1**	**39.2**	**32.8**	**66.4**	**75.6**	**42.4**	**53.7**	**59.8**	**36.2**	**19.1**	**18.5**	**13.4**	**10.2**	**15.8**
FB15k-237	GQE	16.3	N/A	35.0	7.2	5.3	23.3	34.6	10.7	16.5	8.2	5.7	N/A	N/A	N/A	N/A	N/A
	Q2B	20.1	N/A	40.6	9.4	6.8	29.5	42.3	12.6	21.2	11.3	7.6	N/A	N/A	N/A	N/A	N/A
	ConE	23.4	5.9	41.8	12.8	11.0	32.6	47.3	14.0	25.5	14.5	10.8	5.4	8.6	7.8	4.0	3.6
	POINE^2	**24.8**	**6.5**	**42.8**	**13.3**	**11.7**	**35.8**	**50.4**	**15.7**	**27.1**	**14.6**	**11.8**	**6.1**	**9.4**	**8.4**	**4.5**	**4.0**
NELL	GQE	18.6	N/A	32.8	11.9	9.6	27.5	35.2	14.4	18.4	8.5	8.8	N/A	N/A	N/A	N/A	N/A
	Q2B	22.9	N/A	42.2	14.0	11.2	33.3	44.5	16.8	22.4	11.3	10.3	N/A	N/A	N/A	N/A	N/A
	ConE	27.2	6.4	53.1	16.1	13.9	40.0	50.8	17.5	26.3	15.3	11.3	5.7	8.1	10.8	3.5	3.9
	POINE^2	**29.0**	**6.9**	**57.0**	**17.8**	**15.6**	**41.2**	**51.9**	**20.5**	**27.6**	**16.4**	**12.9**	**6.2**	**8.5**	**11.5**	**3.9**	**4.2**
WN18RR	GQE	14.5	N/A	21.1	4.8	3.5	28.8	36.6	14.6	13.6	3.2	4.6	N/A	N/A	N/A	N/A	N/A
	Q2B	21.1	N/A	25.3	5.5	3.9	40.8	70.0	21.1	14.3	4.1	5.1	N/A	N/A	N/A	N/A	N/A
	ConE	34.6	23.5	49.5	19.4	15.9	63.5	81.4	35.8	21.1	10.5	14.5	13.5	67.6	8.9	9.2	18.3
	POINE^2	**36.8**	**25.2**	**51.8**	**21.5**	**18.2**	**65.8**	**83.4**	**37.1**	**25.7**	**11.7**	**16.7**	**15.1**	**70.8**	**10.0**	**9.5**	**20.6**

6.1 Main Results

We compare POINE^2 against baselines on queries with and without negation. We run our model five times with different random seeds and report the average performance.

Comparing with salient baselines Table 2 shows the MRR results when applying a method to answer arbitrary complex queries across evaluation datasets, where POINE^2 outperforms all the baselines. For EPFO queries, we observe that POINE^2 achieves a relative improvement of 8.5%, 6.0%, and 5.5% on average MRR performance compared to ConE on FB15k, FB15k-237, and NELL, respectively. Furthermore, for queries with negation, we observe an average MRR performance improvement of 4.8% on FB15k, 10.1% on FB15k-237, and 4.7% on NELL. Additionally, on dataset WN18RR, POINE^2 achieves a relative performance increase over the baselines for all complex queries. The results reveal two key findings: 1) Our approach, POINE^2, is better at answering complex logical queries across evaluation datasets. 2) Although most of the relationships in FB15k and FB15k-237 lack hierarchy (i.e., these KGs lack the semantic hierarchy), our POINE^2 still outperforms previous state-of-the-art methods for these datasets. This indicates that our model can capture latent hierarchies in the topology of complex queries, which helps in answering complex queries.

Comparing with optimization-based baseline In addition, we compare POINE^2 with the optimization-based model, CQD. As shown in Table 3, POINE^2 is superior to CQD, with 11.1% relative improvement on FB15k-237 and 3.6% relative improvement on NELL on average. It is noteworthy that CQD still outperforms POINE^2 on some query structures, such as $1p$. We suspect that CQD optimizes query atoms to answer logical queries, thereby being more sensitive to shorter and simpler query structures. However, POINE^2 can handle more complex query structures, which allows it to outperform CQD on other query structures.

Comparing with hierarchical-aware baselines To validate the importance of capturing the hierarchical structure of complex logical queries, we conduct a comparison of POINE^2 with HypE [8] and LinE [12]. We follow the experiment setup in LinE by dividing 14 query structures into three categories: relation-heavy, logical-heavy, and negation-related. As illustrated in Table 4, POINE^2 outperforms both HypE and LinE in all cases.

Table 3: MRR performance (%) comparison between CQD [15] and POINE^2 on FB15k-237 and NELL.

Model	Avg_p	1p	2p	3p	2i	3i	ip	pi	2u	up
FB15k-237										
CQD	21.7	**46.3**	9.9	5.9	31.7	41.3	**15.8**	21.8	**14.2**	8.6
POINE^2	**24.1**	42.2	**12.6**	**11.3**	**34.6**	**49.3**	15.1	**26.4**	14.1	**11.2**
NELL										
CQD	28.0	**60.0**	16.5	10.4	40.4	49.6	**20.8**	25.6	**16.8**	12.6
POINE^2	**29.0**	57.0	**17.8**	**15.6**	**41.2**	**51.9**	20.5	**27.6**	16.4	**12.9**

Table 4: MRR performance (%) comparison between POINE^2, HypE [8], and LinE [12] on FB15k-237 and NELL. $\textbf{Avg}_p$, $\textbf{Avg}_l$, and $\textbf{Avg}_n$ indicates the average MRR results on relation-heavy ($1p/2p/3p$), logical-heavy ($2i/3i/ip/pi/2u/up$), and negation-related ($2in/3in/inp/pni/pin$) queries, respectively.

Model	FB15k-237			NELL		
	Avg_p	Avg_l	Avg_n	Avg_p	Avg_l	Avg_n
HypE	17.3	8.9	N/A	18.3	14.5	N/A
LinE	18.5	17.7	5.7	22.3	17.5	5.5
POINE^2	**22.6**	**25.9**	**6.5**	**30.1**	**28.4**	**6.9**

6.2 Quality of Modeling Entity Sets

We hypothesize that our POINE^2 can effectively model the fuzzy entity set. To verify this hypothesis, we evaluate POINE^2's ability to model the cardinality (i.e., the number of elements in the set) of the answer set. This evaluation provides insights into POINE^2's capability to model fuzzy entity set. For POINE^2, the offset embeddings $\boldsymbol{r}_\text{o}$ and $\boldsymbol{\theta}_\text{o}$ can signify the learned cardinality of corresponding answer sets. We estimate the learned cardinality by computing the sum of the L_1 norm of $\boldsymbol{r}_\text{o}$ and $\boldsymbol{\theta}_\text{o}$.

We compute the Spearman's rank correlation (SRC) between the learned cardinality and the ground-truth cardinality (i.e., the number of elements in the answer set $[\![q]\!]$) for each query. The SRC is a commonly used measure to assess the statistical dependence between

Table 5: Spearman's rank correlation scores between learned cardinality and the real size of the corresponding answer set.

Model	Avg	1p	2p	3p	2i	3i	ip	pi
FB15k								
Q2B	0.26	0.30	0.22	0.26	0.33	0.27	0.30	0.14
BetaE	0.46	0.37	0.48	0.47	0.57	0.40	0.52	0.42
ConE	0.62	0.60	0.68	0.70	0.68	0.52	0.59	0.56
POINE^2	**0.76**	**0.75**	**0.75**	**0.73**	**0.82**	**0.75**	**0.70**	**0.79**
FB15k-237								
Q2B	0.29	0.18	0.23	0.27	0.35	0.44	0.36	0.20
BetaE	0.51	0.41	0.50	0.57	0.60	0.52	0.54	0.44
ConE	0.72	0.70	0.71	0.74	0.82	0.72	0.70	0.62
POINE^2	**0.81**	**0.79**	**0.81**	**0.79**	**0.88**	**0.85**	**0.74**	**0.83**
NELL								
Q2B	0.32	0.15	0.29	0.31	0.38	0.41	0.36	0.35
BetaE	0.55	0.42	0.55	0.56	0.59	0.61	0.60	0.54
ConE	0.67	0.56	0.61	0.60	0.79	0.79	0.74	0.58
POINE^2	**0.74**	**0.68**	**0.74**	**0.65**	**0.78**	**0.80**	**0.77**	**0.79**

two variables. A higher correlation indicates that the model is more capable of modeling fuzzy entity sets.

Table 5 displays the Spearman's rank correlation results for POINE^2 and baselines on FB15k, FB15k-237, and NELL. Our POINE^2 significantly outperforms the state-of-the-art baseline ConE, demonstrating its superiority in modeling fuzzy entity sets.

6.3 *Analysis of Modeling Hierarchical Structures of Complex Queries*

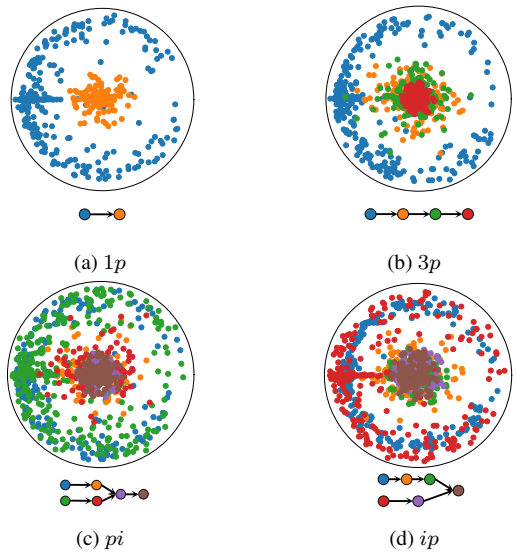
In this section, we visualize the embeddings of several complex queries to further show that POINE^2 can capture the hierarchical structures of complex queries.

Figure 5 visualizes four examples of query embeddings generated during the POINE^2 reasoning procedure on the FB15k dataset. Here, we project the query embeddings onto a 2D space using the following method. We use the center embeddings to indicate the query's position on the embedding space. We map the embeddings in the polar coordinate system to the Cartesian coordinate system and then map them to Poincaré ball via the exponential map function. Note that those embeddings near the bound of the 2D plane have a higher hierarchy level since we use the logarithmic scale to obtain a clearer presentation of the hierarchical gap.

We can observe that the embeddings between different levels have relatively clear boundaries and roughly reflect a hierarchical structure. The observation demonstrates that our POINE^2 can capture the hierarchical structures of complex queries, which accounts for the experiential performance improvements of POINE^2.

7 Conclusion

We have presented POINE^2, a novel geometric-based embedding approach for answering complex queries over incomplete and large knowledge graphs. POINE^2 represents entities and queries as geometric shapes on the Cartesian product space of the Poincaré ball spaces and generates query embeddings via flexible and expressive neural logical operators. POINE^2 can capture hierarchical structures of complex queries and handle FOL operations, including relational projection, intersection, union, and negation. POINE^2 is the first hyperbolic-based query embedding method that supports the negation operation. Extensive experimental results on representative datasets show that POINE^2 outperforms previous geometric-based embedding methods by a large margin.



(a) 1*p* (b) 3*p* (c) *pi* (d) *ip*

Figure 5: Visualization of the query embeddings of four query structures from FB15k. (a): "*List the employees of the Fuji Television.*", (b): "*List the football team that played with the Peruvian national team on the World Cup.*", (c): "*List the artists, origin from the city-town which Brooklyn Law school locates at and Jimmy lives on.*", (d): "*What diseases that occur in the cerebrospinal fluid can be caused by liver cirrhosis.*"

Acknowledgements

We thank the anonymous reviewers for their insightful feedback. This work is supported by the National Natural Science Foundation of China (No.U20B2053).

References

[1] Erik Arakelyan, Daniel Daza, Pasquale Minervini, and Michael Cochez, 'Complex query answering with neural link predictors', in *9th International Conference on Learning Representations, ICLR 2021*, (2021).

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio, 'Neural machine translation by jointly learning to align and translate', in *3rd International Conference on Learning Representations, ICLR 2015*, (2015).

[3] Jiaxin Bai, Zihao Wang, Hongming Zhang, and Yangqiu Song, 'Query2particles: Knowledge graph reasoning with particle embeddings', *arXiv preprint arXiv:2204.12847*, (2022).

[4] Ivana Balazevic, Carl Allen, and Timothy M. Hospedales, 'Multi-relational poincaré graph embeddings', in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019*, pp. 4465–4475, (2019).

[5] Kurt D. Bollacker, Colin Evans, Praveen K. Paritosh, Tim Sturge, and Jamie Taylor, 'Freebase: a collaboratively created graph database for structuring human knowledge', in *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008*, pp. 1247–1250, (2008).

[6] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko, 'Translating embeddings

for modeling multi-relational data', in *Advances in Neural Information Processing Systems 26: Annual Conference on Neural Information Processing Systems 2013, NeurIPS 2013*, pp. 2787–2795, (2013).

[7] Ines Chami, Adva Wolf, Da-Cheng Juan, Frederic Sala, Sujith Ravi, and Christopher Ré, 'Low-dimensional hyperbolic knowledge graph embeddings', in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020*, pp. 6901–6914, (2020).

[8] Nurendra Choudhary, Nikhil Rao, Sumeet Katariya, Karthik Subbian, and Chandan K. Reddy, 'Self-supervised hyperboloid representations from logical queries over knowledge graphs', in *WWW '21: The Web Conference 2021*, pp. 1373–1384, (2021).

[9] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel, 'Convolutional 2d knowledge graph embeddings', in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18)*, pp. 1811–1818, (2018).

[10] Octavian-Eugen Ganea, Gary Bécigneul, and Thomas Hofmann, 'Hyperbolic neural networks', in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018*, pp. 5350–5360, (2018).

[11] William L. Hamilton, Payal Bajaj, Marinka Zitnik, Dan Jurafsky, and Jure Leskovec, 'Embedding logical queries on knowledge graphs', in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018*, pp. 2030–2041, (2018).

[12] Zijian Huang, Meng-Fen Chiang, and Wang-Chien Lee, 'Line: Logical query reasoning over hierarchical knowledge graphs', in *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 615–625, (2022).

[13] Venn M. A J., 'On the diagrammatic and mechanical representation of propositions and reasonings', *Philosophical Magazine Series 5*, **10**(59), 1–18, (1880).

[14] Diederik P. Kingma and Jimmy Ba, 'Adam: A method for stochastic optimization', in *3rd International Conference on Learning Representations, ICLR 2015*, (2015).

[15] Bhushan Kotnis, Carolin Lawrence, and Mathias Niepert, 'Answering complex queries in knowledge graphs with bidirectional sequence encoders', in *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021*, pp. 4968–4977, (2021).

[16] Feng-Lin Li, Minghui Qiu, Haiqing Chen, Xiongwei Wang, Xing Gao, Jun Huang, Juwei Ren, Zhongzhou Zhao, Weipeng Zhao, Lei Wang, Guwei Jin, and Wei Chu, '*AliMe Assist* : An intelligent assistant for creating an innovative e-commerce experience', in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM 2017*, pp. 2495–2498, (2017).

[17] Qian Li, Shu Guo, Cheng Ji, Xutan Peng, Shiyao Cui, and Jianxin Li, 'Dual-gated fusion with prefix-tuning for multi-modal relation extraction', *arXiv preprint arXiv:2306.11020*, (2023).

[18] Xi Victoria Lin, Richard Socher, and Caiming Xiong, 'Multi-hop knowledge graph reasoning with reward shaping', in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 3243–3253, (2018).

[19] Xiao Liu, Shiyu Zhao, Kai Su, Yukuo Cen, Jiezhong Qiu, Mengdi Zhang, Wei Wu, Yuxiao Dong, and Jie Tang, 'Mask and reason: Pre-training knowledge graph transformers for complex logical queries', in *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 1120–1130, (2022).

[20] George A. Miller, 'Wordnet: A lexical database for english', *Commun. ACM*, **38**(11), 39–41, (1995).

[21] Maximilian Nickel and Douwe Kiela, 'Poincaré embeddings for learning hierarchical representations', in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, NeurIPS 2017*, pp. 6338–6347, (2017).

[22] Hongyu Ren, Weihua Hu, and Jure Leskovec, 'Query2box: Reasoning over knowledge graphs in vector space using box embeddings', in *8th International Conference on Learning Representations, ICLR 2020*, (2020).

[23] Hongyu Ren and Jure Leskovec, 'Beta embeddings for multi-hop logical reasoning in knowledge graphs', in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020*, pp. 19716–19726, (2020).

[24] Apoorv Saxena, Aditay Tripathi, and Partha P. Talukdar, 'Improving multi-hop question answering over knowledge graphs using knowledge base embeddings', in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020*, pp. 4498–4507, (2020).

[25] Ryohei Shimizu, Yusuke Mukuta, and Tatsuya Harada, 'Hyperbolic neural networks++', in *9th International Conference on Learning Representations, ICLR 2021*, (2021).

[26] Haitian Sun, Andrew O. Arnold, Tania Bedrax-Weiss, Fernando Pereira, and William W. Cohen, 'Faithful embeddings for knowledge base queries', in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020*, pp. 22505–22516, (2020).

[27] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang, 'Rotate: Knowledge graph embedding by relational rotation in complex space', in *7th International Conference on Learning Representations, ICLR 2019*, (2019).

[28] Atsushi Suzuki, Yosuke Enokida, and Kenji Yamanishi, 'Riemannian transe: Multi-relational graph embedding in non-euclidean space', (2018).

[29] Kristina Toutanova and Danqi 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*, pp. 57–66, (2015).

[30] Zihao Wang, Yangqiu Song, Ginny Y Wong, and Simon See, 'Logical message passing networks with one-hop inference on atomic formulas', *arXiv preprint arXiv:2301.08859*, (2023).

[31] Wenhan Xiong, Thien Hoang, and William Yang Wang, 'Deeppath: A reinforcement learning method for knowledge graph reasoning', in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017*, pp. 564–573, (2017).

[32] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabás Póczos, Ruslan Salakhutdinov, and Alexander J. Smola, 'Deep sets', in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, NeurIPS 2017*, pp. 3391–3401, (2017).

[33] Zhanqiu Zhang, Jie Wang, Jiajun Chen, Shuiwang Ji, and Feng Wu, 'Cone: Cone embeddings for multi-hop reasoning over knowledge graphs', in *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021*, pp. 19172–19183, (2021).