

Query2Particles: Knowledge Graph Reasoning with Particle Embeddings

Jiaxin Bai¹, Zihao Wang¹, Hongming Zhang², and Yangqiu Song¹

¹CSE, HKUST

²Tencent AI Lab

{jbai, zwanggc, yqsong}@cse.ust.hk, hongmzhang@tencent.com

Abstract

Answering complex logical queries on incomplete knowledge graphs (KGs) with missing edges is a fundamental and important task for knowledge graph reasoning. The query embedding method is proposed to answer these queries by jointly encoding queries and entities to the same embedding space. Then the answer entities are selected according to the similarities between the entity embeddings and the query embedding. As the answers to a complex query are obtained from a combination of logical operations over sub-queries, the embeddings of the answer entities may not always follow a uni-modal distribution in the embedding space. Thus, it is challenging to simultaneously retrieve a set of diverse answers from the embedding space using a single and concentrated query representation such as a vector or a hyper-rectangle. To better cope with queries with diversified answers, we propose Query2Particles (Q2P), a complex KG query answering method. Q2P encodes each query into multiple vectors, named particle embeddings. By doing so, the candidate answers can be retrieved from different areas over the embedding space using the maximal similarities between the entity embeddings and any of the particle embeddings. Meanwhile, the corresponding neural logic operations are defined to support its reasoning over arbitrary first-order logic queries. The experiments show that Query2Particles achieves state-of-the-art performance on the complex query answering tasks on FB15k, FB15K-237, and NELL knowledge graphs.

1 Introduction

Reasoning over a factual knowledge graph (KG) is the process of deriving new knowledge or conclusions from the existing data in the knowledge graph (Chen et al., 2020). A recently developed sub-task of knowledge graph reasoning is complex query answering, which aims to answer complex queries over large knowledge graphs (Hamilton et al., 2018;

Logical Knowledge Graph Queries	Interpretations
$q_1 = V_7 . \exists V : \text{Win}(\text{TuringAward}, V) \wedge \neg \text{Citizen}(\text{Canada}, V) \wedge \text{Graduate}(V, V_7)$	Find where the non-Canadian Turing award laureates graduated from.
$q_2 = V_7 . \exists V : \text{Interacts}(V_7, V) \wedge (\text{Assoc}(V, T_1) \vee \text{Assoc}(V, T_2) \vee \text{Assoc}(V, T_3))$	Find the substances that interact with the proteins associated with diseases T1, T2, or T3.
$q_3 = V_7 : \text{MovesTo}(V_7, \text{America}) \wedge \neg \text{IsResident}(V_7, \text{America}) \wedge \text{Wins}(V_7, \text{NobelPrize})$	Find entities, who are not American, were the Nobel Prize winners and eventually moved to the US.

Figure 1: The example logical knowledge graph queries and their interpretations in natural language.

Ren et al., 2020; Ren and Leskovec, 2020). Compared to KG completion tasks (Liu et al., 2016; West et al., 2014), complex query answering requires reasoning over multi-hop relations and logical operations. As shown in Figure 1, complex KG queries are defined in predicate logic forms with relation projection operations, existential quantifiers \exists , logical conjunctions \wedge , disjunctions \vee , and negation \neg . Answering these queries is challenging because real-world knowledge graphs (KG), such as Freebase (Bollacker et al., 2008), NELL (Carlson et al., 2010), and DBPedia (Bizer et al., 2009), are incomplete. Consequently, sub-graph matching methods cannot be used to find the answers.

To address the challenge raised from the incompleteness of knowledge graphs, the query embedding methods are proposed (Hamilton et al., 2018; Ren et al., 2020; Ren and Leskovec, 2020; Sun et al., 2020). In this line of research, the queries and entities are jointly encoded into the same embedding space, and the answers are retrieved based on similarities between the query embedding and entity embeddings. In general, there are two steps in encoding a query to the vector space. First, a query is parsed into a computational graph with a directed acyclic graph (DAG) structure, as shown in Figure 2 (A). Then, the query representation is iteratively computed following the neural logic operations and relation projections in the DAG.

Although the query embedding methods are robust for dealing with the incompleteness of KGs,

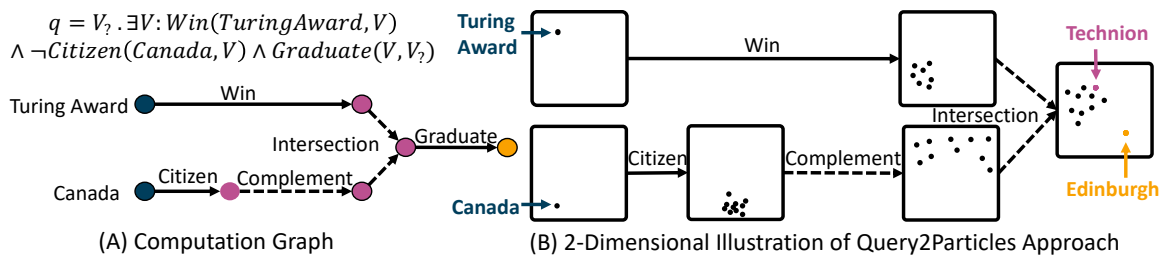


Figure 2: An example of answering a knowledge graph query by using the Query2Particles method. (A) The computational graph corresponds to the query “where did the non-Canadian Turing award laureates graduate from.” (B) The Query2Particles encodes each query into a set of vectors, called particle embeddings. The logical operations iteratively compute particle embeddings following the computational graph. The answers are determined by using the maximum similarities between the entity embeddings and any one of the resulting particle embeddings.

the embedding structure used for encoding the queries can be improved. Because of the multi-hop and compositional nature of complex KG queries, a single query may contain multiple sufficiently diverse answers. Thus, the ideal query embedding may follow a multi-modal distribution¹ in the embedding space. For example, the answers to the query, “Find entities, who are not American, were the Nobel Prize winners and eventually moved to the US,” involve intermediate entities with different attributes, such as gender, nationality, research fields, etc. It is difficult to use a single embedding vector to find all final answer embeddings. Box embedding (Ren et al., 2020) partially solved this problem, but for complicated attributes, a single box may be too coarse, and intermediate entities are distributed far away from each other, so they are more like several disjoint clusters rather than a single big region in the embedding space. So for the query embedding methods, the capability to simultaneously encode a set of answers from different areas is necessary.

To better address the diversity of answers, we propose Query2Particles, a new query embedding method for complex query answering. In this approach, each query is encoded into a set of vectors in the embedding space, called particle embeddings. The particle embeddings of a query are iteratively computed by following the computational graph parsed from the query. Then the answers to this query are determined by using the maximum similarities between the entity embeddings and any one of the resulting particle embeddings. Experimental results show that Query2Particle achieves state-of-the-art performance on complex query answering over three standard knowledge graphs: FB15K, FB15k-237, and NELL. Meanwhile, the inference

¹A multi-modal distribution is a distribution with two or more distinct peaks in the probability density function.

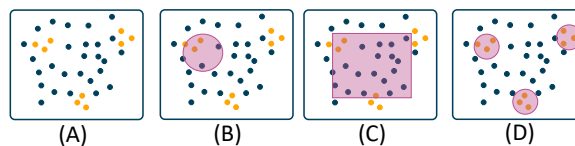


Figure 3: In the example embedding space, the yellow dots are the answer entities, and the blue dots are the non-answer entities. The purple areas in (B), (C), and (D) demonstrate the neighborhoods of the vector embedding, the box embedding, and the particle embeddings respectively. In this case, the particle embeddings are more suitable for finding the answers clustered in different areas in the embedding space.

speed of Query2Particles is comparable to other query embedding methods and is higher than query decomposition methods on multi-hop queries. Further analysis indicates that the optimal numbers of particles for different query types depend on the structures of the queries. Our experimental code is released on github².

2 Related Work

Other query embedding approaches are closely related to our work. These query embedding methods leverage different structures to encode logical KG queries, and they can answer various scopes of logical queries. The GQE method proposed by Hamilton et al. (2018) can answer the conjunctive queries by representing queries as vector representations. Ren et al. (2020) used hyper-rectangles to encode and answer existential positive first-order (EPFO) queries. At the same time, Sun et al. (2020) proposed to improve the faithfulness of the query embedding method by using centroid-sketch representations on EPFO queries. The conjunctive queries and EPFO queries are both subsets of first-order logic (FOL) queries. The Beta Embedding (Ren and Leskovec, 2020) is the first query embed-

²<https://github.com/HKUST-KnowComp/query2particles>

ding method that supports a full set of operations in FOL by encoding entities and queries into probabilistic Beta distributions. In a contemporaneous work, Zhang et al. (2021) uses cone embeddings to encode the FOL queries. As shown in Figure 3, compared to these query embedding approaches, the Q2P method can encode the FOL queries to address the diversity of answers. Note that, Ren et al. (2020) proposed to use the disjunctive normal form (DNF) to address the answer diversities resulting from the union operations. This partly solve the problem, but the diversity of the answers is not solely caused by the union operation, but a joint effort of multi-hop projections, intersection, and complement. As a result, using particle embeddings is a more general solution.

Query decomposition (Arakelyan et al., 2020) is another approach to answering complex knowledge graph queries. In this line of research, a complex query is decomposed into atomic queries, and the probabilities of atomic queries are modeled by link predictors. In the inference process, continuous optimization and beam search are used for finding the answers. Meanwhile, the rule and path-based methods (Guo et al., 2016; Xiong et al., 2017; Lin et al., 2018; Guo et al., 2018; Chen et al., 2019) use pre-defined or learned rules to do multi-hop KG reasoning. These methods explicitly model the intermediate entities in the query. Instead, the query embedding methods directly embed the complex query and retrieve the answers without explicit modeling intermediate entities. So the query embedding methods are more scalable to large knowledge graphs and complex query structures.

Neural link predictors (Wang et al., 2014; Trouillon et al., 2016; Dettmers et al., 2018; Sun et al., 2018) are also related to this work. The link predictors learn the distributed representations of entities and relations in embedding space and use different neural structures to classify whether there exists a certain relation between two entities. The link predictors can be used for one-hop queries, but cannot be directly used for answering complex queries.

3 Preliminaries

In this section, we formally define the complex logical knowledge graph queries and the corresponding computational graphs. The knowledge graph reasoning is conducted on a multi-relational knowledge graph $\mathcal{G} = (\mathcal{V}, \mathcal{R})$, where each vertex $v \in \mathcal{V}$ represents an entity, and each relation $r \in \mathcal{R}$ 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 entities u and v if and only if $r(u, v) = 1$.

3.1 First-Order Logic Query

The complex knowledge graph query is defined in first-order logic form with logical operators such as existential quantifiers \exists , conjunctions \wedge , disjunctions \vee , and negations \neg . In a first-order logic query, there is a set of anchor entities $V_a \in \mathcal{V}$, existential quantified variables $V_1, V_2, \dots, V_k \in \mathcal{V}$, and a unique target variable $V_? \in \mathcal{V}$. The query intends to find the answers $V_? \in \mathcal{V}$, such that there simultaneously exist $V_1, V_2, \dots, V_k \in \mathcal{V}$ satisfying the logical expression in the query. For each FOL query, it can be converted to a disjunctive normal form, where the query is expressed as a disjunction of several conjunctive expressions:

$$q[V_?] = V_?.\exists V_1, \dots, V_k : c_1 \vee c_2 \vee \dots \vee c_n, \quad (1)$$

$$c_i = e_{i1} \wedge e_{i2} \wedge \dots \wedge e_{im}. \quad (2)$$

Each c_i represents a conjunctive expression of several literals e_{ij} , and each e_{ij} is an atomic or the negation of an atomic expression expressed by any of the following expressions: $e_{ij} = r(v_a, V)$, $e_{ij} = \neg r(v_a, V)$, $e_{ij} = r(V, V')$, or $e_{ij} = \neg r(V, V')$. Here $v_a \in V_a$ is one of the anchor entities, and $V, V' \in \{V_1, V_2, \dots, V_k, V_?\}$ are distinct variables satisfying $V \neq V'$.

3.2 Computational Graph and Operations

As shown in Figure 2 (A), for a first-order query, there is a corresponding computational graph. In the computational graph, each node corresponds to an intermediate query embedding, and each edge corresponds to a neural logic operation to be defined in the following section. Both the input and output of these operations are query embeddings. These operations are used for implicitly modeling different set operations over the intermediate answer sets. These set operations include relational projection, intersection, union, and complement: (1) *Relational Projection*: Given a set of entities A and a relation $r \in \mathcal{R}$, the relational projection will return all entities having relation r with at least one of entity $e \in A$. Namely, $P_r(A) = \{v \in \mathcal{V} | \exists v' \in A, r(v', v) = 1\}$; (2) *Intersection*: Given sets of entities $A_1, \dots, A_n \subset \mathcal{V}$, this operation computes their intersection $\cap_{i=1}^n A_i$; (3) *Union*: Given several sets of entities $A_1, \dots, A_n \subset \mathcal{V}$, the union operation calculates their union $\cup_{i=1}^n A_i$; (4) *Complement*:

Given a set of entities A , the complement operation calculates its absolute complement $\mathcal{V} - A$.

4 Query2Particles

In this section, we first introduce the particle embeddings structure and the neural logic operations, and then we present the learning of the model.

4.1 Particles Representations of Queries

In Query2Particles, each query is represented as a set of vectors, called particles. For simplicity, a set of particles $\{p^{(k)}\}_{k=1}^K$ are represented as a matrix P . All the operations discussed in the following sections are invariant to the permutations of the particle vectors in the matrix. Formally, the particle embeddings $P \in R^{d \times K}$ are

$$P = [p^{(1)}, p^{(2)}, \dots, p^{(K)}], \quad (3)$$

where each vector $p^{(k)} \in R^d$ is a particle vector. As shown in Figure 2, the computations along the computation graph start with the anchor entities, such as ‘‘Turing Award’’. Suppose the entity embedding of an anchor entity v is denoted as $e_v \in R^d$. Then, the initial particle embeddings are computed as the sum of e_v and a learnable offset matrix $M \in R^{d \times K}$,

$$P_0 = e_v + M. \quad (4)$$

Here and in the following sections, the addition between the matrix M and the vector e_v is defined as the broadcasted element-wise addition.

4.2 Logical Operations

In this sub-section, we define and parameterize four types of neural logic operations: projection, intersection, negation, and union.

4.2.1 Projection

Suppose the $e_l \in R^d$ is the embedding vector of the relation l . The relation projection f_P is expressed as $P_{i+1} = f_P(P_i, e_l)$, where the P_i and P_{i+1} are input and output particle embeddings. Instead of directly adding the same relation embedding e_l to all particles in P_i to model the relation projection following (Bordes et al., 2013), we incorporate multiple neuralized gates (Chung et al., 2014) to individually adjust the relation transition for each particle in P_i , which are expressed as follows:

$$Z = \sigma(W_z^P e_l + U_z P_i + b_z), \quad (5)$$

$$R = \sigma(W_r^P e_l + U_r P_i + b_r), \quad (6)$$

$$T = \phi(W_h^P e_l + U_h (R \odot P_i) + b_h), \quad (7)$$

$$A_i = (1 - Z) \odot P_i + Z \odot T. \quad (8)$$

Here, σ and ϕ are the sigmoid and hyperbolic tangent functions, and \odot is the Hadamard product. Also, $W_z^P, W_r^P, W_h^P, U_z, U_r, U_h$ are parameter matrices. T is interpreted as the relation transitions for each of the particles given the relation embedding e_l , and Z and R are the update gate and the reset gate used for customizing the relation transitions for each particle. Meanwhile, the relation projection result for each particle should also depend on the positions of other input particles. To allow information exchange among different particles, a scaled dot-product self-attention (Vaswani et al., 2017) module is also incorporated,

$$P_{i+1} = \text{Attn}(W_q^P A_i^T, W_k^P A_i^T, W_v^P A_i^T)^T. \quad (9)$$

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

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

Here, the Q, K , and V represent the input Query, Key, and Value for this attention layer.

4.2.2 Intersection

The intersection operation f_I is defined on multiple sets of particle embeddings $\{P_i^{(n)}\}_{n=1}^N$. It outputs a single set of particle embeddings $P_{i+1} = f_I(\{P_i^{(n)}\}_{n=1}^N)$. The particles from the $\{P_i^{(n)}\}_{n=1}^N$ are first merged into a new matrix $P_i = [P_i^{(1)}, P_i^{(2)}, \dots, P_i^{(N)}] \in R^{d \times NK}$, and this matrix P_i serves as the input of the intersection operation. The operation updates the position of each particle according to the positions of other input particles in $\{P_i^{(n)}\}_{n=1}^N$. This process is modeled by the scaled dot-product self-attention followed by a multi-layer perceptron (MLP) layer,

$$A_i = \text{Attn}(W_q^I P_i^T, W_k^I P_i^T, W_v^I P_i^T)^T, \quad (11)$$

$$P_{i+1} = \text{MLP}(A_i). \quad (12)$$

Here $W_q^I, W_k^I, W_v^I \in R^{d \times d}$ are parameters for the self-attention layer. The MLP here denotes a multi-layer perceptron layer with ReLU activation, and the parameters in the MLP layers in different operations are not shared. To keep the number of particles unchanged, we uniformly sub-sample K particles out of the NK particles in P_{i+1} as the final output of the intersection operation.

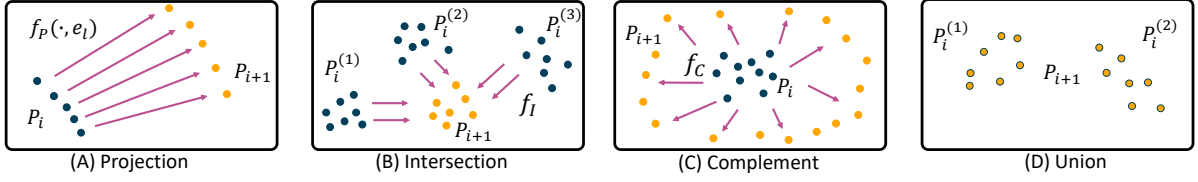


Figure 4: The P_i in blue and the P_{i+1} in yellow are the input and output particles respectively. (A) The embeddings P_i are projected to P_{i+1} by the relation l . (B) The resulting embeddings P_{i+1} are computed from three sets of particles P_i by the intersection f_I . (C) The output P_{i+1} are computed from the input P_i by the complement f_C . (D) The P_{i+1} are directly taken from the all input particles in P_i without any additional parameterization.

4.2.3 Complement

The input of the complement operation is a single set of particle embeddings P_i , and the operation f_C is formulated as $P_{i+1} = f_C(P_i)$. The complement operation updates the position of each particle based on the distributions of other input particles. The operation is then modeled by scaled dot-product attention followed by an MLP layer, and this can be formulated by

$$A_i = \text{Attn}(W_q^C P_i^T, W_k^C P_i^T, W_v^C P_i^T)^T, \quad (13)$$

$$P_{i+1} = \text{MLP}(A_i). \quad (14)$$

Here, the $P_{i+1} \in R^{d \times K}$ are the resulting particle embeddings for the complement operation, and the values in $W_q^C, W_k^C, W_v^C \in R^{d \times d}$ are parameters. Intuitively speaking, the proposed structure can model the complement operation by encouraging the particles to move towards the areas that are not occupied by any of the input particles.

4.2.4 Union

The union operation is directly modeled by all the input particles without extra parameterization. In detail, the particles from the input particle embeddings are directly merged into a new set of particles,

$$f_U(\{P_i^{(n)}\}_{n=1}^N) = [P_i^{(1)}, \dots, P_i^{(N)}]. \quad (15)$$

4.3 Scoring

After the particle embeddings $P_T \in R^{d \times K}$ for the target variable of the query q are computed, the scoring function ϕ between the particle embeddings P_T and each entity embedding e_v is used for calculating the maximal similarities between each particle vectors in $\{p_T^{(k)}\}_{k=1}^K$ and entity embedding vector. Here, the inner product is used to compute the similarity scores between vectors, and the overall scoring function is expressed by

$$\phi(P_T, e_v) = \max_{k \in \{1, 2, \dots, K\}} \langle p_T^{(k)}, e_v \rangle. \quad (16)$$

4.4 Learning Query2Particles

To train the Query2Particles model, we compute the normalized probability of the entity v being the correct answer of query q by using the softmax function on all similarity scores,

$$p(v, q) = \frac{\phi(P_T, e_v)}{\sum_{v' \in V} \phi(P_T, e_{v'})}. \quad (17)$$

Then we construct the cross-entropy loss from the given probabilities to maximize the log probabilities of all correct query-answer pairs:

$$L = -\frac{1}{N} \sum_i \log p(v^{(i)}, q^{(i)}). \quad (18)$$

The $(v^{(i)}, q^{(i)})$ denotes is one of the positive query-answer pairs, and in total there are N such pairs.

5 Experiments

The experiments in this section demonstrate the effectiveness and efficiency of Query2Particles.

5.1 Experimental Setup

The Query2Particles method is evaluated on three commonly used knowledge graphs, FB15K (Bordes et al., 2013), FB15K-237 (Toutanova and Chen, 2015), and NELL995 (Carlson et al., 2010) with the standard training, validation, and testing edges separations. For each of these graphs, the corresponding training graph G_{train} , validation graph G_{valid} , and testing graph G_{test} are created from training edges, training + validation edges, and training + validation + testing edges respectively.

There are two sets of complex logical queries sampled from these knowledge graphs, and the existing methods evaluate their performance on either of them. Specifically, Ren et al. (2020) sample nine different types of existential positive first-order (EPFO) queries. For these queries, five types of them (1p, 2p, 3p, 2i, 3i) are used for training and evaluation in a supervised setting. For the rest of four types of queries (2u, up, ip, pi), they

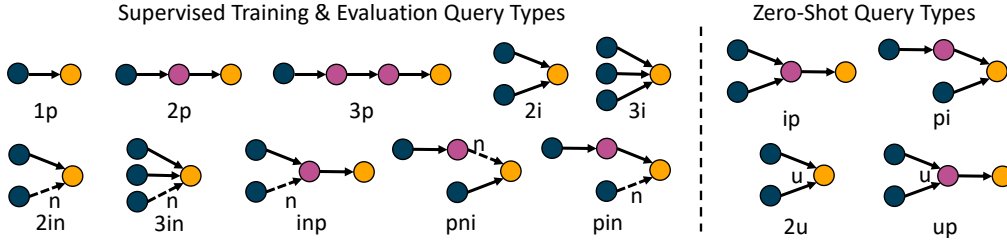


Figure 5: The query structures used for training and evaluation. For brevity, the p , i , n , and u represent the projection, intersection, negation, and union operations respectively. The query types on the left are trained and evaluated under supervised settings. There are not training queries for the four types of queries on the right, and they are directly evaluated at the test time to measure the generalization capability of the models to unseen query types.

MODEL	FB15K										FB15K-237	NELL
	1P	2P	3P	2I	3I	PI	IP	2U	UP	AVG	AVG	AVG
BETA E	65.1	25.7	24.7	55.8	66.5	43.9	28.1	40.1	25.4	41.6	20.9	24.6
Q2B	68.0	21.0	14.2	55.1	66.5	39.4	26.1	35.1	16.7	38.0	20.1	22.9
GQE	54.6	15.3	10.8	39.7	51.4	27.6	19.1	22.1	11.6	28.0	16.3	18.6
Q2P (OURS)	82.6	30.8	25.5	65.1	74.7	49.5	34.9	32.1	26.2	46.8	21.9	25.5

Table 1: The MRR results for existential positive first-order (EPFO) queries used by Ren and Leskovec (2020). The full results are shown in the supplementary materials.

do not appear in the training set and are directly evaluated in a zero-shot way. In another work, Ren and Leskovec (2020) refine these queries by raising the difficulties of the existing nine types of queries. They also include five types of complement queries ($2in$, $3in$, inp , pni , pin) for general first-order logic (FOL) queries. These complement queries are also trained and evaluated in the supervised setting, but their training samples are fewer than other types. More details about the knowledge graphs and sampled queries are shown in the appendix. To demonstrate the performance of Query2Particles, it is evaluated on both sets of queries. Note that, the query-answer pairs used for training are only from the training graph G_{train} . For validation and testing, only the hard answers from validation graph G_{valid} and testing graph G_{test} are evaluated.

5.2 Baselines

The Query2Particles model is compared with the following baselines in the following sections.

Graph Query Embedding (GQE) answers conjunctive logic queries by encoding the logical queries into vectors (Hamilton et al., 2018).

Query2Box (Q2B) answers existential positive first-order logic queries by encoding them into boxes in the embedding space (Ren et al., 2020).

Beta Embedding (BetaE) answers first-order logic queries by modeling them as Beta Distributions (Ren and Leskovec, 2020). This is the current state-of-the-art model on first-order logic queries.

The reported mean reciprocal rank (MRR) scores of these baselines are used by the BetaE paper (Ren and Leskovec, 2020), and Query2Particles (Q2P) is evaluated following with the same metrics under the filtered setting, in which the rankings of answers are computed excluding all other correct answers. Meanwhile, the Q2P method is also compared with other methods on EPFO queries with the queries used by Ren et al. (2020).

Continuous Query Decomposition (CQD) decomposes the complex queries to multiple atomic queries that can be solved by link predictors (Arakelyan et al., 2020).

Embedding Query Language (EmQL) improves the faithfulness in the reasoning process by encoding EPFO queries into centroid-sketch representations (Sun et al., 2020).

The reported Hit@3 results of these two baselines are used by Arakelyan et al. (2020); Sun et al. (2020). Our model is evaluated on FB15K, FB15K-237, and NELL in the same setting.

5.3 Implementation Details

The Query2Particles model is trained on the queries in an end-to-end manner. To fairly compare with previous methods, we set the same size of embedding vectors as four hundred. We use the validation queries to tune hyperparameters for our model by using grid search. In the grid search, we consider the batch size from $\{1024, 2048, 4096, 8192\}$, dropout rate from $\{0.1, 0.2, 0.3\}$, learning rate

DATASET	MODEL	2IN		3IN		INP		PIN		PNI		AVG	
		MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10
FB15k	BETA E	14.3	30.8	14.7	31.9	11.5	23.4	6.5	14.3	12.4	26.3	11.8	25.3
	Q2P (OURS)	21.9	41.3	20.8	40.2	12.5	24.2	8.9	18.8	17.1	33.6	16.4	31.6
FB15k-237	BETA E	5.1	11.3	7.9	17.3	7.4	16.0	3.6	8.1	3.4	7.0	5.4	11.9
	Q2P (OURS)	4.4	10.1	9.7	20.7	7.5	16.7	4.6	9.9	3.8	7.2	6.0	12.9
NELL	BETA E	5.1	11.6	7.8	18.2	10.0	20.8	3.1	6.9	3.5	7.2	5.9	12.9
	Q2P (OURS)	5.1	12.1	7.4	18.2	10.2	21.4	3.3	7.0	3.4	7.6	6.0	13.3
AVERAGE	BETA E	8.2	17.9	10.1	22.5	9.6	20.1	4.4	9.8	6.4	13.5	7.8	16.7
	Q2P (OURS)	10.5	21.2	12.6	26.4	10.1	20.8	5.6	11.9	8.1	16.1	9.4	19.3

Table 2: The MRR and Hit@10 results for complement queries used by Ren and Leskovec (2020). Beta Embedding is the only baseline model that can answer queries with the complement operation.

MODEL	FB15K										FB15K-237	NELL
	1P	2P	3P	2I	3I	IP	PI	2U	UP	AVG	AVG	AVG
EMQL	42.4	50.2	45.9	63.7	70.0	<u>60.7</u>	61.4	9.0	<u>42.6</u>	49.5	<u>35.8</u>	46.8
– SKETCH	50.6	46.7	41.6	61.8	67.3	54.2	53.5	21.6	40.0	48.6	35.5	46.8
CQD-BEAM	91.8	77.9	<u>57.7</u>	<u>79.6</u>	<u>83.7</u>	37.5	<u>65.8</u>	83.9	34.5	<u>68.0</u>	29.0	37.6
CQD-CO	91.8	45.4	19.1	<u>79.6</u>	<u>83.7</u>	33.6	51.3	81.6	31.9	57.6	27.2	36.8
Q2P (OURS)	<u>90.2</u>	<u>74.6</u>	73.4	86.0	89.6	63.7	77.6	<u>83.4</u>	52.7	76.8	43.0	52.2

Table 3: The Hit@3 results for existential positive first-order queries originally used by Ren et al. (2020) and the comparisons are made against the state-of-the-art baselines including EmQL and CQD methods. The best results are marked in **bold** and the second-best ones are marked with underlines. The full results are in the appendix.

from $\{10^{-4}, 3 * 10^{-4}, 10^{-3}\}$, and label smoothing from $\{0.3, 0.5, 0.7\}$. The final hyperparameters are shown in the supplementary materials. Our experiments are conducted on Titan Xp with PyTorch 1.8, and they are repeated three times.

5.4 Comparison with Baselines

First, we compare Query2Particles (Q2P) with GQE, Q2B, and BetaE on the first-order logic queries used by Ren and Leskovec (2020). The results on all fourteen types of queries are reported in Table 1 and Table 2. To fairly compare with the baseline methods, we keep the same number of parameters used in each type of query embedding.

As shown in Tables 1 and 2, the Q2P model can achieve more accurate results than GQE, Q2B, and BetaE on all types of queries except 2u. As we keep the number of query embedding parameters the same, it indicates that the structure of particle embeddings is more suitable for encoding complex queries than boxes or Beta distributions.

Though it is slightly less accurate on the 2u queries, Q2P is more efficient in encoding the queries that include union operations. This is because Q2P is the first embedding method that directly models the union operation. To avoid direct modeling of the union operation, all previous embedding methods pre-process the queries by con-

verting them to DNF forms. However, the DNF forms can be exponentially larger than the original queries, and the conversion also takes exponential time. Meanwhile, BetaE proposes to use De Morgan’s law to replace one union operation with one intersection and three complements, but this substitution still largely increases the query complexity. Instead, Q2P directly models the union operation without any pre-processing or additional parameterization, while achieving the state-of-the-art performance on up, which is more complicated and involving the Union operation.

We also compare our model with EmQL and CQD methods on the queries used by Ren et al. (2020). On average, our model has better Hit@3 scores on all datasets³. Compared to the CQD method, the Q2P method is better at answering multi-hop queries. encodes the complex queries into centroid-sketch representations, which cannot compactly encode sufficiently diverse answers. The Q2P method specifically addresses the diversity of answers, so it has higher empirical performance. CQD performs better on shorter queries like 1p, 2p, and 2u, because it can use the state-of-the-art link predictors. Also, as shown in Figure 6, the Q2P method demonstrates a faster inference

³In this paper, we only focus on the inductive setting, so we skip the comparison with EmQL under the entailment setting, in which the test graph is used for both training and testing.

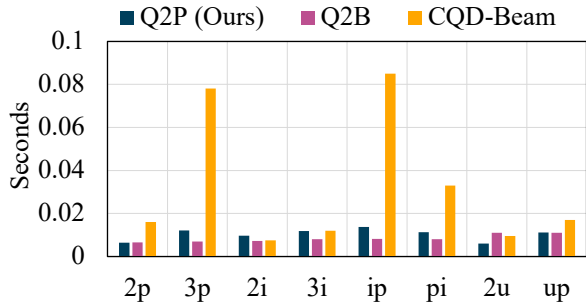


Figure 6: The inference time of different types of queries on FB15k, which has the largest number of edges among three graphs.

speed than the CQD method on multi-hop queries, because CQD uses inference time optimization, which is either a continuous optimization or a beam search. The inference time optimization simplifies the learning of CQD but also slows down the inference efficiency on large graphs.

5.5 The Improvement of Q2P-KP

Experiments show that the performance of the diversified queries can be largely improved by using more particles. To demonstrate the effects, we conduct additional evaluations on the most diversified 10% queries for each query type, as shown in the DIVR columns in Table 4. In doing so, we use the number of answers to measure the diversity of each query. In the same table, we also present the original results in the FULL columns as a comparison.

We can observe that there is a significant performance gap between the FULL and DIVR results, which demonstrates that the diversified queries are harder to answer. Meanwhile, it is also observed that comparing to Q2P-1P, Q2P-KP ($K > 1$) significantly improves the MRR of DIVR queries by 7.8 points. From this perspective, the improvement of Q2P-KP ($K > 1$) over Q2P-1P is significant on those challenging queries.

5.6 Further Ablation Study for Q2P-1P

To better explain the superior performance of Q2P-1P over the baseline models, we conduct further ablations studies in Table 5.

First, we remove all the self-attention layers `Attn`. Then the performance of intersection operations largely decreased. This can be explained that the self-attention structure is important for aggregating the information from multiple sub-queries.

Then, we remove all the neural network structures, including all MLP and `Attn` from all operations, and replace them with the operations defined in the GQE model (Hamilton et al., 2018). Then

MODELS	1P		2I		2U		2IN		AVERAGE	
	FULL	DIVR	FULL	DIVR	FULL	DIVR	FULL	DIVR	FULL	DIVR
Q2P-1P	81.8	44.8	63.4	28.8	33.4	11.3	18.9	15.0	49.4	25.0
Q2P-2P	82.6	49.4	65.1	35.5	32.1	13.3	21.9	20.7	50.4	29.7
Q2P-3P	82.9	53.0	64.4	37.7	33.6	18.6	21.8	21.6	50.6	32.8

Table 4: The FULL columns demonstrate the averaged MRR results, and the DIVR columns demonstrate the averaged MRR on the top ten-percent diversified queries.

MODELS	1P	2P	2I	2U	2IN
Q2P-KP	83.4	31.5	66.0	38.9	22.3
Q2P-1P	81.8	30.7	63.4	33.4	18.9
- SELF ATTENTION	78.5	28.5	30.9	30.3	15.2
- ALL NNS + GQE Ops	56.7	16.1	39.2	20.1	-
Q2P	68.0	21.0	55.1	35.1	-
GQE	54.6	15.3	39.7	22.1	-

Table 5: The ablation study on the Q2P neural networks. The Q2P-KP result shows the highest result when K is ranged from 2 to 6.

the performance of Q2P is also reduced. This indicates that the neural structures in the particle operations are also important to the overall improvement. Thus, we infer that the baseline model underfit the complex queries in the training set, and the performance can be improved by introducing more parameters and non-linearity. This conclusion is also aligned with Sun et al. (2020), in which they found the baselines cannot faithfully answer the queries that are observed in the training time.

However, solely using more complex structures cannot address the problem raised from the diversity of the answers. As shown in Table 4, on the top of Q2P-1P, Q2P-KP ($K > 1$) can still largely improve the performance on the diversified queries.

6 Conclusion

In this paper, we proposed Query2Particles, a query embedding method for answering complex logical knowledge graph queries over incomplete knowledge graphs. The Query2Particle method supports a full set of FOL operations. Specifically, the Q2P method is the first query embedding method that can directly model the union operation without any preprocessing. Experimental results show that the Q2P method achieves state-of-the-art performances on answering FOL queries on three different knowledge graphs while using comparable inference time as the previous methods.

7 Ethical Impacts

This paper introduces a knowledge graph reasoning method, and the experiments are on several publicly available benchmark datasets. As a result, there is

no data privacy concern. Meanwhile, this paper does not involve human annotations, and there is no related ethical concerns.

8 Acknowledgements

The authors of this paper were supported by the NSFC Fund (U20B2053) from the NSFC of China, the RIF (R6020-19 and R6021-20) and the GRF (16211520) from RGC of Hong Kong, the MHKJFS (MHP/001/19) from ITC of Hong Kong and the National Key R&D Program of China (2019YFE0198200) with special thanks to Hong Kong Mediation and Arbitration Centre (HKMAAC) and California University, School of Business Law & Technology (CUSBLT), and the Jiangsu Province Science and Technology Collaboration Fund (BZ2021065).

References

- Erik Arakelyan, Daniel Daza, Pasquale Minervini, and Michael Cochez. 2020. Complex query answering with neural link predictors. In *International Conference on Learning Representations*.
- 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):154–165.
- Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of ACM SIGMOD 2008*, pages 1247–1250.
- Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26.
- Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka, and Tom M Mitchell. 2010. Toward an architecture for never-ending language learning. In *Twenty-Fourth AAAI conference on artificial intelligence*.
- Xiaojun Chen, Shengbin Jia, and Yang Xiang. 2020. A review: Knowledge reasoning over knowledge graph. *Expert Systems with Applications*, 141:112948.
- Xuelu Chen, Muhao Chen, Weijia Shi, Yizhou Sun, and Carlo Zaniolo. 2019. Embedding uncertain knowledge graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3363–3370.
- Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning, December 2014*.
- Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.
- Shu Guo, Quan Wang, Lihong Wang, Bin Wang, and Li Guo. 2016. Jointly embedding knowledge graphs and logical rules. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 192–202.
- Shu Guo, Quan Wang, Lihong Wang, Bin Wang, and Li Guo. 2018. Knowledge graph embedding with iterative guidance from soft rules. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.
- Will Hamilton, Payal Bajaj, Marinka Zitnik, Dan Jurafsky, and Jure Leskovec. 2018. Embedding logical queries on knowledge graphs. *Advances in neural information processing systems*, 31.
- Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2018. Multi-hop knowledge graph reasoning with reward shaping. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3243–3253.
- Zhiyuan Liu, Maosong Sun, Yankai Lin, and Ruobing Xie. 2016. Knowledge representation learning: a review. *Journal of Computer Research and Development*, 53(2):247.
- Hongyu Ren, Weihua Hu, and Jure Leskovec. 2020. Query2box: Reasoning over knowledge graphs in vector space using box embeddings. In *International Conference on Learning Representations*.
- Hongyu Ren and Jure Leskovec. 2020. Beta embeddings for multi-hop logical reasoning in knowledge graphs. *Advances in Neural Information Processing Systems*, 33:19716–19726.
- Haitian Sun, Andrew Arnold, Tania Bedrax Weiss, Fernando Pereira, and William W Cohen. 2020. Faithful embeddings for knowledge base queries. *Advances in neural information processing systems*, 33.
- Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2018. Rotate: Knowledge graph embedding by relational rotation in complex space. In *International Conference on Learning Representations*.
- Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd workshop on continuous vector space models and their compositionality*, pages 57–66.

- Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *International conference on machine learning*, pages 2071–2080. PMLR.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.
- Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28.
- Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. 2014. Knowledge base completion via search-based question answering. In *Proceedings of the 23rd international conference on World wide web*, pages 515–526.
- Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. Deeppath: A reinforcement learning method for knowledge graph reasoning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 564–573.
- Zhanqiu Zhang, Jie Wang, Jiajun Chen, Shuiwang Ji, and Feng Wu. 2021. Cone: Cone embeddings for multi-hop reasoning over knowledge graphs. *Advances in Neural Information Processing Systems*, 34.

DATASET	RELATIONS	ENTITIES	TRAINING	VALIDATION	TESTING	ALL EDGES
FB15k	1,345	14,951	483,142	50,000	59,071	592,213
FB15k-237	237	14,505	272,115	17,526	20,438	310,079
NELL995	200	63,361	114,213	14,324	14,267	142,804

Table 6: The basic information about the three knowledge graph used for the experiments, and their standard training, validation, and testing edges separation according to (Ren and Leskovec, 2020).

DATASET	BATCH SIZE	DROPOUT RATIO	LABEL SMOOTHING	LEARNING RATE
FB15k	8,192	0.1	0.5	0.001
FB15k-237	8,192	0.1	0.5	0.001
NELL995	4,096	0.3	0.7	0.0003

Table 7: The best hyperparameters used by the Query2Particles model for the experiments on the queries originally used by (Ren and Leskovec, 2020).

(REN ET AL., 2020)	TRAINING		VALIDATION		TEST	
DATASET	1P	OTHERS	1P	OTHERS	1P	OTHERS
FB15k	273,710	273,710	59,097	8,000	67,016	8,000
FB15k-237	149,689	149,689	20,101	5,000	22,812	5,000
NELL995	107,982	107,982	16,927	4,000	17,034	4,000

(REN AND LESKOVEC, 2020)	TRAINING		VALIDATION		TEST	
DATASET	1P/2P/3P/2I/3I	2IN/3IN/INP/PIN/PNI	1P	OTHERS	1P	OTHERS
FB15k	273,710	273,71	59,097	8,000	67,016	8,000
FB15k-237	149,689	149,68	20,101	5,000	22,812	5,000
NELL995	107,982	107,98	16,927	4,000	17,034	4,000

Table 8: The detailed information for the queries used for training, validating, and testing all query embedding methods. The upper parts disclose the statistics of the queries taken from the (Ren et al., 2020) paper, while the lower part describes the queries taken from (Ren and Leskovec, 2020). The major differences are that the queries in (Ren and Leskovec, 2020) is harder than (Ren et al., 2020), and include five additional types of queries with the complement operation.

DATASET	MODEL	1P	2P	3P	2I	3I	PI	IP	2U	UP	AVG
FB15k	BETA E	65.1	25.7	24.7	55.8	66.5	43.9	28.1	40.1	25.4	41.6
	Q2B	68.0	21.0	14.2	55.1	66.5	39.4	26.1	35.1	16.7	38.0
	GQE	54.6	15.3	10.8	39.7	51.4	27.6	19.1	22.1	11.6	28.0
	Q2P (OURS)	82.6	30.8	25.5	65.1	74.7	49.5	34.9	32.1	26.2	46.8
FB15k-237	BETA E	39.0	10.9	10.0	28.8	42.5	22.4	12.6	12.4	9.9	20.9
	Q2B	40.6	9.4	6.8	29.5	42.3	21.2	12.6	11.3	7.6	20.1
	GQE	35.0	7.2	5.3	23.3	34.6	16.5	10.7	8.2	5.7	16.3
	Q2P (OURS)	39.1	11.4	10.1	32.3	47.7	24.0	14.3	8.7	9.1	21.9
NELL	BETA E	53.0	13.0	11.4	37.6	47.5	24.1	14.3	12.2	8.6	24.6
	Q2B	42.2	14.0	11.2	33.3	44.5	22.4	16.8	11.3	10.3	22.9
	GQE	32.8	11.9	9.6	27.5	35.2	18.4	14.4	8.5	8.8	18.6
	Q2P (OURS)	56.5	15.2	12.5	35.8	48.7	22.6	16.1	11.1	10.4	25.5
AVERAGE	BETA E	52.4	16.5	15.4	40.7	52.2	30.1	14.3	21.6	14.5	29.1
	Q2B	50.3	14.8	10.7	39.3	51.1	27.7	18.5	19.2	11.5	27.0
	GQE	40.8	11.5	8.6	30.2	40.4	20.8	14.7	12.9	8.7	21.0
	Q2P (OURS)	59.4	19.1	16.0	44.7	57.0	32.0	21.8	17.3	15.2	31.3

Table 9: The MRR result for existential positive first order queries comparing to the BetaE, Q2B, and GQE methods. The results are reported from the queries used by Ren and Leskovec (2020).

DATASET	MODEL	1P	2P	3P	2I	3I	IP	PI	2U	UP	AVG
FB15k	EMQL	42.4	50.2	45.9	63.7	70.0	<u>60.7</u>	61.4	9.0	<u>42.6</u>	49.5
	– SKETCH	50.6	46.7	41.6	61.8	67.3	54.2	53.5	21.6	40.0	48.6
	CQD-BEAM	91.8	77.9	<u>57.7</u>	<u>79.6</u>	<u>83.7</u>	37.5	<u>65.8</u>	83.9	34.5	<u>68.0</u>
	CQD-CO	91.8	45.4	19.1	<u>79.6</u>	<u>83.7</u>	33.6	51.3	81.6	31.9	57.6
	Q2P (OURS)	<u>90.2</u>	<u>74.6</u>	73.4	86.0	89.6	63.7	77.6	<u>83.4</u>	52.7	76.8
FB15k-237	EMQL	37.7	<u>34.9</u>	<u>34.3</u>	44.3	49.4	40.8	<u>42.3</u>	8.7	28.2	35.8
	– SKETCH	43.1	34.6	33.7	41.0	45.5	<u>36.7</u>	37.2	15.3	32.5	<u>35.5</u>
	CQD-BEAM	51.2	28.8	22.1	35.2	45.7	12.9	24.9	<u>28.4</u>	12.1	<u>29.0</u>
	CQD-CO	51.2	21.3	13.1	35.2	45.7	14.6	22.2	28.1	13.2	27.2
	Q2P (OURS)	<u>49.0</u>	44.2	44.6	50.1	57.5	34.1	44.2	32.9	<u>30.6</u>	43.0
NELL	EMQL	41.5	<u>40.5</u>	<u>38.6</u>	62.9	74.5	49.8	64.8	12.6	35.8	<u>46.8</u>
	– SKETCH	48.3	39.5	35.2	<u>57.2</u>	<u>69.0</u>	<u>48.0</u>	<u>59.9</u>	25.9	38.2	<u>46.8</u>
	CQD-BEAM	<u>66.7</u>	35.0	28.8	41.0	52.9	17.1	27.7	53.1	15.6	37.6
	CQD-CO	<u>66.7</u>	26.5	22.0	41.0	52.9	19.6	30.2	53.1	19.4	36.8
	Q2P (OURS)	67.0	53.0	52.6	52.9	<u>69.0</u>	38.0	47.0	<u>52.9</u>	<u>37.0</u>	52.2
AVERAGE	EMQL	40.5	41.9	39.6	57.0	64.6	50.4	<u>56.2</u>	10.1	35.5	44.0
	– SKETCH	47.3	40.3	36.8	53.3	60.6	<u>46.3</u>	50.1	20.9	<u>36.9</u>	43.6
	CQD-BEAM	69.9	<u>47.2</u>	36.2	51.9	60.8	22.5	39.5	<u>55.1</u>	<u>20.7</u>	<u>44.9</u>
	CQD-CO	69.9	31.1	18.1	51.9	60.8	22.6	34.6	54.3	21.5	40.5
	Q2P (OURS)	<u>68.7</u>	57.3	56.9	63.0	72.0	45.3	56.3	56.4	40.1	57.3

Table 10: The Hit@3 results for existential positive first order queries comparing to the EmQL and CQD method over the queries used by Ren et al. (2020).