Neural Multi-hop Logical Query Answering with Concept-level Answers

No Author Given

No Institute Given

Abstract. Neural multi-hop logical query answering (LQA) is a fundamental task to explore relational data such as knowledge graphs, which aims at answering multi-hop queries with logical operations based on distributed representations of queries and answers. Although previous LQA methods can give specific **instance-level** answers, they are not able to provide descriptive **concept-level** answers, where each concept is a description of a set of instances. Concept-level answers are more comprehensible to users and are of great usefulness in the field of applied ontology. In this work, we formulate the problem of LQA with concept-level answers (LQAC), solving which needs to address challenges in **incorporating**, **representing**, and **operating on concepts**. We propose an original solution for LQAC. Firstly, we incorporate description logic-based ontological axioms to provide the source of concepts. Then, we represent concepts and queries as fuzzy sets, i.e., sets whose elements have degrees of membership, to bridge concepts and queries with instances. Moreover, we design operators involving concepts on top of fuzzy set representation of concepts and queries for optimization and inference. Extensive experimental results on three real-world datasets demonstrate the effectiveness of our method for LQAC. In particular, we show that our method is promising in discovering complex logical biomedical facts.

Keywords: Knowledge Representation Learning · Multi-hop Logical Query Answering · Fuzzy Logic · Neuro-symbolic Reasoning.

1 Introduction

Along with the rapid development of high-quality large-scale knowledge infrastructures [35, 2], researchers are increasingly interested in exploiting knowledge bases for real-world applications, such as knowledge graph completion [8, 34] and entity alignment [36]. Recent years have witnessed increasing interest in a fundamental yet challenging task on such relational data, i.e., neural multi-hop logical query answering (LQA), which attempts to answer complex structured queries that include logical operations and multi-hop projections given the facts in knowledge bases using learned distributed representations [15]. Efforts [15, 29, 30] have been made to develop LQA systems by designing strategies to learn geometric or uncertainty-aware distributed query representations, and proposing

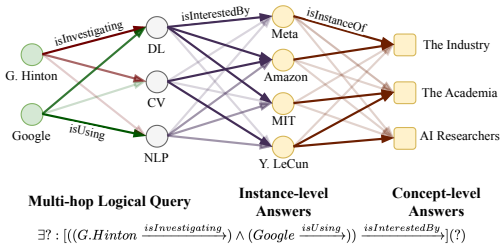

Fig. 1. An example of neural multi-hop logical query answering with concept-level answers (LQAC). The query is "Who will be interested in techniques that G. Hinton is investigating and Google is using?". The answers are not only instance-level answers as yellow circles: *Meta*, *Amazon*, *MIT*, and *Y. LeCun*, but also concept-level answers as squares: *AI Researchers*, *The Academia*, and *The Industry*. Such concept-level descriptive answers are more comprehensible to users.

mechanisms to deal with various logical operations on these distributed representations. However, existing LQA methods are limited to providing instances from knowledge graphs as answers.

In many real-world scenarios, solely providing instance-level answers is insufficient because users may also seek more descriptive concept-level answers, where each of the concepts is a description of a set of instances. For example in online question-answering forums, as shown in Fig. 1, users may ask "Who will be interested in techniques that G. Hinton is investigating and Google is using?" and expect both instance-level answers like *Meta*, *Amazon*, *MIT*, and *Y. LeCun*, as well as concept-level answers such as *AI Researchers*, *The Academia*, and *The Industry*. In this example, the conceptual answer *The Academia* refers to a summary of a set of instances consisting of *Y. LeCun* and *MIT*. Such concept-level answers augment the answer set to make it more comprehensible to users and fulfill their need for both detailed and abstract answers. In biomedical applications, to find the causes for a set of symptoms, scientists expect both instance-level answers (such as *SARS-CoV-2* causing *Fever*) as well as concept-level answers (such as *Viral infections* causing *Fever*). In this case, the answer constitutes a descriptive concept-level answer (e.g., *Viral infections*) that is a summary of a set of instance-level answers. Other downstream tasks like online chatbots [26] and conversational recommender systems [39] also need rich and comprehensive answers to provide better services. Providing both instance-level and concept-level answers can improve their capability of generating more informative responses for users and enriching the semantics in answers for downstream tasks.

Despite the significance of providing concept-level answers to multi-hop logical queries, previous LQA systems [15, 29, 30] do not support that for the following reasons. On the one hand, they reason over knowledge graphs where only instances and relations exist. There are no sources of concepts involved in their systems. On the other hand, mechanisms for exploiting concepts are lacking. Specifically, previous solutions only measure query–instance similarity for LQA,

without considering concept representations and operators involving concepts, such as a proper measurement of query–concept similarity.

Along this line, we propose an initial solution for neural multi-hop logical queries with concept-level answers (LQAC). The key challenges for addressing LQAC are to incorporate, represent, and to operate on concepts. First, we observe that terminological axioms in ontologies include taxonomic hierarchies of concepts, concept definitions, and concept subsumption axioms [14]. To **incorporate** concepts into for LQAC, we thus introduce terminological axioms into the system to provide sources of concepts. Second, we find that fuzzy sets [21], i.e., sets whose elements have degrees of membership, can naturally bridge instances with concepts while providing a notion of vagueness for sets of instances. Therefore, we **represent** concepts as fuzzy sets for LQAC. Meanwhile, properly representing queries is the prerequisite for effectively operating on concepts. We find that fuzzy sets can also bridge instances with queries, i.e., vague sets of instance-level answers. The theoretically supported and unparameterized fuzzy set operations enable us to resolve logical operations within queries. Thus, fuzzy sets are ideal for concept and query representation for LQAC. Then, **operators** involving concepts can also be designed based on fuzzy sets, including query–concept operators for concept retrieval, concept–concept operators for concept subsumption, and instance–concept operators for concept instantiation.

The main contributions of this work are: (1) To the best of our knowledge, we are the first to focus on the LQAC problem that aims at providing not only instance-level but also concept-level answers. LQAC goes beyond LQA over knowledge graphs and better satisfies the need of users, downstream tasks, and ontological applications; (2) We propose an initial solution for LQAC that incorporates, represents, and operates on concepts. We incorporate terminological axioms to provide sources of concepts and employ fuzzy sets as the representations of concepts and queries. Logical operations are supported by the well-established fuzzy set theory and operators involving concepts are designed upon fuzzy sets.

2 Related Work

Neural Multi-hop Logical Query Answering Great efforts have been made to develop LQA systems in recent years. GQE [15] formulated the LQA problem and proposed to use points in the embeddings space to represent logical queries. Q2B [29] use hyper-rectangles that can include multiple points in the embedding space to represent queries. HypE [11], ConE [38], and BetaE [30] extended Q2B by using more sophisticated geometric shapes or Beta distributions for query representation. More recent methods explore fuzzy logic [21] for LQA. CQD [1] used t-norm and t-conorms from the fuzzy logic theory to achieve promising performance on zero-shot settings. QTO [7] further improves CQD by introducing query tree optimization. FuzzQE [10], GNN-QE [40], and LogicE [27] directly represent entities and queries using embeddings with specially designed restrictions and interpreted them as fuzzy sets for LQA. However, these reasoners could only give instance-level answers, while we focus on the more

general LQAC problem that aims at additionally providing descriptive concepts as answers. Furthermore, they either use bit-wise fuzzy logic without fuzzy **sets** or use fuzzy sets with the number of elements that coincides with the embedding dimension, while we adhere to the foundational definition of fuzzy sets [21], allowing us to fully exploit theoretically supported fuzzy set operations.

Ontology Representation Learning Several methods have been developed to exploit ontologies using distributed representation learning [24]. ELEm [23], EmEL [28], and BoxEL [37] learn geometric embeddings for concepts in ontologies. Other approaches [32, 9] rely on graph embeddings or word embeddings and apply them to ontological axioms. Another line of research [16, 17] focuses on jointly embedding instances (entities) and relations in knowledge graphs together with concepts and roles (relations) in ontological axioms. Our work is related in that we incorporate description logic ontological axioms and we exploit concepts with distributed representation learning. However, previous methods are limited to link prediction tasks such as predicting protein–protein interactions or performing knowledge graph completion, which can be regarded as answering *1p* queries in Fig. 2, whereas we focus on more complex multi-hop logical queries.

3 Methodology

Incorporating, representing, and operating on concepts are key to LQAC. In this section, we first formulate the problem along with the process of incorporating concepts into the reasoning system. Then we propose an original solution for LQAC by designing concept representations and operators involving concepts.

3.1 Incorporating Concepts

Multi-hop Logical Query Answering (LQA) is defined on knowledge graphs. A knowledge graph is formulated as $\mathcal{KG} = \{\langle h, r, t\rangle\} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$, where h, r, t denote the head entity (instance), relation, and tail entity (instance) in triple $\langle h, r, t\rangle$, respectively, $\mathcal{E}$ and $\mathcal{R}$ refer to the instance set and the relation set in $\mathcal{KG}$. As shown in Fig. 2.(a), each triple $\langle h, r, t\rangle$ is regarded as a positive sample of the *1p* query $\exists ? : [h \xrightarrow{r}](?)$ with an answer t that satisfies $[h \xrightarrow{r}](t)$, where h is the anchor instance and $\xrightarrow{r}$ is the projection operation with relation r. Furthermore, LQA may also address the intersection, union, and negation operations $\wedge$, $\vee$, and $\neg$ within queries. Thus, infinite types of queries can be found with the combinations of these logical operations. We consider the representative types of queries, which are listed and demonstrated with their graphical structures in Fig. 2. For example, queries of type *pi* in Fig. 2.(b) are to ask $\exists ? : [(h_1 \xrightarrow{r_1} \xrightarrow{r_2}) \wedge (h_2 \xrightarrow{r_3})](?)$. LQA reasoners seek to provide a set of instances that satisfy the query as answers. We predict the possibility of each candidate instance $e \in \mathcal{E}$ satisfying a query $\exists ? : [q](?)$. We then rank the $|\mathcal{E}|$ possibilities and select the top-k instances in $\mathcal{E}$ as the set of answers. Since all the candidate answers are instances, we can only retrieve instance-level answers from LQA systems.

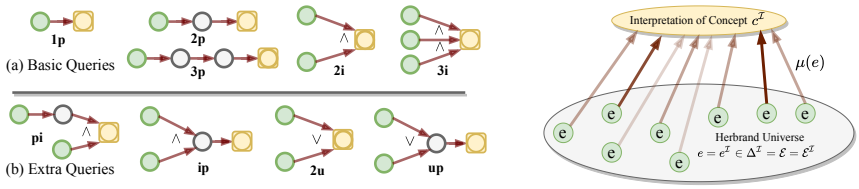

Fig. 2. Left: The considered types of queries are represented with their graphical structures. $\wedge$ and $\vee$ represent the intersection and union logical operations, respectively. Squares denote concepts and circles represent instances. Right: Illustration of the interpretation of a concept $c^{\mathcal{I}}$.

LQA with Concept-level Answers (LQAC) is based on a description logic-based knowledge base $\mathcal{KB}$, i.e., ontology, which is an ordered pair ($\mathcal{T}$, $\mathcal{A}$) for TBox $\mathcal{T}$ and ABox $\mathcal{A}$, where $\mathcal{T}$ is a finite set of terminological axioms and $\mathcal{A}$ is a finite set of assertion axioms. Specifically, terminological axioms within a TBox $\mathcal{T}$ are of the form $c_1 \sqsubseteq c_2$ where the symbol $\sqsubseteq$ denotes subsumption ($subClassOf$). In general, c_1 and c_2 can be concept descriptions that consist of concept names, quantifiers, roles (relations), and logical operators; we limit LQAC to axioms where c_1 and c_2 are concept names that will not involve roles or logical operators [3]. In the following, we do not distinguish between a concept name and a concept description unless there are special needs. In this case, a TBox is:

$$\mathcal{T} \subseteq \{c_i \sqsubseteq c_j | c_i, c_j \in \mathcal{C}\} \tag{1}$$

where $\mathcal{C}$ denotes the set of concept names in $\mathcal{KB}$. $\mathcal{T}$ accounts for the source of concepts and the pairwise concept subsumption information in the LQAC system. Assertion axioms in $\mathcal{A}$ consist of two parts, including role assertions:

$$\mathcal{A}_{ee} \subseteq \{\langle e_1, r, e_2\rangle | e_1, e_2 \in \mathcal{E}, r \in \mathcal{R}\} \tag{2}$$

where $e_1, e_2 \in \mathcal{E}$ denote instances (entities), $\mathcal{E}$ denotes the instance set in $\mathcal{KB}$, $r \in \mathcal{R}$ denotes the role assertion between e_1 and e_2, and $\mathcal{R}$ is the the role set of $\mathcal{A}_{ee}$. $\mathcal{A}_{ee}$ accounts for the triple-wise relational information about instances and roles. $\mathcal{A}$ also include the concept instantiations between an instance $e \in \mathcal{E}$ and a concept $c \in \mathcal{C}$:

$$\mathcal{A}_{ec} = \{e \triangleleft c\} \subseteq \mathcal{E} \times \mathcal{C}, \tag{3}$$

where $e \triangleleft c$ represents e is an element in c. $\mathcal{A}_{ec}$ serves as the bridge between $\mathcal{T}$ and $\mathcal{A}_{ee}$ by providing pairwise links between instances and concepts.

Since we incorporate concepts in the LQAC systems, we are able to ask questions about concepts. In particular, for a query $\exists ? : [q](?)$, we not only provide a set of instances of $\{a_e\}$ as the answers but also infer an explanation for each query result by summarizing instance-level answers with descriptive concepts, yielding another set of concept-level answers $\{a_c\}$. More specifically, as shown in Fig. 2, the answers are no longer restricted to be $e \in \mathcal{E}$ (denoted by

circles), they can also be $c \in \mathcal{C}$ (denoted by squares). To achieve this, we predict the possibility of each candidate instance $e \in \mathcal{E}$ as well as the possibility of each candidate concept $c \in \mathcal{C}$ satisfying a query $\exists ? : [q](?)$. We then rank $|\mathcal{E}|$ predicted scores of candidate instances and $|\mathcal{C}|$ predicted scores of candidate concepts. We select and combine the top-k results from each set of candidates as the final answers of q with instance-and-concept-level answers $\{a\} = \{a_e\} \bigcup \{a_c\}$. Note that LQA is a sub-problem of LQAC. First, an LQA reasoner can only provide a subset of the answers provided by an LQAC system, i.e., $\{a_e\} \subseteq \{a\}$. Also, the entire $\mathcal{KG}$ in the context of LQA corresponds to $\mathcal{A}_{ee}$ in LQAC, which is a subset of the ontology, i.e., $\mathcal{KG} \subseteq \mathcal{KB}$, leaving $\mathcal{T}$ and $\mathcal{A}_{ec}$ with conceptual information in the ontologies not explored. Therefore, LQAC is more general in terms of providing more answers and reasoning over more complex knowledge bases.

3.2 Representing Concepts

We are motivated to represent concepts as fuzzy sets by the relationship between concepts and instances. We gain insights into such a relationship from the definition of the semantics in description logics [6]:

Definition 1. *A terminological interpretation* $\mathcal{I} = \left(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}}\right)$ *over a signature* $(\mathcal{C}, \mathcal{E}, \mathcal{R})$ *consists of:*

- *a non-empty set* $\Delta^{\mathcal{I}}$ *called the domain*
- *an interpretation function* $\cdot^{\mathcal{I}}$ *that maps:*
  - *every instance* $e \in \mathcal{E}$ *to an element* $e^{\mathcal{I}} \in \Delta^{\mathcal{I}}$
  - *every concept* $c \in \mathcal{C}$ *to a subset of* $\Delta^{\mathcal{I}}$
  - *every role (relation)* $r \in \mathcal{R}$ *to a subset of* $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$

As we use a function-free language [3], we set $\Delta^{\mathcal{I}} = \mathcal{E}$, i.e., we focus on the Herbrand universe [25] of our knowledge base. Therefore, according to Definition 1, the interpretation of concept $c^{\mathcal{I}}$ is a subset of $\mathcal{E}$, which is finite. On the other hand, fuzzy sets [21] over the Herbrand Universe are finite sets whose elements have degrees of membership:

$$FS = \{\mu(x_1), \mu(x_2), \cdots, \mu(x_{|FS|})\}, \tag{4}$$

where $\mu(\cdot)$ is the membership function that measures the degree of membership of each element. Therefore, we further interpret all concepts as fuzzy sets over the finite domain $\Delta^{\mathcal{I}} = \mathcal{E} = \{e_1, e_2, \cdots, e_{|\mathcal{E}|}\}$ as the elements of fuzzy sets $\{x_1, x_2, \cdots, x_{|FS|}\}$. Thus, we have:

$$c^{\mathcal{I}} = \{\mu(e_1), \mu(e_2), \cdots, \mu(e_{|\mathcal{E}|})\}. \tag{5}$$

As shown in Fig.2, since the Herbrand universe for our language is always finite, the interpretation of concept $c^{\mathcal{I}}$ is fully determined by the fuzzy membership function $\mu(\cdot)$ that assigns a degree of membership to each instance $e = e^{\mathcal{I}} \in \Delta^{\mathcal{I}} = \mathcal{E} = \mathcal{E}^{\mathcal{I}}$ for $c^{\mathcal{I}} \in \mathcal{C}^{\mathcal{I}}$, where $\mathcal{E}^{\mathcal{I}}$ and $\mathcal{C}^{\mathcal{I}}$ are the interpretation of the instance set and the concept set.

To obtain the degree of membership of instance e_i in $c^{\mathcal{I}}$, i.e., $\mu(e_i)$, we first randomly initialize the embedding matrix of concepts and instances as $\mathbf{E}_c \in$

$\mathbb{R}^{|\mathcal{C}|\times d}$ and $\mathbf{E}_e \in \mathbb{R}^{|\mathcal{E}|\times d}$ with Xavier uniform initialization [13], where d is the embedding dimension. Then we obtain the embedding of each concept $\mathbf{c} \in \mathbb{R}^d$ by looking up the rows of $\mathbf{E}_c$. The embedding then serves as the generator of the fuzzy set representation of each concept FS_c. Thus, we compute the similarities between each concept c and every instance in our universe $e \in \mathcal{E} = \Delta^{\mathcal{I}}$ as the degrees of membership of each instance in the fuzzy set:

$$FS_c = \{\sigma(\mathbf{c} \otimes \mathbf{E}_e^T)\} = c^{\mathcal{I}}, \tag{6}$$

where symbol $\otimes$ denotes matrix multiplication and $\cdot^T$ represents the matrix transposition. The measured similarities are then normalized to $(0, 1)$ using the bit-wise sigmoid function $\sigma(\cdot)$. Here, the set-wise operation to obtain FS_c consists of $|\mathcal{E}|$ pair-wise operations on the instance–concept pairs; we use the same operator for concept instantiation, which we will explain in Section 3.4.

3.3 Representing Instances and Queries

Properly representing instances and queries is the prerequisite for operating on concepts. Fuzzy sets are also particularly suitable to represent instances and queries because both interpretations of instances and queries are essentially interpretations of concepts. Instances are represented as a special type of fuzzy set [31] that assigns the membership function $\mu(\cdot)$ to 1 for exactly one instance and assigns it to 0 to all other instances. Consequently, we can interpret instances as **(singleton) concepts**. Queries can be regarded as **concept (descriptions)** that are more general than concept names in $\mathcal{C}$, where concept names can be combined by logical operations and relations to form concept descriptions. Thus, we can use the same formalism designed for representing concept names to represent queries (concept descriptions). In this way, we can again use description logic semantics [5] to interpret a query: an interpretation function $\cdot^{\mathcal{I}}$ maps every query q to a subset of $\Delta^{\mathcal{I}}$. As the Herbrand universe $\Delta^{\mathcal{I}} = \mathcal{E}$ is finite, the interpretation of query $q^{\mathcal{I}}$ is fully determined by the fuzzy membership function

$$q^{\mathcal{I}} = \{\mu(e_1), \mu(e_2), \cdots, \mu(e_{|\mathcal{E}|})\}. \tag{7}$$

Note that adhering to the fuzzy logic theory [21] enables us to interpret logical operations within queries as vague and unparameterized fuzzy set operations. The preservation of vagueness is important in that LQAC requires uncertainty, rather than deductive reasoning that guarantees correctness. Unparameterized operations are desirable because they require fewer data during training and are often more interpretable. We then explain how to represent queries as fuzzy sets.

Representing Atomic Queries Each multi-hop logical query consists of one or more Atomic Queries (AQ), where an AQ is defined as a query that only contains projection(s) $\xrightarrow{r}$ from an anchor instance without logical operations such as intersection $\wedge$, union $\vee$, and negation $\neg$. Therefore, the first step is to represent AQs. We obtain the embeddings of each instance $\mathbf{e} \in \mathbb{R}^d$ and the i^{th} relation $\mathbf{r} \in \mathbb{R}^d$ by looking up the rows of the randomly initialized instance embedding matrices $\mathbf{E}_e \in \mathbb{R}^{|\mathcal{E}|\times d}$ and $\mathbf{E}_r \in \mathbb{R}^{|\mathcal{R}|\times d}$. Then, the generator for fuzzy

set representation FS_{aq} of a valid AQ $[e \xrightarrow{r_1} \cdots \xrightarrow{r_i}](?)$ is $(\mathbf{e} + \mathbf{r_1} + \cdots + \mathbf{r_i})$. Thus, we obtain the fuzzy set corresponding to the query aq as:

$$FS_{aq} = \{\sigma((\mathbf{e} + \mathbf{r_1} + \cdots + \mathbf{r_i}) \otimes \mathbf{E}_e^T)\} = aq^{\mathcal{I}}. \tag{8}$$

Similar to the process of obtaining fuzzy set representations of concepts, Eq.(8) is to acquire the degrees of membership of every candidate $e \in \mathcal{E}$ being an answer to a given AQ by computing their normalized similarities.

Fusing Atomic Queries AQs are fused by logical operations to form multi-hop logical queries. Since AQs are already represented in fuzzy sets and we are equipped with the theoretically supported fuzzy set operations, we interpret logical operations as fuzzy set operations over concepts to fuse AQs into the final query representations. For two fuzzy sets in the domain $\Delta^{\mathcal{I}} = \mathcal{E}$: $FS_1 = \{\mu_1(e_1), \cdots, \mu_1(e_{|\mathcal{E}|})\}$ and $FS_2 = \{\mu_2(e_1), \cdots, \mu_2(e_{|\mathcal{E}|})\}$, we have the **intersection** $\wedge$ and the **union** $\vee$ over the two fuzzy sets as:

$$FS_\wedge = FS_1 \wedge FS_2 = \{\forall e \in \mathcal{E} : \mu_\wedge(e) = \top(\mu_1(e), \mu_2(e))\}, \tag{9}$$

$$FS_\vee = FS_1 \vee FS_2 = \{\forall e \in \mathcal{E} : \mu_\vee(e) = \bot(\mu_1(e), \mu_2(e))\}, \tag{10}$$

and we have the **negation** $\neg$ over FS as:

$$FS_\neg = \{\mu_\neg(e_1), \cdots, \mu_\neg(e_{|\mathcal{E}|})\} = \{\forall e \in \mathcal{E} : \mu_\neg(e) = 1 - \mu(e)\}, \tag{11}$$

where a t-norm $\top : [0,1] \times [0,1] \mapsto [0,1]$ is a generalisation of logical conjunction [20]. Examples of t-norms include the Gödel t-norm $\top_{\min}(x,y) = \min\{x,y\}$, the product t norm $\top_{\text{prod}}(x,y) = x \cdot y$, and the Łukasiewicz t-norm $\top_{\text{Luk}}(x,y) = \max\{0, x+y-1\}$ [22]. Analogously, a t-conorm $\bot$: $[0,1] \times [0,1] \mapsto [0,1]$ is complementary to t-norm and generalizes logical disjunction, which is defined by $\bot(x,y) = 1 - \top(1-x, 1-y)$ [1]. The choice of the t-norm is a hyperparameter.

Thus, each query can be decomposed into AQs and represented as a fuzzy set with Eq.(8), and then fuzzy set representations of AQs are fused by the fuzzy set operations in Eq.(9), (10), and (11) to obtain the final representation of the query. Note that fuzzy set operations hold the property of *closure*, which means the input and output of these operations remain in fuzzy sets. Thus, the final representation of each query is also a fuzzy set FS_q.

3.4 Operating on Concepts

Here we design operators involving concepts for concept retrieval, instance retrieval, concept subsumption, and concept instantiation based on the fuzzy set representations.

Concept Retrieval is to provide concept-level answers, i.e., $\{a_c\}$ as shown in Section 3.1. We illustrate the overall process of concept retrieval in Fig. 3. After we obtain the fuzzy set of the query FS_q and the answer FS_c, we measure the possibility of each $c \in \mathcal{C}$ being a concept-level answer of a given query upon fuzzy set representations. More specifically, we measure how well can FS_c and FS_q

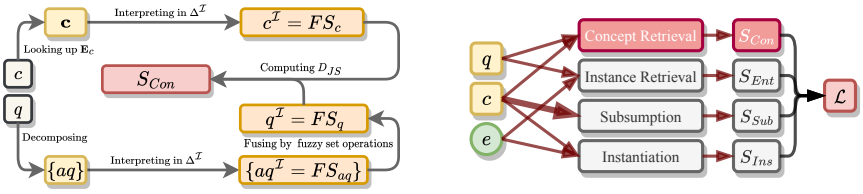

Fig. 3. Left: Representing concepts and queries to retrieve concept-level answers. Right: Illustration of obtaining $\mathcal{L}$ for optimization.

align with each other based on the Jensen-Shannon divergence D_{JS} [12], which is a symmetrized and smoothed version of the Kullback-Leibler divergence D_{KL}. The similarity function S_{Con} is defined by:

$$S_{Con} = -D_{JS}(P\|Q) = -\frac{1}{2}D_{KL}(P\|M) + \frac{1}{2}D_{KL}(Q\|M) \tag{12}$$

where $M = \frac{1}{2}(P+Q)$, P and Q represent the normalized fuzzy set representations of the considered query and concept descriptions, which are given by:

$$P = \frac{FS_c}{\max\left(\|FS_c\|_p, \epsilon\right)}, Q = \frac{FS_q}{\max\left(\|FS_q\|_p, \epsilon\right)}, \tag{13}$$

where ϵ is a small value to avoid division by zero and p is the exponent value in the norm formulation $\|\cdot\|_p$. S_{Con} is then used for model training and inference.

Instance Retrieval aims to provide instance-level answers; for this purpose, only query–instance similarities S_{Ent} need to be measured without the necessity of designing new mechanisms. Therefore, we follow the pioneering work[1] [15] on LQA to represent each query as an embedding $\mathbf{q} = f(q; \Omega)$ and measure query–instance similarity S_{Ent} as:

$$S_{Ent} = \gamma - \|\mathbf{q} - \mathbf{e}\|_1 \tag{14}$$

where γ is the margin. Here, we elaborate on the method to obtain the query embedding $\mathbf{q}$ for providing instance-level answers, i.e., $f(\cdot)$ with parameters Ω. Specifically, the **projection** operation $\mathbf{x} \xrightarrow{r}$ that project an instance or query embedding $\mathbf{x}$ with relation r is resolved by:

$$\mathbf{q} = \mathbf{x} + \mathbf{r}, \tag{15}$$

where $\mathbf{x} \in \mathbb{R}^d$ is another query embedding that is obtained in advance or an instance embedding obtained by looking up $\mathbf{E}_e \in \mathbb{R}^{|\mathcal{E}| \times d}$ by rows. The **intersection** of two query embeddings $\mathbf{q_1}$ and $\mathbf{q_2}$ is resolved by:

$$\mathbf{q} = a(\mathbf{q_1} \oplus \mathbf{q_2}; \Omega)_1 * \mathbf{q_1} + a(\mathbf{q_1} \oplus \mathbf{q_2}; \Omega)_2 * \mathbf{q_2}, \tag{16}$$

where $\oplus$ denotes matrix concatenation over the last dimension, Ω denote s the parameters of $a(\cdot)$, and $a(\cdot)$ is a two-layer feed-forward network with *Relu* activation. $a(\cdot)_1$ and $a(\cdot)_2$ represent the first and second d attention weights, respectively. The **union** of two query embeddings $\mathbf{q_1}$ and $\mathbf{q_2}$ is resolved by:

$$\mathbf{q} = \max(\mathbf{q_1}, \mathbf{q_2})_{-1}, \tag{17}$$

where $\max(\cdot)_{-1}$ denotes the max operation over the last dimension.

[1] https://github.com/snap-stanford/KGReasoning

Concept Subsumption As defined by Eq.(1), $\mathcal{T}$ supplies for relational information among concepts with the form of concept subsumptions. Although concepts are represented in fuzzy sets and we already designed mechanism to measure the similarity between two fuzzy sets, we can not directly apply the method in Section 3.4 for concept subsumptions. It is because we need to measure the degree of inclusion of one concept to another instead of the similarities between them. The degree of inclusion is asymmetrical and more complex than the similarity measurement. Therefore, we employ a neural network $h(\cdot)$ to model the degree of inclusion:

$$S_{Sub} = h\left(\mathbf{c_1} \oplus \mathbf{c_2}; \theta\right) \tag{18}$$

where symbol $\oplus$ denotes matrix concatenation over the last dimension, and θ denotes the parameters of $h(\cdot)$. In this work, $h(\cdot)$ is a two-layer feed-forward network with *Relu* activation. Note that we directly use the embeddings of concepts without interpreting concept in the Herbrand universe of entities $\Delta^{\mathcal{I}} = \mathcal{E}$ because neither concept–instance relationships need to be modeled nor logical operations need to be resolved for modeling degree of subsumption.

Concept Instantiation As defined by Eq.(3), $\mathcal{A}_{ec}$ bridges $\mathcal{T}$ and $\mathcal{A}_{cc}$ by providing links between instances and concepts. Such links instantiate concept with its describing instances and thus offer relational information with the form of concept instantiation. Recall that in Section 3.2, we obtain the fuzzy set representation of concepts by computing the similarities between the given c and every candidate $e \in \mathcal{E}$ with Eq.(6). In the case of concept instatiation, the set-wise computation Eq.(6) is degraded to pair-wise similarity measurement for each concept-instance pair:

$$S_{Ins} = \sigma(\mathbf{c} \otimes \mathbf{e}^T) \tag{19}$$

where $\mathbf{c} \in \mathbb{R}^d$ and $\mathbf{e} \in \mathbb{R}^d$ are concept and instance embeddings, respectively.

3.5 Optimization and Inference

The parameters in our model include the instance embedding matrix $\mathbf{E}_e$, the concept embedding matrix $\mathbf{E}_c$, the relation embedding matrix $\mathbf{E}_r$, θ in Section 3.4, and Ω in Section 3.4. In the training stage, we sample m negative samples for each positive sample of concept-level answering $[q](c^+)$ by corrupting c^+ with randomly sampled $c_i^- \in \mathcal{C}$ $(i = 1, \cdots, m)$. Similarly, negative samples for instance-level answering $[q](e_i^-)$ are obtained by corrupting e^+ in $[q](e^+)$ with randomly sampled $e_i^- \in \mathcal{E}$. For concept subsumption and concept instantiation, both sides of the concept-concept pairs and concept-instance pairs are randomly corrupted following the same procedure. We illustrate the modularized computation procedure for model training as Fig. 3. The loss is defined as

$$\mathcal{L} = -\frac{1}{4m} \sum_{n \in N} \sum_{i=1}^{m} \log \sigma(S_n^+ - S_{n_i}^-) \tag{20}$$

where $N = \{Con, Ent, Sub, Ins\}$ denotes the set of the four included tasks discussed in Section 3.4, S_n^+ (or $S_{n_i}^-$) denotes the predicted similarity or degree

of inclusion of the positive (or negative) sample according to task n. The overall optimization process of $\mathcal{L}$ is outlined in Algorithm 1 in Supplementary Materials.

In the inference stage, we predict S_{Con} (or S_{Ent}) for every candidate concept $c \in \mathcal{C}$ (or instance $e \in \mathcal{E}$) regarding to query q and select the top-k results to be the concept-level answers $\{a_c\}$ (or instance-level answers $\{a_e\}$) for query q. Thus, we are able to achieve LQAC by providing the comprehensive answers $\{a\} = \{a_e\} \bigcup \{a_c\}$. Although concept subsumption and concept instantiation are not included in the inference stage, they empowered our LQAC system to better represent and operate concepts by providing training samples and extra supervision signals of the relational data.

4 Experiments

We conduct extensive experiments to answer the following research questions: How to properly compare our method with LQA systems (**RQ1**)? How does our method perform in concept retrieval (**RQ2**) and instance retrieval (**RQ3**)? How do concept subsumption and instantiation affect reasoning performance (**RQ4**)?

4.1 Experimental Settings

Baselines (RQ1) The considered baselines are representative methods for LQA, namely GQE [15], Q2B [29], and BetaE [30], along with a recent method FuzzQE [10] that directly represent instances and queries using embeddings with specially designed restrictions and interpreted them as fuzzy sets for LQA. Since LQA reasoners can only provide instance-level answers, we need to come up with a strategy to enforce concept retrieval, so as to compare with our method for LQAC. Therefore, we introduce the *One-more-hop* experiment. That is, we exploit all the information given by $\mathcal{KB} = (\mathcal{T}, \mathcal{A})$ and simply degrade concepts to instances in the training stage. Specifically, we first augment $\mathcal{A}_{ec}$ by the transductive links provided by $\mathcal{T}$. Then we combine the augmented $\mathcal{A}_{ec}$ and $\mathcal{A}_{ee}$ to form the new knowledge graph $\mathcal{KG}'$. Note that part of the instances in $\mathcal{KG}'$ are the degraded concepts and $\mathcal{KG}'$ contains an additional relation r_{ec} to describe the *isInstanceOf* relation between an instance and a concept. Thus, we construct training examples with various types of queries using $\mathcal{KG}'$. In the inference stage, two sets of candidate instances are prepared for each query. The first is the instance-level candidate set, which can be handled following the original papers [15, 29, 30]. Another set contains the degraded concepts. To predict the possibility of a concept being an answer to a query $[q](?)$, we add one more projection operation with the relation r_{ec}, i.e., $[q'](?) = [q \xrightarrow{r_{ec}}](?)$. In other words, concept-level reasoning is implicitly achieved by an additional hop asking the *isInstanceOf* upon instance retrieval queries, i.e., the *One-more-hop*.

Table 1. Hit@3 of concept and instance retrieval. The best results are in boldface.

		Concept-level Answers									
		1p	2p	3p	2i	3i	pi	ip	2u	up	avg
YAGO4	GQE	43.7	67.4	44.9	67.1	49.9	15.0	12.3	19.7	9.4	36.6
	Q2B	47.1	75.4	78.2	**70.0**	58.2	2.8	1.9	1.9	1.4	37.4
	BetaE	47.8	74.7	76.7	64.2	52.5	10.1	10.2	2.9	5.0	38.2
	FuzzQE	41.6	69.8	59.4	67.5	52.8	16.9	13.3	23.4	10.4	39.5
	LQAC	**60.3**	**88.8**	**88.7**	66.4	**65.5**	**60.2**	**57.1**	**59.9**	**52.5**	**66.6**
DBpedia	GQE	28.7	42.9	40.0	32.5	36.0	13.0	14.4	7.2	10.7	25.0
	Q2B	28.2	41.4	38.0	32.7	33.5	11.7	13.0	8.4	9.3	24.0
	BetaE	34.1	45.4	50.8	37.2	41.2	14.6	10.9	4.0	8.0	27.4
	FuzzQE	29.0	39.2	38.2	30.5	29.7	15.6	15.9	11.2	12.7	24.7
	LQAC	**62.4**	**83.7**	**83.6**	**50.9**	**43.7**	**45.0**	**29.9**	**67.2**	**67.3**	**59.3**
		Instance-level Answers									
		1p	2p	3p	2i	3i	pi	ip	2u	up	avg
YAGO4	GQE	29.7	17.3	4.6	29.0	31.4	23.5	18.6	13.6	17.4	20.6
	Q2B	28.4	20.0	7.0	29.5	34.4	27.0	21.3	11.6	18.5	22.0
	BetaE	31.4	22.3	12.0	33.5	37.5	31.3	24.1	12.5	**23.8**	25.4
	FuzzQE	30.7	17.3	4.1	30.0	32.3	22.9	18.1	12.5	18.6	20.7
	LQAC	**39.6**	**27.3**	**18.4**	**56.6**	**70.2**	**34.2**	**39.1**	**14.9**	23.7	**36.0**
DBpedia	GQE	25.2	18.9	19.8	26.5	43.6	17.2	33.3	12.6	21.1	24.2
	Q2B	21.3	16.5	17.7	27.5	31.9	19.7	25.7	9.1	15.4	20.7
	BetaE	20.9	23.2	21.7	27.5	32.9	27.5	28.4	13.0	25.1	24.5
	FuzzQE	18.5	16.3	18.5	26.5	33.0	32.6	25.4	8.0	16.0	21.6
	LQAC	**34.6**	**28.0**	**29.0**	**44.6**	**54.8**	**21.4**	**40.6**	**17.8**	**23.0**	**32.6**

Implementation Details We use three real-world large-scale knowledge bases: the English Wikipedia version[2] of YAGO4, 2016-10 release[3] of DBpedia, and the same subset of Gene Ontology[4] used in well-established ontology embedding works [23, 37]. We first filter out low-degree instances in $\mathcal{A}_{ee}$ and $\mathcal{A}_{ec}$ with the threshold of 5. Then we split $\mathcal{A}_{ee}$ to leave 5% out for evaluation. We follow BetaE [30] to construct examples of logical queries from $\mathcal{A}_{ee} = \mathcal{KG}$. We summarize the statistics of datasets in Table 1 and Table 3 in the Supplementary Materials. In the training stage, the initial learning rate of the Adam [19] optimizer, the embedding dimension d, and the batch size, are tuned by grid searching within $\{1e^{-2}, 1e^{-3}, 1e^{-4}, 1e^{-5}\}$, {128, 256, 512}, and {256, 512, 1024}, respectively. We keep the number of corrupted negative samples for each positive sample m, the small value ϵ, the exponent value p, the margin γ, and the adopted type of t-norm as 4, $1e^{-12}$, 1, 12, and $\top_{\text{prod}}$, respectively. We use the filtered setting [29] for testing and report the averaged results of Mean Reciprocal Rank (MRR), Hit@3, and Hit@10 over 3 independent runs.

[2] https://yago-knowledge.org/downloads/yago-4
[3] http://downloads.dbpedia.org/wiki-archive/downloads-2016-10.html
[4] https://bio2vec.cbrc.kaust.edu.sa/data/elembeddings/el-embeddings-data.zip

Table 2. Hit@3 of concept and instance retrieval on the biomedical GO dataset. The best results are in boldface.

		1p	2p	3p	2i	3i	pi	ip	2u	up	avg
Concept	GQE	0.004	0.088	0.067	0.101	0.099	0.010	0.014	0.003	0.006	0.044
	Q2B	0.005	0.085	0.066	0.095	0.109	0.006	0.007	0.005	0.007	0.043
	BetaE	0.004	0.085	0.068	0.102	0.099	0.012	0.013	0.003	0.009	0.044
	FuzzQE	0.008	0.088	0.069	0.107	0.112	0.010	0.014	0.007	0.005	0.047
	LQAC	**0.031**	**0.683**	**0.860**	**0.147**	**0.149**	**0.160**	**0.165**	**0.438**	**0.805**	**0.382**
Instance	GQE	0.130	0.462	**0.641**	0.079	0.068	0.078	0.348	0.064	0.487	0.262
	Q2B	0.137	0.467	0.637	0.087	0.078	0.097	0.358	0.065	**0.508**	0.270
	BetaE	0.129	**0.463**	0.632	0.075	0.065	0.085	0.342	0.059	0.498	0.261
	FuzzQE	0.126	0.461	0.635	0.095	0.090	0.102	0.349	0.051	0.500	0.268
	LQAC	**0.236**	**0.463**	0.628	**0.200**	**0.219**	**0.237**	**0.405**	**0.114**	0.347	**0.317**

4.2 Experimental Results

Concept-level Answers (RQ2) We conduct the *One-more-hop* experiment to answer RQ2. As shown in Table.1, our method consistently outperforms baselines on various evaluation metrics with large margins. For the *basic* queries in Fig. 2 that are simply projections and intersections, our method significantly improves the performance of concept-level reasoning, especially for the multi-hop queries *1p*, *2p*, and *3p*. For *extra* queries in Fig. 2 that are more complex in terms of including unions or combined logical operations, we even boosted the performance exponentially. The average performance of our method is also significantly better than the baselines.

The superior performance can be interpreted in two folds. First, due to the lack of reasoning capabilities when concepts are involved, GQE, Q2B, FuzzQE, and BetaE need to do reasoning over more complicated queries. For example, baselines need to reason over an *ipp* query $[((h_1 \xrightarrow{r_1}) \wedge (h_2 \xrightarrow{r_2})) \xrightarrow{r_{ec}}](?)$ to provide concept-level answers of an *ip* query $[(h_1 \xrightarrow{r_1}) \wedge (h_2 \xrightarrow{r_2})](?)$. Therefore, *1p* queries become *2p* queries for baseline methods, *2p* becomes *3p*, and so on. Thus, the complexity of the transformed queries limits their performance. Second, explicit supervision signals for concept-level reasoning are not provided. Since the concepts are degraded as instances, LQA methods could not explicitly feed the empirical error on concept-level answers back to update the model parameters. It is thus understandable that they cannot perform well in providing concept retrieval, especially on *extra* queries that are more complicated and require supervision signals more eagerly.

Moreover, as shown in Table 2, our method outperforms all baselines in providing concept-level answers. Note that the instances in the GO datasets are proteins, and the concepts are molecular functions, cellular components, or biological processes[5]. Therefore, such results demonstrate that the formulated LQAC task and our solution enable effective complex logical reasoning for crucial biological facts, which is promising to be applied to biomedical applications to facilitate healthcare services.

[5] http://geneontology.org/

Table 3. Ablation study on Concept Subsumptions and Instantiation on DBpedia dataset. The best MRR results are in boldface.

Concept	1p	2p	3p	2i	3i	pi	ip	2u	up	avg
LQAC w/o Sub	53.3	72.4	71.5	19.0	15.8	24.1	19.3	59.9	61.7	44.1
LQAC w/o Ins	52.8	68.8	67.7	38.3	35.4	34.2	19.3	56.2	61.6	48.3
LQAC	**55.0**	**80.8**	**80.7**	**42.9**	**36.7**	**42.0**	**28.5**	**63.4**	**64.9**	**55.0**
Instance	1p	2p	3p	2i	3i	pi	ip	2u	up	avg
LQAC w/o Sub	17.7	20.1	21.0	18.0	18.8	14.9	22.7	9.1	16.0	17.6
LQAC w/o Ins	17.4	18.8	19.0	18.1	18.7	14.3	22.9	8.7	17.7	17.3
LQAC	**28.8**	**24.5**	**24.4**	**38.4**	**46.3**	**20.1**	**33.6**	**14.0**	**20.6**	**27.9**

Instance-level Answers (RQ3) Although our method is designed for LQAC, it is interesting to know its performance on instance-level reasoning only (for answering RQ3). The results in Table 1 and 2 show that our method also outperforms LQA methods on most types of queries on various metrics. The performance gain should be credited to its capability of representing and operating on concepts. It thus encodes the additional information of the relationships among queries, instances, and concepts, which are helpful for instance retrieval. Additional experimental results are provided in Table 2, Table 4, and Table 5 in the Supplementary Materials.

Ablation Study (RQ4) We conduct an ablation study on the effect of incorporating $\mathcal{T}$ and $\mathcal{A}_{ec}$ and our proposed mechanisms of exploiting them to answer RQ4. As shown in Table 3, when the *Subsumption* task is not included, i.e., ontological axioms in $\mathcal{T}$ are not used and S_{Sub} is not computed, *LQAC w/o Sub* underperforms *LQAC* on all types of queries for both tasks. Such results show the importance of incorporating the relational information between concepts, and the effectiveness of the designed operator in Section 3.4 for modeling such information. On the other hand, *LQAC* consistently outperforms *LQAC w/o Ins* on all types of queries. This verifies that the relational information about $isInstanceOf$ in $\mathcal{A}_{ec}$ is vital, and the mechanism introduced in Section 3.4 is effective to tackle with *Instantiation*.

5 Discussion

Instance Realization is closely related to LQAC, which is to retrieve the **most specific** concept among all concepts that a given instance belongs to [18]. We are more interested in specific and fine-grained concepts because they are more informative. For example, from the last column of Table 4, the instance-level answer 'American Academy of Dramatic Arts' is intuitively better to be answered by concept 'Drama School' instead of 'School', where 'Drama School' is substantially more informative than 'School'. Since concept-level answers are descriptions of sets of instances, the most specific one is equivalent to the concept describing the minimal set of instances among all true concepts. However, as the predicted ranks are shown in Table 4, our method is not always able to give

Table 4. Case study on the Instance Realization task for complex logical query [(Manon the Moon $\xrightarrow{\text{actor}}\xrightarrow{\text{alumniOf}}$) $\wedge$ (William Devane $\xrightarrow{\text{alumniOf}}$)](?), which aims to find the *most specific* concept among all true answers.

Concept	*Score*	*Predicted*	*Expected*
Academy	-0.310	5^{th}	3^{rd}
Drama_school	-0.234	3^{rd}	1^{st}
LocalBusiness	-0.238	4^{th}	5^{th}
CollegeOrUniversity	-0.159	1^{st}	2^{nd}
School	-0.219	2^{nd}	4^{th}

the most specific answer. Although our results are stable thanks to the filtered setting, the internal rank among true concepts is not tested. We recognize this as an important future research direction in neuro-symbolic reasoning. Nevertheless, instance realization takes instances instead of multi-hop logical queries as input to retrieve concepts, which fundamentally differs from LQAC.

On the other hand, recall that we define TBox in our system as $\mathcal{T} \subseteq \{c_i \sqsubseteq c_j | c_i, c_j \in \mathcal{C}\}$, where $\mathcal{C}$ denotes the set of concept **names** in $\mathcal{KB}$. However, c can be concept **descriptions** in general. In one of the most widely used description logics $\mathcal{ALC}$ [4], concept descriptions are recursively defined: Every concept name is a concept description; if C and D are concept descriptions and r is a role (relation), then $C \sqcap D$, $C \sqcup D$, $\neg C$, $\forall R.C$, and $\exists r.C$ are concept descriptions. Neural reasoning over $\mathcal{ALC}$ ontologies with concept descriptions, is particularly challenging mainly because the axioms no longer form a graph structure due to their arbitrary depths and the involvement of quantifiers. However, many $\mathcal{ALC}$ ontologies have been developed to provide rich and accurate knowledge about various domains, in particular in the biomedical field where hundreds of ontologies have been created [33]. Therefore, we recognize neural logical reasoning over the full $\mathcal{ALC}$ ontologies as a highly potential future research direction that would be influential in the neuro-symbolic reasoning communities and beyond.

6 Conclusion

We formulated the LQAC problem that is of great importance for users, downstream tasks, and ontological applications. Accordingly, we propose an initial method for LQAC that properly incorporates ontological axioms, represents concepts and queries as fuzzy sets, and operates on concepts based on fuzzy sets. Experimental results and in-depth analysis demonstrate the effectiveness of our method and the significance of LQAC in biomedical knowledge discovery.

Supplemental Material Statement: Source code, statistics of datasets, data preprocessing procedures, and tuned hyperparameter settings are available from our code repository[6]. Detailed computation procedures of model training and additional experimental results are also attached in our code repository and, if accepted, will be published on arXiv in an extended version of the paper.

[6] https://anonymous.4open.science/r/LQAC-F582

References

1. Arakelyan, E., Daza, D., Minervini, P., Cochez, M.: Complex query answering with neural link predictors. In: ICLR (2020)
2. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: Dbpedia: A nucleus for a web of open data. In: The semantic web, pp. 722–735. Springer (2007)
3. Baader, F.: Appendix: description logic terminology. The Description logic handbook: Theory, implementation, and applications pp. 485–495 (2003)
4. Baader, F., Calvanese, D., McGuinness, D., Patel-Schneider, P., Nardi, D., et al.: The description logic handbook: Theory, implementation and applications. Cambridge university press (2003)
5. Baader, F., Horrocks, I., Lutz, C., Sattler, U.: Introduction to description logic. Cambridge University Press (2017)
6. Baader, F., Horrocks, I., Sattler, U.: Description logics. Foundations of Artificial Intelligence **3**, 135–179 (2008)
7. Bai, Y., Lv, X., Li, J., Hou, L.: Answering complex logical queries on knowledge graphs via query tree optimization. arXiv preprint arXiv:2212.09567 (2022)
8. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. NeurIPS **26** (2013)
9. Chen, J., Hu, P., Jimenez-Ruiz, E., Holter, O.M., Antonyrajah, D., Horrocks, I.: Owl2vec*: Embedding of owl ontologies. Machine Learning pp. 1–33 (2021)
10. Chen, X., Hu, Z., Sun, Y.: Fuzzy logic based logical query answering on knowledge graphs. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 36, pp. 3939–3948 (2022)
11. Choudhary, N., Rao, N., Katariya, S., Subbian, K., Reddy, C.K.: Self-supervised hyperboloid representations from logical queries over knowledge graphs. In: WWW. pp. 1373–1384 (2021)
12. Endres, D.M., Schindelin, J.E.: A new metric for probability distributions. IEEE Transactions on Information theory **49**(7), 1858–1860 (2003)
13. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: AISTATS. pp. 249–256. JMLR Workshop and Conference Proceedings (2010)
14. Gruber, T.R.: A translation approach to portable ontology specifications. Knowledge acquisition **5**(2), 199–220 (1993)
15. Hamilton, W.L., Bajaj, P., Zitnik, M., Jurafsky, D., Leskovec, J.: Embedding logical queries on knowledge graphs. In: NeurIPS. pp. 2030–2041 (2018)
16. Hao, J., Chen, M., Yu, W., Sun, Y., Wang, W.: Universal representation learning of knowledge bases by jointly embedding instances and ontological concepts. In: KDD. pp. 1709–1719 (2019)
17. Hao, J., Ju, C.J.T., Chen, M., Sun, Y., Zaniolo, C., Wang, W.: Bio-joie: Joint representation learning of biological knowledge bases. In: Proceedings of the 11th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics. pp. 1–10 (2020)
18. Huitzil, I., Bernad, J., Bobillo, F.: Algorithms for instance retrieval and realization in fuzzy ontologies. Mathematics **8**(2), 154 (2020)
19. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
20. Klement, E.P., Mesiar, R., Pap, E.: Triangular norms. position paper i: basic analytical and algebraic properties. Fuzzy sets and systems **143**(1), 5–26 (2004)

21. Klir, G., Yuan, B.: Fuzzy sets and fuzzy logic, vol. 4. Prentice hall New Jersey (1995)
22. van Krieken, E., Acar, E., van Harmelen, F.: Analyzing differentiable fuzzy implications. In: KR. pp. 893–903. IJCAI Organization (2020)
23. Kulmanov, M., Liu-Wei, W., Yan, Y., Hoehndorf, R.: El embeddings: Geometric construction of models for the description logic el++. IJCAI (2019)
24. Kulmanov, M., Smaili, F.Z., Gao, X., Hoehndorf, R.: Semantic similarity and machine learning with ontologies. Briefings in bioinformatics **22**(4), bbaa199 (2021)
25. Lee, R.C.: Fuzzy logic and the resolution principle. Journal of the ACM (JACM) **19**(1), 109–119 (1972)
26. Liu, S., Chen, H., Ren, Z., Feng, Y., Liu, Q., Yin, D.: Knowledge diffusion for neural dialogue generation. In: ACL. pp. 1489–1498 (2018)
27. Luus, F., Sen, P., Kapanipathi, P., Riegel, R., Makondo, N., Lebese, T., Gray, A.: Logic embeddings for complex query answering. arXiv preprint arXiv:2103.00418 (2021)
28. Mondala, S., Bhatiab, S., Mutharajua, R.: Emel++: Embeddings for description logic (2021)
29. Ren, H., Hu, W., Leskovec, J.: Query2box: Reasoning over knowledge graphs in vector space using box embeddings. In: ICLR (2019)
30. Ren, H., Leskovec, J.: Beta embeddings for multi-hop logical reasoning in knowledge graphs. NeurIPS **33** (2020)
31. Rihoux, B., De Meur, G.: Crisp-set qualitative comparative analysis (csqca). Configurational comparative methods: Qualitative comparative analysis (QCA) and related techniques **51**, 33–68 (2009)
32. Smaili, F.Z., Gao, X., Hoehndorf, R.: Opa2vec: combining formal and informal content of biomedical ontologies to improve similarity-based prediction. Bioinformatics **35**(12), 2133–2140 (2019)
33. Smith, B., Ashburner, M., Rosse, C., Bard, J., Bug, W., Ceusters, W., Goldberg, L.J., Eilbeck, K., Ireland, A., Mungall, C.J., Leontis, N., Serra, P.R., Ruttenberg, A., Sansone, S.A., Scheuermann, R.H., Shah, N., Whetzel, P.L., Lewis, S.: The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration. Nat Biotech **25**(11), 1251–1255 (2007)
34. Tang, Z., Pei, S., Zhang, Z., Zhu, Y., Zhuang, F., Hoehndorf, R., Zhang, X.: Positive-unlabeled learning with adversarial data augmentation for knowledge graph completion. In: IJCAI (2022)
35. Tanon, T.P., Weikum, G., Suchanek, F.: Yago 4: A reason-able knowledge base. In: European Semantic Web Conference. pp. 583–596. Springer (2020)
36. Trisedya, B.D., Qi, J., Zhang, R.: Entity alignment between knowledge graphs using attribute embeddings. In: AAAI. vol. 33, pp. 297–304 (2019)
37. Xiong, B., Potyka, N., Tran, T.K., Nayyeri, M., Staab, S.: Faithful embeddings for el++ knowledge bases. In: The Semantic Web–ISWC 2022: 21st International Semantic Web Conference, Virtual Event, October 23–27, 2022, Proceedings. pp. 22–38. Springer (2022)
38. Zhang, Z., Wang, J., Chen, J., Ji, S., Wu, F.: Cone: Cone embeddings for multi-hop reasoning over knowledge graphs. Advances in Neural Information Processing Systems **34**, 19172–19183 (2021)
39. Zhou, K., Zhao, W.X., Bian, S., Zhou, Y., Wen, J.R., Yu, J.: Improving conversational recommender systems via knowledge graph based semantic fusion. In: KDD. pp. 1006–1014 (2020)
40. Zhu, Z., Galkin, M., Zhang, Z., Tang, J.: Neural-symbolic models for logical queries on knowledge graphs. arXiv preprint arXiv:2205.10128 (2022)

Supplementary Materials

We provide details to ensure reproductivity with Algorithm 1, Table 1, and Table 3. We also show additional experimental results in Table 2, Table 4, and Table 5 to further demonstrate the effectiveness of our proposed method for neural multi-hop logical query answering with concept-level answers.

Algorithm 1 The learning procedure of our method for LQAC.

Require: An ontological knowledge base $\mathcal{KB} = (\mathcal{T}, \{\mathcal{A}_{ee}, \mathcal{A}_{ec}\})$.
$\mathcal{E}$ denotes the set of entities;
$\mathcal{C}$ denotes the set of concept (names) ;
$\mathcal{R}$ denotes the set of relations;
Ensure: $\mathbf{E}_e$ denotes the instance embedding matrix;
$\mathbf{E}_c$ denotes the concept embedding matrix;
$\mathbf{E}_r$ denotes the relation embedding matrix;
Θ denotes the parameters of $h(\cdot)$;
Ω denotes the parameters of $f(\cdot)$.

// Start training.
Initialize $\mathbf{E}_e$, $\mathbf{E}_c$, $\mathbf{E}_r$, Θ and Ω.
for each training episode **do**
  // **Concept Retrieval**.
  for each query $[q](?)$ **do**
    Sample a concept-level answer $c^+ \in \mathcal{C}$ as a positive instance and m non-answer concepts $\{c_1^-, \cdots, c_m^-\}$ as negative instances;
    Represent c^+ as FS_{c^+} by Eq.(6);
    Represent q as FS_q by Eq.(8), (9), (10), and (11);
    Calculate S^+_{Con} by Eq.(12) and (13) given FS_{c^+} and FS_q;
    for each negative instance c_i^- **do**
      Represent c_i^- as $FS_{c_i^-}$ by Eq.(6);
      Calculate $S^-_{Con_i}$ by Eq.(12) and (13) given $FS_{c_i^-}$ and FS_q;
    end for
  end for
  // **Instance Retrieval**.
  for each query $[q](?)$ **do**
    Sample an instance-level answer $e^+ \in \mathcal{E}$ as a positive instance and m non-answer entities $\{e_1^-, \cdots, e_m^-\}$ as negative instances;
    $\mathbf{e}^+ \leftarrow$ Look up $\mathbf{E}_e$ by rows; $\mathbf{q} \leftarrow f(q; \Omega)$;
    Calculate S^+_{Ent} by Eq.(14), (15), (16), (17) given $\mathbf{e}^+$ and $\mathbf{q}$;
    for each negative instance e_i^- **do**
      $\mathbf{e}_i^- \leftarrow$ Look up $\mathbf{E}_e$ by rows;
      Calculate $S^-_{Ent_i}$ by Eq.(14), (15), (16), (17) given $\mathbf{e}_i^-$ and $\mathbf{q}$;
    end for
  end for
  // **Subsumption**.
  for each pair of concepts (c_1, c_2) **do**

Sample m concepts $\{c_1^-, \cdots, c_m^-\}$ as negative instances;
$\mathbf{c}_1$, $\mathbf{c}_2 \leftarrow$ Look up $\mathbf{E}_c$ by rows;
Calculate S_{Sub}^+ by Eq.(18) given $\mathbf{c}_1$ and $\mathbf{c}_2$;
for each negative instance c_i^- **do**
$\mathbf{c}_i^- \leftarrow$ Look up $\mathbf{E}_c$ by rows;
Calculate $S_{Sub_i}^-$ by Eq.(18) given $(\mathbf{c}_1, \mathbf{c}_i^-)$ or $(\mathbf{c}_i^-, \mathbf{c}_2)$ with equal probability;
end for
end for
// **Instantiation**.
for each pair of concept and instance (c, e) **do**
Sample $\frac{m}{2}$ negative concepts $\{c_1^-, \cdots, c_{\frac{m}{2}}^-\}$;
Sample $\frac{m}{2}$ negative entities $\{e_{\frac{m}{2}+1}^-, \cdots, e_m^-\}$;
$\mathbf{c} \leftarrow$ Look up $\mathbf{E}_c$ by rows;
$\mathbf{e} \leftarrow$ Look up $\mathbf{E}_e$ by rows;
Calculate S_{Ins}^+ by Eq.(19) given $\mathbf{c}$ and $\mathbf{e}$;
for each negative concept c_i^- **do**
$\mathbf{c}_i^- \leftarrow$ Look up $\mathbf{E}_c$ by rows;
Calculate $S_{Ins_i}^-$ by Eq.(19) given $(\mathbf{c}_i^-, \mathbf{e})$;
end for
for each negative instance e_i^- **do**
$\mathbf{e}_i^- \leftarrow$ Look up $\mathbf{E}_e$ by rows;
Calculate $S_{Ins_i}^-$ by Eq.(19) given $(\mathbf{c}, \mathbf{e}_i^-)$;
end for
end for
Calculate $\mathcal{L}$ by Eq.(20);
Update $\mathbf{E}_e \leftarrow \partial\mathcal{L}/\partial\mathbf{E}_e$;
Update $\mathbf{E}_c \leftarrow \partial\mathcal{L}/\partial\mathbf{E}_c$;
Update $\mathbf{E}_r \leftarrow \partial\mathcal{L}/\partial\mathbf{E}_r$;
Update $\Theta \leftarrow \partial\mathcal{L}/\partial\Theta$;
Update $\Omega \leftarrow \partial\mathcal{L}/\partial\Omega$;
end for
return updated $\mathbf{E}_e$, $\mathbf{E}_c$, $\mathbf{E}_r$, Θ and Ω.

Table 1. Statistics of instance-level and concept-level candidate answers, and the included relations in the datasets.

Dataset	$\lvert\mathcal{E}\rvert = \lvert FS\rvert$	$\lvert\mathcal{C}\rvert$	$\lvert\mathcal{R}\rvert$
YAGO4	32,465	8,382	75
DBpedia	28,824	981	327
GO	5,586	45,151	3

Table 2. Ablation Study on Subsumptions and Instantiation on DBpedia. The best Hit@3 results are in boldface.

Concept	1p	2p	3p	2i	3i	pi	ip	2u	up	avg
LQAC w/o CC	61.4	73.8	71.8	22.9	20.1	28.8	21.6	**67.5**	62.6	47.8
LQAC w/o EC	58.9	69.7	68.0	42.8	39.6	41.2	22.0	66.4	62.9	52.4
LQAC	**62.4**	**83.7**	**83.6**	**50.9**	**43.7**	**45.0**	**29.9**	67.2	**67.3**	**59.3**
Instance	1p	2p	3p	2i	3i	pi	ip	2u	up	avg
LQAC w/o CC	20.5	22.8	23.0	18.8	20.6	14.7	25.6	10.4	18.4	19.4
LQAC w/o EC	19.4	19.1	20.5	19.1	20.1	15.0	25.7	9.1	20.3	18.7
LQAC	**34.6**	**28.0**	**29.0**	**44.6**	**54.8**	**21.4**	**40.6**	**17.8**	**23.0**	**32.6**

Table 3. The number of query-answer pairs in the split of the training, validation, and test set of the datasets. **Con**, **Ent**, **Sub**, **Ins**, and **NLR** correspond to statistics of the samples for Concept Retrieval, Instance Retrieval, Concept Subsumption, Concept Instantiation, and baseline LQA methods. For *other* query types (-o) in Fig. 2 of the paper, the statistics are the same for each of them.

Dataset	*Partition*	**Con**-1p	**Con**-o	**Ent**-1p	**Ent**-o	**Sub**	**Ins**	**LQA**-1p	**LQA**-o
YAGO4	#Training	189,338	10,000	101,417	10,000	16,644	83,291	184,708	10,000
	#Validation	1,000	1,000	1,000	1,000	0	0	1,000	1,000
	#Test	1,000	1,000	1,000	1,000	0	0	1,000	1,000
DBpedia	#Training	473,924	10,000	136,821	10,000	2,582	225,436	362,257	10,000
	#Validation	1,000	1,000	1,000	1,000	0	0	1,000	1,000
	#Test	1,000	1,000	1,000	1,000	0	0	1,000	1,000
GO	#Training	757,825	5,000	204,299	5,000	77,177	54,052	258,351	5,000
	#Validation	1,000	1,000	1,000	1,000	0	0	1,000	1,000
	#Test	1,000	1,000	1,000	1,000	0	0	1,000	1,000

Table 4. MRR of concept and instance retrieval. The best results are in boldface.

		Concept-level Answers									
		1p	2p	3p	2i	3i	pi	ip	2u	up	avg
YAGO4	GQE	35.3	49.5	33.7	51.3	43.9	11.8	9.1	14.4	6.6	28.4
	Q2B	37.3	53.4	59.6	55.0	47.6	2.1	1.6	1.7	1.1	28.8
	BetaE	39.0	57.0	58.9	52.9	45.8	10.4	8.9	2.7	6.5	31.3
	FuzzQE	34.1	50.3	44.4	52.9	44.0	13.2	11.7	19.0	8.3	30.9
	LQAC	**51.3**	**76.4**	**82.7**	**55.9**	**53.9**	**51.3**	**48.9**	**54.3**	**45.9**	**57.8**
DBpedia	GQE	27.1	35.5	32.5	30.5	32.0	14.0	14.7	9.8	11.8	23.1
	Q2B	26.4	35.7	32.6	30.4	29.9	13.5	14.4	10.3	11.2	22.7
	BetaE	30.4	38.7	40.0	32.9	34.2	14.8	11.4	5.6	9.0	24.1
	FuzzQE	26.7	34.7	32.1	28.1	28.4	16.9	16.5	13.1	14.6	23.5
	LQAC	**55.0**	**80.8**	**80.7**	**42.9**	**36.7**	**42.0**	**28.5**	**63.4**	**64.9**	**55.0**
		instance-level Answers									
		1p	2p	3p	2i	3i	pi	ip	2u	up	avg
YAGO4	GQE	25.8	15.8	4.6	26.4	28.4	21.7	18.1	9.6	18.2	18.7
	Q2B	24.5	17.2	6.8	25.7	28.8	24.5	20.2	8.6	17.7	19.3
	BetaE	28.2	19.7	9.5	30.1	33.4	28.3	22.0	10.6	20.7	22.5
	FuzzQE	26.4	15.6	4.0	26.8	28.3	21.2	18.0	10.4	18.1	18.8
	LQAC	**34.9**	**23.8**	**15.1**	**50.9**	**61.6**	**31.6**	**35.3**	**13.9**	**21.3**	**32.0**
DBpedia	GQE	19.6	16.1	18.2	31.4	38.2	16.8	28.5	10.1	18.0	21.9
	Q2B	16.3	13.7	15.4	22.6	28.5	18.1	22.8	7.4	12.9	17.5
	BetaE	20.2	20.1	19.3	25.4	29.7	**24.2**	25.2	12.3	**24.2**	22.3
	FuzzQE	15.1	14.1	16.3	22.3	29.1	27.6	23.1	6.8	13.7	18.7
	LQAC	**28.8**	**24.5**	**24.4**	**38.4**	**46.3**	20.1	**33.6**	**14.0**	20.6	**27.9**

Table 5. Hit@10 results of concept and instance retrieval on the biomedical GO dataset. The best results are in boldface.

		1p	2p	3p	2i	3i	pi	ip	2u	up	avg
Concept	GQE	0.020	0.241	0.195	0.279	0.266	0.033	0.030	0.011	0.015	0.121
	Q2B	0.019	0.238	0.204	0.274	0.268	0.019	0.026	0.009	0.013	0.119
	BetaE	0.021	0.238	0.200	0.278	0.275	0.033	0.033	0.008	0.014	0.122
	FuzzQE	0.020	0.226	0.203	**0.292**	**0.282**	0.029	0.025	0.016	0.013	0.123
	LQAC	**0.093**	**0.755**	**0.889**	0.205	0.211	**0.217**	**0.229**	**0.619**	**0.860**	**0.453**
Instance	GQE	0.405	0.546	0.703	0.262	0.226	0.265	0.441	0.337	0.603	0.421
	Q2B	0.419	**0.560**	0.712	0.281	0.245	0.264	0.447	**0.354**	**0.615**	0.433
	BetaE	0.405	0.548	0.705	0.263	0.232	0.273	0.441	0.338	0.608	0.424
	FuzzQE	0.403	0.537	**0.719**	0.332	0.290	0.304	0.464	0.329	0.605	0.443
	LQAC	**0.553**	0.549	0.701	**0.573**	**0.616**	**0.511**	**0.494**	0.342	0.419	**0.529**