Improving Knowledge Graph Embeddings with Ontological Reasoning

Nitisha Jain[1,2][0000−0002−7429−7949], Trung-Kien Tran[1], Mohamed H. Gad-Elrab[1][0000−0002−0887−3522], and Daria Stepanova[1][0000−0001−8654−5121]

[1]Bosch Center for Artificial Intelligence, Renningen, Germany
[2]Hasso Plattner Institute, University of Potsdam, Germany
{firstname.lastname}@de.bosch.com,hpi.de

Abstract. Knowledge graph (KG) embedding models have emerged as powerful means for KG completion. To learn the representation of KGs, entities and relations are projected in a low-dimensional vector space so that not only existing triples in the KG are preserved but also new triples can be predicted. Embedding models might learn a good representation of the input KG, but due to the nature of machine learning approaches, they often lose the semantics of entities and relations, which might lead to nonsensical predictions. To address this issue we propose to improve the accuracy of embeddings using ontological reasoning. More specifically, we present a novel iterative approach *ReasonKGE* that identifies dynamically via symbolic reasoning inconsistent predictions produced by a given embedding model and feeds them as negative samples for retraining this model. In order to address the scalability problem that arises when integrating ontological reasoning into the training process, we propose an advanced technique to generalize the inconsistent predictions to other semantically similar negative samples during retraining. Experimental results demonstrate the improvements in accuracy of facts produced by our method compared to the state-of-the-art.

1 Introduction

Motivation. Knowledge Graphs (KG) describe facts about a certain domain of interest by representing them using entities interconnected via relations. Prominent examples of large KGs are DBpedia [4], Yago [31], and WikiData [34]. KGs are widely used for natural question answering, web search and data analytics. Modern KGs store information about millions of facts, however, since they are typically constructed semi-automatically or using crowd-sourcing methods, KGs are often bound to be incomplete.

To address this issue, knowledge graph embedding methods have been proposed for the *knowledge completion* task, i.e. predicting links between entities. Embedding methods learn the representation of the input KG by projecting entities and relations in a low-dimensional vector space so that not only existing triples in the KG are preserved but also new triples can be predicted (see, e.g., [36] for overview of existing approaches). Typically, the training of KG embedding models aims at discerning between correct (positive) and incorrect (negative) triples. A completion model then associates a score with every input triple. The goal of the embedding models is to rank every positive triple higher than all its negative alternatives. Therefore, the quality of

embedding models is heavily impacted by the generated negative triples. Since KGs store explicitly only positive triples, proper negative triple generation is acknowledged to be a challenging problem [11,21,40,39].

State-of-the-Art and Limitations. In the majority of existing methods the generation of negative triples is done either completely at random [9], relying on the (local) closed world assumption [27], or by exploiting the KG structure for the generation of likely true negative samples (e.g. [1,40,2]). However, these methods do not guarantee that the generated negative samples are actually incorrect ones. In [11] this issue is partially addressed by taking as negative examples precomputed triples that are inconsistent with the KG and the accompanied ontology. Since the generation of all such possible inconsistent triples as negative samples is clearly infeasible in practice, only a subset of them is precomputed, and hence certain important inconsistent triples might be missing in the set obtained in [11]. Furthermore, as embedding models rely purely on the data in the input KGs, they often lose the real semantics of entities and relations, and hence provide undesired predictions [37]. This calls for more goal-oriented approaches in which ontological reasoning is used to verify and improve the actual predictions made by embedding models.

Approach and Contributions. To address the presented shortcomings, in this work we propose an iterative method that dynamically identifies inconsistent predictions produced by a given embedding model via symbolic reasoning and feeds them as negative samples for retraining this model. We first start with any available negative sampling procedure (e.g., [21,40]) and train the embedding model as usual. Then, among predictions made by the model, we select those that cause inconsistency when being added to the KG, as negative samples for the next iteration of our method. To avoid predicting similar wrong triples, along with the inconsistent triples explicitly inferred by the embedding model, we also generate triples that are semantically similar via a *generalization procedure*. To address the scalability problem that arises when integrating ontological reasoning into the training process of embedding models, we consider ontologies in an extension of the Description Logic (DL) $DL\text{-}Lite$ [3] so that consistency checking and the generalization procedure can be performed efficiently. Our method can support any embedding model, and with the increasing number of iterations it yields better embeddings that make less inconsistent predictions and achieve higher prediction accuracy w.r.t. standard metrics.

The salient contributions of our work can be summarized as follows.

- We introduce the *ReasonKGE* framework for exploiting ontological reasoning to improve existing embedding models by advancing their negative sampling.
- To efficiently filter inconsistent embedding-based predictions, we exploit the locality property of light-weight ontologies. Moreover, in the spirit of [32] we generalize the computed inconsistent facts to a set of other similar ones to be fed back to the embedding model as negative samples.
- The evaluation of the proposed method on a set of state-of-the-art KGs equipped with ontologies, demonstrates that ontological reasoning exploited in the suggested way indeed improves the existing embedding models with respect to the quality of fact prediction.

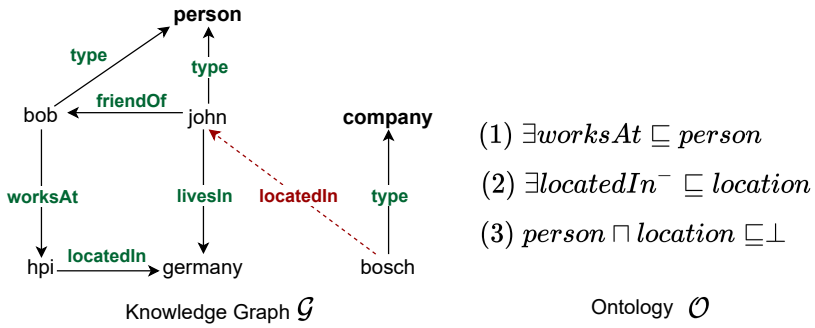

Fig. 1: Example knowledge graph with its ontology, where solid links correspond to the true facts, while the dashed one to a spurious predicted fact.

Organization. The rest of the paper is structured as follows. In Sec. 2 we present necessary background on KGs, ontologies and embedding models. In Sec. 3 our approach is described in detail, and then in Sec. 4 the results of our empirical evaluation are discussed. Finally, in Sec. 5 we present the related work, and conclude in Sec. 6. An extended version of this work[1] contains additional experimental details.

2 Preliminaries

We assume countable pairwise disjoint sets $\mathsf{N_C}$, $\mathsf{N_P}$ and $\mathsf{N_I}$ of class names (*a.k.a.* types), property names (*a.k.a.* relations), and individuals (*a.k.a.* entities). We also assume the standard relation $rdf{:}type$ (abbreviated as $type$) to be included in $\mathsf{N_P}$. A *knowledge graph* (KG) $\mathcal{G}$ is a finite set of *triples* of the form $\langle s, p, o\rangle$, where $s \in \mathsf{N_I}, p \in \mathsf{N_P}, o \in \mathsf{N_I}$, if $p \neq type$, and $o \in \mathsf{N_C}$ otherwise. KGs typically follow Open World Assumption (OWA), meaning that they store only a fraction of positive facts. For instance, given the KG from Fig. 1 $\langle john, type, person\rangle$ and $\langle john, livesIn, germany\rangle$ are true KG facts; however, whether $\langle john, worksAt, bosch\rangle$ holds or not is unknown. Given a triple α, we denote by $\mathsf{Ent}(\alpha)$ a set of all entities occurring in α and extend this notation to a set of triples as $\mathsf{Ent}(\mathcal{G}) = \bigcup_{\alpha\in\mathcal{G}} \mathsf{Ent}(\alpha)$.

An ontology $\mathcal{O}$ (*a.k.a.* TBox) is a set of axioms expressed in a certain Description Logic (DL) [5]. In this work we focus on $DL\text{-}Lite^{\mathcal{S}\sqcup}$, i.e., extension of $DL\text{-}Lite$ [3] with transitive roles and concept disjunctions. Classes C denoting sets of entities, and roles R denoting binary relations between entities, obey the following syntax:

$$C ::= A \mid \exists R \mid A \sqcup B \mid A \sqcap B \mid \neg C$$
$$R ::= P \mid P^{-}$$

Here, $A, B \in \mathsf{N_C}$ are atomic classes and $P \in \mathsf{N_P}$ is an atomic property (i.e., binary relation). An ontology $\mathcal{O}$ is a finite set of axioms of the form $C_1 \sqsubseteq C_2$, $R_1 \sqsubseteq R_2$, $R \circ R \sqsubseteq R$, reflecting the transitivity of the relation R. The summary of the DL syntax

[1] https://github.com/nitishajain/ReasonKGE

DL Syntax	OWL Syntax	Semantics
R	R	$R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
R^{-}	$\texttt{ObjectInverseOf}(R)$	$\{\langle e, d\rangle \mid \langle d, e\rangle \in R^{\mathcal{I}}\}$
A	A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
$\top$	$\texttt{owl:Thing}$	$\Delta^{\mathcal{I}}$
$\bot$	$\texttt{owl:NoThing}$	$\emptyset$
$\neg C$	$\texttt{ObjectComplementOf}(C)$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
$C \sqcap D$	$\texttt{ObjectIntersectionOf}(C, D)$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
$C \sqcup D$	$\texttt{ObjectUnionOf}(C, D)$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
$\exists P$	$\texttt{ObjectSomeValuesFrom}(P, \texttt{owl:Thing})$	$\{d \mid \exists e \in \Delta^{\mathcal{I}}. \langle d, e\rangle \in P^{\mathcal{I}}\}$
$C \sqsubseteq D$	$\texttt{SubClassOf}(C, D)$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
$P \sqsubseteq S$	$\texttt{SubObjectPropertyOf}(P, S)$	$P^{\mathcal{I}} \subseteq S^{\mathcal{I}}$
$P \circ P \sqsubseteq P$	$\texttt{TransitiveObjectProperty}(P)$	$P^{\mathcal{I}} \circ P^{\mathcal{I}} \subseteq P^{\mathcal{I}}$
$\langle a, type, c\rangle$	$\texttt{ClassAssertion}(C, a)$	$a^{\mathcal{I}} \in C^{\mathcal{I}}$
$\langle a, p, b\rangle$	$\texttt{ObjectPropertyAssertion}(P, a, b)$	$\langle a^{\mathcal{I}}, b^{\mathcal{I}}\rangle \in P^{\mathcal{I}}$

Table 1: Syntax and semantics of the ontology language considered in this paper where A, R are a class name and property name, respectively; C and D are class expressions, P, S are property expressions, and a, b are entities.

in $DL\text{-}Lite^{\mathcal{S}\sqcup}$ and its translation to OWL 2[2] is presented in Table 1. In the rest of the paper, we assume that all ontologies in this work are expressed in $DL\text{-}Lite^{\mathcal{S}\sqcup}$.

Our running example of a KG with an ontology given in Figure 1 reflects the domain knowledge about people and their working places. The ontology states that (1) the domain of $worksAt$ relation is $person$, (2) the range of $locatedIn$ is $location$, and (3) $person$ is disjoint with $location$.

Inconsistency and Explanations. The semantics of knowledge graphs and ontologies is defined using the direct model-theoretic semantics via interpretations [26]. An *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a non-empty set $\Delta^{\mathcal{I}}$, the *domain* of $\mathcal{I}$, and an *interpretation function* $\cdot^{\mathcal{I}}$, that assigns to each $A \in \mathsf{N_C}$ a subset $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, to each $R \in \mathsf{N_R}$ a binary relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, and to each $a \in \mathsf{N_I}$ an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. This assignment is extended to (complex) classes and roles as shown in Table 1.

An interpretation $\mathcal{I}$ *satisfies* an axiom α (written $\mathcal{I} \models \alpha$) if the corresponding condition in Table 1 holds. Given a KG $\mathcal{G}$ and an ontology $\mathcal{O}$, $\mathcal{I}$ is a *model* of $\mathcal{G} \cup \mathcal{O}$ (written $\mathcal{I} \models \mathcal{G} \cup \mathcal{O}$) if $\mathcal{I} \models \alpha$ for all axioms $\alpha \in \mathcal{G} \cup \mathcal{O}$. We say that $\mathcal{G} \cup \mathcal{O}$ *entails* an axiom α (written $\mathcal{G} \cup \mathcal{O} \models \alpha$), if every model of $\mathcal{G} \cup \mathcal{O}$ satisfies α. A KG $\mathcal{G}$ is *inconsistent* w.r.t. an ontology $\mathcal{O}$ if no model for $\mathcal{G} \cup \mathcal{O}$ exists. In this case, $\mathcal{G} \cup \mathcal{O}$ is inconsistent. Intuitively, $\mathcal{G} \cup \mathcal{O}$ is inconsistent when some facts of $\mathcal{G}$ contradict some axioms of $\mathcal{O}$.

Under the considered ontology language, KG inconsistency has a locality property, i.e., the problem of checking inconsistency for a KG (w.r.t. an ontology $\mathcal{O}$) can be reduced to checking inconsistency for separated KG *modules* (w.r.t. $\mathcal{O}$) [32].

[2] https://www.w3.org/TR/owl2-overview/

Definition 1 (Modules). *Given a KG $\mathcal{G}$ and an entity $e \in \mathsf{Ent}(\mathcal{G})$, the* module *of e w.r.t. $\mathcal{G}$ is defined as $\mathcal{M}(e, \mathcal{G}) = \{\alpha \mid \alpha \in \mathcal{G}$ and e occurs in $\alpha\}$. We denote the set of all modules for individuals occurring in $\mathcal{G}$ as $\mathcal{M}_{\mathcal{G}} = \{\mathcal{M}(e, \mathcal{G}) \mid e \in \mathsf{Ent}(\mathcal{G})\}$.*

Lemma 1 (Consistency Local Property). *Let $\mathcal{G}$ be a KG and $\mathcal{O}$ an ontology. Then $\mathcal{G} \cup \mathcal{O}$ is consistent iff $\mathcal{M}(a, \mathcal{G}) \cup \mathcal{O}$ is consistent for every $a \in \mathsf{Ent}(\mathcal{G})$.*

An *explanation* for inconsistency of $\mathcal{G} \cup \mathcal{O}$ [20], denoted by $\mathcal{E} = \mathcal{E}_{\mathcal{G}} \cup \mathcal{E}_{\mathcal{O}}$ with $\mathcal{E}_{\mathcal{G}} \subseteq \mathcal{G}$ and $\mathcal{E}_{\mathcal{O}} \subseteq \mathcal{O}$, is a (subset-inclusion) smallest inconsistent subset of $\mathcal{G} \cup \mathcal{O}$.

Example 1. The KG from Fig. 1 with all facts including the dashed red one is inconsistent with the ontology $\mathcal{O}$, and a possible explanation for that is $\mathcal{E} = \mathcal{E}_{\mathcal{G}} \cup \mathcal{E}_{\mathcal{O}}$ with $\mathcal{E}_{\mathcal{G}} = \{\langle bosch, locatedIn, john\rangle, \langle john, type, person\rangle\}$ and $\mathcal{E}_{\mathcal{O}} = \{\exists locatedIn^- \sqsubseteq location, person \sqcap location \sqsubseteq \bot\}$.

KG Embeddings. KG embeddings (see [36] for overview) aim at representing all entities and relations in a continuous vector space, usually as vectors or matrices called *embeddings*. Embeddings can be used to estimate the likelihood of a triple to be true via a scoring function: $f : \mathsf{N_I} \times \mathsf{N_P} \times \mathsf{N_I} \to \mathbb{R}$. Concrete scoring functions are defined based on various vector space assumptions. The likelihood that the respective assumptions of the embedding methods hold, should be higher for triples in the KG than for negative samples outside the KG. The learning process is done through minimizing the error induced from the assumptions given by their respective loss functions. Below we describe widely-used assumptions for KG embeddings:

(i) The translation-based assumption, *e.g.*, TransE [9] embeds entities and relations as vectors and assumes $\mathbf{v_s} + \mathbf{v_p} \approx \mathbf{v_o}$ for true triples, where $\mathbf{v_s}, \mathbf{v_p}, \mathbf{v_o}$ are vector embeddings for subject s, predicate p and object o, respectively. The models that rely on the translation assumption are generally optimised by minimizing the following ranking-based loss function

$$\sum_{\langle s_i, p_i, o_i\rangle \in S^+} \sum_{\langle s'_i, p_i, o'_i\rangle \in S^-} [\gamma - f(s_i, p_i, o_i) + f(s'_i, p_i, o'_i)]_+ \tag{1}$$

where $f(s, p, o) = -\left\|\mathbf{v_s} + \mathbf{v_p} - \mathbf{v_o}\right\|_1$, S^+ and S^- correspond to the sets of positive and negative training triples respectively, that are typically disjoint.

(ii) The linear map assumption, *e.g.*, ComplEx [33] embeds entities as vectors and relations as matrices. It assumes that for true triples, the linear mapping $\mathbf{M_p}$ of the subject embedding $\mathbf{v_s}$ is close to the object embedding $\mathbf{v_o}$: $\mathbf{v_s}\mathbf{M_p} \approx \mathbf{v_o}$. The loss function used for training the linear-map embedding models is given as follows:

$$\sum_{\langle s_i, p_i, o_i\rangle \in S^+} \sum_{\langle s'_i, p_i, o'_i\rangle \in S^-} l(1, f(s_i, p_i, o_i)) + l(-1, f(s'_i, p_i, o'_i))) \tag{2}$$

where $f(s, p, o) = \mathbf{v}_s \mathbf{M}_p \mathbf{v}_o$ and $l(\alpha, \beta) = log(1 - exp(-\alpha\beta))$.

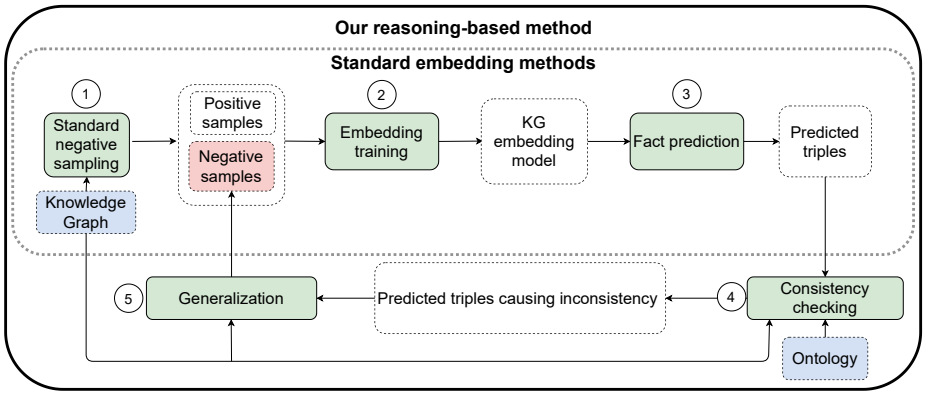

Fig. 2: Standard embedding pipeline (grey dotted frame) and our reasoning-based method (black frame) in a nutshell

3 Ontological Reasoning for Iterative Negative Sampling

While a variety of embedding models exist in the literature [36], one of the major challenges for them to perform accurate fact predictions is finding an effective way for generation of relevant negative samples [29,11,35]. Commonly used approaches for negative sampling randomly corrupt existing triples by perturbing their subject, predicate or object [9,30,13] or rely on the (local) closed world assumption (LCWA). Based on CWA all triples not present in the KG are assumed to be false, while LCWA is a variation of CWA, in which for every $\langle s, p, o\rangle$, only facts of the form $\langle s, p, o'\rangle \notin \mathcal{G}$ are assumed to be false. For instance, given the facts in Figure 1, the corrupted negative triples obtained based on the LCWA could be $\langle john, livesIn, hpi\rangle$ or $\langle bob, worksAt, bosch\rangle$.

However, since KGs follow OWA, the standard sampling methods might often turn out to be sub-optimal, resulting in false positive negative samples [11]. For example, the corrupted triple $\langle bob, worksAt, bosch\rangle$ from above might actually be true in reality.

A natural method to avoid false positives and generate only relevant negative samples is by relying on ontologies with which KGs are typically equipped. A naive approach for that is to generate all facts that can be formed using relations and entities in $\mathcal{G}$ (i.e., construct the Herbrand base) and check which among the resulting candidates are inconsistent with $\mathcal{G} \cup \mathcal{O}$. As modern KGs store millions of facts, the described procedure is infeasible in practice. To still sample some inconsistent triples, in [11] facts $p(s, o) \in \mathcal{G}$ are corrupted by substituting s (resp. o) with s' (resp. o') s.t. s and s' (resp. o and o') belong to disjoint classes and the resulting corrupted triple is inconsistent. For example, given $\mathcal{G}$ and $\mathcal{O}$ in Fig 1, from $\langle bob, worksAt, germany\rangle$ we can obtain $\alpha_1 = \langle germany, worksAt, germany\rangle$ or $\alpha_2 = \langle bob, worksAt, john\rangle$, as $person$ is disjoint with $location$. However, this method might fail to avoid the inconsistent triples that the model actually predicts. E.g., $\langle bosch, locatedIn, john\rangle$ is not generated by this method as a negative example, and the model can in principle still predict it.

Therefore, instead of pre–computing a static set of negative examples, we propose to iteratively generate and improve this set (and subsequently also the embedding model) *dynamically* by computing a collection of negative samples in a guided fashion from

embedding model based on its predictions that are inconsistent with the ontology. We refer to this negative sampling strategy as *dynamic sampling*. On the one hand, this intuitively allows us to overcome the computational challenge of generating all possible negative examples at once, but rather add the most relevant ones on demand to the embedding training process. On the other hand, this approach is capable of reducing frequently encountered errors (in terms of inconsistent predictions) for particularly difficult triples by directly incorporating feedback from incorrect predictions back to the model for further training. Indeed, when trained for increasing number of iterations, such method is capable of generating embeddings that predict fewer inconsistent facts, as empirically demonstrated in Section 4.

3.1 Approach Overview

Next we describe in more details the proposed framework referred to as *ReasonKGE*, whose main steps are depicted in Figure 2. Given a KG, ontology and an embedding method, we aim at generating an enhanced KG embedding, which is trained for predicting facts that are consistent with the KG and the ontology at hand.

The input to our method (represented by blue dashed boxes) is the KG and the ontology, while the output (the red dashed box) is the set of negative samples that is incorporated during the iterative training and tuning of a KG embedding model in each iteration. As negative samples are obtained based on predictions made by an existing embedding, a baseline model is required in the first iteration. For this, in step (1) we obtain the negative samples with **standard negative sampling** using any of the existing methods [11,9,30,13]. We then perform **embedding training** in step (2) to construct the initial KG embedding model.

This model is used for obtaining predictions and computing the set of negative samples for the next training iteration. Specifically, in step (3) the model is used for **fact prediction** as follows. For every triple in the training set, given its subject s and predicate p, we retrieve the top ranked object and obtain the fact $\langle s, p, o\rangle$ as the respective prediction. The same is done inversely for computing the top ranked subject given the object o and predicate p in the training set. Note that only triples that are not in the training set are considered as predictions. In step (4) we check whether the predicted triple complies with the ontology relying on the **consistency checking** procedure. In case the respective triple is found to be inconsistent, in step (5) we generalize it to other semantically similar triples using the **generalization** procedure to obtain an extended set of negative samples. Finally, the computed negative samples, both for subject and object predictions are fed back as input to the next iteration of the embedding training process. The detailed steps are presented in Algorithm 1 and explained in what follows.

3.2 Consistency Checking

The goal of the consistency checking procedure is to verify which predictions made by the embedding model in step (3) are inconsistent with the ontology $\mathcal{O}$ and the original KG $\mathcal{G}$. In principle, any reasoner capable of performing consistency checking effectively for ontologies in the considered $DL\text{-}Lite^{\mathcal{S}\sqcup}$ language can be used in this step. As

Algorithm 1: Training embedding models with negative samples using ontological reasoning

Input : Baseline embedding model $\mathbf{E}$, a knowledge graph $\mathcal{G}$, and an ontology $\mathcal{O}$

/* Step 1 and Step 2 */

Train the baseline embedding model $\mathbf{E}$ for a certain number of epochs.

/* Retrain the baseline model with negative samples derived from reasoning */

Loop

  /* Step 3 */

  foreach *triple* $\alpha = \langle s, p, o\rangle \in \mathcal{G}$ **do**

    Get a set $\mathsf{Predictions}(\alpha)$ of predicted triples of the form $\langle s, p, \hat{o}\rangle$ and $\langle \hat{s}, p, o\rangle$ by giving $\langle s, p\rangle$ and $\langle p, o\rangle$ as inputs to $\mathbf{E}$ and obtaining predicted entities $\hat{o}$ and $\hat{s}$, respectively.

    /* Step 4 */

    $\mathsf{NegSamples}(\alpha) \leftarrow \emptyset$

    foreach *predicted triple* $\beta \in \mathsf{Predictions}(\alpha)$ **do**

      Compute the relevant set $\mathsf{Relv}(\beta, \mathcal{G})$ of β w.r.t. $\mathcal{G}$.

      if $\mathsf{Relv}(\beta, \mathcal{G}) \cup \mathcal{O}$ *is inconsistent* **then**

        /* Step 5 */

        Compute explanations for inconsistency.

        foreach *inconsistency explanation* $\mathcal{E}_\mathcal{G} \cup \mathcal{E}_\mathcal{O}$ **do**

          Compute $\mathsf{GeneralizedSamples}(\beta)$ as defined in Definition 4.

          $\mathsf{NegSamples}(\alpha) \leftarrow \mathsf{NegSamples}(\alpha) \cup \mathsf{GeneralizedSamples}(\beta)$

  Retrain $\mathbf{E}$ in which, for each training step that considers $\alpha \in \mathcal{G}$, $\mathsf{NegSamples}(\alpha)$ is used as negative samples in the loss function, e.g. Equation 1 or Equation 2.

the task that we consider concerns verifying whether a particular triple causes inconsistency, for the target DL when performing the consistency check one does not need to account for the whole KG, but only a small subset of relevant facts. To this end, we define the *relevant sets* as follows.

Definition 2 (Relevant set). *Let $\mathcal{G}$ be a KG and α be a triple. The* relevant set $\mathsf{Relv}(\alpha, \mathcal{G})$ *of α w.r.t. $\mathcal{G}$ is defined as* $\mathsf{Relv}(\alpha, \mathcal{G}) = \{\alpha\} \cup \{\beta \in \mathcal{G} \,|\, \mathsf{Ent}(\beta) \cap \mathsf{Ent}(\alpha) \neq \emptyset\}$.

Example 2. For $\alpha = \langle bosch, locatedIn, john\rangle$ and $\mathcal{G}$ in Fig. 1, we have the following relevant set $\mathsf{Relv}(\alpha, \mathcal{G}) = \{\alpha\} \cup \{\langle john, livesIn, germany\rangle, \langle john, friendOf, bob\rangle, \langle john, type, person\rangle, \langle bosch, type, company\rangle\}$.

The following proposition allows us to reduce the consistency checking of $\alpha \cup \mathcal{G} \cup \mathcal{O}$ to the consistency checking of $\mathsf{Relv}(\alpha, \mathcal{G}) \cup \mathcal{O}$.

Proposition 1. *Let $\mathcal{G}$ be a knowledge graph, $\mathcal{O}$ an ontology such that $\mathcal{G} \cup \mathcal{O}$ is consistent, and α a triple. Then, $\alpha \cup \mathcal{G} \cup \mathcal{O}$ is consistent iff* $\mathsf{Relv}(\alpha, \mathcal{G}) \cup \mathcal{O}$ *is consistent.*

Proof. Since $\mathsf{Relv}(\alpha, \mathcal{G}) \subseteq \mathcal{G}$, we have $\alpha \cup \mathcal{G} \cup \mathcal{O}$ being consistent implies that $\mathsf{Relv}(\alpha, \mathcal{G}) \cup \mathcal{O}$ is also consistent. We start showing the remaining direction by assuming that $\mathsf{Relv}(\alpha, \mathcal{G}) \cup \mathcal{O}$ is consistent and then show that $\alpha \cup \mathcal{G} \cup \mathcal{O}$ is also consistent. Let

$\alpha = \langle s, p, o\rangle$, by Definition 2, we have $\mathsf{Relv}(\alpha, \mathcal{G}) = \mathcal{M}(s, \alpha \cup \mathcal{G}) \cup \mathcal{M}(o, \alpha \cup \mathcal{G})$. Since $\mathcal{G} \cup \mathcal{O}$ is consistent, by Lemma 1, we have $\mathcal{M}(e, \mathcal{G}) \cup \mathcal{O}$ is consistent for every entity in $\mathsf{Ent}(\mathcal{G}) \setminus \{s, o\}$. Since $e \notin \{s, o\}$, we have $\mathcal{M}(e, \mathcal{G}) = \mathcal{M}(e, \alpha \cup \mathcal{G})$, which implies $\mathcal{M}(e, \alpha \cup \mathcal{G}) \cup \mathcal{O}$ is consistent ($\star$). From the assumption that $\mathsf{Relv}(\alpha, \mathcal{G}) \cup \mathcal{O}$ is consistent and $\mathsf{Relv}(\alpha, \mathcal{G}) = \mathcal{M}(s, \alpha \cup \mathcal{G}) \cup \mathcal{M}(o, \alpha \cup \mathcal{G})$, we obtain $\mathcal{M}(s, \alpha \cup \mathcal{G})$ and $\mathcal{M}(o, \alpha \cup \mathcal{G})$ are consistent w.r.t. $\mathcal{O}$ (†). From ($\star$) and (†) we have $\alpha \cup \mathcal{G} \cup \mathcal{O}$ is consistent using Lemma 1. □

Relying on Proposition 1, it is sufficient to check the consistency of a triple α with respect to $\mathcal{G} \cup \mathcal{O}$ using $\mathsf{Relv}(\alpha, \mathcal{G})$ rather than the whole KG. We make use of this property in step (4), and for every prediction produced by the embedding model, we first construct the relevant set for the respective prediction, and then perform the consistency check relying only on the corresponding relevant sets.

Example 3. Assume that the fact $\alpha = \langle bosch, locatedIn, john\rangle$ has been predicted by the embedding model in step (3). Then in the consistency checking step (4) we first construct the relevant set for α as $\mathsf{Relv}(\alpha, \mathcal{G})$ given in Example 2 and check the consistency of $\mathsf{Relv}(\alpha, \mathcal{G}) \cup \mathcal{O}$. Clearly, we have $\mathsf{Relv}(\alpha, \mathcal{G}) \cup \mathcal{O} = \{\langle bosch, locatedIn, john\rangle\} \cup \{\langle john, livesIn, germany\rangle, \langle john, type, person\rangle, \langle john, friendOf, bob\rangle, \langle bosch, type, company\rangle\} \cup \mathcal{O}$ is inconsistent, since $\langle bosch, locatedIn, john\rangle$ and $\{\exists locatedIn^- \sqsubseteq location\} \in \mathcal{O}$ imply that $\langle john, type, location\rangle$, which contradicts the fact that $\langle john, type, person\rangle \in \mathcal{G}$ and $person \sqcap location \sqsubseteq \bot \in \mathcal{O}$. Thus, we have that $\alpha \cup \mathcal{G} \cup \mathcal{O}$ is inconsistent by monotonicity. Proposition 1 further guarantees that it is sufficient to check the consistency of $\alpha \cup \mathcal{G} \cup \mathcal{O}$ this way.

3.3 Negative Sample Generalization

Given each triple of the input KG in the training step, one needs to sample not a single corrupted triple but a set of such triples to train the embedding model at hand. In other words, the inconsistent prediction needs to be *generalized* to obtain a set of similar inconsistent facts within the KG, which ideally have the same structure. Therefore, once an inconsistent prediction for a triple is identified, we proceed with detecting the inconsistency pattern from that prediction and relying on the respective pattern we generate other similar incorrect triples (in step 5 of our method). This allows us to compute sufficient number of negative samples for retraining the embedding model, and to give hints to the embedding model about the wrong patterns that it learned, subsequently avoiding the prediction of similar incorrect triples in next iterations.

A naive approach to obtain the generalized triples of an inconsistent predicted triple, e.g. $\langle s, p, \hat{o}\rangle$, is to replace $\hat{o}$ by another entity o in the input KG such that o has similar KG neighborhood as $\hat{o}$. However, it might happen that only a subset of triples containing $\hat{o}$ is inconsistent w.r.t. the ontology. Therefore, it is sufficient to find such o that it has similar triples as in that subset. This will increase the number of generalized triples as demonstrated in Example 4. To compute a subset of triples of $\hat{o}$ that is inconsistent w.r.t. the ontology, we compute explanations for the inconsistency of $\mathsf{Relv}(\langle s, p, \hat{o}\rangle, \mathcal{G}) \cup \mathcal{O}$.

Example 4. Consider the KG $\mathcal{G}$ and ontology $\mathcal{O}$ as in Figure 1. Assume that $\alpha = \langle bosch, locatedIn, john\rangle$ is the predicted triple, i.e., the embedding model predicted

$john$ as the object entity for the given subject $bosch$ and relation $locatedIn$. The explanation for inconsistency of $\mathsf{Relv}(\alpha, \mathcal{G}) \cup \mathcal{O}$ is $\mathcal{E} = \mathcal{E}_\mathcal{G} \cup \mathcal{E}_\mathcal{O}$, for which it holds that $\mathcal{E}_\mathcal{G} = \{\langle bosch, locatedIn, john\rangle, \langle john, type, person\rangle\}$ and $\mathcal{E}_\mathcal{O} = \{\exists located^- \sqsubseteq location, person \sqcap location \sqsubseteq \bot\}$. Note that there is no other entity in $\mathcal{G}$ that has similar triples as those for $john$. However, if we restrict to the triples in the explanation for inconsistency of $\mathsf{Relv}(\alpha, \mathcal{G}) \cup \mathcal{O}$, then bob has the same neighborhood triple $\langle bob, type, person\rangle$ as $john$ (the predicted triple is ignored). Therefore, we can take $\langle bosch, locatedIn, bob\rangle$ as another negative sample, which together with $\mathcal{G}$ is clearly inconsistent w.r.t. $\mathcal{O}$.

To formally obtain generalized triples as in Example 4, we rely on the notion of *local type* of an entity [16,17,32] as follows.

Definition 3 (Local Types). *Let $\mathbf{T}$ be a set of triples and e an entity occurring in $\mathbf{T}$. Then, the local type of e w.r.t. $\mathbf{T}$, written as $\tau(e, \mathbf{T})$ or $\tau(e)$ when $\mathbf{T}$ is clear from the context, is defined as a tuple $\tau(e) = \langle \tau_i(e), \tau_c(e), \tau_o(e)\rangle$, where $\tau_i(e) = \{p \mid \langle s, p, e\rangle \in \mathcal{G}\}$, $\tau_c(e) = \{t \mid \langle e, type, t\rangle \in \mathcal{G}\}$, and $\tau_o(e) = \{p' \mid \langle e, p', o\rangle \in \mathcal{G}\}$.The local type $t = \langle t_i, t_c, t_o\rangle$ is* smaller than or equal to *the local type $t' = \langle t'_i, t'_c, t'_o\rangle$, written as $t \preceq t'$, iff $t_i \subseteq t'_i, t_c \subseteq t'_c$, and $t_o \subseteq t'_o$.*

Intuitively, a local type of an entity represents a set of types (τ_c) as well as the incoming relations (τ_i) and outgoing relations (τ_o) for that entity in a set of triples.

Example 5 (Example 4 continued). For bob in Fig. 1, we have the local type of bob w.r.t. $\mathcal{G}$ being $\tau(bob) = \langle\{friendOf\}, \{person\}, \{worksAt\}\rangle$. The local type of $john$ w.r.t. $\mathcal{E}_\mathcal{G} \setminus \alpha$ is $\tau(john) = \langle\emptyset, \{person\}, \emptyset\rangle$ and it holds that $\tau(john) \preceq \tau(bob)$.

We now define the set of generalized samples of a given inconsistent predicted triple.

Definition 4 (Generalized Samples). *Let $\mathcal{G}$ be a KG, $\mathcal{O}$ an ontology, and $\alpha = \langle s, p, \hat{o}\rangle$ be a triple in which $\hat{o}$ is predicted by an embedding model given the subject entity s and relation p. Furthermore, let $\mathcal{E} = \mathcal{E}_\mathcal{G} \cup \mathcal{E}_\mathcal{O}$ be an inconsistency explanation of $\mathsf{Relv}(\alpha, \mathcal{G}) \cup \mathcal{O}$. Then, the set of* generalized samples *of α (w.r.t. $\hat{o}$, $\mathcal{E}$, and $\mathcal{G}$) is defined as $\mathsf{GeneralizedSamples}(\alpha, \hat{o}) = \{\langle s, p, o\rangle \mid \tau(\hat{o}, \mathcal{E}_\mathcal{G} \setminus \alpha) \preceq \tau(o, \mathcal{G})\}$. The generalized samples $\mathsf{GeneralizedSamples}(\beta, \hat{s})$ of $\beta = \langle \hat{s}, p, o\rangle$, in which $\hat{s}$ is predicted by an embedding model, is defined analogously. When it is clear from the context, we often write $\mathsf{GeneralizedSamples}(\alpha)$ without mentioning the corresponding entity.*

Example 6 (Example 5 continued). According to Definition 4 and the local types of $john$ and bob computed in Example 5, we have $\mathsf{GeneralizedSamples}(\alpha) = \{\alpha\} \cup \{\langle bosch, LocatedIn, bob\rangle\}$.

The following Lemma guarantees that if a triple is inconsistent (together with the input KG) w.r.t. an ontology $\mathcal{O}$ then all generalized triples of that triple are also inconsistent.

Lemma 2. *Let $\mathcal{G}$ be a KG, $\mathcal{O}$ an ontology, α a triple such that $\mathsf{Relv}(\alpha, \mathcal{G}) \cup \mathcal{O}$ is inconsistent with an explanation $\mathcal{E} = \mathcal{E}_\mathcal{G} \cup \mathcal{E}_\mathcal{O}$, and $\mathsf{GeneralizedSamples}(\alpha)$ is the set of generalized triples of α w.r.t. $\mathcal{E}$, $\mathcal{G}$, and some entity occurring in α. Then, we have $\mathsf{Relv}(\beta, \mathcal{G}) \cup \mathcal{O}$ is inconsistent for every $\beta \in \mathsf{GeneralizedSamples}(\alpha)$.*

Table 2: Knowledge graph statistics.

	LUBM3U	Yago3-10	DBPedia15K
# Entities	127,645	123,182	12,842
# Predicates	28	37	279
# Training Facts	621,516	1,079,040	69,320
# Validation Facts	77,689	5,000	9,902
# Test Facts	77,689	5,000	19,805
# TBox Axioms	325	4,551	3,006

Proof (Sketch). W.l.o.g. let $\alpha = \langle s, p, \hat{o}\rangle$, $\mathsf{GeneralizedSamples}(\alpha)$ is w.r.t. $\hat{o}$, and $\beta = \langle s, p, o\rangle$. Using the result in [32], one can show that if $\langle s, p, \hat{o}\rangle \in \mathcal{E}_\mathcal{G}$ then $\mathcal{E}_\mathcal{G}$ does not contain $\langle s', p, o\rangle$, where $s \neq s'$ due to the minimality of explanations. Together with the condition $\tau(\hat{o}) \preceq \tau(o)$, we can construct a homomorphism from $\mathsf{Relv}(\alpha, \mathcal{G})$ to $\mathsf{Relv}(\beta, \mathcal{G})$, which implies that $\mathsf{Relv}(\beta, \mathcal{G}) \cup \mathcal{O}$ is inconsistent. □

We now describe the details of step (5). For each predicted triple that is inconsistent w.r.t. the input KG and the ontology, we compute explanations for inconsistency, and for each such explanation, we obtain the generalized triples using Def. 4. These generalized triples are then used as negative samples to retrain the embedding model.

4 Experiments

We have implemented the proposed method in a prototype system *ReasonKGE* and evaluated its performance on the commonly used datasets enriched with ontologies. In this section, we present the results of the evaluation in terms of the impact of our method on the quality of fact predictions compared to the baselines.

4.1 Experimental Setup

Datasets. Among commonly used datasets for evaluating embedding models, we chose those datasets that are equipped with ontologies. More specifically, the following datasets with their respective ontologies have been selected.

- **LUBM3U:** A synthesized dataset derived from the Lehigh University Benchmark [18]. The ontology describing the university domain contains 325 axioms. The respective KG stores data for 3 universities.
- **Yago3-10:** A subset of the widely used Yago dataset. We use the ontology with 4551 axioms introduced in [31] based on Yago schema and class hierarchy.
- **DBPedia15K:** A subset of DBPedia KG proposed in [24]. We exploit the general DBPedia ontology enriched with axioms reflecting the disjointness of classes. The ontology comprises of 3006 axioms.

The statistics of the respective datasets is presented in Table 2.

Embedding Models. To demonstrate the benefits of the proposed iterative ontology-driven negative sampling, we apply our method over the following widely used embeddings: ComplEx [33] and TransE [9]. These models have been selected as prominent

examples of translation-based and linear-map embeddings. While more recent embedding models exist in the literature, as shown in [29] classical embeddings are in fact very competitive when combined with effective parameter search. Thus, as baselines we have selected the most widely used and popular embedding models with the best parameters found using the LibKGE library [29].

We also consider another baseline [11] that incorporates background knowledge into the embedding models. We refer to such technique as *static sampling* because in contrast to our proposed *dynamic sampling* method, the approach from [11] generates the negative samples for all triples of the KG in the pre-processing step. Since the authors of [11] only mentioned that they utilized such ontology axioms as *Domain, Range, Functional,* and *Disjointness*, but have not described the exact procedure of how these have been exploited for generating negative samples, we have implemented such static sampling strategy based on our best knowledge, and present the details of the implementation in the extended version.[3]

Measures. We evaluate the performance of the embedding models in terms of the traditional metrics i.e ***MRR*** and ***Hits@k*** in the filtered setting [9]. In addition, we also compute the proportion of inconsistent facts (***Inc@k***) ranked in the *top-k* predictions produced by the presented methods. The measure ***Inc@k*** intuitively reflects how well the model is capable of avoiding inconsistent predictions (the lower the better).

System Configuration. In the experiments, we used HermiT [15] as the reasoner and the explanation method in [20] to compute inconsistency explanations. We run *ReasonKGE* for multiple iterations. In every iteration, the model is trained for $n = 100$ epochs during which, for each subject and object of a triple, $m >= 1$ negative examples are generated. We exploit the optimal value of m tuned for the respective baseline model. In the first iteration, m negative samples are generated using the default random sampling strategy[4]. In the subsequent iterations, we use the trained model to obtain the top $k = 1$ subject and object predictions and compute the inconsistent negative samples to be used for the next iteration of the embedding training as described in Section 3. The number m of negative samples for the next iteration is dynamically computed based on the statistical mean of the size of the generalized samples sets as an indicator.

4.2 Results

The results of the conducted experiments illustrate the benefit of *ReasonKGE* in producing higher quality predictions with less inconsistencies compared to the baselines.

Link Prediction Quality. Table 3 reports the results for the link prediction task obtained by our method and the baselines. Both TransE and ComplEx were trained using the default random sampling strategy [9], the *static sampling* [11], and using *ReasonKGE* for 3 iterations. For fair comparison, the number of the training epochs was kept the same as for *ReasonKGE* in all cases (i.e., 300 epochs).

One can observe that reasoning-based sampling consistently achieves better results than random sampling for training all considered embeddings on all KGs. For the

[3] available at https://github.com/nitishajain/ReasonKGE

[4] For each triple the subject (resp. object) is randomly perturbed to obtain m samples [9]

Table 3: Link prediction results

Model	KG	Default Training			Static Sampling			*ReasonKGE*		
		MRR	*Hits@1*	*Hits@10*	*MRR*	*Hits@1*	*Hits@10*	*MRR*	*Hits@1*	*Hits@10*
TransE	**LUBM3U**	0.119	0.069	0.214	0.125	**0.082**	0.213	**0.135**	0.079	**0.256**
	Yago3-10	0.226	0.044	0.537	0.351	0.183	0.621	**0.367**	**0.197**	**0.629**
	DBPedia15k	0.109	0.061	0.206	0.101	0.073	0.254	**0.118**	**0.101**	**0.299**
ComplEx	**LUBM3U**	0.159	0.119	0.242	0.181	0.136	0.276	**0.233**	**0.195**	**0.313**
	Yago3-10	0.482	0.400	0.643	0.515	0.431	0.665	**0.530**	**0.453**	**0.668**
	DBPedia15k	0.099	0.061	0.174	0.098	0.107	0.193	**0.115**	**0.125**	**0.221**

Table 4: Ratio of inconsistent predictions (the lower, the better).

Model	KG	Prediction	Default Training		Static Sampling		ReasonKGE	
			Inc@1	*Inc@10*	*Inc@1*	*Inc@10*	*Inc@1*	*Inc@10*
TransE	**LUBM3U**	*subject*	0.169	0.270	0.428	0.250	**0.037**	**0.133**
		object	0.095	0.097	0.212	0.104	**0.005**	**0.007**
	YAGO3-10	*subject*	**0.075**	0.280	0.629	0.492	**0.075**	**0.273**
		object	0.026	0.136	0.114	**0.089**	**0.020**	0.117
	DBPedia15K	*subject*	0.311	0.652	0.401	0.663	**0.217**	**0.585**
		object	0.413	0.538	0.428	0.544	**0.170**	**0.460**
ComplEx	**LUBM3U**	*subject*	0.041	0.097	0.177	0.136	**0.036**	**0.069**
		object	0.008	0.012	**0.003**	**0.007**	0.005	**0.007**
	YAGO3-10	*subject*	0.113	0.198	0.169	**0.128**	**0.071**	0.143
		object	0.037	0.115	0.065	0.084	**0.015**	**0.074**
	DBPedia15K	*subject*	0.488	0.667	0.436	0.695	**0.344**	**0.583**
		object	0.397	0.585	0.365	0.528	**0.318**	**0.533**

Yago3-10 dataset the improvements are the most significant, achieving more than 10% enhancement for all measures over TransE. This indicates the advantage of ontology-based reasoning for enhancing the existing KG embeddings.

The comparison of our dynamic sampling method against static sampling [11] presented in Table 3 reveals that *ReasonKGE* outperforms the *static sampling* approach in almost all cases, which illustrates the benefits of exploiting inconsistent predictions as negative samples dynamically using our method, as opposed to their pre-computation.

By keeping the same training configuration and total number of training epochs, we ensure that the reflected performance gains are not merely due to additional training steps, but rather a result of the proposed reasoning-based approach.

Consistency of Predictions. In Table 4, we measure the proportion of inconsistent facts that were obtained when retrieving *top-k* ($k = \{1, 10\}$) predictions for the triples in the test set. We report the inconsistency values both for the prediction of the *subject* and the *object* of the triple separately. From the results, we can observe that for all models in the majority of the cases *ReasonKGE* managed to reduce the ratio of inconsistent predictions over the test sets compared to the results of training the models using *default*

random and *static* sampling. This illustrates that the proposed procedure is effective for improving embeddings with respect to the overall consistency of their predictions.

5 Related Work

Negative Sampling Strategies. The closest to our method is the work [11], in which ontologies are used to generate a selection of negative samples in the pre-processing step for training a certain embedding model. While we use this pre-processing based sampling as a baseline for comparison in Section 4, our method is different in that we do not generate all negative examples at once, but rather compute them iteratively on demand relying on the inconsistent predictions produced by the given embedding. The major advantage of the *ReasonKGE* method compared to [11] is the dynamic and adaptable nature of negative sample generation, wherein, the method is able to specifically target the weaknesses of the previously trained model by leveraging inconsistent predictions to derive negative samples, and use them for re-training of the model in next iterations. This is in contrast to the process of precomputing negative samples using ontology axioms as suggested in in [11].

Another related method is concerned with type-constrained negative sampling [22]. Given a triple from the KG, the negative candidates (subjects or objects) are mined by constraining the entities to belong to the same type as that of the subject or object of the original triple. However, unlike our inconsistency-driven method, the typed-constrained sampling can generate false negatives. This sampling method can be in principle also used as the starting point for our method instead of the random sampling.

More distant random negative samplings generate false candidate triples based on the (local) closed world assumption [27]. Alternatives include Distributional Negative Sampling (DNS) [12] and its variation [2], where during training, given a positive triple, negative examples are generated by replacing it's entity with other similar entities. Unlike in our method, no ontological information is considered in these sampling strategies. The same holds for the triple perturbation or triple corruption approach [30].

Nearest Neighbor and *Near Miss sampling* [21] resp. exploit a pre-trained embedding model for generating negative samples by selecting triples that are close to the positive target triple in vector space. Intuitively, this strategy is supposed to help the model to learn to discriminate between positives and negatives that are very similar to each other. These approaches are similar to ours, in that the embedding training procedure itself is exploited for the generation of negative samples. However, in [21] no ontological knowledge is taken into account which is in contrast to our work.

Another research direction concerns making use of Generative Adversarial Networks (GANs) [39,35,10] for negative sampling. The work [1] presents structure-aware negative sampling (SANS), which utilizes the graph structure by selecting negative samples from a node's neighborhood. The NSCaching sampling method [40] suggests to sample negatives from a cache that can dynamically hold large-gradient samples. While in these works negative triples are updated dynamically like in our method, these approaches are totally different from ours, as they rely purely on the machine learning techniques, and do not consider any extra ontological knowledge. Thus, the proposals are rather complementary in nature.

Integration of Ontological Knowledge into KG Embeddings. Another relevant line of work concerns the integration of ontological knowledge directly into embedding models (e.g., [14,11,25,41,37,22,19]), which is typically done via changes in the loss function, rather than negative sampling.

For example, a related method *Embed2Reason (E2R)* has been proposed by Garg *et al.* [14]. *E2R* relies on the quantum logic, and injects ontology axioms via the loss function, by summing up the terms relevant for these axioms. However, it is unclear how this method captures the interaction among the axioms, which is often the reason for inconsistency. Since the available code of [14] only supports a limited set of axioms, i.e., `SubClassOf, SubPropertyOf, Domain, Range`, which are insufficient for generating inconsistencies, we could not perform a direct comparison of our method to *E2R*. Note that in general, our method is conceptually different from *E2R*. Indeed, in contrast to [14], we focus on ontology-driven targeted improvements of the negative sampling procedure with the goal of teaching a given embedding model to make only consistent predictions, and interactions among the axioms are key to our method. Moreover, our proposed approach can be built on top of any embedding model including [14], making the two methods rather complementary in nature.

The recent work [37] suggests to exploit ontological reasoning for verifying consistency of predictions made by a machine learning method (e.g., embedding or rule learning). However, instead of feeding inconsistent predictions back to the given embedding model, the authors propose to get rid of them and feed other consistent predictions along with the original KG as input to a further KG completion method. In [19] the ontology is explicitly included in the training data to jointly embed entities and concepts. By treating the ontology and KG in the same way, only very restricted ontological knowledge is accounted for.

Our work can be also positioned broadly within neural-symbolic methods, and we refer the reader to [38,6] for other less related neural-symbolic approaches.

Inconsistency in Ontologies. The problems of explaining and handling inconsistency in ontologies have been tackled in different settings [20,8,28,32,7,23]. However, typically these works focus on detecting inconsistency [20,8], scalable reasoning [28,32], or performing reasoning in the presence of such inconsistency [7,23] assuming that the KG is constructed and complete. In other words, these approaches deal purely with data cleaning rather than KG completion. In contrast, our method integrates the reasoning process into the embedding models to improve the accuracy of predicted triples.

6 Conclusion

We have presented a method for ontology-driven negative sampling that proceeds in an iterative fashion by providing at each iteration negative samples to the embedding model on demand from its inconsistent predictions along with their generalizations. The main takeaway message of this work is that targeted negative example generation is beneficial for training the model to predict consistent facts as witnessed by our empirical evaluation on state-of-the-art KGs equipped with ontologies.

While in this work we focused on ontologies in DL-Lite, our method can be adapted to support more expressive ontologies. In this case, the soundness will still be preserved,

but the completeness of the generalized negative sampling step might not be theoretically guaranteed, i.e., not all possible similar negative samples will be obtained based on a given inconsistent prediction of the embedding model. In practice, this will likely have a small impact on the effectiveness of our method, since the majority of useful negative samples will anyway be generated.

There are several exciting directions for future work. First, integrating the developed negative sampling method into the combination of rule learning and embedding-based approaches [37] for KG completion is promising. Second, extending the proposed approach to target other more expressive ontology languages is a relevant future direction. Last but not least, adapting our method to jointly clean and complete KGs can be helpful for facilitating the automatic KG curation.

References

1. Ahrabian, K., Feizi, A., Salehi, Y., Hamilton, W.L., Bose, A.J.: Structure-aware negative sampling in knowledge graphs. In: EMNLP 2020. pp. 6093–6101 (2020)
2. Alam, M.M., Jabeen, H., Ali, M., Mohiuddin, K., Lehmann, J.: Affinity dependent negative sampling for knowledge graph embeddings. In: (DL4KG2020) - (ESWC 2020) (2020)
3. Artale, A., Calvanese, D., Kontchakov, R., Zakharyaschev, M.: The DL-Lite family and relations. CoRR **abs/1401.3487** (2014)
4. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.G.: Dbpedia: A nucleus for a web of open data. In: ISWC. pp. 722–735 (2007)
5. Baader, F., Horrocks, I., Sattler, U.: Description logics. In: Hb. on Ontol., pp. 21–43 (2009)
6. Bianchi, F., Rossiello, G., Costabello, L., Palmonari, M., Minervini, P.: Knowledge graph embeddings and explainable AI. In: Tiddi, I., Lécué, F., Hitzler, P. (eds.) KGs for XAI: Foundations, Applications and Challenges, vol. 47, pp. 49–72. IOS Press (2020)
7. Bienvenu, M.: A short survey on inconsistency handling in ontology-mediated query answering. Künstliche Intell. **34**(4), 443–451 (2020)
8. Bischof, S., Krötzsch, M., Polleres, A., Rudolph, S.: Schema-agnostic query rewriting in SPARQL 1.1. In: ISWC. pp. 584–600 (2014)
9. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: NeurIPS. pp. 2787–2795 (2013)
10. Cai, L., Wang, W.Y.: KBGAN: adversarial learning for knowledge graph embeddings. In: NAACL-HLT 2018. pp. 1470–1480 (2018)
11. d'Amato, C., Quatraro, N.F., Fanizzi, N.: Injecting background knowledge into embedding models for predictive tasks on knowledge graphs. In: ESWC. p. to appear (2021)
12. Dash, S., Gliozzo, A.: Distributional negative sampling for knowledge base completion. CoRR **abs/1908.06178** (2019)
13. Dettmers, T., Minervini, P., Stenetorp, P., Riedel, S.: Convolutional 2d knowledge graph embeddings. In: AAAI. pp. 1811–1818 (2018)
14. Garg, D., Ikbal, S., Srivastava, S.K., Vishwakarma, H., Karanam, H.P., Subramaniam, L.V.: Quantum embedding of knowledge for reasoning. In: Neurips. pp. 5595–5605 (2019)
15. Glimm, B., Horrocks, I., Motik, B., Stoilos, G., Wang, Z.: Hermit: An OWL 2 reasoner. J. Autom. Reasoning **53**(3), 245–269 (2014)
16. Glimm, B., Kazakov, Y., Liebig, T., Tran, T.K., Vialard, V.: ISWC. pp. 180–195 (2014)
17. Glimm, B., Kazakov, Y., Tran, T.: Ontology materialization by abstraction refinement in horn SHOIF. In: AAAI. pp. 1114–1120 (2017)
18. Guo, Y., Pan, Z., Heflin, J.: LUBM: A benchmark for OWL knowledge base systems. J. Web Semant. **3**(2-3), 158–182 (2005)

19. Hao, J., Chen, M., Yu, W., Sun, Y., Wang, W.: Universal representation learning of knowledge bases by jointly embedding instances and ontological concepts. In: KDD. pp. 1709–1719 (2019)
20. Horridge, M., Parsia, B., Sattler, U.: Explaining inconsistencies in owl ontologies. In: Scalable Uncertainty Management. pp. 124–137 (2009)
21. Kotnis, B., Nastase, V.: Analysis of the impact of negative sampling on link prediction in knowledge graphs. CoRR **abs/1708.06816** (2017), http://arxiv.org/abs/1708.06816
22. Krompaß, D., Baier, S., Tresp, V.: Type-constrained representation learning in knowledge graphs. In: ISWC. pp. 640–655 (2015)
23. Lembo, D., Lenzerini, M., Rosati, R., Ruzzi, M., Savo, D.F.: Inconsistency-tolerant query answering in ontology-based data access. J. Web Semant. **33**, 3–29 (2015)
24. Liu, Y., Li, H., Garcia-Duran, A., Niepert, M., Onoro-Rubio, D., Rosenblum, D.S.: Mmkg: multi-modal knowledge graphs. In: ESWC. pp. 459–474 (2019)
25. Minervini, P., Demeester, T., Rocktäschel, T., Riedel, S.: Adversarial sets for regularising neural link predictors. In: UAI (2017)
26. Motik, B., Patel-Schneider, P.F., Grau, B.C.: OWL 2 Web Ontology Language Direct Semantics (Second Edition). Tech. rep. (2012), https://www.w3.org/TR/owl-direct-semantics/
27. Nickel, M., Murphy, K., Tresp, V., Gabrilovich, E.: A review of relational machine learning for knowledge graphs. Proc. IEEE **104**(1), 11–33 (2016)
28. Paulheim, H., Gangemi, A.: Serving dbpedia with dolce – more than just adding a cherry on top. In: The Semantic Web - ISWC 2015. pp. 180–196. Springer (2015)
29. Ruffinelli, D., Broscheit, S., Gemulla, R.: You CAN teach an old dog new tricks! on training knowledge graph embeddings. In: ICLR (2020)
30. Socher, R., Chen, D., Manning, C.D., Ng, A.Y.: Reasoning with neural tensor networks for knowledge base completion. In: NIPS. pp. 926–934 (2013)
31. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: A Core of Semantic Knowledge. In: WWW
32. Tran, T., Gad-Elrab, M.H., Stepanova, D., Kharlamov, E., Strötgen, J.: Fast computation of explanations for inconsistency in large-scale kgs. In: WWW 2020. pp. 2613–2619 (2020)
33. Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., Bouchard, G.: Complex embeddings for simple link prediction. In: ICML. pp. 2071–2080 (2016)
34. Vrandecic, D., Krötzsch, M.: Wikidata: a free collaborative knowledgebase. Commun. ACM **57**(10), 78–85 (2014)
35. Wang, P., Li, S., Pan, R.: Incorporating GAN for negative sampling in knowledge representation learning. In: AAAI. pp. 2005–2012 (2018)
36. Wang, Q., Mao, Z., Wang, B., Guo, L.: Knowledge graph embedding: A survey of approaches and applications. IEEE Trans. Knowl. Data Eng. **29**(12), 2724–2743 (2017)
37. Wiharja, K., Pan, J.Z., Kollingbaum, M.J., Deng, Y.: Schema aware iterative knowledge graph completion. J. Web Semant. **65**, 100616 (2020)
38. Zhang, J., Chen, B., Zhang, L., Ke, X., Ding, H.: Neural-symbolic reasoning on knowledge graphs. CoRR **abs/2010.05446** (2020)
39. Zhang, Y., Yao, Q., Chen, L.: Efficient, simple and automated negative sampling for knowledge graph embedding. CoRR **abs/2010.14227** (2020), https://arxiv.org/abs/2010.14227
40. Zhang, Y., Yao, Q., Shao, Y., Chen, L.: Nscaching: Simple and efficient negative sampling for knowledge graph embedding. In: ICDE. pp. 614–625 (2019)
41. Ziegler, K., Caelen, O., Garchery, M., Granitzer, M., He-Guelton, L., Jurgovsky, J., Portier, P., Zwicklbauer, S.: Injecting semantic background knowledge into neural networks using graph embeddings. In: 26th IEEE, WETICE. pp. 200–205 (2017)

Table 5: Link prediction results for different iterations

Model	KG	*ReasonKGE*- Iteration 2				*ReasonKGE*- Iteration 3			
		MRR	*Hits@1*	*Hits@3*	*Hits@10*	*MRR*	*Hits@1*	*Hits@3*	*Hits@10*
TransE	**LUBM3U**	0.133	0.078	0.159	0.242	0.135	0.079	0.162	0.256
	Yago3-10	0.356	0.184	0.493	0.627	0.367	0.197	0.511	0.629
	DBPedia15k	0.116	0.091	0.130	0.287	0.118	0.101	0.132	0.299
ComplEx	**LUBM3U**	0.229	0.190	0.237	0.310	0.233	0.195	0.240	0.313
	Yago3-10	0.521	0.442	0.569	0.664	0.530	0.453	0.577	0.668
	DBPedia15k	0.111	0.119	0.154	0.216	0.115	0.125	0.162	0.221

7 Extended Experiments

7.1 Intermediate Training Results

In this section, we present the complete results obtained by the *ReasonKGE* method at all iterations. As explained in Section 4.1, in the first iteration the negative samples generated using the default random sampling technique are exploited for training the embedding model. Thereafter, further iterations leverage the trained model from the previous iteration to predict subjects for given relations and objects, as well as similarly, to predict objects for given relations and subjects. Predicted triples that are found to be inconsistent w.r.t. the existing KG and ontology are then subsequently used for the generation of further negative samples for the next round of model training. This process is repeated for multiple iterations until no significant improvement in the performance of the embedding model is observed.

In Table 5 we present the results of *all* iterations of our method. With every iteration, the model is trained for additional 100 epochs, i.e., in the second iteration the model training has been performed for 200 epochs, while in the third iteration altogether for 300 epochs respectively.

It can be seen that the improvement from iteration 2 to iteration 3 is below approximately 1% for all datasets. Therefore, in our experiments reported in Table 3 of Section 4 the training has been stopped at the third iteration, and the best results obtained have been compared to the results for the default training approach run for the same number of epochs (i.e., 300).

In general, the differences in the results of the model obtained with the increasing number of iterations can be used as a stopping criteria for finalizing the training process, i.e., the small difference witnesses the convergence of the training process.

7.2 Static Sampling vs. *ReasonKGE* Negative Sampling

As discussed in Section 4.2, we have performed the experiments to compare our *ReasonKGE* method with related approaches for generating ontology-aware negative samples proposed in previous work [11]. The major advantage of the *ReasonKGE* method compared to [11] is the dynamic and adaptable nature of negative sample generation, wherein, the method is able to specifically target the weaknesses of the previously trained model by leveraging inconsistent predictions to derive negative samples and use them for re-training of the model in next iterations. This is in contrast to the process of precomputing negative samples using ontology axioms [11], to which we refer as *static sampling*.

Since the authors in [11] only mentioned that they utilized ontology axioms such as *Domain, Range, Functional,* and *Disjointness* without describing the exact procedure of how these were actually exploited to generate negative samples, we implemented such static sampling strategy

Algorithm 2: Precomputing negative samples using ontology axioms

Input : A knowledge graph $\mathcal{G}$, and an ontology $\mathcal{O}$
Output: A set of negative samples $\mathsf{NegSamples}(\langle s, p, o\rangle)$ for each triple $\langle s, p, o\rangle$ in $\mathcal{G}$

/* Compute classes/types for each entity in $\mathcal{G}$ */
foreach *entity* e *occurring in* $\mathcal{G}$ **do**
  $\mathsf{TypeSet}(e) \leftarrow \emptyset$
  Compute local type $\tau(e) = \langle \tau_i(e), \tau_c(e), \tau_o(e)\rangle$ of e w.r.t. $\mathcal{G}$
  /* Compute super classes/types of e */
  $\tau'_c(e) = \{B \mid \mathcal{O} \models A \sqsubseteq B,$ and $A \in \tau_c(e)\}$.
  /* Compute incoming and outgoing super-properties/relations of e by calling a reasoner */
  $\tau'_i(e) = \{S \mid \mathcal{O} \models R \sqsubseteq S,$ and $R \in \tau_i(e)\}$
  $\tau'_o(e) = \{S \mid \mathcal{O} \models R \sqsubseteq S,$ and $R \in \tau_o(e)\}$
  /* Calculate the TypeSet (e) for the entity e */
  $\mathsf{TypeSet}(e) = \tau'_c(e) \cup \{A \mid \mathcal{O} \models DomainOf(R) \sqsubseteq A, R \in \tau'_o(e)\} \cup \{B \mid \mathcal{O} \models RangeOf(P) \sqsubseteq B, P \in \tau'_i(e)\}$

/* Compute the set of corrupted entities for e */
foreach *entity* e *occurring in* $\mathcal{G}$ **do**
  $\mathsf{DisjointType}(e) \leftarrow \emptyset$
  foreach $A \in \mathsf{TypeSet}(e)$ **do**
    $\mathsf{DisjointType}(e) = \mathsf{DisjointType}(e) \cup \{B \mid \mathcal{O} \models A \sqcap B \sqsubseteq \bot\}$
  $\mathsf{CorruptedEntities}(e) \leftarrow \{e' \mid \mathsf{DisjointType}(e) \cap \mathsf{TypeSet}(e') \neq \emptyset\}$

/* Compute negative samples for each triple in $\mathcal{G}$ */
foreach $\langle s, p, o\rangle \in \mathcal{G}$ **do**
  $\mathsf{NegSamples}(\langle s, p, o\rangle) \leftarrow \{\langle s', p, o\rangle \mid s' \in \mathsf{CorruptedEntities}(s)\} \cup \{\langle s, p, o'\rangle \mid o' \in \mathsf{CorruptedEntities}(o)\}$

based on our best knowledge and present detailed steps in Algorithm 2. Intuitively, for each entity e we first compute both asserted classes (explicitly known in the input KG) and derived classes (using ontology axioms together with the input KG) to which the entity belongs. Based on the *DisjointClasses* axioms, we then identify a set of entities that belong to any class known to be disjoint with one of the classes of e and refer to these entities as a set of *corrupted entities* for e. Such corrupted entities for e are then subsequently used to generate negative samples in the training step that considers triples in which e occurs.

One of the main steps of Algorithm 2 is to compute the TypeSet of an entity, which is the set of classes to which the entity belongs. For each entity in the KG, the *local type* of the entity (as defined in 3) is leveraged. A TypeSet of each entity is created as follows - the set of types (τ_c) is extended by adding the respective superclasses (parent types) of the classes present in (τ_c). For the set of incoming relations (τ_i), the super-relations of all incoming relations in the set (τ_i) are calculated, and the classes belonging to the range of these relations are extracted. Similarly, for the set of outgoing relations (τ_o), the super-relations of all outgoing relations in the set (τ_o) are computed and the classes belonging to the domain of these relations are extracted. The TypeSet is constructed by taking the union of the extracted classes. Thereafter, the set of classes that are disjoint with any of the classes in TypeSet are retrieved. For each entity e, every entity e' is added to the set of corrupted entities of e if e' has some type that is disjoint with at least one type in TypeSet of e.

Table 6: Link predictions results for *static sampling* and *ReasonKGE*

Model	KG	Static Sampling				*ReasonKGE*			
		MRR	*Hits@1*	*Hits@3*	*Hits@10*	*MRR*	*Hits@1*	*Hits@3*	*Hits@10*
TransE	**LUBM3U**	0.125	0.082	0.140	0.213	0.135	0.079	0.162	0.256
	Yago3-10	0.351	0.183	0.482	0.621	0.367	0.197	0.511	0.629
	DBPedia15k	0.101	0.073	0.116	0.254	0.118	0.101	0.132	0.299
ComplEx	**LUBM3U**	0.181	0.136	0.190	0.276	0.233	0.195	0.240	0.313
	Yago3-10	0.515	0.431	0.564	0.665	0.530	0.453	0.577	0.668
	DBpedia15k	0.098	0.107	0.129	0.193	0.115	0.125	0.162	0.221

Table 7: Ratio of inconsistent predictions (the lower, the better).

Model	KG	Prediction	Static Sampling		ReasonKGE	
			Inc@1	*Inc@10*	*Inc@1*	*Inc@10*
TransE	**LUBM3U**	*subject*	0.428	0.250	0.037	0.133
		object	0.212	0.104	0.005	0.007
	YAGO3-10	*subject*	0.629	0.492	0.075	0.273
		object	0.114	0.089	0.020	0.117
	DBpedia15K	*subject*	0.401	0.663	0.217	0.585
		object	0.428	0.544	0.170	0.460
ComplEx	**LUBM3U**	*subject*	0.177	0.136	0.036	0.069
		object	0.003	0.007	0.005	0.007
	YAGO3-10	*subject*	0.169	0.128	0.071	0.143
		object	0.065	0.084	0.015	0.074
	DBpedia15K	*subject*	0.436	0.695	0.344	0.583
		object	0.365	0.528	0.318	0.533

During the training steps of embedding models, for each KG triple, the set of negative samples can be obtained by replacing the subject or object entity with the corresponding set of precomputed corrupted entities.

Results. Table 6 presents the link prediction performance using the static sampling approach and our proposed *ReasonKGE* method for the LUBM3 and Yago3-10 datasets. For fair comparison, the numbers are reported on the models that were trained for 300 epochs with the static samples. The results with *ReasonKGE* are found to be consistently better for the considered datasets and embeddings techniques, especially for ComplEx model. Table 7 reports on the inconsistency of predictions obtained by the static sampling in comparison to *ReasonKGE* method. Static sampling predicts higher ratio of inconsistent predictions overall, which is reduced by the *ReasonKGE* model. These results highlight the benefit of the dynamic generation of negative samples as advocated by our method.