NEURO-SYMBOLIC ONTOLOGY-MEDIATED QUERY ANSWERING

Anonymous authors
Paper under double-blind review

ABSTRACT

Recently, low-dimensional vector space representations of Knowledge Graphs (KGs) have been applied to find answers to logical queries over incomplete KGs. However, the current methods only focus on inductive reasoning, i.e. answering such queries by predicting facts based on patterns learned from the data, and lack the ability of deductive reasoning, the task of computing logical entailments using expert domain knowledge. To address this shortcoming, we investigate how existing embedding models for query answering over incomplete KGs can be adapted to incorporate domain knowledge in the form of ontologies. We propose two novel datasets, based on LUBM and NELL KGs, as well as various training strategies to integrate domain knowledge into prominent representatives of embedding models for query answering. Our strategies involve (1) different ontology-driven data augmentation techniques and (2) adaptation of the loss function using query-rewriting methods. The achieved improvements in the settings that require both inductive and deductive reasoning, are from 20% to 50% in HITS@3.

1 INTRODUCTION

Answering complex logical queries over Knowledge Graphs (KGs) has recently received a lot of attention due to the relevance of this task in various applications such as natural question answering, web search or data analytics. For example, the query *Who works for Amazon and has a degree from MIT?* over the KG in Figure 1 can be formulated as $q(X) \leftarrow \mathrm{degreeFrom}(X, \mathrm{mit}) \wedge \mathrm{worksFor}(X, \mathrm{amazon})$. Answering such a query is very challenging when KGs are incomplete, which is often the case due to their (semi-) automatic construction, and obtaining complete answers typically requires further domain knowledge. For instance, mary is a missing but desired answer of q. Due to the data distribution in the KG, link prediction models might only be able to derive $\mathrm{managerAt}(\mathrm{mary}, \mathrm{amazon})$. Therefore, in this case, further domain knowledge that $\mathrm{managerAt}$ implies $\mathrm{worksFor}$ in ontology $\mathcal{O}$ of Figure 1 would be required to derive $\mathrm{worksFor}(\mathrm{mary}, \mathrm{amazon})$ and retrieve mary as an answer for q.

Recently, *Knowledge Graph Embedding* (KGE) techniques (Nickel et al., 2016; Wang et al., 2017) that are able to predict missing facts have been proposed for answering logical queries over incomplete KGs. The existing methods can be broadly divided into two categories: *query-based* (Ren et al., 2020; Ren & Leskovec, 2020; Liu et al., 2021; Choudhary et al., 2021; Kotnis et al., 2021) and *atom-based* (Arakelyan et al., 2021). The former compute continuous query embedding representations, and use them for answering queries, while the latter compute answers to a query by identifying the most likely answers to all its atoms using *neural link predictors* (Nickel et al., 2016), and then aggregating those answers using t-norms.

While being promising, such existing embedding-based methods do not account for ontologies, regarded as KG schema that enriches the KG by describing dependencies between types and/or relations. Exploiting ontologies when querying KGs is beneficial, e.g., for simplifying query formulation and obtaining more complete answers. The task of answering logical queries in the presence of ontologies is referred to as *Ontology-Mediated Query Answering* (OMQA) (Bienvenu & Ortiz, 2015). On the one hand, the use of ontologies requires *deductive reasoning*, i.e., inferring new facts by applying ontology rules to existing facts, but ignoring missing true facts. On the other hand, embedding methods are essentially tailored towards *inductive reasoning*, i.e. learning from examples: Given a number of queries and their answers, they are used to predict answers to other similar

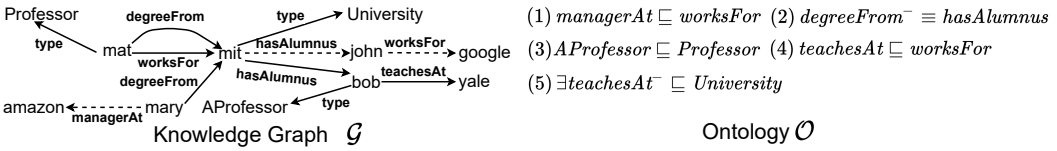

Figure 1: An exemplary KG in which solid edges illustrate existing facts in the KG, while dashed edges indicate missing facts. The rules in $\mathcal{O}$ state that (1) managers at companies also work there; (2) the inverse of relation $\mathrm{degreeFrom}$ is $\mathrm{hasAlumnus}$; (3) assistant professors are professors; (4) teachers at organizations also work there; (5) the range of the relation $\mathrm{teachesAt}$ is $\mathrm{University}$.

queries, but they typically cannot perform ontology reasoning. Since large portions of expert knowledge can be conveniently encoded using ontologies, the benefits of coupling ontology reasoning and embedding methods for KG completion are evident, and have been acknowledged (e.g. see Bianchi et al., 2020; Zhang et al., 2020; Gutiérrez-Basulto & Schockaert, 2018; Kulmanov et al., 2019). However, to the best of our knowledge, such coupling has not been studied for OMQA.

A natural attempt is to interchangeably complete the KG using ontology reasoning and embedding methods, and then perform query answering on top of the result. This naive procedure comes with a big scalability challenge: In practice, we need to restrict ourselves to computing merely small subsets of likely fact predictions required for answering a given query; thus more sophisticated proposals are required. To this end, we investigate three open questions: (1) How to adapt existing OMQA techniques to the setting of KGE? (2) How do different data augmentation strategies impact the accuracy of existing embedding models for OMQA task? and (3) Does the enforcement of ontology axioms in the embedding space via loss function help to improve inductive and deductive reasoning performance? We answer these questions by making the following contributions:

- We formally define the task of *Embedding-Based OMQA* (E-OMQA) and empirically show that existing off-the-shelf KGE models applied naively perform poorly on this task.
- We propose novel ontology-driven strategies for sampling training queries as well as loss function modification to enforce the ontology within the embedding space, and demonstrate the effectiveness of these proposals on popular representatives of query-based and atom-based models.
- Since no previous benchmarks exist for E-OMQA, we design two datasets using LUBM and NELL, which are well-known benchmarks for OMQA and embedding models, respectively.
- Extensive evaluation demonstrates improvements (20% to 50% in HITS@3) in the accuracy of E-OMQA by our methods compared to the baselines, and allows us to obtain and analyze answers to the above questions.

2 PRELIMINARIES

Knowledge Graphs and Ontologies. We assume a signature $\Sigma = \langle \mathbf{E}, \mathbf{C}, \mathbf{R} \rangle$ consisting of countable pairwise disjoint sets $\mathbf{E}, \mathbf{C}$, and $\mathbf{R}$ of constants (entities), concepts (types), and roles (binary relations) respectively. A knowledge graph $\mathcal{G}$ (*a.k.a.* ABox) is a set of triples, such as $(\mathrm{mit}, \mathrm{type}, \mathrm{University})$ and $(\mathrm{bob}, \mathrm{worksFor}, \mathrm{mit})$ formalized using Σ. These triples can also be represented as $\mathrm{type}(\mathrm{mit}, \mathrm{University})$ and $\mathrm{worksFor}(\mathrm{bob}, \mathrm{mit})$. An ontology $\mathcal{O}$ (*a.k.a.* TBox), e.g. $\mathcal{O}$ in Figure 1, is a set of axioms in Description Logics (Baader et al., 2009) over Σ. We focus on *DL-Lite*$_\mathcal{R}$ (Artale et al., 2009) which has the following syntax: $A \sqsubseteq A', A \sqsubseteq \exists p, \exists p \sqsubseteq A, \exists p^- \sqsubseteq A, p \sqsubseteq s, p^- \sqsubseteq s$, where $A, A' \in \mathbf{C}$, $p, s \in \mathbf{R}$, and p^- denotes the inverse relation of p. The deductive *closure* $\mathcal{O}^\infty(\mathcal{G})$, contains all (possibly infinitely many) new facts derived from $\mathcal{G}$ using axioms from $\mathcal{O}$ (e.g., $\mathrm{type}(\mathrm{bob}, \mathrm{Professor})$ follows from (3) and $\mathrm{type}(\mathrm{bob}, \mathrm{AProfessor})$).

Ontology-Mediated Query Answering. A *query atom* is an expression of the form $p(T_1, T_2)$, where $p \in \mathbf{R}$, and each $T_i \in \mathbf{V} \cup \mathbf{E}$ is called a *term*, with $\mathbf{V}$ disjoint with $\mathbf{E}, \mathbf{C}$, and $\mathbf{R}$ being a set of variables. A *monadic conjunctive query* (CQ) $q(X)$ is a First-Order (FO) formula of the form $q(X) \leftarrow \exists \vec{Y}.p_1(\vec{T_1}) \wedge \cdots \wedge p_n(\vec{T_n})$ where each $p_i(\vec{T_i})$ is a query atom, and $vars(q) = X \cup \vec{Y}$ denotes the set of variables appearing in q, with $X \notin \vec{Y}$ being the *answer variable*. A *monadic Existential Positive FO* (EPFO) query is a union of monadic CQs (Dalvi & Suciu, 2007). For a

query $q(X)$ and a KG $\mathcal{G}$, a constant a from $\mathcal{G}$ is an answer to $q(X)$ if there exists a mapping $\pi : var(q) \mapsto \mathbf{E}$ that maps the body (the right-hand side) of q to a sub-graph of $\mathcal{G}$. We denote by $q[\mathcal{G}]$ the answers of q on $\mathcal{G}$. *Ontology-Mediated Query Answering (OMQA)* concerns answering queries by accounting for both the KG and the accompanying ontology. Given a KG $\mathcal{G}$ and an ontology $\mathcal{O}$, an entity a from $\mathcal{G}$ is a *certain answer* of $q(X)$ over $(\mathcal{G}, \mathcal{O})$ if a is an answer to $q(X)$ over $\mathcal{O}^\infty(\mathcal{G})$. We use $q[\mathcal{G}, \mathcal{O}]$ to denote the set of *certain answers of* q *over* $(\mathcal{G}, \mathcal{O})$. Let q and q' be two monadic queries over $(\mathcal{G}, \mathcal{O})$, then q is *contained* in q' w.r.t. $\mathcal{O}$ if $q[\mathcal{G}, \mathcal{O}] \subseteq q'[\mathcal{G}, \mathcal{O}]$; we call q a *specialization* of q' (written as $q' \overset{\mathsf{s}}{\rightsquigarrow} q$), and q' a *generalization* of q (written as $q \overset{\mathsf{g}}{\rightsquigarrow} q'$). Query generalizations and specializations can be obtained by exploiting ontology axioms; such process (and result) is referred to as *query rewriting*.

Example 1. *Consider $\mathcal{G}$ in Figure 1 and* $q(X) \leftarrow \mathrm{type}(X, \mathrm{Professor}) \wedge \mathrm{degreeFrom}(X, \mathrm{mit})$. *Since* $\mathrm{mat} \in q[\mathcal{G}]$*, it is a certain answer. Moreover, according to $\mathcal{O}$,* AProfessor *is a sub-type of* Professor *and* degreeFrom *is inverse of* hasAlumnus*, thus* bob *is also a certain answer. Query* $q'(X) \leftarrow \mathrm{type}(X, \mathrm{AProfessor}) \wedge \mathrm{degreeFrom}(X, \mathrm{mit})$ *is a specialization of q as* $\mathrm{mat} \notin q'[\mathcal{G}, \mathcal{O}]$.

Embedding-Based Query Answering. Recent works on KGEs for answering logical queries can be divided into two categories: *query-based* (Ren et al., 2020; Ren & Leskovec, 2020; Liu et al., 2021; Choudhary et al., 2021; Kotnis et al., 2021) and *atom-based* (Arakelyan et al., 2021) models. A neural QA model maps entities and relations into a d-dimensional embedding space. It then computes a score of each entity c for being an answer to a given query q via a scoring function $\phi_q(\mathbf{c}) : \mathbb{R}^d \mapsto [0, 1]$, where $\mathbf{c}$ denotes the embedding vector of c.[1] Using these scoring functions, the final *embedding QA function* $\mathcal{E}_\mathcal{G}$ takes as input a query and returns answers to that query. We describe below how this is done for Query2Box and Continuous Query Decomposition (CQD). In Query2Box, entities and queries are embedded as *points* and *boxes*, respectively, in a d-dimensional vector space. A *d-dimensional embedding* is a function φ that maps $c \in \mathbf{E} \cup \mathbf{C}$ to $\mathbf{c} \in \mathbb{R}^d$ and a query q to $\mathbf{q}{=}(\mathbf{cen}_q, \mathbf{off}_q){\in}\mathbb{R}^d \times \mathbb{R}^d_{\geq 0}$, which is used to define a *query box* as $\mathrm{box}_q = \{\mathbf{v} \in \mathbb{R}^d \mid \mathbf{cen}_q - \mathbf{off}_q \preceq \mathbf{v} \preceq \mathbf{cen}_q + \mathbf{off}_q\}$, where $\preceq$ is the element-wise inequality, $\mathbf{cen}_q$ is the center of the box, and $\mathbf{off}_q$ is the positive offset of the box, modeling its size. The score for an entity c being an answer to q is computed based on the distance from $\mathbf{c}$ to box_q. A prominent representative of the second category, Continuous Query Decomposition (Arakelyan et al., 2021) reduces the task of answering a complex query to that of answering each of its sub-queries. It relies on neural link predictors for answering atomic sub-queries, and aggregates the resulting scores via t-norms.

3 Embedding-Based Ontology-Mediated Query Answering

Inductive and deductive reasoning complement each other, thus combining both yields more complete answers to queries. To target such combination, we define an embedding-based QA function that can additionally apply ontology rules to answer queries.

Definition 1 (E-OMQA)**.** *Let $\mathcal{G}$ be a KG, let $\mathcal{O}$ be an ontology, and let $\mathcal{G}^i$ be an ideal completion of $\mathcal{G}$. An embedding QA function $\mathcal{E}_\mathcal{G}$ is* reliable *if for any query q and entity a we have that $a \in \mathcal{E}_\mathcal{G}(q)$ iff $a \in q[\mathcal{G}^i]$. Moreover, $\mathcal{E}_\mathcal{G}$ is* ontology-aware *iff $a \in q[\mathcal{G}^i, \mathcal{O}]$. The* problem of embedding-based OMQA *is to obtain an embedding QA function that is both reliable and ontology-aware.*

Note that, $q[\mathcal{G}^i, \mathcal{O}]$ subsumes both $q[\mathcal{G}^i]$, the answers requiring inductive reasoning, and $q[\mathcal{G}, \mathcal{O}]$, the answers computed via deductive reasoning. We proceed to present several methods for E-OMQA.

Query Rewriting over Pre-trained Models. In the traditional OMQA setting, each query q can be evaluated by first rewriting q into a set of FO-queries $q_\mathcal{O}$, and then evaluating each query in $q_\mathcal{O}$ over $\mathcal{G}$ alone. In our case, this amounts to constructing an embedding QA function $\mathcal{E}_\mathcal{G}$ aware of $\mathcal{G}$ alone, and using it to compute the answers to all queries in $q_\mathcal{O}$ rather than only to the query q.

Example 2. *For $\mathcal{G}$, $\mathcal{O}$ in Figure 1 and queries* $q(X) \leftarrow \mathrm{degreeFrom}(X, \mathrm{mit}) \wedge \mathrm{worksFor}(X, \mathrm{amazon})$ *and* $q'(X) \leftarrow \mathrm{degreeFrom}(X, \mathrm{mit}) \wedge \mathrm{managerAt}(X, \mathrm{amazon})$*, $q_\mathcal{O}$ contains q and q' among others, and to approximate $q[\mathcal{G}^i, \mathcal{O}]$ we take the $\mathcal{E}_\mathcal{G}$-based answers of all queries in $q_\mathcal{O}$.*

Ontology-Aware Models. An alternative to query rewriting is to develop an embedding QA function that accounts for axioms in $\mathcal{O}$. To the best of our knowledge, there are no KGE models that di-

[1] Bold small letters denote vector representations.

Table 1: Rules to specialize and generalize an atom β from $q(X) \leftarrow \alpha \wedge \beta$, where $A, B \in \mathbf{C}$, $p, r, s \in \mathbf{R}$ and $T, T_1, T_2 \in vars(q) \cup \mathbf{E}$. The operators $\overset{\mathbf{s}}{\rightsquigarrow}$ and $\overset{\mathbf{g}}{\rightsquigarrow}$ are used for constructing specializations and generalizations respectively of a given query.

(R1) If $A \sqsubseteq B \in \mathcal{O}$	then:	$\alpha \wedge \text{type}(T, B) \overset{\mathbf{s}}{\rightsquigarrow} \alpha \wedge \text{type}(T, A)$	$\alpha \wedge \text{type}(T, A) \overset{\mathbf{g}}{\rightsquigarrow} \alpha \wedge \text{type}(T, B)$
(R2) If $\exists p \sqsubseteq A \in \mathcal{O}$	then:	$\alpha \wedge \text{type}(T_1, A) \overset{\mathbf{s}}{\rightsquigarrow} \alpha \wedge p(T_1, T_2)$	$\alpha \wedge p(T_1, T_2) \overset{\mathbf{g}}{\rightsquigarrow} \alpha \wedge \text{type}(T_1, A)$
(R3) If $A \sqsubseteq \exists p \in \mathcal{O}$	then:	$\alpha \wedge p(T_1, T_2) \overset{\mathbf{s}}{\rightsquigarrow} \alpha \wedge \text{type}(T, A)$	$\alpha \wedge \text{type}(T_1, A) \overset{\mathbf{g}}{\rightsquigarrow} \alpha \wedge p(T_1, T_2)$
(R4) If $\exists p^- \sqsubseteq A \in \mathcal{O}$	then:	$\alpha \wedge A(T) \overset{\mathbf{s}}{\rightsquigarrow} \alpha \wedge p(T_2, T_1)$	$\alpha \wedge p(T_2, T_1) \overset{\mathbf{g}}{\rightsquigarrow} \alpha \wedge \text{type}(T_1, A)$
(R5) If $A \sqsubseteq \exists p \in \mathcal{O}$	then:	$\alpha \wedge p(T_1, T_2) \overset{\mathbf{s}}{\rightsquigarrow} \alpha \wedge \text{type}(T, A)$	$\alpha \wedge \text{type}(T_1, A) \overset{\mathbf{g}}{\rightsquigarrow} \alpha \wedge p(T_1, T_2)$
(R6) If $p \sqsubseteq s \in \mathcal{O}$	then:	$\alpha \wedge s(T_1, T_2) \overset{\mathbf{s}}{\rightsquigarrow} \alpha \wedge p(T_1, T_2)$	$\alpha \wedge p(T_1, T_2) \overset{\mathbf{g}}{\rightsquigarrow} \alpha \wedge s(T_1, T_2)$
(R7) If $s^- \sqsubseteq p \in \mathcal{O}$	then:	$\alpha \wedge p(T_1, T_2) \overset{\mathbf{s}}{\rightsquigarrow} \alpha \wedge s(T_2, T_1)$	$\alpha \wedge s(T_1, T_2) \overset{\mathbf{g}}{\rightsquigarrow} \alpha \wedge p(T_2, T_1)$
(R8) If $\theta{:}vars(q) \mapsto vars(q) \cup \mathbf{E}$ $s.t.\ \theta(T_i) = \theta(T_i')$	then:	$\alpha \wedge p(T_1, T_2) \wedge p(T_1', T_2') \in q \overset{\mathbf{s}}{\rightsquigarrow} \alpha\theta \wedge p(T_1, T_2)\theta$ $\alpha \wedge p(T_1, T_2)$ s.t. T_1 or $T_2 \in \mathbf{E} \overset{\mathbf{g}}{\rightsquigarrow} \alpha \wedge p(Z, T_2)$ or $\alpha \wedge p(T_1, Z)$	

rectly address the problem of E-OMQA. Therefore, we suggest the following two options: (1) Train existing embedding models for logical QA on the data derived from $\mathcal{O}^\infty(\mathcal{G})$ instead of $\mathcal{G}$;(2) Develop an *ontology-aware* embedding model that will be trained on $\mathcal{G}$, but will have special terms in the training objective structurally enforcing $\mathcal{O}$.

While the proposed approaches can be realized on top of any embedding model for logical QA, in this work we verify their effectiveness on the two prominent recent embedding models: *Query2Box* and *CQD*. Regarding (1), in Section 3.1 we present several methods for effective ontology-driven training. As for (2), building on *Query2Box*, in Section 3.2 we develop an ontology-aware embedding model. Finally, we use the query-rewriting method over embeddings as a baseline in Section 4.

3.1 Ontology-Driven Data Sampling

Let $\mathcal{Q}_\mathcal{G}$ be the set of all possible EPFO monadic queries that can be formed using signature Σ. To answer any arbitrary such query, existing embedding models are trained on a set of sampled queries of certain shapes and their answers over the KG $\mathcal{G}$. For instance, queries in (Ren et al., 2020) have multiple atoms while in (Arakelyan et al., 2021) they are atomic (e.g., $q(X) \leftarrow \text{worksFor}(X, \text{mit})$). Usually, the set of training queries does not take the schema into account. For example, in (Ren et al., 2020), the queries are randomly selected from $\mathcal{Q}_\mathcal{G}$ and used for training the model along with their answers over $\mathcal{G}$ as positive examples and randomly generated non-answers as negative examples. However, if the ontology is present along with the KG, this procedure is not guaranteed to capture the ontology axioms, and using all possible queries from $\mathcal{Q}_\mathcal{G}$ may be infeasible in practice. In the following, we discuss various options for sampling queries to train ontology-aware KGE models.

Certain Answer and Query Rewriting-Based Sampling. The first natural approach for query sampling is to select queries along with their certain answers instead of the standard answers. For ontology languages such as those in the *DL-Lite* family (Artale et al., 2009) computing certain answers can be done efficiently. An example of this training case is to randomly sample query $q(Y) \leftarrow \exists X.\text{hasAlumnus}(\text{mit}, X) \wedge \text{worksFor}(X, Y)$ and, given ($\mathcal{G}$, $\mathcal{O}$) in Figure 1, use it along with all its certain answers: mit, yale during training. To account for the ontology, we can randomly sample queries over the KG, and add also all of their generalizations and specializations obtained using the rules in Table 1. To rewrite a query we select an atom and apply an ontology axiom, e.g., the first rule **(R1)** applies a concept inclusion axiom, while **(R6)** applies a role inclusion. The specializations of a given query q (denoted as $Spec(q)$), incorporate specific information regarding the answers of q, while the generalizations of q (i.e. $Gen(q)$) incorporate additional related entities.

Example 3. *Consider* $q_1(X) \leftarrow \exists Y.\text{type}(X, \text{University})$, $q_2(X) \leftarrow \exists Y, Z.\text{teachesAt}(Z, X)$. *Using* ***R2*** *in Table 1 and (5) in Figure 1 we get* $q_1 \overset{\mathbf{s}}{\rightsquigarrow} q_2$. *Similarly, in Example 2,* $q' \overset{\mathbf{g}}{\rightsquigarrow} q$ *using* ***R6*** *and (1).*

In general, there are exponentially many rewritings thus to keep the training size reasonable, we fix a rewriting depth, up to which the respective training queries are generated, via a dedicated parameter.

Strategic Ontology-Based Sampling. While adding generalizations and specializations of randomly selected queries should capture some parts of the ontological background knowledge, many relevant queries might still be missed. For example, if $\mathcal{O}$ contains also $\exists \text{worksFor}^- \sqsubseteq$

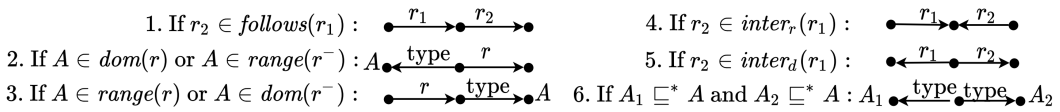

Figure 2: Ontology-driven rules to label query shapes; r^- denotes any of $inv(r)$.

Organization, queries such as $q(X) \leftarrow \exists Y \, \text{managerAt}(Y, X) \wedge \text{type}(X, \text{Organization})$ are likely to be disregarded during training. Therefore, another training approach that we propose is to leverage the ontology to strategically generate the train queries.

For that first, we formalize the set of target queries by means of a *query template graph*, i.e., a directed acyclic graph (DAG) (N, E), where N is a set of nodes and $E \subseteq N \times N$ is a set of directed edges. Such DAG captures the shape of each query. The set of target queries is then obtained by applying a labeling function to assign symbols in Σ to nodes and edges.

Definition 2 (Query Shape)**.** *A* query shape S *is a tuple* (N, E, n) *such that* (N, E) *is a DAG and* $n \in N$ *is the* distinguished node *of* S *(i.e., the node corresponding to the answer variable). For a given set of relations and constants in* Σ*,* a labeling function $f : N \cup E \mapsto \Sigma \cup \mathbf{V}$ *maps each node to either a variable or an entitiy and each edge to a relation symbol in* Σ*.*

Our goal is to exploit the ontology to label query shapes to create semantically meaningful queries. Towards that, let $\sqsubseteq^*$ be the reflexive and transitive closure of $\sqsubseteq$. Then, for a given relation p :

- $inv(p) = \{p' \mid p \sqsubseteq p'^- \in \mathcal{O}\}$, $\quad dom(p) = \{A \mid \exists p' \sqsubseteq A' \in \mathcal{O}$ s.t. $p \sqsubseteq^* p', A \sqsubseteq^* A'$ or $A' \sqsubseteq^* A\}$,
- $range(p) = \{A \mid \exists p'^- \sqsubseteq A' \in \mathcal{O}$ s.t. $p \sqsubseteq^* p', A \sqsubseteq^* A'$ or $A' \sqsubseteq^* A\}$,
- $follows(p) = \{p' \mid range(p) \cap dom(p') \neq \emptyset$ or $\}$,
- $inter_r(p) = \{p' | range(p) \cap range(p') \neq \emptyset$ or $p_1 \in inv(p), p_2 \in inv(p')$ and $dom(p_1) \cap dom(p_2) \neq \emptyset\}$,
- $inter_d(p) = \{p' | dom(p) \cap dom(p') \neq \emptyset$ or $p_1 \in inv(p), p_2 \in inv(p')$ and $range(p_1) \cap range(p_2) \neq \emptyset\}$.

Intuitively, for a given relation p, the set $inv(p)$ contains all inverse relations of p, $dom(p)$ contains all domain types for p, $range(p)$ all range types for p, $follows(p)$ stores all relations p' which can follow p, and $inter_r(p)$, $inter_d(p)$ contain resp. all relations p' which can intersect p on range and domain positions. Then, for each shape we label nodes and edges to create queries that are valid w.r.t. $\mathcal{O}$ as illustrated in Figure 2. Note that this query sampling process uses only the ontology, thus it is data independent. However, if the ontology does not capture additional data patterns we can proceed in a bottom-up fashion. We randomly take some labeled query shapes which produce answers, and construct their generalizations as before.

3.2 An Ontology-Aware Training Objective

In this section, we present our novel training objective function employed by Query2Box. Recall that when embedding a query, the Query2Box model defines a box in an embedding space, s.t. the answer entities of the given query are mapped to points located inside of the box. Note that for every ontological axiom its both left- and right-hand side can be turned into queries. We observe that when embedding those queries as boxes, ontological axioms can be naturally injected into the model if in the vector space the inclusion of the boxes corresponding to the respective queries is ensured.

Example 4. *In Figure 3, the entities and relations are embedded into the vector space as points and projection operators, resp. The embedding of* $q(Y) \leftarrow \exists X.\text{hasAlumnus}(\text{mit}, X) \wedge \text{worksFor}(X, Y)$ *is represented by the larger grey box, obtained by applying the projection* hasAlumnus *to the embedding of entity* mit *followed by the projection on* worksFor*. To enforce* $\text{teachesAt} \sqsubseteq \text{worksFor}$ *we ensure that the box corresponding to* $q'(Y) \leftarrow \exists X.\text{hasAlumnus}(\text{mit}, X) \wedge \text{teachesAt}(X, Y)$*, is contained in the box corresponding to* q*.*

The goal is to learn the embedding of queries, such that the *distance* between the box, corresponding to the query, and its answers is minimized, while the *distance* to this box from other negative samples is maximized. Similarly to Ren et al. (2020), we define the distance between $\mathbf{q} \in \mathbb{R}^d \times \mathbb{R}^d_{\geq 0}$ and $\mathbf{v} \in \mathbb{R}^d$ as $d(\mathbf{q}, \mathbf{v}) = \|\mathbf{cen}_q - \mathbf{v}\|_1$, namely the L_1 distance from the entity $\mathbf{v}$ to the center of the box. Using the sigmoid function we transform the distance into the $(0, 1)$ interval, that is, $p(\mathbf{v} \mid \mathbf{q}) = \sigma\big(-(d(\mathbf{q}, \mathbf{v}) - \gamma)\big)$, where $\gamma > 0$ is a margin, which denotes the probability of $v \in q[\mathcal{O}, \mathcal{G}^i]$.

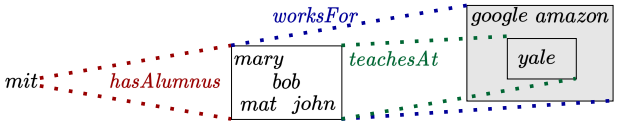

Figure 3: Our extension of Query2Box, where query embeddings capture the ontology axiom $teachesAt \sqsubseteq worksFor$, represented by the inclusion of the respective boxes.

For a query q, let $Gen(q) = \{q_1 \dots q_n\}$ be the set of all generalizations of q based on $\mathcal{O}$. Given a train query q and its certain answer $v \in q[\mathcal{G}, \mathcal{O}]$, we aim at maximizing $\prod_{i=1}^{n} p(\mathbf{v} \mid \mathbf{q}_i)^{\beta_i}$, where $\beta_i \geq 0$ is a weighting parameter for all $i = 1, \dots, n$. This is achieved by minimizing the negative log-likelihood:[2] $-\log\left(\prod_{i=1}^{n} p(\mathbf{v} \mid \mathbf{q}_i)^{\beta_i}\right) = -\sum_{i=1}^{n} \beta_i \log\big(p(\mathbf{v} \mid \mathbf{q}_i)\big)$. By exploiting the fact that $\sigma(x) = 1 - \sigma(-x)$, for any $\mathbf{v}'_j \notin q[\mathcal{G}, \mathcal{O}]$, we have that $p(\mathbf{v}' \mid \mathbf{q}) = 1 - p(\mathbf{v} \mid \mathbf{q}_i) = \sigma(d(\mathbf{q}, \mathbf{v}) - \gamma)$.

Our goal is to ensure that if q' is a generalization of a given train query q w.r.t. $\mathcal{O}$, then the box of q' contains the box of q. In other words, if a is an answer to the query q then the distance not only between a and q should be minimized, but also between a and q' as well as between a and all other generalizations of q. The following training objective reflects our goal:

$$L = -\sum_{i=1}^{n} \beta_i \log \sigma\big(\gamma - d(\mathbf{v}, \mathbf{q}_i)\big) - \sum_{j=1}^{k} \frac{1}{k} \log \sigma(d(\mathbf{v}'_j; \mathbf{q}) - \gamma),$$

where $\mathbf{v}'_j \notin q[\mathcal{G}, \mathcal{O}]$ is a random entity for all $j = 1, \dots, k$ obtained via negative sampling. In our experiments, we use $\beta_i = |Gen(q)|^{-1} = 1/n$.

Example 5. *Consider* $q(Y) \leftarrow \exists X.\mathrm{hasAlumnus}(\mathrm{mit}, X) \wedge \mathrm{type}(X, \mathrm{AProfessor}) \wedge \mathrm{teachesAt}(X, Y)$. *We have* $Gen(q) = \{q_1, q_2, q_3\}$, *where* q_1 *is obtained from* q *by substituting* $\mathrm{teachesAt}$ *with* $\mathrm{worksAt}$, *while* q_2 *is* q *with* $\mathrm{type}(X, \mathrm{Professor})$ *instead of* $\mathrm{type}(X, \mathrm{AProfessor})$. *In* q_3 *the first, second and third atoms are resp. the same as in* q, q_1 *and* q_2. *It holds that* $q[\mathcal{G}, \mathcal{O}] = \{\mathrm{yale}\}$, *hence our training objective is to minimize the distance between* $\mathbf{yale}$ *(the embedding of* yale*), and* $\mathbf{q}$ *as well as the distance between* $\mathbf{yale}$ *and the boxes of* q_1, q_2 *and* q_3 *(denoted by* $\mathbf{q_1}$, $\mathbf{q_2}$ *and* $\mathbf{q_3}$*).*

Note that conceptually, our training data sampling techniques and the loss function modifications are flexible in terms of the Description Logic in which the ontology is encoded. The only restriction is the existence of efficient query rewriting algorithms for this DL. In this work, we focused on $DL\text{-}Lite_{\mathcal{R}}$, since the majority of available ontologies are belong to this language.

4 Evaluation

In this section, we evaluate the proposed training strategies on the two recent embedding models for QA: Query2Box model ($Q2B$, Ren et al., 2020) and Continuous Query Decomposition (CQD, Arakelyan et al., 2021). We also measure the effectiveness of the newly introduced training objective function of $Q2B$ model (called $O2B$). All models are evaluated in different settings to measure their ability to perform inductive reasoning, deductive reasoning, and their combination.[3]

4.1 Experimental Setup

We have configured both $Q2B$ and $O2B$ systems as follows: We set the size of the embedding dimension to 400, and trained the models for 15×10^4 steps using Adam (Kingma & Ba, 2015) with an initial learning rate of 10^{-4} and the batch size of 512. We evaluated the models periodically and reported the test results of the models which have the best performance on the validation dataset. For CQD we have used the following configuration: we used ComplEx-N3 (Lacroix et al., 2018) as the underlying neural link predictor, where the embedding size was set to 1000, and the regularisation weight was selected based on the validation set by searching in $\{10^{-3}, 5 \times 10^{-3}, \dots, 10^{-1}\}$.

[2]The log is strictly monotonically increasing, thus, it will not change the maximization. It only changes the product to a summation. During training we consider a minimization, which motivates the negative sign.

[3]Code and data are available at https://tinyurl.com/66hbhppc.

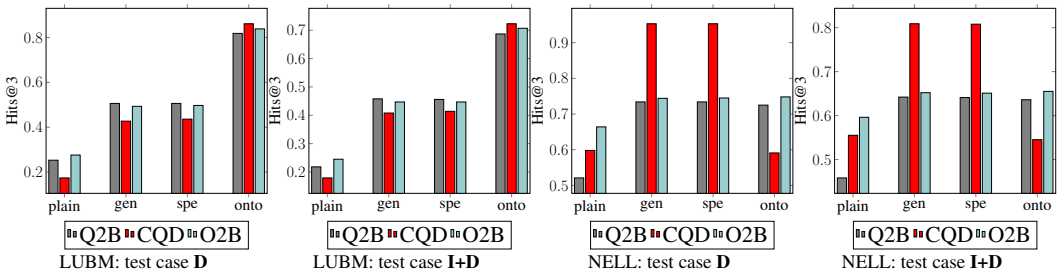

Figure 4: Comparison of $Q2B$,$O2B$ and CQD in each training setting for test cases (**D**) and (**I+D**)

Table 2: HITS@3 metric per query shape for deductive (**D**) and inductive+deductive (**I+D**)

Model	Test Case D										Test Case I+D									
	Avg.	**1p**	**2p**	**3p**	**2i**	**3i**	**ip**	**pi**	**2u**	**up**	**Avg.**	**1p**	**2p**	**3p**	**2i**	**3i**	**ip**	**pi**	**2u**	**up**
LUBM																				
$Q2B_{plain}$	0.253	**0.318**	**0.12**	**0.103**	0.464	0.588	0.181	0.242	0.160	0.104	0.218	0.173	0.101	**0.107**	0.433	0.546	0.167	0.200	0.133	0.100
$O2B_{plain}$	**0.276**	0.317	0.113	0.094	**0.512**	**0.63**	**0.189**	**0.263**	**0.257**	**0.110**	**0.245**	**0.235**	**0.109**	0.095	**0.488**	**0.584**	**0.176**	**0.218**	**0.200**	**0.103**
CQD_{plain}	0.174	0.101	0.051	0.100	0.364	0.509	0.133	0.199	0.076	0.040	0.179	0.109	0.058	0.104	0.384	0.502	0.130	0.187	0.092	0.046
$Q2B_{gen}$	**0.506**	0.619	**0.242**	**0.113**	**0.887**	**0.936**	**0.426**	**0.333**	**0.671**	**0.327**	**0.458**	**0.592**	**0.267**	**0.129**	**0.789**	**0.870**	**0.360**	**0.282**	**0.552**	**0.279**
$O2B_{gen}$	0.493	**0.641**	0.221	0.100	0.876	0.921	0.399	0.317	0.66	0.301	0.447	0.577	0.257	0.114	0.777	0.859	0.359	0.27	0.546	0.264
CQD_{gen}	0.427	0.460	0.150	0.079	0.770	0.830	0.343	0.342	0.618	0.252	0.408	0.539	0.214	0.098	0.710	0.791	0.304	0.302	0.513	0.208
$Q2B_{spec}$	**0.506**	**0.677**	**0.229**	**0.107**	**0.893**	**0.936**	**0.408**	**0.327**	0.66	**0.313**	**0.456**	**0.590**	**0.263**	**0.122**	**0.791**	**0.872**	0.359	**0.286**	**0.548**	**0.275**
$O2B_{spec}$	0.497	0.646	0.228	0.104	0.873	0.919	0.407	0.317	**0.666**	0.310	0.447	0.576	0.258	0.113	0.776	0.857	**0.360**	0.27	0.544	0.265
CQD_{spec}	0.436	0.459	0.193	0.097	0.764	0.825	0.378	0.342	0.616	0.252	0.414	0.539	0.240	0.114	0.705	0.787	0.330	0.298	0.511	0.210
$Q2B_{onto}$	0.818	0.929	0.760	0.482	0.988	0.994	0.877	0.646	0.932	0.751	0.687	0.762	0.617	0.447	0.868	0.915	0.693	0.555	0.732	0.600
$O2B_{onto}$	0.838	**0.960**	0.771	0.514	**0.991**	**0.996**	0.879	0.697	**0.963**	0.768	0.707	**0.771**	0.629	0.476	**0.878**	**0.927**	0.694	0.619	**0.752**	**0.618**
CQD_{onto}	**0.861**	0.901	**0.857**	**0.552**	0.961	0.979	**0.896**	**0.879**	0.942	**0.788**	**0.723**	0.752	**0.681**	**0.481**	0.870	0.924	**0.735**	**0.728**	0.738	0.604
NELL																				
$Q2B_{plain}$	0.521	0.763	0.401	0.325	0.776	0.832	0.452	0.535	0.337	0.272	0.458	0.516	0.343	0.286	0.747	0.81	0.404	0.447	0.325	0.241
$O2B_{plain}$	**0.664**	**0.816**	**0.483**	**0.413**	**0.961**	**0.975**	**0.535**	**0.648**	**0.792**	**0.355**	**0.596**	**0.79**	**0.409**	**0.359**	**0.904**	**0.936**	**0.479**	**0.521**	**0.666**	0.303
CQD_{plain}	0.598	0.710	0.412	0.341	0.891	0.929	0.522	0.593	0.649	0.331	0.555	0.664	0.383	0.304	0.853	0.903	0.471	0.512	0.599	**0.306**
$Q2B_{gen}$	0.734	0.974	0.559	0.466	0.99	0.99	0.622	0.685	0.94	0.377	0.642	0.858	0.485	0.397	0.928	0.95	0.538	0.539	0.768	0.312
$O2B_{gen}$	0.744	0.962	0.572	0.492	0.989	0.990	0.639	0.712	0.944	0.396	0.652	0.859	0.494	0.420	0.928	0.953	0.552	0.559	0.77	0.329
CQD_{gen}	**0.953**	**1.000**	**0.996**	**0.601**	**1.000**	**1.000**	**0.997**	**0.999**	**1.000**	**0.988**	**0.809**	**0.903**	**0.775**	**0.473**	**0.957**	**0.969**	**0.821**	**0.757**	**0.886**	**0.743**
$Q2B_{spec}$	0.734	0.974	0.559	0.464	0.99	0.991	0.622	0.684	0.940	0.377	0.641	0.859	0.483	0.397	0.927	0.950	0.538	0.538	0.766	0.313
$O2B_{spec}$	0.745	0.967	0.573	0.493	0.988	0.990	0.639	0.711	0.944	0.397	0.651	0.859	0.494	0.42	0.928	0.954	0.551	0.558	0.771	0.329
CQD_{spec}	**0.953**	**1.000**	**0.996**	**0.599**	**1.000**	**1.000**	**0.998**	**0.999**	**1.000**	**0.988**	**0.808**	**0.902**	**0.774**	**0.474**	**0.958**	**0.968**	**0.820**	**0.755**	**0.885**	**0.741**
$Q2B_{onto}$	0.725	**0.973**	0.567	0.466	0.985	0.986	0.606	0.654	0.909	0.384	0.636	0.858	0.472	0.398	0.927	0.948	0.529	0.524	0.747	0.317
$O2B_{onto}$	**0.748**	0.968	**0.598**	**0.496**	**0.989**	**0.988**	**0.646**	**0.697**	**0.941**	**0.412**	**0.655**	**0.862**	**0.498**	**0.423**	**0.933**	**0.953**	**0.557**	**0.555**	**0.773**	**0.340**
CQD_{onto}	0.591	0.659	0.404	0.329	0.896	0.943	0.515	0.611	0.663	0.302	0.545	0.667	0.368	0.293	0.848	0.904	0.453	0.506	0.595	0.275

Query and Answers Sampling. We use the same type of queries (corresponding to directed acyclic graphs with entities as the source nodes, also known as *anchors*) as Ren et al. (2020) (see Figure 5 in Appendix). We consider each input KG $\mathcal{G}$ to be the ideal completion (i.e. $\mathcal{G}^i$) and then partition it into $\mathcal{G}_{valid}$ for validation and $\mathcal{G}_{train}$ for training by discarding 10% of edges at each step; this yields $\mathcal{G}_{train} \subsetneq \mathcal{G}_{valid} \subsetneq \mathcal{G}$. We then create several training sets of queries according to our ontology-aware data sampling strategies from Section 3.1. More specifically, these include:

- $plain$: the training queries are randomly sampled based on the signature of $\mathcal{G}_{train}$, and their plain answers, i.e. over $\mathcal{G}_{train}$.
- gen: queries from $plain$ augmented with their ontology-based generalizations [4]; all answers are certain, i.e. over $\mathcal{O}^{\infty}(\mathcal{G}_{train})$.
- $spec$: queries from gen augmented with their ontology-based specializations; all answers are certain answers as well.
- $onto$: queries constructed relying on the ontology axioms as introduced in Section 3.1, for which we randomly choose a percentage of valid entities as anchors; all answers are certain.

[4]This setting is similar to random sampling over $\mathcal{O}^{\infty}(\mathcal{G}_{train})$ but unlike the deductive closure, our procedure is guaranteed to terminate. We used the rewriting depth of up to 10.

Following Ren & Leskovec (2020), the training query shapes are the first five ones in Figure 5 (1p–3i); non-compliant specializations and generalizations are discarded. The $Q2B$ and $O2B$ are trained on all five query shapes, while CQD is trained only on 1p queries (Arakelyan et al., 2021).

Evaluation Procedure. For each trained model we measure its performance using standard metric HITS at K for K=3 (HITS@3), which indicates the frequency that the correct answer is ranked among the top-3 results (the higher, the better). We use such metric for measuring the reliability and ontology-awareness of the resulting models (as in Definition 1):

Inductive case (I). Evaluating the inductive reasoning ability (accounts for the standard test case): Is the model able to predict missing answers to queries over the ideal completion $\mathcal{G}^i$?

Deductive case (D). Evaluating the deductive reasoning ability: Is the model able to predict answers that can be inferred from the known triples in $\mathcal{G}_{train}$ using ontology axioms?

Inductive + Deductive case (I+D). The combination of **I** and **D**: Is the model able to predict missing answers that are inferred from the ideal completion $\mathcal{G}^i$ using axioms from $\mathcal{O}$?

For **I**, we randomly generate validation and test queries over $\mathcal{G}_{valid}$, and input $\mathcal{G}$, in such a way that for each validation query q we have that $q[\mathcal{G}_{train}] \subsetneq q[\mathcal{G}_{valid}]$, and for each test query q we have $q[\mathcal{G}_{valid}] \subsetneq q[\mathcal{G}]$. For **D**, we randomly generate evaluation queries over $\mathcal{O}^\infty(\mathcal{G}_{train})$ s.t. they are not trivially answered over $\mathcal{G}_{train}$. Moreover, each validation query is unseen during training, and each test query is unseen during training and validation. For **I+D**, we proceed as for **I**, but use $\mathcal{O}^\infty(\mathcal{G}_{valid})$ and $\mathcal{O}^\infty(\mathcal{G})$ to sample validation and test queries and their answers. In each test case all shapes in Figure 5 are sampled, and we measure accuracy based on so-called *hard answers*, which cannot be trivially retrieved from $\mathcal{G}_{train}$ and require prediction of missing edges and/or the application of some ontology axioms. A hard answer for the case **I** is an answer in $q[\mathcal{G}] \setminus q[\mathcal{G}_{valid}]$, for **D** it is an answer in $q[\mathcal{O}^\infty(\mathcal{G}_{train})] \setminus q[\mathcal{G}_{train}]$, while for **I+D** it is an answer in $q[\mathcal{O}^\infty(\mathcal{G})] \setminus q[\mathcal{O}^\infty(\mathcal{G}_{valid})]$.

Models and Datasets. We consider $Q2B$, $O2B$ and CQD trained in each described setting: i.e., M_x, where $M \in \{Q2B, O2B, CQD\}$ and $x \in \{plain, gen, spec, onto\}$. Additionally, we consider the use of the query-rewriting method on top of each model pre-trained using $plain$ strategy, denoted by M^{rew}_{plain}, i.e., ${Q2B}_{plain}$, $Q2B^{rew}_{plain}$ and ${CQD}_{plain}$, CQD^{rew}_{plain} respectively are used as baselines. We evaluate the proposed training methods, as well as the novel training objective, on two datasets: NELL (Carlson et al., 2010), a general purpose real world KG, and LUBM (Guo et al., 2005), a domain specific synthetic dataset describing the university domain. We selected these datasets, as they are among few large KGs that have ontologies (see Appendix for statistics).

4.2 Evaluation Results

For the standard test case **I**, the baseline ${Q2B}_{plain}$ performs best on LUBM, while ${CQD}_{plain}$ outperforms the other models and configurations on NELL. This is not surprising: ontologies are not effective when coping with missing edges and facts in a KG beyond those that they can deductively infer. In fact, if statistically, the patterns reflected by ontologies do not hold in the data, ontology-aware training strategies might worsen the prediction quality. The second observation is that the query rewriting over embedding models only slightly improves the prediction accuracy; $Q2B^{rew}_{plain}$, $O2B^{rew}_{plain}$ and CQD^{rew}_{plain} result in at most 10% enhancement on test cases **D** and **I+D** over ${Q2B}_{plain}$, ${O2B}_{plain}$ and ${CQD}_{plain}$ respectively. These limited improvements are likely due to the incompleteness of the rewriting procedure caused by the restriction of the queries supported by the models. The results on **I** and query-rewriting over embeddings are in Appendix E.1.

Next, we discuss our main observations for the other more interesting test cases **D** and **I+D**. The results are reported in Table 2 and visually illustrated in Figure 4. Overall, the effectiveness of the proposed solutions is evident: for **I+D** on LUBM the improvements are of almost 50% for Query2Box and 54% for CQD, while for NELL of almost 20% for Query2Box and 25% for CQD.

Performance of Training Strategies. The results on certain answer prediction (**D** and **I+D**) show that none of the baselines is able to capture the domain knowledge expressed in the ontology, and thus cannot be used directly for OMQA. Our ontology-aware model–${O2B}_{plain}$ outperforms the other models trained on $plain$, but incorporation of certain answers and generalizations via our training strategies leads to better results. The proposed training methods from Section 3.1 significantly improved the accuracy for test cases **D** and **I+D**. For all models generating training queries by taking the ontology into account yields improvements. This observation holds already when augmenting

the set of random queries by choosing their generalizations, though the addition of specializations does not seem to have a major impact. We observed that randomly selecting training queries, as usually done in the literature, does not result in the most accurate models. On LUBM, for all models, the advantage of the ontology-driven query sampling (i.e. $onto$ setting) is significant compared to all other settings. Remarkably, for LUBM CQD_{onto} trained on less data than CQD_{gen} or CQD_{spe} results in higher accuracy. This shows that random sampling is not adequate for OMQA. For NELL, to keep the size of the training set reasonable, we chose a much lower number of anchors obtaining a significantly lower number of atomic queries (details in the Appendix), however since $Q2B$ and $O2B$ use information from other query shapes $onto$ setting still outperforms all others, unlike for CQD which only relies on atomic queries.

Evaluation of the Ontology-Aware Training Objective. The model $O2B_{onto}$ has far better accuracy on cases **D** and **I+D** than the $Q2B$ baseline. This shows that the enforcement of ontology axioms in the embedding space together with strategic ontology-driven training provides significant improvement, especially for LUBM which has a more expressive ontology. Furthermore, the improvement of $O2B_{plain}$ over $Q2B_{plain}$ shows that we are able to partially incorporate the domain knowledge into the embedding model without explicitly training on certain answers.

5 Related Work

The task of answering queries that involve multiple atoms using embedding techniques has recently received a lot of attention. The existing proposals can be divided into *query-based* (Ren et al., 2020; Ren & Leskovec, 2020; Liu et al., 2021; Choudhary et al., 2021; Kotnis et al., 2021; Sun et al., 2020) and *atom-based* (Arakelyan et al., 2021). Friedman & den Broeck (2020) and Borgwardt et al. (2019) study the relation between the problem of conjunctive QA in the embedding space and over probabilistic databases. Our work is different from the above proposals in that along with the data we also rely on ontologies to answer queries.

Integration of ontologies into KG embeddings has been studied by e.g. Krompaß et al. (2015); Minervini et al. (2017); Hao et al. (2019); Guo et al. (2016); Rocktäschel et al. (2015); Demeester et al. (2016); Kazemi & Poole (2018); Fatemi et al. (2019); Abboud et al. (2020), but these works do not capture all supported axioms and focus on link prediction rather than QA. The capability of embeddings to model hierarchical data has been explored by Patel et al. (2020); Idahl et al. (2019); Gutiérrez-Basulto & Schockaert (2018). In particular, Idahl et al. (2019) aim at interpreting embeddings by finding concept spaces in node embeddings and linking them to a simple external type hierarchy; this is different from our method for OMQA over embeddings. In Gutiérrez-Basulto & Schockaert (2018), conceptual space representations of known concepts are learned by associating a Gaussian distribution with each concept over a learned vector space. Constructing models for $\mathcal{EL}$ ontologies in the embedding space (Kulmanov et al., 2019) is another relevant direction. While Gutiérrez-Basulto & Schockaert (2018); Kulmanov et al. (2019) are related to our work, they do not touch upon the problem of OMQA. The OMQA problem has been actively studied (see e.g. Schneider & Simkus (2020) for an overview), but available methods only focus on purely logic-based deductive reasoning, without aiming at simultaneously handling missing links.

6 Conclusion

We have presented methods for Ontology-Mediated Query Answering that operate in the embedding space to enable simultaneous inductive and deductive reasoning over the incomplete data. To the best of our knowledge, this is the first work on embedding-based OMQA. We have empirically demonstrated that embedding-based methods for QA applied naively or combined with query rewriting techniques are not effective. In our work, we have proposed solutions for making the existing models ontology-aware via ontology-driven training sampling strategies and loss function modifications. The improvements in the accuracy on prominent query-based and atom-based models range from 20% to 50% compared to the baselines. We believe that this work opens interesting perspectives for combining OMQA methods, with roots in knowledge representation, and embedding techniques from the machine learning area.

Reproducibility Statement. Code, data, and instructions for reproducing all experiments are available at https://tinyurl.com/66hbhppc. The hyperparameters are presented in Appendix F.

REFERENCES

Ralph Abboud, İsmail İlkan Ceylan, Thomas Lukasiewicz, and Tommaso Salvatori. Boxe: A box embedding model for knowledge base completion. In *NeurIPS*, 2020.

Erik Arakelyan, Daniel Daza, Pasquale Minervini, and Michael Cochez. Complex query answering with neural link predictors. In *ICLR*, 2021.

Alessandro Artale, Diego Calvanese, Roman Kontchakov, and Michael Zakharyaschev. The dl-lite family and relations. *J. Artif. Intell. Res.*, 36:1–69, 2009.

Franz Baader, Ian Horrocks, and Ulrike Sattler. Description logics. In *Handbook on Ontologies*, pp. 21–43. 2009.

Federico Bianchi, Gaetano Rossiello, Luca Costabello, Matteo Palmonari, and Pasquale Minervini. Knowledge graph embeddings and explainable AI. In *Knowledge Graphs for eXplainable Artificial Intelligence: Foundations, Applications and Challenges*, pp. 49–72. 2020.

Meghyn Bienvenu and Magdalena Ortiz. Ontology-mediated query answering with data-tractable description logics. In *Reasoning Web*, volume 9203 of *Lecture Notes in Computer Science*, pp. 218–307. Springer, 2015.

Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Neurips*, pp. 2787–2795, 2013.

Stefan Borgwardt, İsmail İlkan Ceylan, and Thomas Lukasiewicz. Ontology-mediated query answering over log-linear probabilistic data. In *AAA*, pp. 2711–2718. AAAI Press, 2019.

Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *J. Autom. Reason.*, 39(3):385–429, 2007.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell. Toward an architecture for never-ending language learning. In *AAAI*, 2010.

Nurendra Choudhary, Nikhil Rao, Sumeet Katariya, Karthik Subbian, and Chandan K. Reddy. Self-supervised hyperboloid representations from logical queries over knowledge graphs. In *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*, pp. 1373–1384, 2021.

Nilesh N. Dalvi and Dan Suciu. Efficient query evaluation on probabilistic databases. *VLDB J.*, 16 (4):523–544, 2007.

Thomas Demeester, Tim Rocktäschel, and Sebastian Riedel. Lifted rule injection for relation embeddings. In *EMNLP*, pp. 1389–1399. The Association for Computational Linguistics, 2016.

Bahare Fatemi, Siamak Ravanbakhsh, and David Poole. Improved knowledge graph embedding using background taxonomic information. In *IAAI*, pp. 3526–3533, 2019.

Tal Friedman and Guy Van den Broeck. Symbolic querying of vector spaces: Probabilistic databases meets relational embeddings. In Ryan P. Adams and Vibhav Gogate (eds.), *UAI*, pp. 1268–1277, 2020.

Shu Guo, Quan Wang, Lihong Wang, Bin Wang, and Li Guo. Jointly embedding knowledge graphs and logical rules. In *EMNLP*, pp. 192–202, 2016.

Yuanbo Guo, Zhengxiang Pan, and Jeff Heflin. LUBM: A benchmark for OWL knowledge base systems. *J. Web Semant.*, 3(2-3):158–182, 2005.

Víctor Gutiérrez-Basulto and Steven Schockaert. From knowledge graph embedding to ontology embedding? an analysis of the compatibility between vector space representations and rules. In *KR*, pp. 379–388. AAAI Press, 2018.

Junheng Hao, Muhao Chen, Wenchao Yu, Yizhou Sun, and Wei Wang. Universal representation learning of knowledge bases by jointly embedding instances and ontological concepts. In *SIGKDD*, pp. 1709–1719, 2019.

Maximilian Idahl, Megha Khosla, and Avishek Anand. Finding interpretable concept spaces in node embeddings using knowledge bases. *CoRR*, abs/1910.05030, 2019.

Seyed Mehran Kazemi and David Poole. Simple embedding for link prediction in knowledge graphs. In *Neurips*, pp. 4289–4300, 2018.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR (Poster)*, 2015.

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

Denis Krompaß, Stephan Baier, and Volker Tresp. Type-constrained representation learning in knowledge graphs. In *The Semantic Web - ISWC 2015 - 14th International Semantic Web Conference, Bethlehem, PA, USA, October 11-15, 2015, Proceedings, Part I*, pp. 640–655, 2015.

Maxat Kulmanov, Wang Liu-Wei, Yuan Yan, and Robert Hoehndorf. EL embeddings: Geometric construction of models for the description logic EL ++. *CoRR*, abs/1902.10499, 2019.

Timothée Lacroix, Nicolas Usunier, and Guillaume Obozinski. Canonical tensor decomposition for knowledge base completion. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2869–2878. PMLR, 2018.

Lihui Liu, Boxin Du, Heng Ji, ChengXiang Zhai, and Hanghang Tong. Neural-answering logical queries on knowledge graphs. In *KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, August 14-18, 2021*, pp. 1087–1097, 2021.

Pasquale Minervini, Thomas Demeester, Tim Rocktäschel, and Sebastian Riedel. Adversarial sets for regularising neural link predictors. In *UAI*. AUAI Press, 2017.

Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A review of relational machine learning for knowledge graphs. *Proc. IEEE*, 104(1):11–33, 2016.

Dhruvesh Patel, Shib Sankar Dasgupta, Michael Boratko, Xiang Li, Luke Vilnis, and Andrew McCallum. Representing joint hierarchies with box embeddings. In *AKBC*, 2020.

Hongyu Ren and Jure Leskovec. Beta embeddings for multi-hop logical reasoning in knowledge graphs. In *Neurips*, 2020.

Hongyu Ren, Weihua Hu, and Jure Leskovec. Query2box: Reasoning over knowledge graphs in vector space using box embeddings. In *ICLR*. OpenReview.net, 2020.

Hongyu Ren, Hanjun Dai, Bo Dai, Xinyun Chen, Michihiro Yasunaga, Haitian Sun, Dale Schuurmans, Jure Leskovec, and Denny Zhou. LEGO: latent execution-guided reasoning for multi-hop question answering on knowledge graphs. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, pp. 8959–8970, 2021.

Tim Rocktäschel, Sameer Singh, and Sebastian Riedel. Injecting logical background knowledge into embeddings for relation extraction. In *HLT-NAACL*, pp. 1119–1129, 2015.

Thomas Schneider and Mantas Simkus. Ontologies and data management: A brief survey. *Künstliche Intell.*, 34(3):329–353, 2020.

Haitian Sun, Andrew O. Arnold, Tania Bedrax-Weiss, Fernando Pereira, and William W. Cohen. Faithful embeddings for knowledge base queries. In *Neurips*, 2020.

Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE Trans. Knowl. Data Eng.*, 29(12):2724–2743, 2017.

Jing Zhang, Bo Chen, Lingxi Zhang, Xirui Ke, and Haipeng Ding. Neural-symbolic reasoning on knowledge graphs. *CoRR*, abs/2010.05446, 2020.

A DESCRIPTION LOGICS ONTOLOGIES

Table 3: Ontology axioms in the Description Logic $DL\text{-}Lite_{\mathcal{R}}$ and their translation to First Order syntax.

DL syntax	**FO syntax**
$A \sqsubseteq A'$	$\text{type}(X, \texttt{A}) \rightarrow \text{type}(X, \texttt{A'})$
$A \sqsubseteq \exists p$	$\text{type}(X, \texttt{A}) \rightarrow \exists Y.p(X, Y)$
$\exists p \sqsubseteq A$	$p(X,\, Y) \rightarrow \text{type}(X, \texttt{A})$
$\exists p^- \sqsubseteq A$	$p(Y, X) \rightarrow \text{type}(X, \texttt{A})$
$p \sqsubseteq s$	$p(X,\, Y) \rightarrow s(X,\, Y)$
$p^- \sqsubseteq s$	$p(Y, X) \rightarrow s(X, Y)$

The syntax of $DL\text{-}Lite_{\mathcal{R}}$ ontologies and its translation into rule-based syntax are given in Table 3.

The semantics of DL ontologies is defined using FO interpretations $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consisting of a non-empty domain $\Delta^{\mathcal{I}}$ and an interpretation function $\cdot^{\mathcal{I}}$, which maps each entity e to an element $e^{\mathcal{I}} \in \Delta^{\mathcal{I}}$, each concept name A to a subset $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, and each role name r to a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The interpretation function $\cdot^{\mathcal{I}}$ is extended to *complex concepts* as follows: $(\exists p)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \exists d', (d, d') \in p^{\mathcal{I}}\}$, $(p^-)^{\mathcal{I}} = \{(d', d) \mid (d, d) \in p^{\mathcal{I}}\}$.

An interpretation $\mathcal{I}$ satisfies a concept inclusion $C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, and $\mathcal{I}$ satisfies a role inclusion $p \sqsubseteq s$ iff $p^{\mathcal{I}} \subseteq s^{\mathcal{I}}$. Finally, $\mathcal{I}$ is a *model* of an ontology $\mathcal{O}$ if it satisfies all concept and role inclusions in $\mathcal{O}$. The notion of modelhood is applied also to a KG $\mathcal{G}$ as follows: An interpretation $\mathcal{I}$ satisfies a fact $A(c)$ (i.e., $\text{type}(c, A)$), resp. $p(c, c')$, if $c \in A^{\mathcal{I}}$, resp. $(c, c') \in p^{\mathcal{I}}$.

Given a KG $\mathcal{G}$ and an ontology $\mathcal{O}$, an interpretation $\mathcal{I}$ *is a model of* $\mathcal{G}$ *w.r.t* $\mathcal{O}$ if $\mathcal{I}$ satisfies each fact in $\mathcal{G}$ and each axiom in $\mathcal{O}$. In OMQA setting, to answer a given query, we need to evaluate it over each such model; in the case of $DL\text{-}Lite_{\mathcal{R}}$ ontologies, for computing answers to ontology-mediated queries, we can rely on the deductive closure $\mathcal{O}^{\infty}(\mathcal{G})$, since the model constructed from $\mathcal{O}^{\infty}(\mathcal{G})$ can be homomorphically mapped to every other model.

B TRACTABILITY OF REWRITING-BASED QUERY GENERATION

For an arbitrary $DL\text{-}Lite_{\mathcal{R}}$ ontology $\mathcal{O}$ and an arbitrary existential positive FO query q, let $Spe(q, \mathcal{O}) = \{q' \mid q \overset{\mathbf{s}}{\rightsquigarrow}^* q'\}$ be the set of all specializations of q w.r.t. $\mathcal{O}$, modulo variable renamings, obtained by exhaustively applying $\overset{\mathbf{s}}{\rightsquigarrow}^*$ rules from Table 1. Similarly, let $Gen(q, \mathcal{O}) = \{q' \mid q \overset{\mathbf{g}}{\rightsquigarrow}^* q'\}$ be the set of all generalizations of q w.r.t. $\mathcal{O}$, modulo variable renamings, obtained by exhaustively applying $\overset{\mathbf{g}}{\rightsquigarrow}^*$ rules.

The following proposition, states that our training strategies based on query-rewriting are tractable.

Proposition 1. *Let $\mathcal{O}$ be an arbitrary $DL\text{-}Lite_{\mathcal{R}}$ ontology, and let q be an existential positive FO query. Then, $Spe(q, \mathcal{O})$ and $Gen(q, \mathcal{O})$ are finite and can be computed in time that is polynomial in the size of $\mathcal{O}$.*

Proof (Sketch). The rewriting rules we propose are simulating the standard rewriting for $DL\text{-}Lite_{\mathcal{R}}$. Thus, it follows from Lemma 34 in Calvanese et al. (2007) that $Spe(q, \mathcal{O})$ is finite. Moreover, based on Lemma 42 in Calvanese et al. (2007) it follows that there exists a procedure to compute $Spe(q, \mathcal{O})$ in time that is polynomial in the size of $\mathcal{O}$. Since the generalization procedure is similar, only applying the axioms in the other direction, we also conclude that $Gen(q, \mathcal{O})$ is finite and polynomially bounded by $\mathcal{O}$. □

C QUERY2BOX GEOMETRIC OPERATIONS

We now describe the geometric operators employed in the Query2Box model.

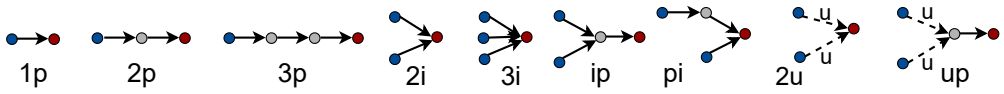

Figure 5: Query shapes considered in our experiments, where blue nodes correspond to anchor entities and red ones to answer variables; p stands for projection, i for intersection and u for union. The first five shapes are used in training.

Table 4: The number of axioms in the ontology $|\mathcal{O}|$, the number of each type of axiom, the size of the input KG $|\mathcal{G}|$, the number of entities $|\mathbf{E}|$, the number of relations $|\mathbf{R}|$, and the number of materialized triples $|\mathcal{O}^\infty(\mathcal{G})|$.

Dataset	**Ontology $\mathcal{O}$**						**KG $\mathcal{G}$**			
	$\lvert\mathcal{O}\rvert$	$A \sqsubseteq A'$	$p \sqsubseteq s$	$p^- \sqsubseteq s$	$\exists p \sqsubseteq A$	$\exists p^- \sqsubseteq A$	$\lvert\mathcal{G}\rvert$	$\lvert\mathbf{E}\rvert$	$\lvert\mathbf{R}\rvert$	$\lvert\mathcal{O}^\infty(\mathcal{G})\rvert$
LUBM	68	13	5	28	11	11	284k	55684	28	565k
NELL	307	–	92	215	–	–	285k	63361	400	497k

Projection. Let $S \subseteq \mathbf{E} \cup \mathbf{C}$ be a set of entities, and $r \in \mathbf{R}$ a relation. Intuitively, the *projection* operator performs graph traversal, e.g. given an embedding of entity e, the projection operator for the relation r provides the box corresponding to the set $\{e' \in \mathbf{E} \cup \mathbf{C} \mid r(e, e') \in \mathcal{G}\}$. Given the embedding $\mathbf{r} = (\mathbf{cen}_r, \mathbf{off}_r) \in \mathbb{R}^d \times \mathbb{R}^d_{\geq 0}$ for the relation r, we model the projection of a box $\mathbf{v} = (\mathbf{cen}_v, \mathbf{off}_v)$ by applying element-wise summation $\mathbf{v} + \mathbf{r} = (\mathbf{cen}_v + \mathbf{cen}_r, \mathbf{off}_v + \mathbf{off}_r)$. This relational translation Bordes et al. (2013) operation corresponds to the translation and enlargement of the box $\mathbf{v}$.

Intersection. Given a set of entity sets $\{S_1, \ldots, S_n\}$, the *intersection* operator computes the intersection of these sets. Recall that each set of entities is represented by a box in Query2Box model. The intersection $\mathbf{w} = (\mathbf{cen}_w, \mathbf{off}_w)$ of a set of boxes $\{(\mathbf{cen}_{v_1}, \mathbf{off}_{v_1}), \ldots, (\mathbf{cen}_{v_n}, \mathbf{off}_{v_n})\}$ corresponding to the set $\{S_1, \ldots, S_n\}$ is modeled by applying the following operations:

$$\mathbf{cen}_w = \sum_{i=1}^{n} \Phi\big(\mathrm{NN}(\mathbf{cen}_{v_1}), \ldots, \mathrm{NN}(\mathbf{cen}_{v_n})\big)_i \odot \mathbf{cen}_{v_i},$$

$$\mathbf{off}_w = \min(\mathbf{off}_{v_1}, \ldots, \mathbf{off}_{v_n}) \odot \sigma\big(\Psi(\mathbf{off}_{v_1}, \ldots, \mathbf{off}_{v_n})\big),$$

where $\odot$ and $\min$ denote the element-wise multiplication and minimum, respectively. $\mathrm{NN}\colon \mathbb{R}^d \to \mathbb{R}^d$ is a 2-layer feed-forward neural network having the same dimensionality for the hidden layers as for the input layer. Φ and σ stand for the softmax and sigmoid functions, resp., applied in a dimension-wise manner. Ψ is a permutation invariant function composed of a 2-layer feed-forward network followed by element-wise mean operation and a linear transformation. The center $\mathbf{cen}_w$ is calculated as the weighted mean of the box centers $\mathbf{cen}_{v_1}, \ldots, \mathbf{cen}_{v_n}$.

This geometric intersection provides a smaller box that lies inside a given set of boxes – for more details we refer to Ren et al. (2020).

D Data and Query Statistics

Following the procedure in the literature, each input KG is completed w.r.t. inverse edges. For the considered datasets, in Table 4 we present the number of ontology axioms of various types as well as the number of (materialized) triples, entities and relations. In our experiments, we have considered both complex and simple ontologies. Indeed, LUBM has a rich ontology including domain and range axioms as well as concept and role inclusions, while the NELL KG is accompanied with a more simple ontology containing only (inverse) role inclusions.

The size of each training/testing set, as well as the number of queries per shape for each of the considered settings is presented in Table 5, while each query shape is illustrated in Figure 5. Note that for NELL, the *plain* data is exactly the one from Ren et al. (2021). We observe that the number of 1p queries obtained for *gen* and *spe* settings are identical. This is probably because the set of 1p

Table 5: Queries statistics.

Dataset	Train/Test	Query Shape								
		1p	2p	3p	2i	3i	ip	pi	2u	up
LUBM	*Plain*	110000	110000	110000	110000	110000	–	–	–	–
	Gen	117124	136731	150653	181234	208710	–	–	–	–
	Spe	117780	154851	173678	271532	230085	–	–	–	–
	Onto	116893	166159	333406	212718	491707	–	–	–	–
	I	8000	8000	8000	8000	8000	8000	8000	8000	8000
	D	1241	4701	6472	3829	4746	7393	7557	4986	7122
	I+D	8000	8000	8000	8000	8000	8000	8000	7986	8000
NELL	*Plain*	107982	107982	107982	107982	107982	–	–	–	–
	Gen	174310	408842	864268	398412	930787	–	–	–	–
	Spe	174310	419664	906609	401954	936537	–	–	–	–
	Onto	114614	542923	864268	629144	930787	–	–	–	–
	I	15688	3910	3918	3828	3786	3932	3895	3940	3966
	D	346	4461	4294	4842	5996	7295	5862	5646	6894
	I+D	8000	8000	8000	8000	8000	8000	8000	7990	8000

queries in $plain$ covers all edges in the train KG. This explains the high accuracy of CQD_{gen} and CQD_{spe} on the test case **D**. Moreover, the NELL ontology does not contain interesting axioms that can be leveraged by ontology-driven query sampling technique, thus to obtain $onto$ we had to rely on the patterns from the data alone. Since there are too many queries to chose from, due to the large number of relations, we had to select a smaller number of valid entities as anchors, namely 20-30%. This explains the small number of 1p queries.

For the LUBM dataset, we have created the training and testing sets from scratch, and the 1p queries in $plain$ do not contain the entire training KG. The $onto$ set of queries leverages the proposed ontology-driven technique, given that the ontology covers all relations and concepts in the KG and describes how they interact, i.e. the ontology axioms support all the constructed queries, and we chose 50 % of valid entities as anchors.

E Extended Discussion of Experimental Results

In this section, we present more insights into our results on the inductive case **I** and performance of the query-rewriting baselines.

E.1 Results on Inductive Test Case

In Figure 6, we present the average HITS@3 for the inductive test case **I**. As previously discussed, we see no improvement of ontology-injection methods upon answering queries over incomplete KGs without taking certain answers into account. Indeed, $Q2B_{plain}$ outperforms all other models on LUBM, while CQD_{plain} performs best on NELL for this test case. This behaviour is expected, since ontologies cannot handle missing edges and facts in a KG that are not inferred from the data using ontological reasoning.

E.2 Query Rewriting over Pre-trained Embedding Models

In order to evaluate the target procedure that performs query rewriting over pre-trained embeddings for QA, for each hard answer a we take the best (i.e., minimum) ranking among all rankings generated by all queries in the rewriting of each test query. In other words, we take the minimal distance between the embedding of a and all rewritings of q. Note that, for measuring the performance we use the pre-trained models $Q2B_{plain}$, CQD_{plain} and $O2B_{plain}$, obtained after 450K training steps.

Due to the reliance on particular query shapes of the respective models, the complete rewriting for each query is not guaranteed. In Table 6, we present the results for this method compared to the $plain$ setting. Minor improvements of only at most 10% are observed.

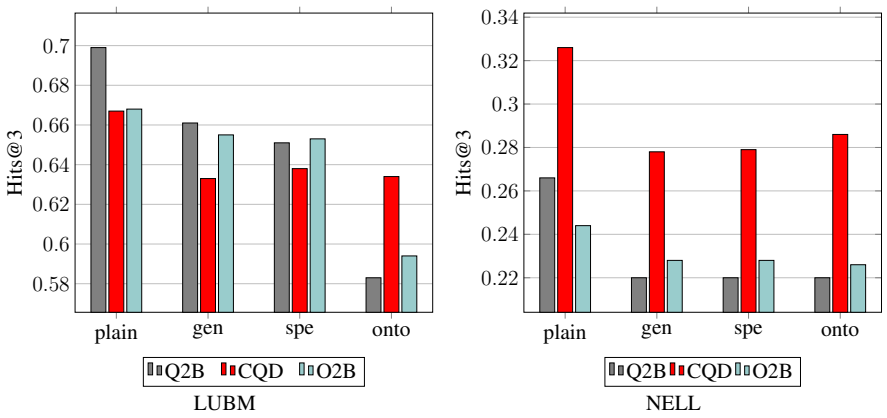

Figure 6: Comparison of $Q2B$, $O2B$, CQD in each train setting for test case **I**

Table 6: Avg. HITS@3 metric on answering queries of shapes 1p, 2p, 3p, 2i, 3i using rewriting on top of pre-trained $plain$ model versus the $plain$ model alone.

Models	**Test Case D**		**Test Case I+D**	
	LUBM	NELL	LUBM	NELL
$Q2B_{plain}$	0.189	0.617	0.193	0.539
$Q2B^{rew}_{plain}$	0.248	0.683	0.261	0.639
Gain	+0.059	+0.066	+0.068	+ 0.1
CQD_{plain}	0.225	0.656	0.231	0.621
CQD^{rew}_{plain}	0.228	0.743	0.249	0.708
Gain	+0.003	+0.087	+0.018	+ 0.087
$O2B_{plain}$	0.245	0.731	0.264	0.680
$O2B^{rew}_{plain}$	**0.255**	**0.760**	**0.273**	**0.711**
Gain	+0.01	+0.029	+0.009	+ 0.031

We have also used our pre-trained $O2B_{plain}$ model as a possible way to cope with this issue, and indeed it outperforms all other baselines. In fact, on NELL $O2B_{plain}$ becomes relatively competitive even compared to the other ontology-aware models that have been trained using more advanced ontology-driven training strategies. However, for richer ontology that comes with the LUBM KG, the improvements are still not sufficient.

E.3 Data Augmentation

In Figure 7, we present the performance of each model and the number of training queries needed in each training setting. In general, naturally, the increase of the number of training queries leads to better performance, with the exception of the setting spe, for which the training data contains all queries from gen, but the performance is comparable or slightly decreases. The $onto$ setting boosts the performance for almost all models. In particular, on LUBM, which has a richer ontology, the increase in performance is much higher compared to that for the setting when query generalizations and certain answers are included. It is worth noting that the number of 1p queries is smaller for $onto$ than for gen, but CQD_{onto} performs much better than CQD_{gen}, which demonstrates the effectiveness of our proposed ontology-driven training strategy.

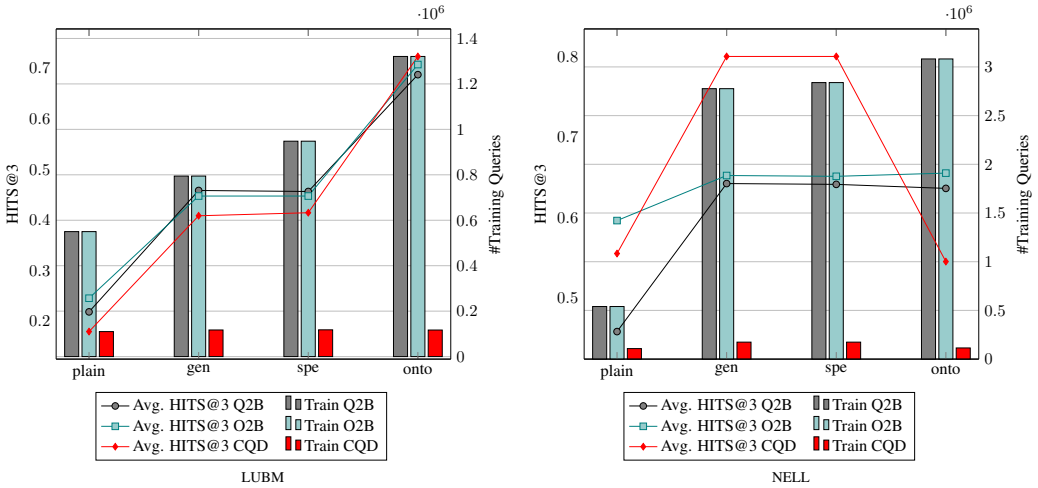

Figure 7: Performance of Q2B, O2B and CQD on **I+D** and size of the training set for each setting *plain*, *gen*, *spe*, *onto*. The number of training queries is scaled by multiplying with 10^6.

F HYPERPARAMETERS FOR Q2B, O2B AND CQD

For *Q2B* we have used the code[5] from Ren & Leskovec (2020). Our extension of this code with the implementation of the novel training objective is available online.[6].

The systems *Q2B* and *O2B* have been configured as follows: We set the size of the embedding dimension to 400, and trained the models for 15×10^4 steps using Adam (Kingma & Ba, 2015) with an initial learning rate of 10^{-4} and the batch size of 512. The rest of the parameters were set in the same way as in Ren et al. (2020). We evaluated the models periodically and reported the test results of the models which have the best performance on the validation dataset.

For CQD, we used the code shared by Arakelyan et al. (2021) [7], using ComplEx-N3 (Lacroix et al., 2018) as the base model, where the embedding size was set to 1000, and the regularisation weight was selected based on the validation set by searching in $\{10^{-3}, 5 \times 10^{-3}, \ldots, 10^{-1}\}$. For LUBM, the regularization weight was set to 0.1 in the *gen*, *spe*, and *onto* settings, and to 0.01 in the *plain* setting. For NELL, the regularization weight was set to 0.005 in the *plain* setting, to 0.001 in the *gen* and *spe* settings, and to 0.05 in the *onto* setting.

[5] https://github.com/snap-stanford/KGReasoning
[6] https://tinyurl.com/66hbhppc
[7] Available at https://github.com/pminervini/KGReasoning/