Knowledge Graph Reasoning over Entities and Numerical Values

Jiaxin Bai*
Department of CSE, HKUST
Hong Kong SAR, China
jbai@connect.ust.hk

Chen Luo
Amazon.com Inc
Palo Alto, USA
cheluo@amazon.com

Zheng Li
Amazon.com Inc
Palo Alto, USA
amzzhe@amazon.com

Qingyu Yin
Amazon.com Inc
Palo Alto, USA
qingyy@amazon.com

Bing Yin
Amazon.com Inc
Palo Alto, USA
alexbyin@amazon.com

Yangqiu Song†
Department of CSE, HKUST
Hong Kong SAR, China
yqsong@cse.ust.hk

ABSTRACT

A complex logic query in a knowledge graph refers to a query expressed in logic form that conveys a complex meaning, such as *where did the Canadian Turing award winner graduate from?* Knowledge graph reasoning-based applications, such as dialogue systems and interactive search engines, rely on the ability to answer complex logic queries as a fundamental task. In most knowledge graphs, edges are typically used to either describe the relationships between entities or their associated attribute values. An attribute value can be in categorical or numerical format, such as dates, years, sizes, etc. However, existing complex query answering (CQA) methods simply treat numerical values in the same way as they treat entities. This can lead to difficulties in answering certain queries, such as *which Australian Pulitzer award winner is born before 1927*, and *which drug is a pain reliever and has fewer side effects than Paracetamol*. In this work, inspired by the recent advances in numerical encoding and knowledge graph reasoning, we propose numerical complex query answering. In this task, we introduce new numerical variables and operations to describe queries involving numerical attribute values. To address the difference between entities and numerical values, we also propose the framework of Number Reasoning Network (NRN) for alternatively encoding entities and numerical values into separate encoding structures. During the numerical encoding process, NRN employs a parameterized density function to encode the distribution of numerical values. During the entity encoding process, NRN uses established query encoding methods for the original CQA problem. Experimental results show that NRN consistently improves various query encoding methods on three different knowledge graphs and achieves state-of-the-art results.

---

* Work done during an internship at Amazon.
† Visiting academic scholar at Amazon.

CCS CONCEPTS

• **Information systems** → **Data mining**; • **Computing methodologies** → **Semantic networks**; **Logic programming and answer set programming**.

KEYWORDS

knowledge graph, complex query answering, numerical attribute

1 INTRODUCTION

Reasoning over knowledge graphs (KG) is the process of deriving new knowledge or drawing new conclusions from the existing ones in the KG [9]. Complex query answering (CQA) is a recently developed knowledge graph reasoning task, which aims to answer complex knowledge graph queries [15, 24, 25]. As shown in Figure 1, a complex knowledge graph query, or in short, complex query, targets finding the entities from the KG such that the logic expression can be satisfied [15, 24]. The logic expression of a complex query usually contains multiple terms connected by logic connectives, thus it is able to carry a complicated meaning. For example, in Figure 1, the logic form of q_1 includes terms like *Win(TuringAward,V)* and operations like conjunctions ∧ and disjunctions ∨, and it carries the meaning of *Find where the Canadian Turing award laureates graduated from*. The complex query answering (CQA) task is considered a KG reasoning task because it needs to deal with the incompleteness problem of the KGs. The KGs, like Freebase [8] and YAGO [26], are typically sparse and incomplete. They often include missing relations and attribute values, although these missing edges can be inferred from other edges in the KG. Consequently, subgraph matching algorithms cannot directly be used to find the answers that need to be inferred from missing edges and attributes.

Query encoding methods [4, 14, 15, 20, 24, 25, 27, 39] are proposed to address this incompleteness challenge in CQA. In query encoding methods, queries and entities are simultaneously encoded into the same embedding space. Then the answers are retrieved according to the similarity between the query embedding and entity embedding. In detail, there are two steps in encoding a complex query. First, the query is parsed into a computational graph as a

directed acyclic graph (DAG). Then, the query encoding is iteratively computed along the computational graph, by using neural parameterized logic operations and relational projections. Different query encoding methods encode their queries into different structures, and they have different parameterizations for their logic and projection operations as well. For example, GQE [15] uses vectors, Query2Box [24] uses hyper-rectangles, and Query2Particles [4] uses multiple vectors in the embedding space.

On the other hand, numerical values are also important components of various knowledge graphs like Freebase [8], DBpedia [7], and YAGO [26]. Though existing query encoding methods achieve good performance on complex query answering, they typically focus only on the relations between entities while being unable to reasonably deal with attribute values. There are in general two reasons. First, the query encoding methods are unable to encode the semantic information of numerical values. For q_2 in Figure 1, to find its corresponding answers, we need to first encode the semantics of *the year of 1927* such that we can compute the distributions of *the years before 1927*. However, existing query encoding methods can only learn the graph features of *1927*, and ignore its numerical meaning. Second, the existing query encoding method is unable to describe the distributional changes of values when there are computations and comparisons between numerical values. In Figure 1, the complex queries q_2, q_3, and q_4 involve numerical comparisons *GreaterThan* or *SmallerThan*. Meanwhile, q_4 includes the numerical computation of *TimesByTwo*. As the existing methods only have relational projections between entities, they are insufficient to deal with the relations involving numerical values.

To formally define and evaluate the problem of complex query answering with numerical values, we propose the task of numerical complex query answering (Numerical CQA). In Numerical CQA, we extend the existing problem scope to further include the numerical variables. We also introduce new logic terms expressing numerical relationships. In Figure 1, the logic terms expressing numerical attributes are underlined. Three example numerical queries are given in Figure 1, and their definitions will be given in §2. Meanwhile, we also create the corresponding benchmark dataset to evaluate the performance of Numerical CQA. To do this, we use three public knowledge graphs: Freebase, YAGO, and DBpedia.

To address the challenges of reasonably encoding both entities and numerical values in a query, we propose the Number Reasoning Network (NRN) as a solution. In NRN, there are two encoding phases, which are respectively used for encoding entities and numerical values, following the computation graph. The descriptions of NRN will be given in §3. Experiment results show that the NRN can constantly improve the performance on Numerical CQA when combined with different types of query encoding methods.[1] The major contributions of this paper are three-fold:

- We propose a new problem of numerical complex query answering (Numerical CQA) aiming at answering the complex queries that need reasoning over numerical values.
- We create a new benchmark dataset based on three widely used knowledge graphs: Freebase, Dbpedia, and YAGO, for evaluating the performance of Numerical CQA.

Table 1: The table for frequent notations with definitions.

Notations	Definitions
$\mathcal{G}$	The knowledge graph
$\mathcal{V}$	The set of knowledge graph vertices in KG
$\mathcal{R}$	The set of relations between entities in KG
$\mathcal{A}$	The set of attribute types in KG
$\mathcal{N}$	The set of numerical values in KG
V_a, X_a	The anchor variables that are known in a KG query
V_i, X_i	The existentially quantified variables in a KG query
$r(V, V')$	Whether the relation r holds between V and V'
$a(V, X)$	Whether V has an attribute a with value X
$f(X, X')$	Whether values X and X' satisfy numerical relation f

- We propose Number Reasoning Network (NRN) for CQA. It iteratively encodes a numerical complex query into two different structures for entities and values respectively and achieves state-of-the-art Numerical CQA.

2 DEFINITION OF NUMERICAL CQA

To precisely describe the problem, we mathematically define numerical complex query answering in this section. A multi-relational knowledge graph with numerical attributes is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{R}, \mathcal{A}, \mathcal{N})$. $\mathcal{V}$ is the set of the nodes or vertices, $\mathcal{R}$ is the set of relation types, $\mathcal{A}$ is the set of numerical attribute types, and $\mathcal{N}$ is the set of numerical attribute values. Each vertex $v \in \mathcal{V}$ represents an entity. Each of the relations $r \in \mathcal{R}$ is a binary function defined as $r : \mathcal{V} \times \mathcal{V} \rightarrow \{0, 1\}$. For any $r \in \mathcal{R}$, and $u, v \in \mathcal{V}$, there is a relation r between u and v if and only if $r(u, v) = 1$. Meanwhile, each attribute $a \in \mathcal{A}$ is also a binary function describing numerical attribute values, and it is defined on $\mathcal{V} \times \mathcal{N} \rightarrow \{0, 1\}$. The entity $v \in \mathcal{V}$ has an attribute $a \in \mathcal{A}$ of value $x \in \mathcal{N}$, if and only if $a(v, x) = 1$. In some queries, we also need to express the numerical relations, like comparison and computation between two numerical values, so we use another set of binary functions $f(x_1, x_2) : \mathcal{N} \times \mathcal{N} \rightarrow \{0, 1\}$ to describe whether two numerical values satisfy certain constraints, such as $x_1 < x_2$, $x_1 > x_2 + 20$, or $x_1 = 2x_2 + 3$. The relations are defined as binary functions so that they can be directly used to express boolean values in complex queries, and we will introduce the definition of the complex queries in the following paragraph.

A complex numerical query is defined in existential positive first-order logic form, which is a type of logic expression including existential quantifiers $\exists$, logic conjunctions $\land$, and disjunctions $\lor$. In the logic query, there is a set of anchor entities V_a and values X_a. These anchor entities and values are the given entities and values in a query, like *Beijing* in q_3 and *California* in q_4 in Figure 1. Meanwhile, there are existential quantified entity variables $V_1, ..., V_k$, and numerical variables $X_1, ..., X_l$ in a complex query. There is a unique target variable, $V_?$ in each logic query. The query intends to find the values of $V_?$, such that there simultaneously exist $V_1, ..., V_k \in \mathcal{V}$ and $X_1, ..., X_l \in \mathcal{N}$ satisfying the logic expression. As every logic query can be converted into a disjunctive normal form, the complex numerical queries can be expressed in the following formula:

$$q[V_?] = V_?.V_1, ..., V_k, X_1, ..., X_l : c_1 \lor c_2 \lor ... \lor c_n$$
$$c_i = e_{i,1} \land e_{i,2} \land ... \land e_{i,m}. \tag{1}$$

Here, c_i are conjunctive expressions of several atomic logic expressions $e_{i,j}$, where each $e_{i,j}$ is any of the following expressions:

Query Category	Complex Queries	Interpretations
CQA	$q_1 = V_? . \exists V: Win(V, TuringAward)$ $\wedge IsCitizenOf(V, Canada) \wedge GraduatedFrom(V, V_?)$	Find where the Canadian Turing award laureates graduated from.
Numerical CQA	$q_2 = V_? . \exists X_1, X_2: Win(V_?, PulitzerAward)$ $\wedge \underline{GreaterThan}(1927, X_2) \wedge \underline{BornIn}(V_?, X_2)$	Find the Pulitzer award winners that <u>is born before</u> the year of 1927.
Numerical CQA	$q_3 = V_? . \exists X_1, X_2: Treat(V_?, Pain) \wedge \underline{NumSideEffect}(V_?, X_1)$ $\wedge \underline{SmallerThan}(X_1, X_2) \wedge \underline{NumSideEffect}(Paracetamol, X_2)$	Find the drug that can relieve pain and have <u>fewer side effects</u> than Paracetamol.
Numerical CQA	$q_4 = V_? . \exists X_1, X_2, X_3: LocatedIn(V_?, UnitedStates) \wedge \underline{HasPopulation}(V_?, X_1)$ $\wedge \underline{SmallerThan}(X_1, X_2) \wedge \underline{TimesByTwo}(X_2, X_3) \wedge \underline{HasPopulation}(California, X_3)$	Find the states in US that have a <u>twice smaller population</u> than California?

Figure 1: Examples of entity relations and attribute complex queries. Existing complex query answering research only focuses on entities and their relations. The queries involving numerical values are not defined and studied in the previous literature. Here we underline the logic terms that involve at least one numerical variable in this figure.

(A) *Find the cities that have a <u>higher latitudes than</u> Japanese cities.*

(B) $q = V_? . \exists V_1, X_1, X_2: \underline{HasLatitude}(V_?, X_2)$ $\wedge \underline{GreaterThan}(X_2, X_1)$ $\wedge \underline{HasLatitude}(V_1, X_1)$ $\wedge LocatedIn(V_1, Japan)$
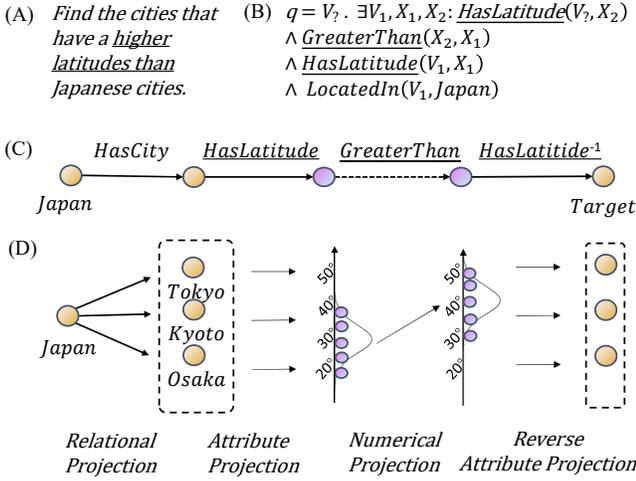

Figure 2: An example query demonstrating the Number Reasoning Network (NRN) for complex query with numerical values. (A) The query interpretation; (B) The logic knowledge graph query involving attribute value reasoning; (C) The computational graph parsed from the logic query; (D) The illustrations of four types of projections. The attribute projection, the numerical projection, and the reverse attribute projections are newly introduced for reasoning over values.

$e_{i,j} = r(V, V')$, $e_{i,j} = a(V, X)$, or $e_{i,j} = f(X, X')$, where the r, a, and f are the binary expressing entity relation, attribute, and numerical relation respectively. They are defined in the previous subsection. V, V', X, X' are either anchor entities or values, or existentially quantified variables in $\{V_1, V_2, ..., V_k\}$ or $\{X_1, X_2, ..., X_l\}$.

3 NUMBER REASONING NETWORK FOR CQA

To deal with the complex queries that involve both entities and numerical values, in this section, we propose the number reasoning network (NRN) method. Different from the original query encoding methods, in NRN, there are two phases in the encoding process to encode the entities and numerical values separately. One of the phases is the entity encoding phase, where we use the existing query encoding structure, like Box Embedding [24], to encode intermediate entity set representation. The other phase is the numerical

encoding phase, in which we use parameterized distribution to encode the intermediate numerical representation. Here is a sample encoding process of NRN. For example, we want to answer the logic query in Figure 2 (B), expressing the meaning of *Find cities that have a higher latitude than Japanese cities.* The query will first be parsed into the computational graph as shown in (C). To start with, we are given the entity embedding of the anchor entity *Japan*, and then a *relational projection* is used to compute the representation of the *cities in Japan*. In these steps, the encoding process is in the entity phase. Then, we will compute the numerical values indicating the *latitude values of cities in Japan* by using *attribute projection*. Then we use another encoding structure to encode the set of numerical values in the numerical encoding phase. Specifically, we use parameterized probabilistic distributions, to estimate the intermediate numerical values. When we obtain the intermediate representation of *latitudes of cities in Japan*, we will estimate the distribution of the latitude values that are larger than previous results by using a *numerical projection* of *GreaterThan*. Finally, a *reverse attribute projection* is computed to find the cities that have such latitude values from the previous step, and the encoding process switches back to the entity phase. In the following part of this section, we are going to first give the full definition of the NRN computational graph, and then introduce the query encoding structure for both entity and numerical variables. Finally, we discuss how to parameterize the operations in the computational graph of NRN.

3.1 Computational Graph for NRN

In the query encoding process for NRN, each intermediate state in the encoding process corresponds to either a set of entities or a set of attribute values. For the computational graph of entity-only complex queries, only relational projections and logic operations appear. Their definitions are as follows:

- *Relational Projection*: Given a set of entities S and a relation between entities $r \in \mathcal{R}$, the relational projection will produce all the entities that hold a relation r with any of the entities in S. Thus the relational projection can be described as $P_r(S) = \{v \in \mathcal{V} | v' \in S, r(v', v) = 1\}$;
- *Intersection*: Given a set of entities or values $S_1, S_2, ..., S_n$, this logic operation of intersection computes the intersection of those sets as $\cap_{i=1}^{n} S_i$;

- *Union*: Given a set of entities or values $S_1, S_2, ..., S_n$, the operation computes the union of $\cup_{i=1}^{n} S_i$.

As shown in Figure 2 (D), to compute the computational graph involving numerical attributes, we need to define three types of new projections. An *attribute projection* is to compute a set of attribute values of a given set of entities. A *reverse attribute projection* is to find a set of entities that have the attribute values in a given set of values. A *numerical projection* is used to find a set of values that holds certain constraints with another set of values, for example, *GreatThan*, *EqualTo*, etc. Their formal definitions are as follows:

- *Attribute Projection*: Given a set of entities S and a numerical attribute type a, the attribute projection gives all the values that can be achieved under the attribute type a by any of the entities in S. Then the attribute projection can be described by $P_a(S) = \{x \in \mathcal{N} | v \in S, a(v, x) = 1\}$;
- *Reverse Attribute Projection*: Given a set of values $S \subset \mathcal{N}$ and a numerical attribute type a, the reverse attribute projection gives all the entities whose attribute a has a value in S. As a result, the reverse attribute projection can be described by the following formula $P_a^{-1}(S) = \{v \in \mathcal{V} | x \in S, a(v, x) = 1\}$;
- *Numerical Projection*: A numerical projection is used to compute a set of values from a set of existing values. This computation depends on the numerical relation function f, and it can be described by $P_f(S) = \{x \in \mathcal{N} | x' \in S, f(x', x) = 1\}$;

3.2 Representations and Operations in NRN

When a complex query is iteratively encoded through the computation graph, as shown in Figure 2 (C), it is either in the entity encoding phase or in the numerical encoding phase. Correspondingly, the intermediate state either represents a set of entities in the knowledge graph or a set of attribute values.

In previous work on query encoding, various methods were proposed to encode entity variables, but they cannot reasonably encode numerical attributes. In this paper, we will focus on how to compute and represent numerical values and how to use them to answer complex queries. For simplicity, suppose in the step i and the encoding process is entity phase, then the query embedding is denoted as: $q_i \in R^d$. Otherwise, if the encoding process is in the numerical phase, queries are represented in a more sophisticated way. Previous research shows that directly using the numerical value itself as input to a reasoning model is not effective. As a result, we employ static numerical encoding methods to project the numerical values from the real space R to the d-dimensional real space R^d, and this numerical encoding is denoted as $\psi(\cdot)$. Meanwhile, we further assume the intermediate numerical answers are drawn from a multi-variant probabilistic distribution in the d-dimensional space.

If the query encoding process is in the numerical encoding phase, the query encoding is represented by a family of probability $p_{\theta_i}(x)$ density functions on R_d, where the parameters are denoted as $\theta_i \in R^k$. Meanwhile, when we have queries like *Find latitudes of the Canadian cities*, we are not only asking for some numerical quantity, but also the types or the units of measure of the quantity, like *Degrees*, *Meters*, *Dates*, or *Years*. Suppose the distribution of type t in the numerical space is described by another distribution ϕ_t.

3.2.1 Relational Projection and Logic Operations on Entities.
In the number reasoning network, we adopt established query encoding methods as the backbones to conduct relational projections and logic operations for entities. In the implementation, we used three different encoding structures to encode the entity phase [4, 15, 24]. Despite their differences, their query embeddings can be uniformly flattened to vectors. Then, the query encoding in the entity phase is expressed as a d-dimensional vector: $q_i \in R^d$.

3.2.2 Attribute Projection.
In the attribute projection, suppose we have query embedding of the entity variable at the i-th step of q_i, and we want to obtain its representation on the attribute values on the attribute type a, we can use a neural structure to parameterize the distribution parameters for the $i + 1$-th step,

$$\theta_{i+1} = F_p(q_i, a), \tag{2}$$

where the F_p is a neural structure whose weights will be optimized in the learning process. A typical parameterization of F_p is gated transition, which is originally inspired by the recurrent neural network unit [4]. Here are the equations for our parameterization:

$$
\begin{aligned}
p_i &= W_p^p q_i + b_p^p \\
z_i &= \sigma(W_z^p e_a + U_z^p p_i + b_z^p), \\
r_i &= \sigma(W_r^p e_a + U_r^p p_i + b_r^p), \\
t_i &= \phi(W_h^p e_a + U_h(r_i \odot p_i) + b_h^p), \\
\theta_{i+1} &= (1 - z_i) \odot p_i + z_i \odot t_i.
\end{aligned}
\tag{3}
$$

Here, $e_a \in R^k$ is the embedding used to encode the attribute type a. Meanwhile, σ and ϕ are the sigmoid and hyperbolic tangent functions, and $\odot$ is the Hadamard product. Also, $W_p^p \in R^{d \cdot k}$ and $W_z^p, W_r^p, W_h^p, U_z^p, U_r^p, U_h^p \in R^{k \cdot k}$ are parameter matrices.

3.2.3 Reverse Attribute Projection.
Suppose we want to find entities that have a numerical attribute value from a given set of values. Then we compute the query embedding for the resulting entity set by using the reverse attribute projection operation:

$$q_{i+1} = F_r(\theta_i, a), \tag{4}$$

where the F_r is a neural structure whose weights will also be optimized in the learning process. Equation (3)'s gated transition function can serve for reverse attribute projection. While the weights are substituted with a different set of parameters $W_p^r, W_z^r, W_r^r, W_h^r \in R^{k \cdot d}$ and $U_z^r, U_r^r, U_h^r \in R^{d \cdot d}$.

3.2.4 Numerical Projection.
The numerical projections are directly applied to the parameters of distributions. As the numerical constraint of f could be an arbitrary binary function from R^2 to $\{0, 1\}$, for now, we use the function F_f to express the projection operation:

$$\theta_{i+1} = F_f(\theta_i). \tag{5}$$

In a typical parameterization, we can use an embedding e_f to encode the characteristics of the numerical function f. The transition of distribution caused the function f to be learned in the training process. For simplicity, we can still use gated transition in Equation (3) while replacing the embedding e_r with e_f, and replacing their weights with $W_j^f, U_j^f \in R^{k \cdot k}$, where $j \in \{z, r, h\}$ respectively.

Table 2: The statistics of the three knowledge graphs used to construct the benchmark.

Graphs	Data Split	#Nodes	#Rel.	# Attr.	#Rel. Edges	#Attr. Edges	#Num. Edges	#Edges
FB15k	Training	25,106	1,345	15	947,540	20,248	27,020	1,015,056
	Validation	26,108	1,345	15	1,065,982	22,779	27,376	1,138,916
	Testing	27,144	1,345	15	1,184,426	25,311	27,389	1,262,437
DB15k	Training	31,980	279	30	145,262	33,131	25,495	237,019
	Validation	34,191	279	30	161,978	37,269	25,596	262,112
	Testing	36,358	279	30	178,394	41,411	25,680	286,896
YAGO15k	Training	32,112	32	7	196,616	21,732	26,616	266,696
	Validation	33,078	32	7	221,194	22,748	26,627	293,317
	Testing	33,610	32	7	245,772	23,520	26,631	319,443

3.2.5 Intersection and Union on Attribute Values. Suppose we have n attribute variables $\theta_i^1, \theta_i^2, ..., \theta_i^n$ as inputs for intersection operations. Their intersection and union are expressed as:

$$\theta_{i+1} = F_i(\theta_i^1, \theta_i^2, ..., \theta_i^n),$$
$$\theta_{i+1} = F_u(\theta_i^1, \theta_i^2, ..., \theta_i^n). \quad (6)$$

For functions F_i and F_u, we use DeepSet [38] functions to parameterize them respectively.

$$\theta_{i+1} = \text{DeepSet}([\theta_i^1, \theta_i^2, ..., \theta_i^n]). \quad (7)$$

The intersection and union operations are thus invariant to input variable permutations. In our implementation, we use a self-attention network followed by a feed-forward network to implement such permutation invariant DeepSet function. Suppose $\theta_i = [\theta_i^1, \theta_i^2, ..., \theta_i^n]$, then their formulas are as follows:

$$a_{i+1} = \text{Attn}(W_q\theta_i^T, W_k\theta_i^T, W_v\theta_i^T)^T \quad (8)$$
$$\theta_{i+1} = \text{MLP}(a_i). \quad (9)$$

The $W_q, W_k, W_v \in R^{k \times k}$ are parameters used for modeling the input Query, Key, and Value for the attention module Attn. The Attn represents the scaled dot-product attention,

$$\text{Attn}(Q, K, V) = \text{Softmax}(\frac{QK^T}{\sqrt{d}})V. \quad (10)$$

Here, the Q, K, and V represent the input Query, Key, and Value for this attention layer. The MLP here denotes a multi-layer perceptron layer with ReLU activation.

3.2.6 Learning Attribute Reasoning. As there are two types of queries with different types of target variables, we define two loss functions respectively. Suppose there are total I steps in the computational graphs, the target variable is an entity variable, and its query embedding is q_I, then we compute the normalized probability of the entity v being the correct answer to the query q by using the softmax function on all similarity scores,

$$p(q, v) = \frac{e^{<q_I, e_v>}}{\sum_{v' \in N} e^{<q_I, e_{v'}>}}. \quad (11)$$

Then the objective function of entities is expressed as

$$L_E = -\frac{1}{N} \sum_{j=1}^{N} \log p(q^{(j)}, v^{(j)}). \quad (12)$$

Here, each $(q^{(i)}, v^{(i)})$ is one of the positive query-answer parties, and there are N such pairs.

Meanwhile, for the numerical variables, we are going to use the maximize a posteriori probability (MAP) estimation to derive an objective function for type-aware attribute reasoning. Suppose the parameters of the attribute distribution of the final step are θ_I, the positive answer is v, and the answer has a value type t. The θ_I is computed from θ_i and q_i where $i \in \{1, 2, 3, ..., I - 1\}$ by using a series of projections like F_p, F_r, F_f, and logic operations F_i and F_u. Then our goal is to optimize θ_I given v and t. Then suppose f and g are the conditional distribution of v given θ_I and t, and θ_I given t, and meanwhile v is conditional independent of t given θ_I. Then according to MAP estimation,

$$\begin{aligned} \hat{\theta}_I(v, t) &= \arg\max_{\theta_I} f(\theta_I|v, t) \\ &= \arg\max_{\theta_I} \frac{f(v|\theta_I, t)g(\theta_I|t)}{\int_{\theta_I} f(x|\theta, t)g(\theta|t)d\theta} \\ &= \arg\max_{\theta_I} f(v|\theta_I, t)g(\theta_I|t) \\ &= \arg\min_{\theta_I} (-\log f(v|\theta_I) - \log g(\theta_I|t)). \end{aligned} \quad (13)$$

Suppose $(q^{(j)}, v^{(j)})$ is a pair of positive query-answer pairs, and we have totally M such pairs. Suppose the value type of $v^{(i)}$ is $t^{(i)}$. Using the previous defined notions, then $f(v^{(j)}|\theta_I^{(j)})$ can be written as $p_{\theta_I^{(j)}}(v^{(j)})$, and $g(\theta_I|t))$ can be written as $\phi_{t^{(j)}}(\theta_I^{(j)})$. Then, the loss of explicit queries can be described by

$$L_A = \frac{1}{M} \sum_{j=1}^{M} (-\log p_{\theta_I^{(j)}}(\psi(v^{(j)})) - \log \phi_{t^{(j)}}(\theta_I^{(j)})), \quad (14)$$

where the $\psi(\cdot)$ is numerical encoding defined on the domain of R with the range of R^d. In the training process, we alternatively optimize these two loss functions to train all parameters in different neural structures together.

4 BENCHMARK CONSTRUCTION

In this section, we construct three benchmark datasets for evaluating the performance of numerical complex query answering.

Table 3: The statistics of the queries sampled from the FB15k, DB15k, and YAGO15k knowledge graphs.

Graphs	Data Split	1p	2p	2i	3i	pi	ip	2u	up	All
FB15k	Training	304,633	138,192	226,729	288,874	260,057	233,834	284,301	284,931	2,021,551
	Validation	8,271	15,860	23,359	28,836	25,081	22,930	29,187	29,210	182,734
	Testing	7,969	15,431	23,346	28,865	24,810	22,232	29,212	29,274	181,139
DB15k	Training	124,851	99,698	140,427	190,413	171,353	163,687	190,364	194,244	1,275,037
	Validation	3,529	10,388	9,792	13,817	14,594	16,651	19,512	19,792	108,075
	Testing	3,387	10,047	9,914	14,603	14,642	15,897	19,504	19,773	107,767
YAGO15k	Training	84,014	76,238	136,282	183,850	162,712	145,994	183,963	183,459	1,156,512
	Validation	2,833	7,986	10,757	16,884	13,485	13,899	18,444	19,105	103,393
	Testing	2,713	7,949	10,935	17,171	13,481	13,526	18,433	18,997	103,205
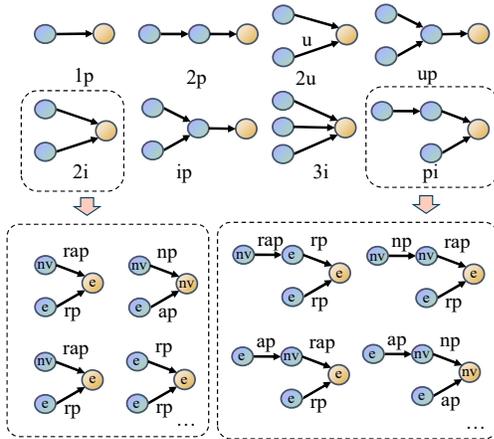
Figure 3: Eight general query types. Each general query type corresponds to multiple specific query types.

Figure 4: The distribution of the types of numerical values on three different knowledge graphs.

4.1 Knowledge Graphs

We use FB15k, DB15k, and YAGO15k [17] as the knowledge graphs to create our benchmarks. These KGs are publicly available, and they include both triples describing entity relations and numerical values. For each knowledge graph, we first randomly divide the edges into training, validation, and testing edges with a ratio of 8:1:1 respectively. Then, the training graph $\mathcal{G}_{train}$, validation graph $\mathcal{G}_{val}$, and testing graph $\mathcal{G}_{test}$ are aggregated from the training edges, training+validation edges, and training+validation+testing edges respectively. The detailed statistics of the knowledge graphs are shown in Table 2. The columns #Rel. and #Attr. denote the number of entity relation types and the number of attribute types respectively. #Rel. Edges, #Attr. Edges, and #Num. Edges represent the total number of edges describing entity relations, attribute values, and numerical relations between values.

4.2 Query Types

Following previous work [15], we used the following eight types of query as general query types. For each general query type, each edge represents a projection, and each node either represents either a set of entities or numerical values. Based on these query types and knowledge graph structures, we further derive various specific sub-types. For example, the query type *pi* can be instantiated

into various types of specific types of queries. As shown in Figure 3, *e* stands for an entity, *nv* stands for numerical values, and *ap*, *rap*, *rp*, and *np* are the abbreviations of *attribute projection*, *reverse attribute projection*, *relational projection*, and *numerical projections* respectively. The general query types are shown in Table 4.

4.3 Sampling of Complex Attribute Queries

In this section, we show how to sample numerical complex queries. Before sampling, we added dummy numerical edges to the graphs to evaluate the performance on numerical relations. For simplicity, we consider seven types of numerical relations: *EqualTo*, *SmallerThan*, *GreaterThan*, *TwiceEqualTo*, *ThreeTimesEqualTo*, *TwiceGreaterThan*, and *ThreeTimesGreaterThan*. These numerical relations are randomly added to the attribute value pairs with the same value type and satisfy the corresponding constraints. For each type of numerical relation, we set an upper limit of four thousand edges on each graph. The total number of numerical edges added is shown in Table 2. There are two general steps in sampling the complex attribute queries. In the first step, we are given eight types of general queries with abbreviations of *1p,2p,2i,3i,ip,ip,2u*, and *up*. Their definitions are given in Table 3. Given the general query type and a knowledge graph G, a sampling algorithm is used to generate a complex attribute query that has at least one answer. The details of the algorithm are given in the Appendix. In the second step, given a complex attribute query from the first step, we conduct a graph search and find its corresponding answers on the KG.

For each knowledge graph, we have constructed their corresponding Training, Validation, and Testing graphs. Then, the first

Table 4: The abbreviations of the query types and structures.

Abbr.	General Query Type	Specific Query Type Example
1p	(p,(e))	(rap,(nv))
2p	(p,(p,(e)))	(rap, (np,(nv)))
2i	(i,(p,(e)),(p,(e)))	(i, (np,(nv)), (ap, (e)))
3i	(i,(p,(e)),(p,(e)),(p,(e)))	(i, (np,(nv)), (ap, (e)), (ap, (e)))
ip	(p,(i,(p,(e)),(p,(e))))	(rp, (i, (np,(nv)), (ap, (e))))
pi	(i,(p,(e)),(p,(p,(e))))	(i, (np,(nv)), (ap, (rp, (e))))
2u	(u,(p,(e)),(p,(e)))	(u, (np,(nv)), (ap, (e)))
up	(p,(u,(p,(e)),(p,(e))))	(rp, (u, (np,(nv)), (ap, (e))))

step of the previously described query sampling process is conducted on these three data splits independently. After this, we obtained the training queries, validation queries, and test queries respectively. Then, For the training queries, we conduct a graph search on the training graph to find corresponding training answers. For the validation queries, we search their answers on both the training graph and the validation graph, and we only use the queries that have a different number of training and validation answers. Finally, for the testing queries, the graph searches are conducted on training, validation, and testing respectively. Similarly, we only use the testing queries that have a different number of answers on validation and test graphs. After these two steps are done, the statistics of the resulting training, validation, and testing queries of three knowledge graphs are shown in Table 3.

5 EXPERIMENT

In this section, we implement the proposed number reasoning network (NRN) and compare it to existing query encoding models.

5.1 Baseline Methods

NRN uses previous query encoding methods to encode entities during the entity encoding phase. It can be combined with a wide range of entity encoding structures that can be transformed into a fixed length of real-valued vectors. We use the following methods:

- GQE [15]: The graph query encoding model encodes a complex logic query into a single vector;
- Q2B [24]: The Query2Box model encodes a complex query into hyper-rectangles in the embedding space;
- Q2P [4]: The Query2Particles model encodes a complex logic query into multiple vectors.

5.2 Numerical Encoding

As explained in §3.2, we use numerical encoding methods to project the attribute values from R to R^D to obtain number embedding for NRN. We use the following two deterministic numerical encodings.

- DICE [28]: DICE is a deterministic independent-of-corpus word embedding of numbers. DICE managed to use a linear mapping from the numerical values v to an angle $\alpha \in [0, \pi]$, and then conduct a polar-to-Cartesian transformation in the D-dimensional space according to the following formula:

$$\psi(v)_d = \begin{cases} \sin^{d-1}(\alpha)\cos(\alpha), & 1 \le d < D \\ \sin^D(\alpha), & d = D \end{cases} \quad (15)$$

Table 5: The main experiment results of numerical complex query answering. NRN constantly outperforms the baseline methods on Hit@1, Hit@3, Hit@10, and MRR.

Dataset	Entity	Value	H@1	H@3	H@10	MRR
FB15k	GQE	Baseline	13.56	23.47	34.80	20.80
		NRN + DICE	14.16	24.69	36.18	21.72
		NRN + Sinusoidal	**14.33**	**24.81**	**36.25**	**21.86**
	Q2B	Baseline	14.49	25.96	38.20	22.56
		NRN + DICE	15.55	27.31	39.24	23.72
		NRN + Sinusoidal	**15.78**	**27.65**	**39.54**	**23.96**
	Q2P	Baseline	15.54	25.50	36.88	22.84
		NRN + DICE	16.98	27.02	38.36	24.27
		NRN + Sinusoidal	**17.72**	**27.96**	**39.23**	**25.09**
DB15k	GQE	Baseline	6.86	12.98	21.28	11.73
		NRN + DICE	7.26	13.36	21.81	12.15
		NRN + Sinusoidal	**7.30**	**13.63**	**22.32**	**12.34**
	Q2B	Baseline	7.47	14.17	22.81	12.66
		NRN + DICE	**8.07**	**15.21**	**24.22**	**13.53**
		NRN + Sinusoidal	8.05	15.04	24.05	13.45
	Q2P	Baseline	8.41	14.57	22.84	13.31
		NRN + DICE	8.81	15.10	23.27	13.73
		NRN + Sinusoidal	**9.13**	**15.69**	**24.22**	**14.25**
YAGO15k	GQE	Baseline	10.56	18.12	27.65	16.35
		NRN + DICE	11.68	19.49	29.05	17.59
		NRN + Sinusoidal	**11.78**	**19.74**	**29.12**	**17.73**
	Q2B	Baseline	13.47	22.67	32.56	20.02
		NRN + DICE	13.94	23.74	33.56	20.77
		NRN + Sinusoidal	**14.41**	**23.96**	**33.78**	**21.13**
	Q2P	Baseline	6.70	11.98	20.11	11.29
		NRN + DICE	9.80	**16.98**	**26.74**	**15.53**
		NRN + Sinusoidal	**9.90**	16.83	26.43	15.49

- Sinusoidal [30]: The Sinusoidal encoding of value is first proposed in encoding token position in the transformer model. In this paper, we found that this function is also effective in encoding numerical values. Its formula is expressed by:

$$\psi(v)_d = \begin{cases} \sin(\frac{v}{n^{d/D}}), & d \equiv 0 \quad \mod 2 \\ \cos(\frac{v}{n^{(d-1)/D}}), & d \equiv 1 \quad \mod 2 \end{cases} \quad (16)$$

These two functions are both deterministic and preserve geometric properties from the original value space to the embedding space.

5.3 Evaluation Metrics

For a testing query q, the training, validation, and testing answers are denoted as $[q]_{train}$, $[q]_{val}$, and $[q]_{test}$ respectively. In the experiment, we evaluate the generalization capability of the attribute reasoning models by computing the rankings of the answers that cannot be directly searched from an observed knowledge graph. Suppose $[q]_{val}/[q]_{train}$ represents the answer set of the query q that can be searched from the validation graph but cannot be searched from the training graph. Similarly, $[q]_{test}/[q]_{val}$ is the set of answers that can be found on the test graph but cannot be found in the validation graph. Then, the evaluation metrics of the test query q can be expressed in the following equation:

$$\text{Metric}(q) = \frac{1}{|[q]_{test}/[q]_{val}|} \sum_{v \in [q]_{test}/[q]_{val}} m(rank(v)). \quad (17)$$

When the evaluation metric is Hit@K, the $m(r)$ in Equation (17) is defined as $m(r) = \mathbf{1}[r \le K]$. In other words, $m(r) = 1$ if $r \le K$,

Table 6: The mean reciprocal ranking (MRR) results for different types of numerical complex queries.

Dataset	Entity Encoding	Values Encoding	1p	2p	2i	3i	pi	ip	2u	up
FB15k	GQE	Baseline	30.63	7.54	32.70	39.61	21.57	8.87	7.18	4.17
		NRN + DICE	**31.75**	**8.16**	33.96	40.32	22.70	9.80	**8.45**	**4.66**
		NRN + Sinusoidal	31.69	8.15	**33.99**	**41.19**	**23.10**	**10.10**	7.57	4.51
	Q2B	Baseline	35.16	7.12	35.39	41.31	22.99	9.17	12.76	4.02
		NRN + DICE	36.96	7.66	36.50	42.30	24.17	**10.41**	14.53	**4.79**
		NRN + Sinusoidal	**37.01**	**7.74**	**36.88**	**42.72**	**24.25**	10.09	**15.61**	4.54
	Q2P	Baseline	39.29	12.45	30.40	35.29	20.09	11.05	21.25	7.80
		NRN + DICE	41.45	12.65	32.91	37.61	22.02	12.85	21.74	**8.03**
		+ Sinusoidal	**42.75**	**12.87**	**33.71**	**38.75**	**23.14**	**13.23**	**23.16**	7.89
DB15k	GQE	Baseline	9.83	2.41	18.83	34.48	11.21	2.11	1.86	1.94
		NRN + DICE	**10.46**	**2.58**	**20.17**	34.80	11.88	2.48	1.94	2.03
		NRN + Sinusoidal	10.29	2.53	20.14	**35.46**	**12.50**	**2.52**	**2.08**	**2.14**
	Q2B	Baseline	10.18	2.53	20.81	36.29	12.57	2.72	2.06	1.67
		NRN + DICE	**11.01**	2.66	22.38	**37.47**	**14.17**	**3.09**	**2.62**	**2.20**
		NRN + Sinusoidal	10.96	**2.71**	**22.60**	37.44	13.81	3.05	2.41	2.13
	Q2P	Baseline	14.44	**3.96**	20.67	33.85	13.64	**3.48**	4.66	**2.96**
		NRN + DICE	14.58	3.81	22.15	35.36	13.90	2.83	**4.94**	2.67
		NRN + Sinusoidal	**14.71**	3.81	**23.75**	**36.66**	**14.47**	2.96	4.63	2.81
YAGO15k	GQE	Baseline	14.28	3.18	32.83	37.83	15.91	4.89	3.62	1.82
		NRN + DICE	**15.21**	4.20	35.08	**40.32**	17.01	**5.71**	3.95	1.95
		NRN + Sinusoidal	14.79	**4.23**	**35.68**	39.64	**18.29**	5.65	**4.23**	**1.96**
	Q2B	Baseline	18.84	3.91	38.62	44.67	18.72	7.31	6.90	2.47
		NRN + DICE	20.83	4.53	38.87	**45.19**	**20.61**	7.69	8.72	2.73
		NRN + Sinusoidal	**21.40**	**4.59**	**39.72**	45.16	19.62	**7.90**	**9.05**	**2.82**
	Q2P	Baseline	19.84	4.64	17.42	19.34	11.14	4.33	9.92	3.07
		NRN + DICE	22.68	5.36	25.45	**29.70**	14.07	**5.67**	11.73	3.47
		NRN + Sinusoidal	**22.97**	**5.70**	**25.70**	29.04	**14.38**	5.40	**11.87**	**3.51**

otherwise $m(r) = 0$. Meanwhile, if the evaluation metric is mean reciprocal ranking (MRR), then the $m(r)$ is defined as $m(r) = \frac{1}{r}$. Similarly, if q is a validation query, its generalization capability is evaluated on $[q]_{val}/[q]_{train}$ instead of $[q]_{test}/[q]_{val}$. In our experiment, we train all the models by using the training queries, and we select the hyper-parameters by using the validation queries. The evaluation is then finally conducted on the testing queries.

5.4 Experiment Results

As discussed in §5.1, we use three different query encoding methods GQE [15], Q2B [24], and Q2P [4] as the backbone for entities in our framework. Meanwhile, to evaluate the performance of NRN, we also use the original version of the GQE, Q2B, and Q2P methods as baselines. In these baselines, all attribute values in the knowledge graphs are treated in the same way as other entities in the knowledge graph. In our proposed NRN framework, we use two ways of incorporating attribute value information in the reasoning model. First, we use the DICE [28] and Sinusoidal [30] function to encode attribute values as the numerical encoding function $\psi(v)$ in Equation (14). Meanwhile, for simplicity, we choose to use the multi-variant Gaussian distributions as the distribution family used for the distribution of numerical queries $p_\theta(x)$ and the distribution of the parameters given the value type $\phi_t(\theta)$ in Equation (14). Experiment results are reported in Table 5 and Table 6.

Table 5 reports the averaged results in Hit@1, Hit@3, Hit@10, and MRR. The experiment results show that the number reasoning networks constantly outperform their baseline reasoning methods on three different knowledge graphs under four different evaluation metrics. The difference between the baseline query encoding method and our number reasoning network is that we use parameterized distributions to encode the numerical attribute values instead of using a unified query encoding structure. As a result, from the empirical results, we can conclude that parameterized distributions are effective encoding structures for numerical attributes. On the other hand, the Sinusoidal function performs on par or even better than DICE on different KGs under various metrics. Thus Sinusoidal is a simple yet effective method to encode the magnitude of knowledge graph attribute values. Moreover, the mean reciprocal rank (MRR) scores for different types of queries are shown in Table 6. Empirical results show that NRN performs better than the baselines on almost every type of query on the three datasets.

5.5 Computing Time Analysis

Further analysis on computing time is also conducted. In this experiment, we measure the average training and inference time on FB15k, DB15k, and YAGO15K with the Q2P. The experiment results are shown in Table 7. We run our experiments on the RTX2080ti graphics card with a batch size of 1024. Then we compute the average training and inference time per query using the batch processing time divided by the batch size. The empirical results show that the average inference time of NRN is 0.11 milliseconds longer than the baseline, and the training time is 0.16 milliseconds longer. Although

Table 7: The average training time and inference time per query in milliseconds (ms) of complex numerical queries. Although the training and inference are slower than the baseline, NRN still achieves high training and inference speed.

Time (ms)	FB15k		DB15k		YAGO15k		Average	
	Training	Inference	Training	Inference	Training	Inference	Training	Inference
Baseline	0.090	0.042	0.078	0.031	0.088	0.040	0.085	0.037
NRN	0.253	0.149	0.229	0.134	0.255	0.152	0.245	0.145

the computing time is longer than the baseline model, the NRN is still fast and effective.

6 RELATED WORK

The query encoding methods are closely related to this work. Some methods use different geometric to encode queries. The GQE [15] model encodes a KG query as a vector in embedding space. Query2Box [24] encodes a KG query as a hyper-rectangle in an embedding space. It can be used for answering existential positive first-order (EPFO) complex queries. EmQL [27] proposes to use count sketches to improve the faithfulness of the query embeddings by using count-min sketches. ConE [39] improves the Q2B by using cone embeddings. The cone embeddings are able to express negation and encode arbitrary first-order logic queries. CylE [22] proposes to use cylinders to encode complex queries. Q2P [4] encodes a logic query into multiple vectors in the embeddings space. Meanwhile, HypE [11] encodes the complex queries as hyperbolic query embedding. Wang et al. [32] propose Wasserstein-Fisher-Rao Embedding for CQA. Meanwhile, some methods try to encode logic queries into specific probabilistic structures. Beta Embedding [25] proposes to use multiple Beta distributions to encode logic knowledge graph queries expressed in First-order logic queries. Yang et al. [37] propose to use Gamma distributions to encode complex queries. Then, PERM [10] embeddings encode the complex queries by using a mixture of Gaussians. Recently, Line Embedding [16] is proposed to relax the distributional assumptions by using a logic space transformation layer to convert probabilistic embeddings to LinE space embedding. Meanwhile, there is some work attempting to use neural structures to encode complex queries. The transformer structure with specially designed graph hierarchy encoding is proposed by BiQE [18] for complex queries. Meanwhile, Bai et al. [5] propose to linearize the computational graph and then use sequence encoders to encode queries. Newlook [20] proposes to use different types of neural networks to iteratively encode complex queries. Simultaneously, [1] managed to use MLP and MLP-Mixers [29] to achieve improved performance. ENeSy [36] uses an entangled method of the neural and symbolic method to conduct query encoding. QE-GNN [40] uses the message passing over KG to conduct query encoding. The Mask-and-Reasoning [21] proposes to use a knowledge graph transformer, which is a type of graph neural network, to conduct pre-training on the knowledge graph. The pre-trained KG transformer is then fine-tuned to answer complex queries.

Meanwhile, query decomposition [2] is another line of research for answering complex KG queries. In the query decomposition method, a complex query is first decomposed into atomic queries. Then the probabilities of atomic queries are computed by a neural link predictor. Recently, Wang et al. [33] propose to use one-hop message passing on query graphs to conduct complex query answering. In the inference process, either continuous optimization

or beam search is used for finding the answers. More recently, SMORE [23] is proposed as a framework to train and evaluate large scaled knowledge graph completion and complex query answering models. The compositional generalization capability of complex query answering is bench-evaluated by the EFO-1 dataset [34]. On the other hand, ROMA [35] is proposed to answer complex logic queries on multi-view knowledge graphs. Though existing methods can effectively conduct reasoning on entities and their relations on a knowledge graph, they cannot be directly used to reasonably answer numerical complex queries. Recently, Bai et al. [3] propose to use memory-enhanced query encoding for complex query answering on eventuality knowledge graphs.

Prior research has explored the treatment of numerical attributes in knowledge graphs. KBLRN [13] proposes to learn the representation of knowledge bases by jointly using latent, relational, and numerical features. KR-EAR [19] proposes a knowledge representation model to jointly learn the entity, relations, and attributes in knowledge graphs. Numerical value propagation [17] is proposed to predict the value of entities' numerical attributes. Multi-relational attribute propagation [6] is then proposed to improve attribute completion by using message passing on a knowledge graph. However, these methods are not able to answer complex logic queries. Duan et al. [12] propose to use the KG embedding methods to train the representations of numerical values over synthetic numerical relations, and this method can is an alternative to the DICE and Sinusoidal for encoding real numbers for NRN. Meanwhile, Neural-Num-LP [31] proposes to learn multi-hop rules involving numerical attributes from a knowledge graph. However, it still cannot deal with complex logic queries.

7 CONCLUSION

In this work, we proposed the new task of numerical complex attribute query answering (Numerical CQA). Meanwhile, we constructed a benchmark dataset based on three public KGs: FB15k, YAGO15k, and DBpedia15k. Finally, we proposed a new framework that can conduct numerical attribute value answering. Experiments show that our number encoding network (NRN) with two-phase query encoding can significantly outperform the previous query encoding methods and achieve state-of-the-art Numerical CQA.

REFERENCES

[1] Alfonso Amayuelas, Shuai Zhang, Susie Xi Rao, and Ce Zhang. 2022. Neural Methods for Logical Reasoning over Knowledge Graphs. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net. https://openreview.net/forum?id=tgcAoUVHRIB

[2] Erik Arakelyan, Daniel Daza, Pasquale Minervini, and Michael Cochez. 2021. Complex Query Answering with Neural Link Predictors. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net. https://openreview.net/forum?id=Mos9F9kDwkz

[3] Jiaxin Bai, Xin Liu, Weiqi Wang, Chen Luo, and Yangqiu Song. 2023. Complex Query Answering on Eventuality Knowledge Graph with Implicit Logical Constraints. arXiv:2305.19068 [cs.CL]

[4] Jiaxin Bai, Zihao Wang, Hongming Zhang, and Yangqiu Song. 2022. Query2Particles: Knowledge Graph Reasoning with Particle Embeddings. In *Findings of the Association for Computational Linguistics: NAACL 2022*. Association for Computational Linguistics, Seattle, United States, 2703–2714. https://doi.org/10.18653/v1/2022.findings-naacl.207

[5] Jiaxin Bai, Tianshi Zheng, and Yangqiu Song. 2023. Sequential Query Encoding For Complex Query Answering on Knowledge Graphs. *CoRR* abs/2302.13114 (2023). https://doi.org/10.48550/arXiv.2302.13114 arXiv:2302.13114

[6] Eda Bayram, Alberto García-Durán, and Robert West. 2021. Node attribute completion in knowledge graphs with multi-relational propagation. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 3590–3594.

[7] Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. 2009. Dbpedia-a crystallization point for the web of data. *Journal of web semantics* 7, 3 (2009), 154–165.

[8] Kurt D. Bollacker, Colin Evans, Praveen K. Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008*, Jason Tsong-Li Wang (Ed.). ACM, 1247–1250. https://doi.org/10.1145/1376616.1376746

[9] Xiaojun Chen, Shengbin Jia, and Yang Xiang. 2020. A review: Knowledge reasoning over knowledge graph. *Expert Syst. Appl.* 141 (2020). https://doi.org/10.1016/j.eswa.2019.112948

[10] Nurendra Choudhary, Nikhil Rao, Sumeet Kataria, Karthik Subbian, and Chandan K. Reddy. 2021. Probabilistic Entity Representation Model for Reasoning over Knowledge Graphs. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (Eds.). 23440–23451. https://proceedings.neurips.cc/paper/2021/hash/c4d2ce3f3ebb5393a77c33c0cd95dc93-Abstract.html

[11] Nurendra Choudhary, Nikhil Rao, Sumeet Kataria, Karthik Subbian, and Chandan K. Reddy. 2021. Self-Supervised Hyperboloid Representations from Logical Queries over Knowledge Graphs. In *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*, Jure Leskovec, Marko Grobelnik, Marc Najork, Jie Tang, and Leila Zia (Eds.). ACM / IW3C2, 1373–1384. https://doi.org/10.1145/3442381.3449974

[12] Hanyu Duan, Yi Yang, and Kar Yan Tam. 2021. Learning Numeracy: A Simple Yet Effective Number Embedding Approach Using Knowledge Graph. In *Findings of the Association for Computational Linguistics: EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 16-20 November, 2021*, Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (Eds.). Association for Computational Linguistics, 2597–2602. https://doi.org/10.18653/v1/2021.findings-emnlp.221

[13] Alberto García-Durán and Mathias Niepert. 2018. KBlrn: End-to-End Learning of Knowledge Base Representations with Latent, Relational, and Numerical Features. In *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence, UAI 2018, Monterey, California, USA, August 6-10, 2018*, Amir Globerson and Ricardo Silva (Eds.). AUAI Press, 372–381. http://auai.org/uai2018/proceedings/papers/149.pdf

[14] Jia Guo and Stanley Kok. 2021. BiQUE: Biquaternionic Embeddings of Knowledge Graphs. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (Eds.). Association for Computational Linguistics, 8338–8351. https://doi.org/10.18653/v1/2021.emnlp-main.657

[15] William L. Hamilton, Payal Bajaj, Marinka Zitnik, Dan Jurafsky, and Jure Leskovec. 2018. Embedding Logical Queries on Knowledge Graphs. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (Eds.). 2030–2041. https://proceedings.neurips.cc/paper/2018/hash/ef50c335cca9f340bde656363ebd02fd-Abstract.html

[16] Zijian Huang, Meng-Fen Chiang, and Wang-Chien Lee. 2022. LinE: Logical Query Reasoning over Hierarchical Knowledge Graphs. In *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*, Aidong Zhang and Huzefa Rangwala (Eds.). ACM, 615–625. https://doi.org/10.1145/3534678.3539338

[17] Bhushan Kotnis and Alberto García-Durán. 2019. Learning Numerical Attributes in Knowledge Bases. In *1st Conference on Automated Knowledge Base Construction, AKBC 2019, Amherst, MA, USA, May 20-22, 2019*. https://doi.org/10.24432/C5Z59Q

[18] Bhushan Kotnis, Carolin Lawrence, and Mathias Niepert. 2021. Answering Complex Queries in Knowledge Graphs with Bidirectional Sequence Encoders. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*. AAAI Press, 4968–4977. https://ojs.aaai.org/index.php/AAAI/article/view/16630

[19] Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2016. Knowledge Representation Learning with Entities, Attributes and Relations. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, Subbarao Kambhampati (Ed.). IJCAI/AAAI Press, 2866–2872. http://www.ijcai.org/Abstract/16/407

[20] Lihui Liu, Boxin Du, Heng Ji, ChengXiang Zhai, and Hanghang Tong. 2021. Neural-Answering Logical Queries on Knowledge Graphs. In *KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, August 14-18, 2021*, Feida Zhu, Beng Chin Ooi, and Chunyan Miao (Eds.). ACM, 1087–1097. https://doi.org/10.1145/3447548.3467375

[21] Xiao Liu, Shiyu Zhao, Kai Su, Yukuo Cen, Jiezhong Qiu, Mengdi Zhang, Wei Wu, Yuxiao Dong, and Jie Tang. 2022. Mask and Reason: Pre-Training Knowledge Graph Transformers for Complex Logical Queries. In *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*, Aidong Zhang and Huzefa Rangwala (Eds.). ACM, 1120–1130. https://doi.org/10.1145/3534678.3539472

[22] Chau Nguyen, Tim French, Wei Liu, and Michael Stewart. 2023. CylE: Cylinder Embeddings for Multi-hop Reasoning over Knowledge Graphs. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2023, Dubrovnik, Croatia, May 2-6, 2023*, Andreas Vlachos and Isabelle Augenstein (Eds.). Association for Computational Linguistics, 1728–1743. https://aclanthology.org/2023.eacl-main.127

[23] Hongyu Ren, Hanjun Dai, Bo Dai, Xinyun Chen, Denny Zhou, Jure Leskovec, and Dale Schuurmans. 2022. SMORE: Knowledge Graph Completion and Multi-hop Reasoning in Massive Knowledge Graphs. In *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*, Aidong Zhang and Huzefa Rangwala (Eds.). ACM, 1472–1482. https://doi.org/10.1145/3534678.3539405

[24] Hongyu Ren, Weihua Hu, and Jure Leskovec. 2020. Query2box: Reasoning over Knowledge Graphs in Vector Space Using Box Embeddings. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net. https://openreview.net/forum?id=BJgr4kSFDS

[25] Hongyu Ren and Jure Leskovec. 2020. 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, December 6-12, 2020, virtual*, Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (Eds.). https://proceedings.neurips.cc/paper/2020/hash/e43739bba7cdb577e9e3e4e42447f5a5-Abstract.html

[26] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*, Carey L. Williamson, Mary Ellen Zurko, Peter F. Patel-Schneider, and Prashant J. Shenoy (Eds.). ACM, 697–706. https://doi.org/10.1145/1242572.1242667

[27] Haitian Sun, Andrew O. Arnold, Tania Bedrax-Weiss, Fernando Pereira, and William W. Cohen. 2020. Faithful Embeddings for Knowledge Base Queries. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (Eds.). https://proceedings.neurips.cc/paper/2020/hash/fe74074593f21197b7b7be3c08678616-Abstract.html

[28] Dhanasekar Sundararaman, Shijing Si, Vivek Subramanian, Guoyin Wang, Devamanyu Hazarika, and Lawrence Carin. 2020. Methods for Numeracy-Preserving Word Embeddings. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Online, 4742–4753. https://doi.org/10.18653/v1/2020.emnlp-main.384

[29] Ilya O. Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, and Alexey Dosovitskiy. 2021. MLP-Mixer: An all-MLP Architecture for Vision. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (Eds.). 24261–24272. https://proceedings.neurips.cc/paper/2021/hash/cba0a4ee5ccd02fda0fe3f9a3e7b89fe-Abstract.html

[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 5998–6008. https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

[31] Po-Wei Wang, Daria Stepanova, Csaba Domokos, and J. Zico Kolter. 2020. Differentiable learning of numerical rules in knowledge graphs. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net. https://openreview.net/forum?id=rJleKgrKwS

[32] Zihao Wang, Weizhi Fei, Hang Yin, Yangqiu Song, Ginny Y. Wong, and Simon See. 2023. Wasserstein-Fisher-Rao Embedding: Logical Query Embeddings with Local Comparison and Global Transport. *CoRR* abs/2305.04034 (2023). https://doi.org/10.48550/arXiv.2305.04034 arXiv:2305.04034

[33] Zihao Wang, Yangqiu Song, Ginny Wong, and Simon See. 2023. Logical Message Passing Networks with One-hop Inference on Atomic Formulas. In *The Eleventh International Conference on Learning Representations*.

[34] Zihao Wang, Hang Yin, and Yangqiu Song. 2021. Benchmarking the Combinatorial Generalizability of Complex Query Answering on Knowledge Graphs. In *NeurIPS Datasets and Benchmarks Track*. https://openreview.net/forum?id=pX4x8f6Km5T

[35] Zhaohan Xi, Ren Pang, Changjiang Li, Tianyu Du, Shouling Ji, Fenglong Ma, and Ting Wang. 2022. Reasoning over Multi-view Knowledge Graphs. *CoRR* abs/2209.13702 (2022). https://doi.org/10.48550/arXiv.2209.13702 arXiv:2209.13702

[36] Zezhong Xu, Wen Zhang, Peng Ye, Hui Chen, and Huajun Chen. 2022. Neural-Symbolic Entangled Framework for Complex Query Answering. In *NeurIPS*. http://papers.nips.cc/paper_files/paper/2022/hash/0bcfb525c8f8f07ae10a93d0b2a40e00-Abstract-Conference.html

[37] Dong Yang, Peijun Qing, Yang Li, Haonan Lu, and Xiaodong Lin. 2022. GammaE: Gamma Embeddings for Logical Queries on Knowledge Graphs. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (Eds.). Association for Computational Linguistics, 745–760. https://aclanthology.org/2022.emnlp-main.47

[38] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabás Póczos, Ruslan Salakhutdinov, and Alexander J. Smola. 2017. Deep Sets. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 3391–3401. https://proceedings.neurips.cc/paper/2017/hash/f22e4747da1aa27e363d86d40ff442fe-Abstract.html

[39] Zhanqiu Zhang, Jie Wang, Jiajun Chen, Shuiwang Ji, and Feng Wu. 2021. 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, December 6-14, 2021, virtual*, Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (Eds.). 19172–19183. https://proceedings.neurips.cc/paper/2021/hash/a0160709701140704575d499c997b6ca-Abstract.html

[40] Zhaocheng Zhu, Mikhail Galkin, Zuobai Zhang, and Jian Tang. 2022. Neural-Symbolic Models for Logical Queries on Knowledge Graphs. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA (Proceedings of Machine Learning Research, Vol. 162)*, Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato (Eds.). PMLR, 27454–27478. https://proceedings.mlr.press/v162/zhu22c.html

---

Algorithm 1 Ground General Type

Require: G is a knowledge graph.
   function GROUNDGENERALTYPE(T, v)
        T is an arbitrary node of the computation graph.
        v is an arbitrary knowledge graph vertex
        if $T.operation = p$ **then**
            $u \leftarrow$ SAMPLE($\{u | (u, v)$ is an edge in $G\}$)
            $RelType \leftarrow$ type of (u, v) in G
            if $v.type = Entity$ & $u.type = Entity$ **then**
                $ProjectionType \leftarrow rp$
            else if $v.type = Entity$ & $u.type = Values$ **then**
                $ProjectionType \leftarrow rap$
            else if $v.type = Values$ & $u.type = Entity$ **then**
                $ProjectionType \leftarrow ap$
            else
                $ProjectionType \leftarrow np$
            end if
            $SubQuery \leftarrow$ GROUNDGENERALTYPE($T.child, u$)
            return $(ProjectionType, RelType, SubQuery)$
        else if $T.operation = i$ **then**
            $IntersectionResult = (i)$
            for $child \in T.Children$ **do**
                $SubQuery \leftarrow$ GROUNDGENERALTYPE($T.child, v$)
                $IntersectionResult$.PUSHBACK($child, v$)
            end for
            return $IntersectionResult$
        else if $T.operation = u$ **then**
            $UnionResult = (u)$
            for $child \in T.Children$ **do**
                if $UnionResult.length > 2$ **then**
                    $v \leftarrow$ SAMPLE(G)
                end if
                $SubQuery \leftarrow$ GROUNDGENERALTYPE($T.child, v$)
                $UnionResult$.PUSHBACK($child, v$)
            end for
            return $UnionResult$
        else if $T.operation = e$ **then**
            if $T.type = Entity$ **then**
                return $(e, T.value)$
            else
                return $(nv, T.value)$
            end if
         end if
   end function

A SAMPLING ALGORITHM

In this section, we introduce the algorithm used for sampling the numerical complex queries from a given knowledge graph. The detailed algorithm is described in Algorithm 1. For a given knowledge G and a general query type t, we start with a random node v to reversely find a query that has answer v with the corresponding structure t. Basically, this process is conducted in a recursion process. In this recursion, we first look at the last operation in this query. If the operation is projection, we randomly select one of its predecessors u that holds the corresponding relation to v as the answer of its sub-query. Then we call the recursion on node u and the sub-query type of t again. Similarly, for intersection and union, we will apply recursion on their sub-queries on the same node v. Differently for union, only one of the sub-queries is keeping u, while the others will use a randomly selected node, due to the nature of the union operation. The recursion will stop when there are no more operations for the current node. The specific query types will be determined according to the relation types during the process of sampling projection operation.