LitCQD: Multi-Hop Reasoning in Incomplete Knowledge Graphs with Numeric Literals

Caglar Demir[0000−0001−8970−3850], Michel Wiebesiek, Renzhong Lu, Axel-Cyrille Ngonga Ngomo[0000−0001−7112−3516], and Stefan Heindorf[0000−0002−4525−6865] (✉)

Paderborn University, Germany
{caglar.demir@, renzhong@mail., axel.ngonga@, heindorf@}upb.de, michel.wiebesiek@mailbox.org

Abstract. Most real-world knowledge graphs, including Wikidata, DBpedia, and Yago are incomplete. Answering queries on such incomplete graphs is an important, but challenging problem. Recently, a number of approaches, including complex query decomposition (CQD), have been proposed to answer complex, multi-hop queries with conjunctions and disjunctions on such graphs. However, these approaches only consider graphs consisting of entities and relations, neglecting literal values. In this paper, we propose LitCQD—an approach to answer complex, multi-hop queries where both the query and the knowledge graph can contain numeric literal values: LitCQD can answer queries having numerical answers or having entity answers satisfying numerical constraints. For example, it allows to query (1) persons living in New York having a certain age, and (2) the average age of persons living in New York. We evaluate LitCQD on query types with and without literal values. To evaluate LitCQD, we generate complex, multi-hop queries and their expected answers on a version of the FB15k-237 dataset that was extended by literal values.

1 Introduction

Knowledge Graphs (KGs) such as Wikidata [30], DBpedia [3], and YAGO [25] have been of increasing interest in both academia and industry, e.g., for major question answering systems [1, 9, 27] and for intelligent assistants such as Amazon Alexa, Siri, and Google Now. Natural language questions on such KGs are typically answered by translating them into subsets of First-Order Logic (FOL) involving conjunctions ($\wedge$), disjunctions ($\vee$), and existential quantification ($\exists$) of multi-hop path expressions in the KGs. However, this approach to modeling queries has an important intrinsic flaw: Almost all real-world KGs are incomplete [8, 10, 20]. Traditional symbolic models, which rely on sub-graph matching, are unable to infer missing information on such incomplete KGs [12]. Hence, they often return empty answer sets to queries that can be answered by predicting missing information. Hence, several approaches (e.g., GQE [12], Query2Box [22], and CQD [2]) have recently been proposed that can query incomplete KGs by performing neural reasoning over Knowledge Graph Embeddings (KGEs).

However, all the aforementioned models operate solely on KGs consisting of *entities and relations* and none of them supports KGs with *literal values* such as the age of a person, the height of a building, or the population of a city. Taking literal values into account, however, has been shown to improve predictive performance in many tasks [13, 18].

In this paper, we remedy this drawback and propose LitCQD, a neural reasoning approach that can answer queries involving *numerical literal values* over incomplete KGs. LitCQD extends CQD by combining a KGE model (e.g., ComplEx-N3 [19]) that predicts missing entities/relations with a literal KGE model (e.g., TransEA [31]) able to predict missing numerical literal values. Therewith, LitCQD can mitigate missing entities/relations as well as missing numerical values to answer various types of queries. Moreover, we *increase the expressiveness of queries* that can be answered on KGs with literal values by allowing queries (1) to contain filter restrictions involving literals and (2) to ask for predictions of numeric values (see Example 1).

Example 1. The query "*Who ($P_?$) is married to somebody (P) younger than 25?*" with a filter restriction "younger than 25" can be rewritten as $P_?.\exists P, C : \text{hasAge}(P, C) \wedge \text{lt}(C, 25) \wedge \text{married}(P, P_?)$.

To answer this query, we predict the age of all persons P in the knowledge graph and check whether the condition "less than 25" is fulfilled. Then, all persons $P_?$ married to persons P are returned.

To evaluate filter expressions such as "less than 25" on incomplete knowledge graphs, we introduce continuous attribute filter functions (Section 4.1, Equations 8–10) and improve them by introducing attribute existence checks (Equations 11–12). We predict attribute values for a subset of entities that are obtained via beam search with an attribute predictor (Section 4.2).

In our experiments (Section 5), we use a similar setup to Arakelyan et al. [2], García-Durán and Niepert [11], Hamilton et al. [12] and use the FB15k-237 dataset augmented with literals [11]. However, as previous work did not contain queries with literal values, we generate such queries and their expected answers. Our experiments suggest that LitCQD can effectively answer various types of queries involving literal values, which was not possible before (Tables 3, 4). Moreover, our results show that including literal values during the training process improves the query answering performance even on standard queries in our benchmark (Table 2). Our contributions can be summarized as follows:

- *Filter restrictions with literals:* We propose an approach that can answer multi-hop queries where numeric literals are used to filter valid answers (e.g., "return entities whose age is less than 25").
- *Prediction of literal values:* We propose an approach that can *predict the numeric values* of literals (e.g., "return mean age of married people').
- *Benchmark construction:* We generate multi-hop queries *with numeric literals* and their expected answers.
- *Embeddings with literals:* We show that using knowledge graph embeddings that support literal values even yields better results for traditional queries without literal values.

2 Background and Preliminaries

In this section, we introduce knowledge graphs without literals and queries on them, before introducing our approach with literals in Section 4.

2.1 Knowledge Graph without Literals

A knowledge graph (KG) without literals is defined as $\mathcal{G} = \{(h, r, t)\} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$, where $h, t \in \mathcal{E}$ denote entities and $r \in \mathcal{R}$ denotes a relation [12, 22]. $\mathcal{G}$ can be regarded as a FOL knowledge base, where a relation $r \in \mathcal{R}$ corresponds to a binary function $\hat{r} : \mathcal{E} \times \mathcal{E} \to \{1, 0\}$ and a triple (h, r, t) corresponds to an atomic formula $\alpha = \hat{r}(h, t)$ [2]. When it is clear from the context that $\hat{r}$ denotes a binary function, we may simply write r as in the following definitions.

2.2 Multihop Queries without Literals

Conjunctive Queries. A conjunctive graph query [2, 12, 22, 23] $q \in \mathcal{Q}(\mathcal{G})$ over $\mathcal{G}$ is defined as

$$q = E_? \,.\, \exists E_1, \ldots, E_m : \alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n, \tag{1}$$

where

- $\alpha_i = r(e, E)$, with $E \in \{E_?, E_1, \ldots, E_m\}$, $r \in \mathcal{R}$, $e \in \mathcal{E}$ or
- $\alpha_i = r(E, E')$, with $E, E' \in \{E_?, E_1, \ldots, E_m\}$, $E \neq E'$, $r \in \mathcal{R}$.

In the query, the target variable $E_?$ and the existentially quantified variables $E_1, \ldots, E_m$ are bound to subsets of *entities* $\mathcal{E}$. The entities bound to $E_?$ represent the answer nodes of the query. The conjunction $\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n$ consists of n atoms defined over relations $r \in \mathcal{R}$, anchor entities $e \in \mathcal{E}$ and variables $E, E' \in \{E_?, E_1, \ldots, E_m\}$.

Example 2. The question "*Which ($D_?$) drugs are to interact with (P) proteins associated with the diseases e_1 and e_2?*" can be represented as the query

$$q = D_?.\exists P : \mathit{assoc}(e_1, P) \wedge \mathit{assoc}(e_2, P) \wedge \mathit{interacts}(P, D_?), \tag{2}$$

where $D_?, P$ are bound to subsets of entities $\mathcal{E}$, $e_1, e_2 \in \mathcal{E}$ are anchor entities, and $\mathit{interacts}, \mathit{assoc} \in \mathcal{R}$ are relations.

The dependency graph of a query $q \in Q(\mathcal{G})$ is defined over its query edges α_1, α_2, $\ldots, \alpha_n$ with nodes being either anchor entities or variables [12]. Following Hamilton et al. [12] and Arakelyan et al. [2], we focus on queries whose dependency graph forms a Directed Acyclic Graph (DAG) with anchor entities being source nodes and the target variable being the unique sink node (such queries are called *valid* queries in previous work [2, 12]). Figure 2 (left) represents the dependency graph of the query in Equation (2). Note that for simplicity, we use the term of an entity in a KG interchangeably with a node in a dependency graph.

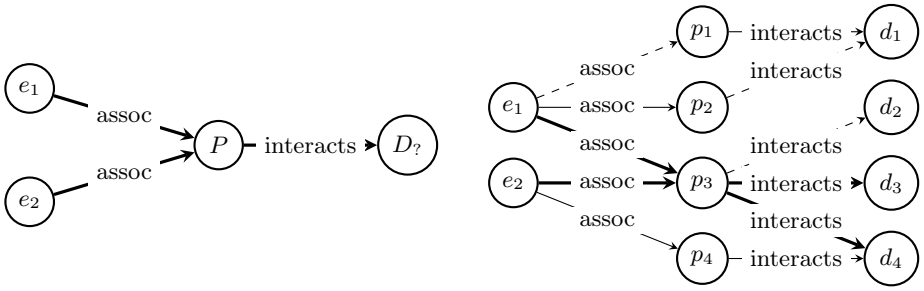

Fig. 1. Example query without literals (see Equation (2)). Dependency graph of query (left) and symbolic query answering on an incomplete graph (right). Solid bold lines represent paths leading to answer entities. Dashed lines represent missing triples.

The dependency graph of a query encodes the *computation graph* to obtain the answer set $[\![q]\!]$ via *projection* $\mathcal{P}$ and *intersection* $\mathcal{I}$ operators [22]. Starting from a set of anchor nodes (e.g., e_1, e_2), $[\![q]\!]$ is derived by iteratively applying $\mathcal{P}$ and/or $\mathcal{I}$ until the unique sink target node (e.g., $D_?$) is reached. Given a set of entities $S \subseteq \mathcal{E}$ and a relation $r \in \mathcal{R}$, the projection operator is defined as $\mathcal{P}(S, r) := \cup_{e \in S} \{x \in \mathcal{E} : \hat{r}(e, x) = 1\}$ where the binary function $\hat{r} : \mathcal{E} \times \mathcal{E} \to \{1, 0\}$ indicates whether the triple (e, r, x) exists in $\mathcal{G}$.[1] Given a set of entity sets $\{S_1, S_2, \ldots, S_n\}, S_i \subseteq \mathcal{E}$, the intersection operator $\mathcal{I}$ is defined as $\mathcal{I}(\{S_1, S_2, \ldots, S_n\}) := \cap_{i=1}^{n} S_i$. Therefore, the conjunctive query defined in Equation (2) can be answered via the computation

$$\mathcal{P}\Big(\mathcal{I}\big(\{\mathcal{P}(\{e_1\}, assoc), \mathcal{P}(\{e_2\}, assoc)\}\big), interacts\Big). \tag{3}$$

In the example of Figure 2 (right), a traditional, symbolic approach yields the answer set $[\![q]\!] = \{d_3, d_4\}$ although the complete answer set taking missing triples into account would be $[\![q]\!] = \{d_2, d_3, d_4\}$. The result is obtained as follows: Starting at the anchor entities e_1 and e_2, the entity p_3 is the only entity for which both $assoc(e_1, p_3)$ and $assoc(e_2, p_3)$ hold. Moving on from p_3, a traditional, symbolic approach can only reach the entities d_3, d_4 via the "interacts" relation, but not the entity d_2 because the edge $(p_3, \text{interacts}, d_2)$ is missing. Note that d_1 is not part of the answer set because both p_1 and p_2 are only associated with e_1.

Existential Positive First-order (EPFO) Queries. An EPFO query q in its Disjunctive Normal Form (DNF) is a disjunction of conjunctive queries [2, 22]:

$$q = E_? \,.\, \exists E_1, \ldots, E_m : (\alpha_1^1 \wedge \cdots \wedge \alpha_{n_1}^1) \vee \cdots \vee (\alpha_{n_1}^d \wedge \cdots \wedge \alpha_{n_d}^d), \tag{4}$$

where α_i^j are defined as above. Its dependency graph is a DAG having three types of directed edges: *projection*, *intersection*, and *union*; the union $\mathcal{U}$ of entity sets $S_1, S_2, \ldots, S_n \subseteq \mathcal{E}$ is $\mathcal{U}(\{S_1, S_2, \ldots, S_n\}) := \cup_{i=1}^{n} S_i$.

[1] When computing the ground truth answer on the complete graph, we check whether $(e, r, x) \in \mathcal{G}$ (see details on query generation below and in Hamilton et al. [12]). When performing neural reasoning, $\hat{r}$ is approximated with a link predictor yielding a score between 0 and 1.

3 Related Work

In this section, we overview the state of the art with regards to knowledge graph embeddings and neural query answering on incomplete knowledge graphs.

3.1 Knowledge Graph Embeddings and Literals

In the last decade, a plethora of knowledge graph embedding (KGE) models have been successfully applied to tackle various tasks, including link prediction, relation prediction, community detection, fact checking, and class expression learning [15–17, 20, 24, 29]. KGE research has mainly focused on learning embeddings for entities and relations tailored towards predicting missing entities/links, i.e., tackling single-hop queries [4, 6–8, 20, 26, 29, 33, 34]. Despite their effectiveness in tackling single-hop queries, KGE models cannot be directly applied to answer multi-hop queries because multi-hop query answering is a strict generalization [21]. Most KGE models do not incorporate literals (e.g., age of a person, height of a person, or date of birth), but there has been a growing interest in designing such models. For instance, Wu and Wang [32] propose TransEA by extending the translation loss used in TransE [5] by adding the attribute loss as a weighted regularization term. García-Durán and Niepert [11] propose KBLRN that is based on relation features, numerical literals, and a KGE model. Kristiadi et al. [18] propose LiteralE, which applies a non-linear parameterized function to merge entity embeddings with numerical literals. Thereby, LiteralE is computationally less demanding than KBLRN as it does not require any rule generation for relation features and is more expressive than TransEA as TransEA integrates the impact of literals linearly.

3.2 Neural Query Answering on Incomplete Knowledge Graphs

In recent years, significant progress has been made on querying incomplete KGs. Hamilton et al. [12] laid the foundations for multi-hop reasoning with graph query embeddings (GQE). Given a conjunctive query (e.g., Equation (2)), they learn continuous vector representations for queries, entities, and relations and answer queries by performing projection $\mathcal{P}$ and intersection $\mathcal{I}$ operations in the embedding vector space. Ren et al. [22] show that GQE cannot answer EPFO queries (see Equation (4)) since GQE does not model the union operator $\mathcal{U}$. Hence, they propose Query2Box that represents an EPFO query with a set of box embeddings, where one box embedding is constructed per conjunctive subquery. A query is answered by returning the entities whose minimal distance to one of the box embeddings is smallest.

All the aforementioned models learn query embeddings and answer queries via nearest neighbor search in the embedding space. However, learning embeddings for complex, multi-hop queries involving conjunctions and disjunctions can be computationally demanding. Towards this end, Arakelyan et al. [2] propose complex query decomposition (CQD). They answer EPFO queries by decomposing

them into single-hop subqueries and aggregate the scores of a pre-trained single-hop link predictor (e.g., ComplEx-N3). Scores are aggregated using a t-norm and t-conorm—continuous generalizations of the logical conjunction and disjunction [2, 14]. Their experiments suggest that CQD outperforms GQE and Query2Box; it generalizes well to complex query structures while requiring orders of magnitude less training data. Zhu et al. [35] highlight that CQD is the only interpretable model among the aforementioned models as it produces intermediate results. In this work, we extend CQD to answer multi-hop queries involving literals.

4 LitCQD: Multi-hop Reasoning with Literals

A knowledge graph with numeric literals (i.e., with scalar values), can be defined as $\mathcal{G}_A = \{(h, r, t)\} \subset (\mathcal{E} \times \mathcal{R} \times \mathcal{E}) \cup (\mathcal{E} \times \mathcal{A} \times \mathbb{R})$, where $\mathcal{R} \cap \mathcal{A} = \emptyset$ and $\mathcal{A}$ and $\mathbb{R}$ denote numeric attributes and real numbers, respectively [18]. The binary function $\hat{a} : \mathcal{E} \times \mathbb{R} \mapsto \{1, 0\}$ indicates whether an entity has attribute $a \in \mathcal{A}$ and we might just write a instead of $\hat{a}$ when this is clear from context. We categorize EPFO queries $q \in \mathcal{Q}(\mathcal{G}_A)$ involving literals depending on the type of their answer sets $[\![q]\!]$: In Section 4.1, we define queries with entities as answer set $[\![q]\!] \subseteq \mathcal{E}$; in Section 4.2, we define queries with a literal value as answer $[\![q]\!] \in \mathbb{R}$.

4.1 Multihop Queries with Literals and Entity Answers

An EPFO query q on a KG with numeric literals ($\mathcal{G}_\mathcal{A}$) can be defined as

$$q = E_? \,.\, \exists E_1, \ldots, E_m : (\alpha_1^1 \wedge \cdots \wedge \alpha_{n_1}^1) \vee \cdots \vee (\alpha_1^d \wedge \cdots \wedge \alpha_{n_d}^d), \tag{5}$$

where

- $\alpha_i^j = r(e, E)$, with $E \in \{E_?, E_1, \ldots, E_m\}$, $r \in \mathcal{R}$, $e \in \mathcal{E}$ or
- $\alpha_i^j = r(E, E')$, with $E, E' \in \{E_?, E_1, \ldots, E_m\}$, $E \neq E'$, $r \in \mathcal{R}$ or
- $\alpha_i^j = a(E, C) \wedge af(C, c)$, with $E \in \{E_?, E_1, \ldots, E_m\}$, $C \in \{C_1, \ldots, C_l\}$, $a \in \mathcal{A}$, $af \in \{\mathrm{lt}, \mathrm{gt}, \mathrm{eq}\}$, $c \in \mathbb{R}$.

In the query, the target variable $E_?$ and the variables $E_1, \ldots, E_m$ are bound to subsets of *entities* $\mathcal{E}$ and the variables $C_1, \ldots, C_l$ are bound to numeric values from $\mathbb{R}$. The binary function $r : \mathcal{E} \times \mathcal{E} \mapsto \{1, 0\}$ denotes whether a relation exists between the two entities, $a : \mathcal{E} \times \mathbb{R} \mapsto \{1, 0\}$ whether an attribution relation exists, and $af : \mathbb{R} \times \mathbb{R} \mapsto \{1, 0\}$ is one of the attribute filter conditions lt (*less-than*), gt (*greater-than*), or eq (*equal-to*). For example, $lt(20, 25)$ returns 1 because $20 \leq 25$. To approximately answer queries defined with Equation (5) and assuming an incomplete knowledge graph, we propose the following optimization problem:

$$\underset{E_?, E_1, \ldots, E_m}{\arg\max} \; \left(\alpha_1^1 \top \ldots \top \alpha_{n_1}^1\right) \perp \ldots \perp \left(\alpha_1^d \top \ldots \top \alpha_{n_d}^d\right) \tag{6}$$

where

- $\alpha_i^j = \phi_r(e, E)$, with $E \in \{E_?, E_1, \ldots, E_m\}$, $r \in \mathcal{R}$, $e \in \mathcal{E}$ or

Fig. 2. Example query with literals and entity answer (see Equation (7)). On the left, the query's dependency graph is shown and on the right, symbolic query answering on an incomplete graph with literal values. Bold lines represent paths leading to answer entities, dashed lines represent missing triples, solid existing triples.

- $\alpha_i^j = \phi_r(E, E')$, with $E, E' \in \{E_?, E_1, \ldots, E_m\}$, $E \neq E'$, $r \in \mathcal{R}$ or
- $\alpha_i^j = \phi_{af,a}(\phi_a(E), c)$, with $E \in \{E_?, E_1, \ldots, E_m\}$, $a \in \mathcal{A}$, $af \in \{\mathrm{lt}, \mathrm{gt}, \mathrm{eq}\}$, $c \in \mathbb{R}$,

and $\phi_r : \mathcal{E} \times \mathcal{E} \mapsto [0, 1]$ is a link predictor that predicts a *likelihood* of a link between two entities via a relation r. $\phi_a : \mathcal{E} \mapsto \mathbb{R}$ is an attribute predictor that predicts a *value* of an attribute a given an entity. An attribute filter predictor $\phi_{af,a} : \mathbb{R} \times \mathbb{R} \mapsto [0, 1]$ predicts a *likelihood* that the filter condition is met given the predicted attribute value $\hat{c} := \phi_a(\cdot)$ and the constant value $c \in \mathbb{R}$ specified in the query. All three predictors are derived from a KGE model as described below. A t-norm $\top : [0, 1] \times [0, 1] \mapsto [0, 1]$ is considered as a continuous generalization of the logical conjunction [2, 14]. Given a t-norm $\top$, the complementary t-conorm can be defined as $\bot(a, b) = 1 - \top(1 - a, 1 - b)$ [2]. Numerically, the *Gödel t-norm* $\top_{\min}(x, y) = \min\{x, y\}$, the *product t-norm* $\top_{\mathrm{prod}}(x, y) = x \cdot y$, or the *Łukasiewicz t-norm* $\top_{\mathrm{Luk}}(x, y) = \max\{0, x + y - 1\}$ can be used to aggregate predicted likelihoods to obtain a query score [2]. With this formulation, various questions involving numerical values can be asked on incomplete $\mathcal{G}_\mathcal{A}$. For example, the question "*Which entities are younger than 25?*" can be represented as

$$q = E_? \,.\, \exists C : \mathit{hasAge}(E_?, C) \wedge \mathit{lt}(C, 25). \tag{7}$$

The dependency graph of this query is visualized in Figure 2 (left). Let $S_?$ be the entities bound to variable $E_?$. Then the projection of $S_?$ with *hasAge* is performed by an attribute prediction model $\phi_{\mathit{hasAge}}(S_?) \in \mathbb{R}^{|E|}$ that predicts the value of the attribute a for each entity in $e \in E$. The answer set is obtained by filtering entities via ϕ_{lt}. A subgraph of $\mathcal{G}_\mathcal{A}$ satisfying this query is visualized in Figure 2 (right). While a symbolic approach only yields the answer set $[\![q]\!] = \{e_1\}$, our approach involving link predictors can identify the full answer set $[\![q]\!] = \{e_1, e_2\}$.

We solve the optimization problem in Equation (6) approximately with a variant of beam search by greedily searching for sets of entities $S_?, S_1, \ldots S_m$ substituting the variables $E_?, E_1, \ldots, E_m$ in a fashion akin to CQD [2]. In the example in Equation (7), given the *hasAge* attribute, attribute values $\hat{c} = \phi_{\mathit{hasAge}}(e) \in \mathbb{R}$ are predicted for all entities $e \in \mathcal{E}$. Next, likelihoods of fulfilling the filter condition "less than 25" can be inferred via $\phi_{\mathrm{lt}}(\hat{c}, 25)$. Finally, all entities are sorted by

their query scores in descending order and the top k entities are considered to be answers of q. It is important to note that LitCQD like CQD not only computes the final answer but also intermediate steps leading to this answer. In this sense, LitCQD can be considered an interpretable model.

Joint Training of Link and Attribute Predictors. Following Arakelyan et al. [2], we use ComplEx-N3 [19] as entity predictor $\phi_r(\cdot,\cdot)$. As attribute predictor $\phi_a(\cdot)$, we employ TransEA [31]. We jointly train the KGE models underlying both models. The link predictor ComplEx-N3 has previously been found to work well for multi-hop query answering [2] and to perform better than DistMult [2, 33]. In a pilot study, we also experimented with the attribute predictor MTKGNN [28]. Overall, it achieved similar performance to TransEA, but we decided to move forward with TransEA, because it slightly outperformed MTKGNN in terms of MRR and required less parameters. KBLRN [11] and LiteralE [18] only compute knowledge graph embeddings based on literal information, but they do not predict the value of attributes which is required in our framework.

Attribute Filter Function without Existence Check. The attribute filter function returns a score indicating the likelihood that the filter condition is met. First, we define a preliminary version $\phi'_{af,a}$ of the function, which does not check whether the attribute relation a actually exist for an entity. The function is defined case by case. For the *equal-to* condition, i.e., for $af = \text{eq}$, we define it as

$$\phi'_{\text{eq},a}(\hat{c}, c) := \frac{1}{\exp(|\hat{c} - c|/\sigma_a)}, \tag{8}$$

where $\hat{c} = \phi_a(e), e \in \mathcal{E}$, $c \in \mathbb{R}$ is a numeric literal (e.g., 25 in Figure 2, left) and σ_a denotes the standard deviation of $\mathcal{C}_a$ where $\mathcal{C}_a := \{c \in \mathbb{R} | \hat{a}(e, c) = 1, e \in \mathcal{E}\}$ are all literal values found on $\mathcal{G}_\mathcal{A}$ given an attribute a. With $\phi'_{\text{eq},a}(\hat{c}, c)$, we map the difference between the predicted attribute value $\hat{c}$ and the constant value $\hat{c}$ specified in the query into the unit interval $[0, 1]$. As the difference $\mid \hat{c} - c \mid$ approaches 0, $\phi_{\text{eq},a}(\hat{c}, c)$ approaches 1. The division by the standard deviation σ_a normalizes the difference $\mid \hat{c} - c \mid$. For the attribute filter function with *less-than* ($af = \text{lt}$), we define

$$\phi'_{\text{lt}}(\hat{c}, c) := \frac{1}{1 + \exp((\hat{c} - c)/\sigma_a)}. \tag{9}$$

As $\hat{c} - c \to -\infty$, $\phi_{\text{lt}}(\hat{c}, c) \to 1$. Following Equation (9), the attribute filter function with *greater-than* is defined as

$$\phi'_{\text{gt}}(\hat{c}, c) := 1 - \phi_{\text{lt}}(\hat{c}, c). \tag{10}$$

We also experimented with a version where the standard deviation σ_a was not computed per attribute but for all literal values independent of a, i.e., σ was computed for $\bigcup_{a \in \mathcal{A}} C_a$. We picked the latter variant as default for our LitCQD approach as it outperformed the former variant in our experiments.

Attribute Filter Function with Existence Check. The preliminary attribute filter function $\phi'_{af,a}$ assumes that the attribute relation a exists for each entity in the knowledge base which is clearly not the case. Hence, we employ a model $\phi_{\text{exists},a}(e)$ that scores the likelihood that the attribute relation a exists for entity e. Then the final attribute filter function $\phi_{af,a}$ is obtained by combining the attribute existence predictor $\phi_{\text{exist},a}(e)$ with the preliminary filter predictor $\phi'_{af,a}$:

$$\phi_{af,a}(\hat{c}, c) := \phi_{\text{exists},a}(e) \cdot \phi'_{af,a}(\hat{c}, c) \tag{11}$$

Technically, the attribute existence predictor is realized by adding a dummy entity e_{exists} to the knowledge base along with dummy edges $r_a(e, e_{\text{exists}})$ if entity e has an attribute relation a. Then, the existence of an attribute is predicted with the link predictor as

$$\phi_{\text{exists},a}(e) := \phi_{r_a}(e, e_{\text{exists}}) \tag{12}$$

Note that the dummy entity and the dummy relations are only added to the train set but not the validation or test set.

4.2 Multihop Queries with Literals and Literal Answers

Here, we define an EPFO query q on an incomplete $\mathcal{G}_{\mathcal{A}}$, whose answer $[\![q]\!] \in \mathbb{R}$ is a real number (instead of a subset of entities) as follows

$$q = \psi(C_?) \,.\, \exists E_?, E_1, \ldots, E_m : (\alpha_1^1 \wedge \cdots \wedge \alpha_{n_1}^1) \vee \cdots \vee (\alpha_1^d \wedge \cdots \wedge \alpha_{n_d}^d), \tag{13}$$

where $\psi : 2^{\mathbb{R}} \mapsto \mathbb{R}$ is a permutation-invariant aggregation function and

- $\alpha_i^j = r(e, E)$, with $E \in \{E_?, E_1, \ldots, E_m\}$, $r \in \mathcal{R}$, $e \in \mathcal{E}$ or
- $\alpha_i^j = r(E, E')$, with $E, E' \in \{E_?, E_1, \ldots, E_m\}$, $E \neq E'$, $r \in \mathcal{R}$ or
- $\alpha_i^j = a(E, C) \wedge af(C, c)$, with $E \in \{E_?, E_1, \ldots, E_m\}$, $C \in \{C_?, C_1, \ldots, C_l\}$ $a \in \mathcal{A}$, $af \in \{\text{lt}, \text{gt}, \text{eq}\}$, $c \in \mathbb{R}$.

Variable bindings $S_?, S_1, \ldots, S_m$ for $E_?, E_1, \ldots, E_m$ are obtained via the same optimization problem as in Section 4.1. Then the set of values $C_?$ can be computed by applying the attribute value predictor ϕ_a on the entities in $S_?$.

With this formulation, various questions can be asked on incomplete $\mathcal{G}_{\mathcal{A}}$. For instance, the question "*What is the average age of Turing Award (TA) winners?*" can be answered by computing the mean of a set of numeric literals $C_?$:

$$\text{mean}(C_?).\exists E_? : \text{winner}(E_?, \text{turingAward}) \wedge \text{hasAge}(E_?, C_?) \tag{14}$$

Similarly, the question "*What is the minimum age of Turing Award (TA) winners?*" can be answered by computing the minimum of a set of numeric literals $C_?$:

$$\min(C_?).\exists E_? : \text{winner}(E_?, \text{turingAward}) \wedge \text{hasAge}(E_?, C_?) \tag{15}$$

Figure 3 visualizes a subgraph of $\mathcal{G}_{\mathcal{A}}$ to answer q defined in Equation (14). Having found the binding $S_? = \{e_1, e_2\}$ for $E_?$, to each $e \in S_?$, we apply the attribute predictor $\phi_{\text{winner}}(e, \text{turingAward})$ and average the results, yielding the answer $[\![q]\!] = \frac{22+24}{2} = 23$—in contrast to $[\![q]\!] = 22$ by a symbolic approach that neglects missing information.

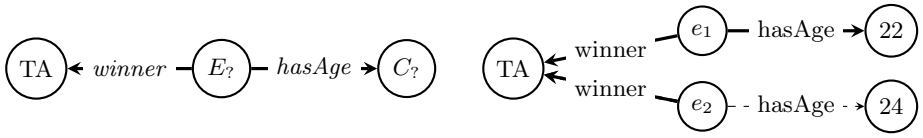

Fig. 3. Example of a query predicting attribute values (see Equation (14)). On the left, the dependency graph of the query is shown, on the right a subgraph to answer q. Dashed lines represent missing information. Bold lines represent paths leading to the symbolic answer $[\![q]\!] = 22$.

5 Experimental Results

After a brief description of the experimental setup, we evaluate the performance of LitCQD on the query types shown in Table 1. Finally, we show the answers of LitCQD for an example query. Our code is publicly available.[2]

5.1 Experimental Setup

Dataset and Query Generation. We use the FB15k-237 dataset augmented with attributes as done by García-Durán and Niepert [11]. The dataset contains 12,390 entities, 237 entity relations, 115 attribute relations, and 29,229 triples. Queries and their expected answers are generated as by Hamilton et al. [12]. The newly introduced attribute filter conditions (af) are handled as follows: When checking for *equality* ($af(C, c) = eq(C, c)$), we consider all entities whose attribute value lies within one standard deviation from c as correct where the standard deviation is computed per attribute relation a; when checking the *less-than* or *greater-than* criterion, the criterion is checked exactly, i.e., all entities with attribute value "$\leq c$" or "$\geq c$" are considered correct. In a preprocessing step, we normalize all values of an attribute to the unit interval via min-max scaling. Table 1 gives an overview of the newly introduced query types along with previous query types.

Hyperparameters. Per query type, we tried 16 different configurations on the validation set and chose the best before applying the model to the test set. As our framework is derived from the CQD framework, it allows two different optimization algorithms: Continuous optimization (Co), Combinatorial optimization (Beam); two t-norms: Gödel (min), product (prod); and 7 different beam sizes $k \in \{2^2, 2^3, \ldots, 2^8\}$ for the combinatorial optimization algorithm. Each optimization algorithm is computed for both of the t-norms resulting in 2 configurations using the continuous optimization algorithm and 14 using the combinatorial optimization algorithm as every beam size is evaluated for both t-norms.

5.2 Multihop Queries without Literals

In a first experiment (Table 2), we compare the performance of our approach LitCQD to CQD [2] and Query2Box [22] on multihop entity queries without

[2] https://github.com/dice-group/LitCQD

Table 1. Overview of different query types. Entity queries without literals were proposed by Ren et al. [22]. Entity queries with literals and queries with literal answers are newly proposed in this paper.

Multihop queries without literals	
1p	$E_? \,.\, r(e, E_?)$
2p	$E_? \,.\, \exists E_1 : r_1(e, E_1) \wedge r_2(E_1, E_?)$
3p	$E_? \,.\, \exists E_1 E_2 . r_1(e, E_1) \wedge r_2(E_1, E_2) \wedge r_3(E_2, E_?)$
2i	$E_? \,.\, r_1(e_1, E_?) \wedge r_2(e_2, E_?)$
3i	$E_? \,.\, r_1(e_1, E_?) \wedge r_2(e_2, E_?) \wedge r_3(e_3, E_?)$
ip	$E_? \,.\, \exists E_1 . r_1(e_1, E_1) \wedge r_2(e_2, E_1) \wedge r_3(E_1, E_?)$
pi	$E_? \,.\, \exists E_1 . r_1(e_1, E_1) \wedge r_2(E_1, E_?) \wedge r_3(e_2, E_?)$
2u	$E_? \,.\, r_1(e_1, E_?) \vee r_2(e_2, E_?)$
up	$E_? \,.\, \exists E_1 . [r_1(e_1, E_1) \vee r_2(e_2, E_1)] \wedge r_3(E_1, E_?)$
Multihop queries with literals and entity answers	
ai	$E_? \,.\, \exists C_1 . a(E_?, C_1) \wedge af(C_1, c)$
2ai	$E_? \,.\, \exists C_1 C_2 . a_1(E_?, C_1) \wedge af_1(C_1, c_1) \wedge a_2(E_?, C_2) \wedge af_2(C_2, c_2)$
pai	$E_? \,.\, \exists C_1 . r(e, E_?) \wedge a(E_?, C_1) \wedge af(C_1, c_1)$
aip	$E_? \,.\, \exists E_1 C_1 . a(E_1, C_1) \wedge af(C_1, c_1) \wedge r(E_1, E_?)$
au	$E_? \,.\, \exists C_1 C_2 . a_1(E_?, C_1) \wedge af_1(C_1, c_1) \vee a_2(E_?, C_2) \wedge af_2(C_2, c_2)$
Multihop queries with literals and literal answers	
1ap	$\text{mean}(C_?) \,.\, a(e, C_?)$
2ap	$\text{mean}(C_?) \,.\, \exists E_1 . r(e, E_1) \wedge a(E_1, C_?)$
3ap	$\text{mean}(C_?) \,.\, \exists E_1 E_2 . r_1(e, E_1) \wedge r_2(E_1, E_2) \wedge a(E_2, C_?)$

literals, which can be answered by all three models—in contrast to more expressive queries that can only be answered by LitCQD. While CQD does not utilize literal information and employs the vanilla ComplEx-N3 [19] model, LitCQD employs a model combining ComplEx-N3 [19] with TransEA [31]. Table 2 shows that LitCQD clearly outperforms CQD and Query2Box in terms of the mean reciprocal rank (MRR), and Hits@k for $k \in \{1, 3, 10\}$.

5.3 Multihop Queries with Literals and Entity Answers

Table 3 shows the evaluation results for the new query types with filter restrictions introduced in Section 4.1 (second block in Table 1). For the simple ai query, each filtering expression (*less-than, equals, greater-than*) is evaluated separately; the other query types contain all three filtering expressions. Except for aip queries, all query types with literals can be answered with a performance of at least 0.256 which is comparable to query types without literals (cf. Table 2).

Moreover, we experimented with different variants of our model and performed an ablation study. As described in Section 4.1, Equation (11), the attribute filter predictor $\phi_{af,a}$ is a product of $\phi_{\text{exists},a}(e)$ and $\phi'_{af,a}(\hat{c}, c)$. We performed three experiments, where we replaced each/both of the two scoring functions by the

Table 2. Query answering results for multihop queries without literals. Results were computed for test queries over the FB15k-237 dataset and evaluated in terms of mean reciprocal rank (MRR) and Hits@k for $k \in \{1, 3, 10\}$.

Method	Average	1p	2p	3p	2i	3i	ip	pi	2u	up
MRR										
Query2Box	0.213	0.403	0.198	0.134	0.238	0.332	0.107	0.158	0.195	0.153
CQD	0.295	0.454	0.275	0.197	0.339	0.457	0.188	0.267	0.261	0.214
LitCQD (ours)	**0.301**	**0.457**	**0.285**	**0.202**	**0.350**	**0.466**	**0.193**	**0.274**	**0.266**	**0.215**
HITS@1										
Query2Box	0.124	0.293	0.120	0.071	0.124	0.202	0.056	0.083	0.094	0.079
CQD	0.211	0.354	0.198	0.137	0.235	0.354	**0.130**	0.186	0.165	**0.137**
LitCQD (ours)	**0.215**	**0.355**	**0.206**	**0.141**	**0.245**	**0.365**	0.129	**0.193**	**0.168**	0.135
HITS@3										
Query2Box	0.240	0.453	0.214	0.142	0.277	0.399	0.111	0.176	0.226	0.161
CQD	0.322	0.498	0.297	0.208	0.380	0.508	0.195	0.290	0.287	0.230
LitCQD (ours)	**0.330**	**0.506**	**0.309**	**0.214**	**0.395**	**0.517**	**0.204**	**0.296**	**0.295**	**0.235**
HITS@10										
Query2Box	0.390	0.623	0.356	0.259	0.472	0.580	0.203	0.303	0.405	0.303
CQD	0.463	0.656	0.422	0.312	0.551	0.656	0.305	0.425	0.465	0.370
LitCQD (ours)	**0.472**	**0.660**	**0.439**	**0.323**	**0.561**	**0.663**	**0.315**	**0.434**	**0.475**	**0.379**

constant value 1. Table 3 shows that both components are crucial and the performance drops drastically if one of them is removed.

Moreover, the Equation (8) and Equation (9) normalize the difference $\hat{c} - c$ by dividing by the standard deviation. Per default (first line), LitCQD employs the universal standard deviation across all attributes of the knowledge base, i.e., the standard deviation σ of $\bigcup_{a \in \mathcal{A}} C_a$. As an alternative, we computed attribute-specific standard deviations σ_a per C_a. Table 3 (last line) shows that using an attribute-specific standard deviation instead of a universal standard deviation leads to a lower performance on five query types, to the same performance on one query type, and to a higher performance on only one query type.

5.4 Multihop Queries with Literals and Literal Answers

Table 4 evaluates the performance of queries asking for literal answers. The predicted numeric values are compared to the actual numeric values in terms of mean absolute error (MAE) and mean squared error (MSE). Interestingly, we notice that the mean absolute error for the 2ap queries is lower than for 1ap queries. This can be explained by the fact that for 1ap queries a single prediction of an attribute value is made whereas 2ap queries average multiple predictions (the number of the beam width). For 3ap queries, the mean absolute error increases again because the relation path becomes longer and errors accumulate.

Table 3. Query answering results for multihop queries with literals and entity answers. Our best-performing model LitCQD is compared to variations thereof. Results were computed for test queries over the FB15k-237 dataset and evaluated in terms of Hits@10.

Method	**ai-lt**	**ai-eq**	**ai-gt**	**2ai**	**aip**	**pai**	**au**
LitCQD	**0.405**	**0.361**	0.317	**0.336**	**0.182**	0.463	**0.256**
- w/o attribute filter predictor	0.280	0.005	0.237	0.148	0.124	0.421	0.054
- w/o attribute existence predictor	0.206	0.137	0.128	0.104	0.167	**0.470**	0.120
- w/o both	0.015	0.001	0.003	0.001	0.051	0.412	0.003
- with attribute-specific stdev	**0.405**	0.232	**0.329**	0.216	0.174	0.320	0.212

Table 4. Query answering results for multihop queries with literals and literal answers. Results were compute for test queries over the FB15k-237 dataset and evaluated in terms of mean absolute error (MAE) and mean squared error (MSE).

Method	**1ap**		**2ap**		**3ap**	
	MAE	MSE	MAE	MSE	MAE	MSE
LitCQD	0.050	0.011	0.034	0.005	0.041	0.007
Mean Predictor	0.338	0.141	0.344	0.140	0.359	0.151

As a simple baseline, we also report the results of the model that always predicts the mean value $\frac{1}{|\mathcal{C}_a|}\sum_{c\in\mathcal{C}_a} c$ of the attribute a in the whole knowledge graph (mean predictor in the table).

5.5 Example Query and Answers

As an illustration of the model's query-answering ability, consider the query "What are musicians from the USA born before 1972?" and its logical representation

$$E_? \,.\, \exists C_1./\text{music/artist/origin}(\text{USA}, E_?) \wedge \\ /\text{people/person/date\_of\_birth}(E_?, C_1) \wedge lt(C_1, 1972). \tag{16}$$

Table 5 lists the top 10 returned answers. Although the model confuses the band *Funkadelic* as musicians with a date of birth, the model is able to produce a reasonable ranking of entities. Out of these 10 entities, the entity *Robert E. Lee* receives the highest score of 0.95 for the attribute portion of the query. The model is confident that the entity has the attribute `/people/person/date_of_birth` and that its value is less than 1972. The entities *Dio*, *Rob Thomas*, and *Donna Summer* only receive a score of 0.39 for the attribute portion of the query because their predicted values are closer to the threshold of 1972. The model is more certain that the connection `/music/artist/origin`, *USA* exists for *John Denver* compared to *Robert E. Lee*. While Linus Pauling is a chemist rather than a musician and the dataset does not contain the connection `/music/artist/origin`, *USA*, the learned embeddings implicitly encode that *Linus Pauling* has another connection to the entity *USA* via the `/people/person/nationality` relation.

Table 5. Ranking of LitCQD's top 10 answers to the query in Equation (16) including their expected and predicted attribute value for `date_of_birth`. The star (*) indicates attribute values unseen during training and the double star (**) refers to attribute values not part of the dataset at all. The dash (–) indicates that an entity does not have a date of birth.

Rank	Answer	Expected Attr.	Predicted Attr.
1	Linus Pauling	1901.17	1900.06
2	John Denver	1944.00	1941.52
3	Funkadelic	–	1925.21
4	Friedrich Hayek	1899.42	1900.04
5	Robert E. Lee	1807.08	1794.49
6	Dio	1942**	1935.59
7	Marvin March	1930.42	1922.07
8	Rob Thomas	1972*	1943.72
9	Ezra Pound	1885.83	1882.00
10	Donna Summer	1949.00	1948.55

6 Conclusion

In this paper, we propose LitCQD, a novel approach to answer multihop queries on incomplete knowledge graphs with numeric literals. Our approach allows answering queries that could not be answered before, e.g., queries involving literal filter restrictions and queries predicting the value of numeric literals. Moreover, our experiments suggest that even the performance of answering multihop queries that could be answered before improves as the underlying knowledge graph embedding models now take literal information into account. This is an important finding as most real-world knowledge graphs contain millions of entities with numerical attributes. In future work, we plan to further increase the expressiveness of our queries, e.g., by supporting string literals, Boolean literals, and datetime literals.

Acknowledgements This work has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 860801, the Horizon Europe research and innovation programme under the Marie Skłodowska-Curie grant agreement No 101073307, and the Horizon Europe research and innovation programme under grant agreement No 101070305. This work has also been supported by the Ministry of Culture and Science of North Rhine-Westphalia (MKW NRW) within the project SAIL under the grant No NW21-059D and by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation): TRR 318/1 2021 – 438445824.

Bibliography

[1] Adolphs, P., Theobald, M., Schäfer, U., Uszkoreit, H., Weikum, G.: YAGO-QA: answering questions by structured knowledge queries. In: ICSC, pp. 158–161, IEEE Computer Society (2011)

[2] Arakelyan, E., Daza, D., Minervini, P., Cochez, M.: Complex query answering with neural link predictors. In: ICLR, OpenReview.net (2021)
[3] Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.G.: DBpedia: A nucleus for a web of open data. In: ISWC/ASWC, Lecture Notes in Computer Science, vol. 4825, pp. 722–735, Springer (2007)
[4] Balazevic, I., Allen, C., Hospedales, T.M.: TuckER: tensor factorization for knowledge graph completion. In: EMNLP/IJCNLP (1), pp. 5184–5193, Association for Computational Linguistics (2019)
[5] Bordes, A., Usunier, N., García-Durán, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: NIPS, pp. 2787–2795 (2013)
[6] Demir, C., Moussallem, D., Heindorf, S., Ngonga Ngomo, A.: Convolutional hypercomplex embeddings for link prediction. In: ACML, Proceedings of Machine Learning Research, vol. 157, pp. 656–671, PMLR (2021)
[7] Demir, C., Ngonga Ngomo, A.: Convolutional complex knowledge graph embeddings. In: ESWC, Lecture Notes in Computer Science, vol. 12731, pp. 409–424, Springer (2021)
[8] Dettmers, T., Minervini, P., Stenetorp, P., Riedel, S.: Convolutional 2d knowledge graph embeddings. In: AAAI, pp. 1811–1818, AAAI Press (2018)
[9] Diefenbach, D., Tanon, T.P., Singh, K.D., Maret, P.: Question answering benchmarks for Wikidata. In: ISWC (Posters, Demos & Industry Tracks), CEUR Workshop Proceedings, vol. 1963, CEUR-WS.org (2017)
[10] Färber, M., Bartscherer, F., Menne, C., Rettinger, A.: Linked data quality of DBpedia, Freebase, OpenCyc, Wikidata, and YAGO. Semantic Web **9**(1), 77–129 (2018)
[11] García-Durán, A., Niepert, M.: KBLRN: end-to-end learning of knowledge base representations with latent, relational, and numerical features. In: UAI, pp. 372–381, AUAI Press (2018)
[12] Hamilton, W., Bajaj, P., Zitnik, M., Jurafsky, D., Leskovec, J.: Embedding logical queries on knowledge graphs. Advances in neural information processing systems **31** (2018)
[13] Heindorf, S., Blübaum, L., Düsterhus, N., Werner, T., Nandkumar Golani, V., Demir, C., Ngonga Ngomo, A.: EvoLearner: learning description logics with evolutionary algorithms. In: WWW, pp. 818–828, ACM (2022)
[14] Klement, E., Mesiar, R., Pap, E.: Triangular norms. position paper I: basic analytical and algebraic properties. Fuzzy Sets Syst. **143**(1), 5–26 (2004)
[15] Kouagou, N.J., Heindorf, S., Demir, C., Ngonga Ngomo, A.: Learning concept lengths accelerates concept learning in ALC. In: ESWC, Lecture Notes in Computer Science, vol. 13261, pp. 236–252, Springer (2022)
[16] Kouagou, N.J., Heindorf, S., Demir, C., Ngonga Ngomo, A.: Neural class expression synthesis. In: ESWC, Lecture Notes in Computer Science, vol. 13870, pp. 209–226, Springer (2023)
[17] Kouagou, N.J., Heindorf, S., Demir, C., Ngonga Ngomo, A.: Neural class expression synthesis in ALCHIQ(D). In: ECML, Lecture Notes in Computer Science, Springer (2023)

[18] Kristiadi, A., Khan, M.A., Lukovnikov, D., Lehmann, J., Fischer, A.: Incorporating literals into knowledge graph embeddings. In: ISWC, Lecture Notes in Computer Science, vol. 11778, pp. 347–363, Springer (2019)
[19] Lacroix, T., Usunier, N., Obozinski, G.: Canonical tensor decomposition for knowledge base completion. In: ICML, Proceedings of Machine Learning Research, vol. 80, pp. 2869–2878, PMLR (2018)
[20] Nickel, M., Murphy, K., Tresp, V., Gabrilovich, E.: A review of relational machine learning for knowledge graphs. Proc. IEEE **104**(1), 11–33 (2016)
[21] Ren, H., Dai, H., Dai, B., Chen, X., Zhou, D., Leskovec, J., Schuurmans, D.: SMORE: knowledge graph completion and multi-hop reasoning in massive knowledge graphs. In: KDD, pp. 1472–1482, ACM (2022)
[22] Ren, H., Hu, W., Leskovec, J.: Query2box: Reasoning over knowledge graphs in vector space using box embeddings. In: ICLR, OpenReview.net (2020)
[23] Ren, H., Leskovec, J.: Beta embeddings for multi-hop logical reasoning in knowledge graphs. In: NeurIPS (2020)
[24] Morim da Silva, A.A., Röder, M., Ngonga Ngomo, A.: Using compositional embeddings for fact checking. In: ISWC, Lecture Notes in Computer Science, vol. 12922, pp. 270–286, Springer (2021)
[25] Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: a core of semantic knowledge. In: WWW, pp. 697–706, ACM (2007)
[26] Sun, Z., Deng, Z., Nie, J., Tang, J.: RotatE: knowledge graph embedding by relational rotation in complex space. In: ICLR (Poster), OpenReview.net (2019)
[27] Tahri, A., Tibermacine, O.: DBPedia based factoid question answering system. International Journal of Web & Semantic Technology **4**(3), 23 (2013)
[28] Tay, Y., Tuan, L.A., Phan, M.C., Hui, S.C.: Multi-task neural network for non-discrete attribute prediction in knowledge graphs. In: CIKM, pp. 1029–1038, ACM (2017)
[29] Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., Bouchard, G.: Complex embeddings for simple link prediction. In: ICML, JMLR Workshop and Conference Proceedings, vol. 48, pp. 2071–2080, JMLR.org (2016)
[30] Vrandecic, D., Krötzsch, M.: Wikidata: a free collaborative knowledgebase. Commun. ACM **57**(10), 78–85 (2014)
[31] Wu, Y., Wang, Z.: Knowledge graph embedding with numeric attributes of entities. In: Rep4NLP@ACL, pp. 132–136, Association for Computational Linguistics (2018)
[32] Wu, Y., Wang, Z.: Knowledge graph embedding with numeric attributes of entities. In: Rep4NLP@ACL, pp. 132–136, Association for Computational Linguistics (2018)
[33] Yang, B., Yih, W., He, X., Gao, J., Deng, L.: Embedding entities and relations for learning and inference in knowledge bases. In: ICLR (Poster) (2015)
[34] Zhang, S., Tay, Y., Yao, L., Liu, Q.: Quaternion knowledge graph embeddings. In: NeurIPS, pp. 2731–2741 (2019)

[35] Zhu, Z., Galkin, M., Zhang, Z., Tang, J.: Neural-symbolic models for logical queries on knowledge graphs. In: ICML, Proceedings of Machine Learning Research, vol. 162, pp. 27454–27478, PMLR (2022)