Journal of Artificial Intelligence Research 68 (2020) 503–540 Submitted 09/2019; published 07/2020

Ontology Reasoning with Deep Neural Networks

Patrick Hohenecker PATRICK.HOHENECKER@CS.OX.AC.UK
Department of Computer Science
University of Oxford, UK

Thomas Lukasiewicz THOMAS.LUKASIEWICZ@CS.OX.AC.UK
Department of Computer Science
University of Oxford, UK

arXiv:1808.07980v4 [cs.AI] 8 Jan 2021

Abstract

The ability to conduct logical reasoning is a fundamental aspect of intelligent human behavior, and thus an important problem along the way to human-level artificial intelligence. Traditionally, logic-based symbolic methods from the field of knowledge representation and reasoning have been used to equip agents with capabilities that resemble human logical reasoning qualities. More recently, however, there has been an increasing interest in using machine learning rather than logic-based symbolic formalisms to tackle these tasks. In this paper, we employ state-of-the-art methods for training deep neural networks to devise a novel model that is able to learn how to effectively perform logical reasoning in the form of basic ontology reasoning. This is an important and at the same time very natural logical reasoning task, which is why the presented approach is applicable to a plethora of important real-world problems. We present the outcomes of several experiments, which show that our model is able to learn to perform highly accurate ontology reasoning on very large, diverse, and challenging benchmarks. Furthermore, it turned out that the suggested approach suffers much less from different obstacles that prohibit logic-based symbolic reasoning, and, at the same time, is surprisingly plausible from a biological point of view.

1. Introduction

Implementing human-like logical reasoning has been among the major goals of artificial intelligence research ever since, and has recently also enjoyed increasing attention in the field of machine learning. However, a noticeable commonality of previous approaches in this area is that they, with a few very recent exceptions (Serafini & d'Avila Garcez, 2016; Cai et al., 2017; Rocktäschel & Riedel, 2017; Cingillioglu & Russo, 2018; Dai et al., 2018; Evans et al., 2018; Manhaeve et al., 2018), entertain a quite informal notion of reasoning, which is often simply identified with a particular kind of prediction task. This contrasts the (traditional) understanding of reasoning as an application of mathematical proof theory, like it is used in the context of logic-based knowledge representation and reasoning (KRR). Interestingly, however, it can be observed that, under certain provisions, even the best reasoning models based on machine learning are still not in a position to compete with their symbolic counterparts. To close this gap between learning-based and KRR methods, we develop a novel model architecture, called *recursive reasoning network* (RRN), which makes use of recent advances in the area of deep neural networks (Bengio, 2009). By design, this model is much closer to logic-based symbolic methods than most of the other learning-based approaches, but the fact that it employs machine learning allows for overcoming many of the

©2020 AI Access Foundation. All rights reserved.

obstacles that we encounter with KRR methods in practice. Furthermore, while it might seem paradoxical to elect logic-based symbolic methods as a starting point for implementing human-like logical reasoning, blending those concepts with deep learning leads to compelling results, just as it opens up interesting new perspectives on the considered problem.

Articles on reasoning based on machine learning commonly assume a particular application, such as reasoning about natural language (Henaff et al., 2017; Santoro et al., 2017) or visual inputs (Santoro et al., 2017). Here, we take a different approach, and consider a logic-based formal reasoning problem as a starting point instead. The choice of the particular problem turns out to be a critical one, though. While logic-based formal reasoning defines a certain conceptual frame, the specific nature of and extent to which inferences may be drawn are highly dependent on the concrete formalism employed. Therefore, it is generally sensible to choose an approach that presents the right balance between expressiveness on the one hand and complexity on the other hand. This criterion as well as its vast importance in practice made us look into the problem of ontology reasoning with OWL 2 RL. Ontology reasoning refers to a common scenario where the inference rules to be used for reasoning, called the ontology in this context, are specified along with the factual information that we seek to reason about. Figure 1 provides an example of this setting. The rationale behind this separation of ontology and facts is that it allows for adopting the same set of rules for reasoning about different, independent data. What makes this task particularly interesting, though, is that the application of generic rules to concrete problem settings happens to describe a reoccurring pattern in all of our lives, which is why human beings are in general very good at handling problems of this kind. Furthermore, ontology reasoning is an incredibly pliant tool, which allows for modeling a plethora of different scenarios, and as such meets our desire for a system that is applicable to a wide range of applications.

One may ask why we would like to set about this problem by means of machine learning in the first place. For one thing, such combinations of deep learning technologies with symbolic methods for logical reasoning are commonly regarded as a prerequisite for further substantial progress in AI. In particular, they are expected to allow for leveraging symbolic background knowledge in learning deep neural networks (and so for knowledge transfer and for learning from smaller amounts of data) as well as explainable symbolic inference in computing predictions for improved explainability of the learned neural systems. Furthermore, most of the KRR formalisms that are used for reasoning today are rooted in symbolic logic, and thus, as mentioned above, employ mathematical proof theory to answer queries about a given problem. However, while this, in theory, allows for answering any kind of (decidable) question accurately, most of these approaches suffer from a number of issues in practice, like difficulties with handling incomplete, conflicting, or uncertain data, to name just a few. In contrast to this, machine learning models are often highly scalable, more resistant to disturbances in the data, and capable of providing predictions even if the formal effort fails. There is one salient aspect, though, that KRR methods benefit from compared to learning-based approaches: every conclusion derived via formal inference is correct with certainty, and, under optimal circumstances, formal reasoning identifies all inferences that can be drawn altogether. These characteristics do, in general, not apply to methods of machine learning. However, »*optimal*« is the operative word right here, as these advantages can often not come into play due to the obstacles mentioned above. By employing state-of-the-art techniques of deep learning, we aim to manage the balancing act between approximating the highly

Ontology:	human(X) ← holds(X,_)	Only human beings can hold things.
	object(Y) ← holds(_,Y)	Only objects can be held.
	⊥ ← human(X) ∧ object(X)	Objects are not human beings and vice versa.
	isAt(Y,Z) ← holds(X,Y) ∧ isAt(X,Z)	Objects are at the same location as the one holding them.
	⊥ ← isAt(X,Y) ∧ isAt(X,Z) ∧ Y≠Z	Nobody/nothing can be at two locations at the same time.
Facts:	holds(mary,apple)	Mary holds the apple.
	isAt(mary,kitchen)	Mary is in the kitchen.
Queries:	?human(apple)	Is the apple a human being? (Evaluates to **false**.)
	?isAt(apple,kitchen)	Is the apple in the kitchen? (Evaluates to **true**.)
	?isAt(mary,bedroom)	Is Mary in the bedroom? (Evaluates to **false**.)

Figure 1: This figure provides a simple example of an ontology, which was inspired by the well-known bAbI tasks (Weston, Bordes, et al., 2015). An ontology is a collection of generic rules, which are combined with a set of facts. Here, the ontology describes a few rules for reasoning over human beings, objects, and their locations. Combined with the stated facts, it allows for answering queries like »*Is the apple a human being?*« or »*Is Mary in the bedroom?*«.

desirable (theoretical) properties of the formal approach, on the one hand, and utilizing the robustness of machine learning, on the other.

The main contributions of this paper are summarized as follows:

- We present a novel deep neural architecture for a model that is able to effectively perform logical reasoning in the form of basic ontology reasoning.
- We present, and make freely available, several very large, diverse, and challenging datasets for learning and benchmarking machine learning approaches to basic ontology reasoning.
- We present extensive experimental evaluations on the above benchmarks, which show that our model is able to learn to perform highly accurate ontology reasoning.

The rest of this article begins with a description of the reasoning problem that served as the starting point of our research. After this, we introduce the RRN model, explain how it has been evaluated in several experiments, and present the outcomes of our experimental evaluation. Finally, we review relevant related work, and conclude with a discussion.

2. Problem Description

The major part of the knowledge bases that are used for ontology reasoning today formalizes information in terms of individuals, classes, and binary relations, any of which may thus be considered as a directed knowledge graph, where individuals correspond to vertices, relations to labeled directed edges, and classes to binary vertex labels.[1] In KRR terminology, a setting like this may be described as a basic ontological knowledge base with a function-free signature that contains unary and binary predicates only. The facts that define such a knowledge graph are usually stated in terms of triples of the form $\langle \mathit{subject}, \mathit{predicate}, \mathit{object} \rangle$, and specify either a relation between two individuals, *subject* and *object*, or an individual's membership of a class, in which case *subject* refers to an individual, *predicate* to a special membership relation, and *object* to a class. Notice that we assume a fixed vocabulary, which means that all of the considered classes and relations are known beforehand, and are thus regarded as part of the ontology. The rules of the ontology are usually specified in terms of a knowledge representation language, such as a logic program or OWL.

We now provide a formal description of the reasoning problem that we consider in this article, and thus put our work in a formal context. Throughout this paper, we consider ontology reasoning that corresponds to an (extension of a) subset of Datalog where relations have an arity of at most two, but are not partitioned into database and derived relations, and where rules with falsity $\bot$ in the head, called *negative constraints*, are allowed. To that end, we assume an infinite set of *constants* Δ, an infinite set of *variables* $\mathcal{V}$, and a *relational schema* $\mathcal{R}$, which is a finite set of names for unary and binary relations. A *term* t is a constant or a variable, and an *atomic formula* (or *atom*) has the form of either $p(t)$ or $p(t_1, t_2)$, where p is a relation name, and t, t_1, and t_2 are terms. A *rule* r has the form

$$\beta_1 \wedge \cdots \wedge \beta_n \rightarrow \alpha, \tag{1}$$

where $\alpha, \beta_1, \ldots, \beta_n$ are atoms and $n \geq 0$. Such a rule is *safe*, if each of the variables in α also occurs in at least one β_i (for $1 \leq i \leq n$). A *negative constraint* γ has the form

$$\beta_1 \wedge \cdots \wedge \beta_n \rightarrow \bot, \tag{2}$$

where $\beta_1, \ldots, \beta_n$ are atoms and $n \geq 0$. A *program* Σ is a finite set of safe rules of the form (1) and negative constraints of the form (2) with $n \geq 1$. A safe rule with $n = 0$ is a *fact*, and a *database* is a finite set of facts. A *literal* ℓ is either a fact α or a negated fact $\neg\alpha$.

A (Herbrand) *interpretation* I is a (possibly infinite) set of facts. An interpretation I *satisfies* a variable-free rule r of the form (1), if $\{\beta_1, \ldots, \beta_n\} \subseteq I$ implies $\alpha \in I$. An interpretation I *satisfies* a variable-free negative constraint γ of the form (2), if $\{\beta_1, \ldots, \beta_n\} \not\subseteq I$. Furthermore, I *satisfies* a set R of rules and negative constraints, if it satisfies all their variable-free instances. We say that R is *satisfiable*, if it has a satisfying interpretation. Finally, a fact ζ (resp., negated fact $\neg\zeta$) is *logically entailed* by R, denoted $R \models \zeta$ (resp., $R \models \neg\zeta$), if it is part (resp., not part) of every interpretation that satisfies R. For satisfiable sets R of rules and negative constraints, since each such R has a unique least satisfying interpretation, denoted M_R, and this is equivalent to $\zeta \in M_R$ (resp., $\{\zeta\} = \{\beta_1, \ldots, \beta_n\} \setminus M_R$ for some variable-free instance of the form (2) of a negative constraint in R). In the sequel, we implicitly assume that all considered sets R of rules and negative constraints are satisfiable.

Further negated facts can be derived, if we additionally make the *closed-world assumption* (CWA) or the *local CWA* (LCWA). Formally, a negated fact $\neg\alpha$ is *logically entailed by*

[1] In the context of relational learning, knowledge graphs are typically simplified by viewing classes as individuals as well and memberships as ordinary relations. For our purposes, however, a clear distinction between classes and relations is important, which is why we adopt this slightly different view here.

R *under the CWA*, if and only if α is not entailed by the same, that is, $R \models \neg\alpha$ if and only if $R \not\models \alpha$. *Logical entailment under the LCWA* is a restricted subset of logical entailment under the CWA and superset of logical entailment that additionally allows for deriving negated atoms about binary predicates that are similar (»*local*«) to entailed positive facts in the following sense: if $R \not\models p(u_1, u_2)$ and $R \not\models \neg p(u_1, u_2)$, then, under the LCWA, $R \models \neg p(u_1, u_2)$ if and only if either $t_1 = u_1$ or $t_2 = u_2$ for some $p(t_1, t_2)$ with $R \models p(t_1, t_2)$. We denote by »$\mid\!\sim$« one of these three logical entailment relations.

We are now ready to define the problem that this article aims to solve, the *logical entailment problem* (of facts from databases and programs): given a database D, a program Σ, and a literal ℓ, decide whether $\Sigma \cup D \mid\!\sim \ell$. In this work, we consider the most common case where D is variable and of size k, Σ is fixed, and ℓ is variable as well. The goal is to generate a neural network $N[\Sigma, k]$ with binary output that, given an arbitrary database D of size at most k as well as an arbitrary literal ℓ, decides the logical entailment problem, that is, $N[\Sigma, k](D, \ell) = 1$ if and only if $\Sigma \cup D \mid\!\sim \ell$.

3. The Recursive Reasoning Network (RRN)

To tackle the problem specified in the previous section, we introduce a novel model architecture, the *recursive reasoning network* (RRN). To that end, this section starts with a high-level outline of the RRN model, and then provides a detailed description of the same.

3.1 Intuition

With the introduction of RRNs, we replace formal ontology reasoning with computing a learned deep neural network to remedy the issues outlined above. Thereby, following the spirit of the considered problem, every RRN is trained relative to a particular ontology, and is thus, like the formal counterpart, independent of the specific facts that it is provided with. To that end, as described in detail below, the vocabulary of classes and relations used by the ontology determines the recursive layers that are available in an RRN, and hence the structure of the same. In contrast to this, however, the rules that are used for reasoning are not provided to the model directly, but have to be learned from the training data. When a trained model is applied to a particular set of facts, then, on the face of it, it operates in two stages (each of which is explained subsequently): first, it generates vector representations, so-called embeddings, for all individuals that appear in the considered data, and second, it computes predictions for queries solely based on these generated vectors.

RRNs are based on the idea that we can encode all the information that we have about an individual, both specified and inferable, in its embedding. A similar idea is employed, for example, in the context of natural language processing, where real vectors are used to represent the meaning of text (Mikolov et al., 2013). Given a set of facts, specified as triples, we start by randomly generating initial embeddings for all the individuals that appear in any of them. After this, the model iterates over all the triples, and, for each of them, updates the embeddings of the individuals involved. Any such update is supposed to store the considered triple in the individuals' embeddings for one thing, but also to encode possible inferences based on the same. So, intuitively, a single update step conducts local reasoning based on what is encoded in the embeddings already as well as on the new information that was gained through the provided fact. An obvious necessity implied by this local reasoning scheme is

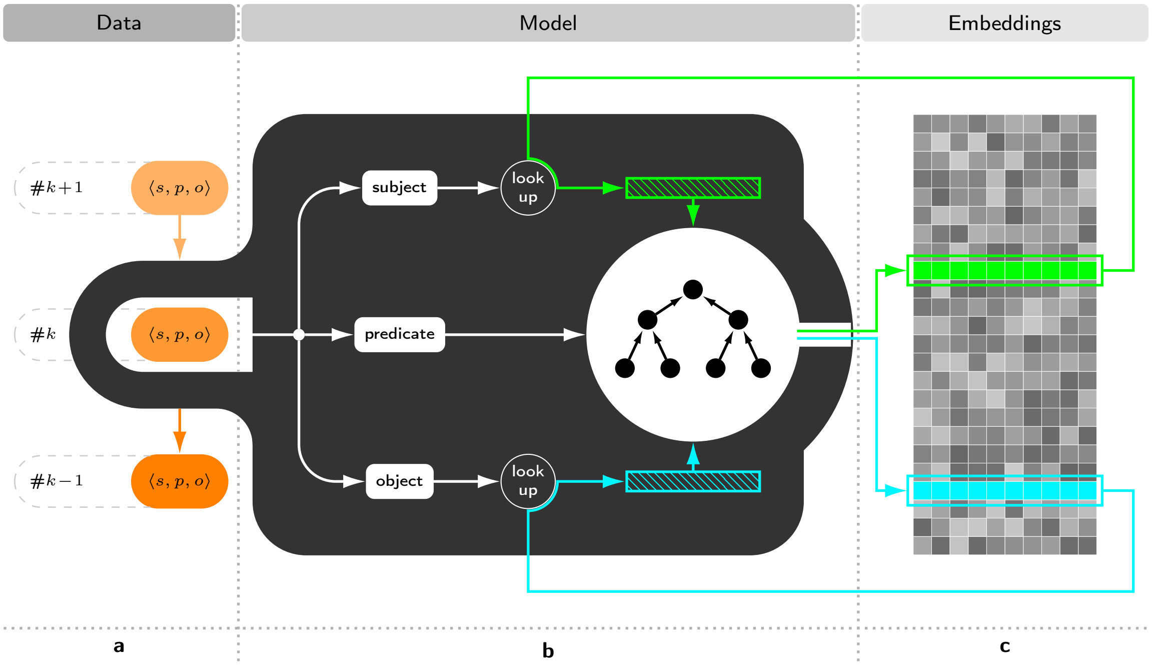

Data
Model
Embeddings
#k+1
$\langle s, p, o \rangle$
#k
$\langle s, p, o \rangle$
#k−1
$\langle s, p, o \rangle$
subject
predicate
object
look up
look up
a
b
c

Figure 2: **(a)** To generate embeddings for the individuals in a knowledge base, the model iterates over all its facts. To that end, it considers one triple at a time in arbitrary order, and, depending on the respective dataset, might repeat this process several times. **(b)** Whenever a fact is read, the model fetches the embeddings of the individuals that appear in the according triple, and feeds them into an update layer for the respective predicate. This layer yields updated versions of the embeddings that have been provided, which are then stored in place of the previous ones. The figure illustrates such an update for the case that a triple describes a relation between two individuals. In contrast to this, updates based on facts specifying class memberships involve a single individual embedding only. **(c)** Starting from randomly generated vectors, embeddings are updated step by step in order to encode both facts and inferences about the individuals that they represent.

that the model, in general, has to sift through all the data multiple times. This is essential in order to allow for multi-step, also called multi-hop, reasoning, which is based on several triples at the same time, since information might need to transpire from one individual's embedding to another's. The actual number of iterations required depends on the respective dataset, though. Figure 2 summarizes this process.

From a technical perspective, the outlined procedure for generating embeddings corresponds to computing a recursive neural network (Pollack, 1990) that receives the randomly generated initial embeddings as input and provides the final embeddings as output. The structure of the network depends on the set of facts being processed, and its recursive layers compute the update operations described above, which is why we refer to them as update layers.

Once the desired embeddings are generated, they can be used to answer atomic queries about the data that they are computed from. To that end, the model provides various multilayer perceptrons (MLPs) for computing predictions about relations between two individuals and class memberships of a single individual. Notice that the only inputs provided to these MLPs are the embeddings of the individuals that are involved in a particular query, which is why the model has to ensure that all the information that is needed for answering such is effectively encoded during the first step. Thus, the second step is just needed to uncover the knowledge that is encoded in individual embeddings, while the actual reasoning happens before.

A notable characteristic of the RRN is that it performs deductive inference over entire knowledge bases, and, like its symbolic stencil, aims to encode *all* possible inferences in the created individual embeddings, rather than answering just a single query. Because of this, the model is able to unravel complex relationships that are hard to detect if we try to evaluate the inferability of an isolated triple of interest only. Furthermore, the fact that the RRN conducts logical inference over all classes and relations simultaneously allows for leveraging interactions between any of them, and thus further adds to improving the model's predictive performance.

Another noteworthy aspect is that, as we will see below, an RRN does *not* treat triples as text. Instead, the individuals that appear in a triple are mapped to their embeddings before the same is provided to any layers of the used model. However, this means that RRNs are agnostic to the individual names that are used in a database, which makes perfect sense, as it is only the structure of a knowledge base that determines possible inferences.

3.2 Terminology

In the sequel, we always assume that we are facing a dataset that is given as an ontological knowledge base $KB = \langle \Sigma, D \rangle$, consisting of a program Σ (also called the *ontology*) and a database D, as defined above. We do not impose any restrictions on the formalism that has been used to specify KB, but we do require that it makes use of a fixed vocabulary that contains unary and binary predicates only, which we refer to as classes and relations, respectively, and that the individuals considered have been fixed as well. The sets of all classes and individuals of KB are denoted as $\mathit{classes}(KB)$ and $\mathit{individuals}(KB)$, respectively. Notice that these assumptions allow for viewing D as a set of triples, since we may represent any relation $R(i, j)$ as $\langle i, R, j \rangle$ and any class membership $C(i)$ as $\langle i, \mathtt{MemberOf}, C \rangle$, where `MemberOf` is a special symbol that does not appear in the vocabulary. Furthermore, we can represent negated facts as triples with a different relation symbol, e.g., $\neg R(i, j)$ and $\neg C(i)$ as $\langle i, \neg R, j \rangle$ and $\langle i, \neg\mathtt{MemberOf}, C \rangle$, respectively.

For every knowledge base KB, we define an indicator function

$$\mathbb{1}_{KB} : \mathit{individuals}(KB) \to \{-1, 0, 1\}^{|\mathit{classes}(KB)|}$$

that maps the individuals that appear in KB to 3-valued incidence vectors. Any such vector summarizes all the information about the respective individual's class memberships that are given *explicitly* as part of the facts D. So, if the considered classes are specified via the predicates $C_1, C_2, \dots, C_m$ and $i \in \mathit{individuals}(KB)$, then $\mathbb{1}_{KB}(i)$ yields an m-dimensional vector, and for $\ell \in \{1, \ldots, m\}$,

$$\left[\mathbb{1}_{KB}(i)\right]_\ell = \begin{cases} 1 & \text{if } \langle i, \texttt{MemberOf}, C_\ell \rangle \in D, \\ -1 & \text{if } \langle i, \neg\texttt{MemberOf}, C_\ell \rangle \in D, \text{ and} \\ 0 & \text{otherwise.} \end{cases}$$

3.3 Formal Definition of the Model

The purpose of the RRN model is to solve the logical entailment problem defined above, and to that end, we train an RRN to reason relative to a fixed ontology. More formally, suppose that RRN_Σ is a model that has been trained to reason with the ontology Σ, and D, like before, denotes a finite set of facts. Furthermore, let the triple $T = \langle s, P, o \rangle$ be an arbitrary query whose entailment is to be checked. Then, the network RRN_Σ defines a function such that

$$RRN_\Sigma(D, T) = \mathbb{P}\left\{T \text{ is true} \mid \langle \Sigma, D \rangle\right\}. \tag{3}$$

There are a few important aspects to notice about Equation 3. First and foremost, D may be any set of facts that use the same vocabulary as Σ, and does *not* have to be the one that has been used to train RRN_Σ. Similarly, T may be an arbitrary triple, once again, sharing the same vocabulary, and does *not* have to appear in the training data—neither as fact nor as inference. This means that RRN_Σ actually performs ontology reasoning, and does not just »*memorize*« the ontological knowledge base(s) that it has been trained on. The probability on the right-hand side of Equation 3 is used to express the model's belief in whether the queried triple is true based on the ontological knowledge base that is formed implicitly by the ontology that the model was trained on together with the provided set of facts, that is, $\langle \Sigma, D \rangle$. Interpreting the model's output as a probability allows for training it via a cross-entropy error term, but during evaluation, we predict queries to be true as soon as the model provides a probability of at least 50%. Note that we deliberately chose the wording »*T is true*«, rather than $\Sigma \cup D \mathrel{|\!\!\sim} T$, as an RRN provides a prediction even if T is neither provably entailed nor refutable based on $\langle \Sigma, D \rangle$.

We illustrate these ideas with a simple example. Suppose that we require a model for reasoning with respect to some ontology Σ. Then, as a first step, we train an RRN for reasoning with this particular ontology, denoted RRN_Σ. To train this model, we can either use an existing database at hand or simply generate (consistent) sets of facts that make use of the same vocabulary, that is, the same set of predicate symbols, as training samples. After this, RRN_Σ can be used to perform ontology reasoning (relative to Σ) over arbitrary databases that use the same vocabulary as Σ. For instance, suppose that we need to check whether $\Sigma \cup D_1 \mathrel{|\!\!\sim} T$ for some database D_1 and triple T. In this case, we employ RRN_Σ to first generate an embedding matrix $\mathbf{E}_{D_1}$ for the individuals in D_1, and second compute a probability for whether the considered entailment holds from $\mathbf{E}_{D_1}$. Taken together, these two steps constitute the function specified in Equation 3. We emphasize again that the actual reasoning step happens as part of the embedding generation, and entailment probabilities are computed solely based on generated embedding matrices. Notice further that RRN_Σ is not tied to D_1 in any way, which means that we can use the same model to perform ontology reasoning over another (different) database D_2. Finally, note that, in practice, we do not generate new embeddings whenever we check an entailment based on one and the same set of facts, but instead, create and reuse just a single embedding matrix for each database.

3.4 Individual Embeddings

We represent individuals as unit vectors, with respect to the Euclidean norm, of the space $\mathbb{R}^d$, which means that all embeddings live in the d-dimensional unit hypersphere. To that end, d is a hyperparameter that, in general, has to be chosen based on the expressiveness of the ontology, the size of the used vocabulary, and the total number of individuals considered. In this work, we denote the embedding of an individual i as $\mathbf{e}_i$.

Initially, individual embeddings are generated randomly following some probability distribution, which was chosen to be a uniform distribution in our experiments, and normalized subsequently, that is,

$$\hat{\mathbf{e}}_{i_1}, \hat{\mathbf{e}}_{i_2}, \dots \overset{\text{iid}}{\sim} \mathcal{U}(-\mathbf{1}_d, \mathbf{1}_d) \qquad \text{and}$$

$$\mathbf{e}_{i_\ell} = \frac{\hat{\mathbf{e}}_{i_\ell}}{\|\hat{\mathbf{e}}_{i_\ell}\|_2} \qquad (1 \le \ell \le |\mathit{individuals}(\mathit{KB})|),$$

for $\mathit{individuals}(\mathit{KB}) = \{i_1, i_2, \dots\}$.

3.5 Update Layers

At the heart of the RRN model, there are two kinds of update operations, one based on relations and one based on class memberships. Both of these are very similar in nature, and serve the same essential purpose: given a fact together with the embeddings of the individuals involved, update these embeddings to incorporate the knowledge that was gained through the provided fact (under the used ontology Σ as well as the information that is encoded in the embeddings already).

We start with the update operation for relations, which means that we aim to update the embeddings $\mathbf{e}_s$ and $\mathbf{e}_o$ based on some triple $\langle s, P, o\rangle$ with P being a relation type, that is, neither `MemberOf` nor `¬MemberOf`. Notice, however, that P might as well be a negated relation type $\neg R$. To define a proper update operation, one has to mind the fact that relations are, in general, not symmetric, and hence it makes a difference whether we update a triple's subject or object. Therefore, we define two (equal) recursive update layers for every (possibly negated) relation type P that appears in KB, one for updating the subject's embedding, which we denote as $\mathit{Update}^{\lhd P}$, and one for updating the object's, denoted $\mathit{Update}^{P \rhd}$. The following equations describe $\mathit{Update}^{\lhd P}$—$\mathit{Update}^{P \rhd}$ is defined analogously:

$$\mathbf{g}^{\lhd P}(s, o) = \sigma\Big(\mathbf{V}_1^{\lhd P}\mathbf{e}_s + \mathbf{V}_2^{\lhd P}\mathbf{e}_o\Big), \tag{4}$$

$$\hat{\mathbf{e}}_s^{(1)} = \mathit{ReLU}\Big(\mathbf{W}_1^{\lhd P}\mathbf{e}_s + \mathbf{W}_2^{\lhd P}\mathbf{e}_o + \mathbf{e}_s{\mathbf{e}_o}^T\mathbf{w}^{\lhd P}\Big), \tag{5}$$

$$\hat{\mathbf{e}}_s^{(2)} = \mathbf{e}_s + \hat{\mathbf{e}}_s^{(1)} \circ \mathbf{g}^{\lhd P}(s, o),$$

$$\mathbf{e}_s = \mathit{Update}^{\lhd P}(s, o) = \frac{\hat{\mathbf{e}}_s^{(2)}}{\|\hat{\mathbf{e}}_s^{(2)}\|_2}.$$

As usual, $\sigma(x)$ denotes the logistic function $1/(1+\exp(-x))$ and $\mathit{ReLU}(x)$ the ramp function $\max\{0, x\}$, both of which are applied elementwise. $\circ$ denotes the elementwise vector product, and $\mathbf{V}_1^{\lhd P}, \mathbf{V}_2^{\lhd P}, \mathbf{W}_1^{\lhd P}, \mathbf{W}_2^{\lhd P} \in \mathbb{R}^{d \times d}$ as well as $\mathbf{w}^{\lhd P} \in \mathbb{R}^d$ are parameters of the model. As can be seen from these equations, the recursive layers of the RRN model define a gated network

```
Input: an ontological knowledge base $KB = \langle \Sigma, D \rangle$ with $individuals(KB) = \{i_1, i_2, \ldots, i_M\}$,
       a number of update iterations $N$, and (optionally) a matrix of initial embeddings $\mathbf{E}$.
Output: the generated embeddings $\mathbf{E}$.
if no embedding matrix $\mathbf{E}$ was provided then
    Randomly initialize $\mathbf{E} = [\mathbf{e}_{i_1}, \mathbf{e}_{i_2}, \ldots, \mathbf{e}_{i_M}]^T$;
end
for $iter = 1, \ldots, N$ do
    foreach $i \in individuals(KB)$ do
        $\mathbf{e}_i = Update(i)$;
    end
    foreach $\langle s, P, o \rangle \in F$ with $P$ being a (possibly negated) relation type do
        $\mathbf{e}_s, \mathbf{e}_o = Update(\langle s, P, o \rangle)$;
    end
end
return $\mathbf{E}$.
```

Algorithm 1: Generating individual embeddings.

architecture. To that end, Equation 5 describes the calculation of a »*candidate*« update step, and Equation 4 specifies a gate that controls how much of this is actually applied. Based on these one-sided updates, we can now define the following simultaneous update operation:

$$\langle \mathbf{e}_s, \mathbf{e}_o \rangle = Update(\langle s, P, o \rangle) = \big\langle Update^{\triangleleft P}(s, o), Update^{P \triangleright}(s, o) \big\rangle.$$

Next, we consider updates based on class memberships. In this case, there are no interactions between individuals, which is why updates of this kind may be batched. Also, we do not have to pay special attention to the »*direction*« of an update, neither to whether a membership assertion is positive or negative. Instead, we simply define a single recursive update layer, which expects an individual i as input, as follows:

$$\mathbf{g}(i) = \sigma\Big(\mathbf{V} \cdot [\mathbf{e}_i : \mathbb{1}_{KB}(i)]\Big),$$

$$\hat{\mathbf{e}}_i^{(1)} = ReLU\big(\mathbf{W} \cdot [\mathbf{e}_i : \mathbb{1}_{KB}(i)]\big),$$

$$\hat{\mathbf{e}}_i^{(2)} = \mathbf{e}_i + \hat{\mathbf{e}}_i^{(1)} \circ \mathbf{g}(i),$$

$$\mathbf{e}_i = Update(i) = \frac{\hat{\mathbf{e}}_i^{(2)}}{\|\hat{\mathbf{e}}_i^{(2)}\|_2}.$$

This layer updates an individual's embedding based on *all* the information that has been provided about its class memberships. Notice that, in these equations, the colon denotes the concatenation of two vectors. Like before, the layer makes use of a gated architecture, and $\mathbf{V}, \mathbf{W} \in \mathbb{R}^{d \times (d + |classes(KB)|)}$ are parameters of the model.

Algorithm 1 summarizes how these layers are used to generate embeddings for all individuals in the considered knowledge base.

3.6 Prediction

For computing predictions based on previously generated individual embeddings, we make use of several MLPs, a single one for all classes and one for each relation R in the vocabulary, denoted $\mathit{MLP}^{(\text{classes})}$ and $\mathit{MLP}^{(R)}$, respectively. Notice that, in this case, R is indeed a relation type, rather than its negation. $\mathit{MLP}^{(\text{classes})}$ expects a single individual embedding as input, and computes probabilities for the individual's memberships with respect to all classes. If we assume that KB contains the classes $C_1, \dots, C_m$, this means that $\mathit{MLP}^{(\text{classes})}$ computes a function

$$\mathit{MLP}^{(\text{classes})} : \mathbb{R}^d \to [0,1]^m$$

and

$$\left[\mathit{MLP}^{(\text{classes})}(\mathbf{e}_i)\right]_\ell = \mathbb{P}\left\{\langle i, \texttt{MemberOf}, C_\ell \rangle \mid \mathit{KB}\right\}.$$

Similarly, $\mathit{MLP}^{(R)}$ requires two individual embeddings as input, and provides a probability for the according relation to exist between the respective individuals. Assuming that KB consists of the relations $R_1, \dots, R_n$, this means that

$$\mathit{MLP}^{(R_\ell)} : \mathbb{R}^{2d} \to [0,1]$$

and

$$\mathit{MLP}^{(R_\ell)}(\mathbf{e}_i, \mathbf{e}_j) = \mathbb{P}\left\{\langle i, R_\ell, j \rangle \mid \mathit{KB}\right\}.$$

Notice that, at the same time,

$$\mathbb{P}\left\{\langle i, \neg R_\ell, j \rangle \mid \mathit{KB}\right\} = 1 - \mathbb{P}\left\{\langle i, R_\ell, j \rangle \mid \mathit{KB}\right\} = 1 - \mathit{MLP}^{(R_\ell)}(\mathbf{e}_i, \mathbf{e}_j).$$

3.7 Training

The training of an RRN is straightforward, and a high-level description is provided in Algorithm 2. An important detail to notice is that all training samples are knowledge bases that share the same ontology, namely, the one that the RRN is being trained for, but usually differ with respect to the facts that they contain. In contrast to this, many approaches to learning-based reasoning and knowledge-base completion make use of one large knowledge graph for training a model only, which is usually also the one that has to be completed. As stressed above already, this is a major advantage of the RRN, since it allows for training a model with respect to an ontology, but independently of the knowledge base(s) that it is supposed to be applied to later on. To that end, one may simply *generate* sample knowledge bases relative to the ontology of interest, and train the RRN on this synthetic dataset. Once a model has been trained, it can be applied directly to any previously unseen knowledge base without the need for being retrained. Another important advantage of the possibility to train on generated data is that it allows for the model to encounter predicates during training that appear as inferences, but not as facts in the knowledge base that it is eventually applied to. As opposed to this, models that are trained on the same knowledge base that has to be completed have no means to learn about such predicates, and thus cannot identify according inferences.

Intuitively, this approach might not seem sound, as it strongly disagrees with the usual way of training a model, and one may challenge whether an RRN is trained on the right

```
Input: a sequence of training samples $T = \langle KB_1, KB_2, \dots, KB_M \rangle$, where $KB_\ell = \langle \Sigma, D_\ell \rangle$, and
       a number of update iterations $N$.
while evaluation error has not converged do
    Randomly shuffle $T$;
    for $KB_\ell \in T$ do
        Generate embedding matrix $\mathbf{E}$ for $KB_\ell$;
        Compute cross-entropy loss for predicting triples in $KB_\ell$ from $\mathbf{E}$ (both facts and
          inferences);
        Update model parameters to minimize the loss;
    end
end
```

Algorithm 2: RRN training.

data distribution this way. We argue that this is indeed the case, though. Training a model to perform ontology reasoning requires the same to learn how to make use of the rules comprised by the considered ontology. Furthermore, an RRN is not provided these rules directly, but has to learn them from the training data as well. Notice, however, that the application of any such rule is exactly the same for a synthetically generated knowledge base and another one encountered as part of a real-world problem, assuming that both rely on the same ontology, and thus there is no reason to favor real-world data in this context. In fact, synthetic datasets offer the possibility for us to control that all rules in an ontology are used to, more or less, the same extent, while real-world knowledge bases tend to be skewed towards a part of the rules only. Therefore, we have to make sure that all of the ontology is faithfully represented by a real-world dataset, while a sufficiently large synthetic one is clearly capable of accomplishing this.

Besides that, the training procedure for an RRN closely resembles the common pattern for training supervised models. This means that we sample a knowledge base from the training data, generate individual embeddings for the same, and compute predictions for all facts and inferences that can be drawn from the considered knowledge base. Notice that every fact in a knowledge base is trivially entailed by the same, and hence, strictly speaking, part of the inferences that the model learns to predict. For evaluating our approach, a clear distinction of facts and inferences is useful, though, which is why we use the terms »*inference*« and »*inferable*« in the sense of inferable but not specified as fact. Finally, we compute a cross-entropy loss for the computed predictions, and adjust the model via stochastic gradient descent (SGD). After every training epoch, the model is evaluated on a hold-out set, and the training procedure is stopped as soon as the error that was computed on the same has converged.

A salient detail of the outlined training procedure is that we consider just a single training sample at a time, and thus seem to use an online version of SGD. However, as samples are knowledge bases, they usually induce a multitude of triples that have to be predicted in each training iteration, and hence effectively correspond with minibatches of varying sizes.

3.8 Computational Aspects

Now that we have defined all parts of the RRN model, we need to ask the question of whether the presented approach is sensible from a practical point of view. For most problems that we encounter in practice, we can consider the ontology Σ to be fixed, while the database of known facts D may be subject to frequent change. Therefore, while we can safely disregard the cost for training a model RRN_{Σ}, the complexity of computing the same in order to reason over D (relative to Σ) turns out to be critical.

We start with investigating the cost of generating individual embeddings. As stated in Algorithm 1, each triple in D triggers (at most) one update operation per update iteration.[2] Since every update can be computed in constant time and the number of update iterations is fixed, this means that the (time) cost for generating an embedding matrix $\mathbf{E}_D$ is (at most) linear in the size of D, that is, $\mathcal{O}(|D|)$. This can be further improved by batching update operations for triples that describe facts about relations, and parallelizing the computation of any such batch of updates. One way to find appropriate batches is to compute maximal matchings in the considered knowledge graph restricted to a single relation type, which can be realized by means of fast greedy algorithms. Notice further that, as pointed out above already, we usually do not compute new embeddings every time we need to check an entailment, but only if the database D changed.

The second operation involved in using RRN_{Σ} is computing predictions from $\mathbf{E}_D$. However, this requires constant time for each query, and once again, multiple queries may be batched in order to further reduce the time of computation.

4. Evaluation

To assess the suggested approach, we trained and evaluated an RRN on five different datasets, two out of which were artificially generated toy datasets, two were extracted from real-world databases, and one was a generated dataset based on a real-world ontology. Toy problems, generally, have the great advantage that it is immediately evident how difficult certain kinds of inferences are, and thus provide us with a fairly good impression of the model's abilities. Nevertheless, evaluating an approach in a real-world setting is, of course, an indispensable measure of performance, which is why we considered both kinds of data for our experiments. Table 1 provides a few summary statistics of the used datasets. In this table as well as in the sequel, we use the terms specified or inferable classes and relations as a short form of triples that describe class memberships and relations, respectively. This means that »*inferable classes*«, for example, refers to triples that are inferable from (but not given as facts in) a knowledge base, and that specify a class membership of an individual. Notice that we always assume ontologies to be fixed, which is why there is no indication for talking about specified or inferable class or relation types. Notice that the stated counts of individuals, classes, and relations in Table 1 are average values per sample knowledge base. Furthermore, triples that describe classes and relations, respectively, appear both in positive and negative from. For example, $\langle i, \texttt{MemberOf}, C_{\ell} \rangle$ specifies the positive literal $C_{\ell}(i)$, while

[2] In Algorithm 1, updates based on triples that specify information with respect to class memberships (lines 5 to 7) are grouped by individual and batched. In practice, we would perform such a batch update only if there is at least one fact concerning the respective individual. This means that, in the worst case, we have to perform one update for each fact about a class membership per update iteration.

dataset	# sample KBs		avg. ind.	vocabulary size	
	train	test	per sample	# class types	# relation types
family trees	5,000	500	23	2	29
countries (S1)	5,000	20	240	3	2
countries (S2)	5,000	20	240	3	2
countries (S3)	5,000	20	240	3	2
DBpedia	5,000	500	200	101	518
Claros	5,000	500	200	33	77
UMLS-reasoning	5,000	500	60	127	53

dataset	avg. classes per sample		avg. relations per sample	
	specified	inferable	specified	inferable
	(pos./neg.)	(pos./neg.)	(pos./neg.)	(pos./neg.)
family trees	23 / —	— / 23	28 / —	240 / 16,160
countries (S1)	— / —	238 / 478	820 / —	20 / 49,261
countries (S2)	— / —	238 / 478	782 / —	39 / 49,279
countries (S3)	— / —	238 / 478	757 / —	68 / 49,275
DBpedia	— / —	183 / —	642 / —	156 / —
Claros	— / —	1,499 / —	518 / —	17,072 / —
UMLS-reasoning	43 / 39	1,194 / —	28 / 44	59 / 345

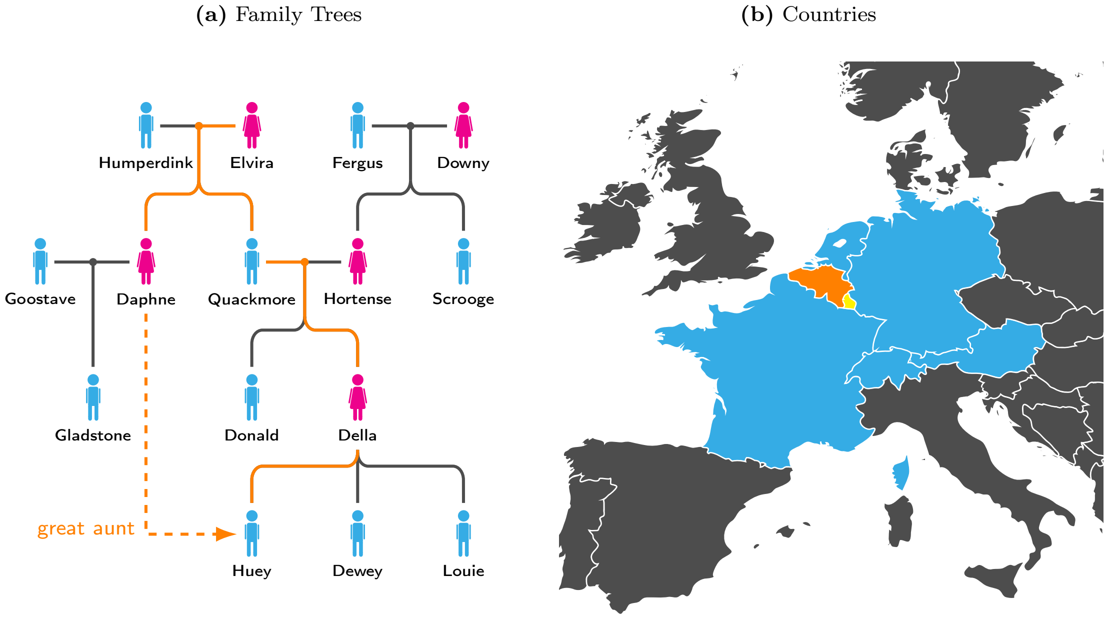
Table 1: This table breaks down the datasets that we used in our experiments according to the total numbers of samples, individuals, and triples that they contain. For triples, it lists both class memberships and relations specified as facts as well as such that are just inferable. The numbers of positive and negated triples are quoted separately. Notice that we trained and evaluated an RRN on 20 independent datasets for each countries task in order to obtain a larger amount of evaluation data. To that end, the number of training samples are average values over all these datasets, while the number of test samples was summed up over all of them.

$\langle i, \neg\texttt{MemberOf}, C_\ell \rangle$ describes a negative one $\neg C_\ell(i)$. Accordingly, Table 1 provides separate counts for triples specifying positive and negative literals, respectively, and in the sequel, we commonly refer to positive or negatives facts and inferences, respectively, which shall be interpreted this way.

Each of the considered datasets is a collection of sample knowledge bases that share a common ontology. During training and evaluation, the model was provided with all the facts in the considered samples, and had to compute predictions for both facts and inferences that were derivable from the same, based on the used ontologies. Since neural networks are notoriously data-hungry, there has been a recent interest in training models on limited amounts of data. Taking the same line, we confined all our training sets to a total of 5000 training samples.

4.1 Datasets

We generated two toy datasets that pose reasoning tasks of varying difficulty with respect to family trees, on the one hand, and countries, on the other. In the family-trees dataset (Figure 3a), samples describe pedigrees of different sizes such that the only facts that are available in any of them are the genders of the people involved as well as the immediate



(a) Family Trees

(b) Countries

Ontology:
Grandparent0f(G, C) ← ParentOf(G, P), ParentOf(P, C).
GrandfatherOf(G, C) ← GrandparentOf(G, C), Male(G).
SiblingOf(C, C') ← ParentOf(P, C), ParentOf(P,C'), $C \neq C'$.
GreatAuntOf(G, C) ← SiblingOf(I, I'), GrandparentOf(I', C), Female(G).
MotherOf(M, C) ← ParentOf(M, C), Female(M).
...

Facts:
Female(*Daphne*)
Male(*Quackmore*)
ParentOf(*Elvira*, *Daphne*)
ParentOf(*Elvira*, *Quackmore*)
ParentOf(*Quackmore*, *Della*)
ParentOf(*Della*, *Huey*)
...

Ontology:
LocatedIn(C,R) ← LocatedIn(C, S), LocatedIn(S, R).
LocatedIn(C,R) ← NeighborOf(C, C'), LocatedIn(C', R).
Region(R) ← Subregion(S), LocatedIn(S, R).
Country(C) ← NeighborOf(C, N).
...

Facts:
LocatedIn(*WesternEurope*, *Europe*)
LocatedIn(*France*, *WesternEurope*)
LocatedIn(*Belgium*, *WesternEurope*)
LocatedIn(*Luxemburg*, *WesternEurope*)
NeighborOf(*France*, *Belgium*)
NeighborOf(*Belgium*, *Luxemburg*)
...

Inference Examples:

GrandparentOf(*Quackmore*, *Huey*) ←
ParentOf(*Quackmore*, *Della*),
ParentOf(*Della*, *Huey*).

SiblingOf(*Daphne*, *Quackmore*) ←
ParentOf(*Elvira*, *Daphne*),
ParentOf(*Elvira*, *Quackmore*).

GreatAuntOf(*Daphne*, *Huey*) ←
SiblingOf(*Daphne*, *Quackmore*),
GrandparentOf(*Quackmore*, *Huey*),
Female(*Daphne*).

Inference Examples:

LocatedIn(*Belgium*, *Europe*) ←
LocatedIn(*Belgium*, *WesternEurope*),
LocatedIn(*WesternEurope*, *Europe*).

LocatedIn(*Belgium*, *WesternEurope*) ←
NeighborOf(*Belgium*, *Luxemburg*),
LocatedIn(*Luxemburg*, *WesternEurope*).

Figure 3: **(a)** An example of a family tree, like it is found in the first toy dataset that the RRN model was tested on. While the genders of all people involved as well as the immediate relations between parents and their children are provided as facts, all other family relations have to be inferred. The figure depicts an example of a four-hop inference, namely, Daphne being a great aunt of Huey. **(b)** A sample like the ones in the countries dataset. It illustrates how the location of Belgium (orange) is inferred from the known location of its neighbor Luxemburg (yellow). Given that Luxemburg is known to be part of Western Europe (blue), it follows that Belgium is situated in Western Europe as well. Furthermore, since Western Europe is a subregion of Europe, Belgium has to be in Europe too. The outlined course of reasoning does not work in every case, however. It fails, for example, if we try to infer Spain's region based on facts abouts its neighbor France.

ancestors, that is, parent-of relations, among them. Besides this, the used ontology specifies rules for reasoning about 28 different kinds of family relations, ranging from »*easy*« inferences, such as fatherhood, to more elaborate ones, like being a girl first cousin once removed.[3] What is particularly challenging about this dataset, however, is that seemingly small samples allow for great numbers of inferences. For instance, an actual sample that consists of 12 individuals with 12 parent-of relations specified among them admits no less than 4032 inferences, both positive and negative. Furthermore, the training data is highly skewed, as, for most of the relation types, only less than three percent of all inferences are positive ones. Appendix B contains additional details of how we created this dataset.

The second toy dataset is based on the countries knowledge base (Bouchard et al., 2015), which describes the adjacency of countries together with their locations in terms of regions and subregions. In every sample, some of the countries' regions and subregions, respectively, are not stated as facts, but supposed to be inferred from the information that is provided about their neighborhoods (Figure 3b). Following Nickel et al. (2016), we constructed three different versions of the dataset, S1 (easy), S2, and S3 (hard). In contrast to the original work, however, we created an ontology that allows for reasoning not just over countries' locations, but also about their neighborhood relations. Furthermore, we introduced classes for countries, regions, and subregions, all of which have to be inferred from the provided set of facts. An interesting characteristic of the latter two versions of the dataset is that the sample knowledge bases are constructed such that parts of the missing relations cannot be inferred by means of the ontology at all. The same is true for some of the class memberships in all three versions of the problem. This challenges the model's ability to generalize beyond pure ontology reasoning. Additional details about how we generated the countries datasets can be found in Appendix C.

To evaluate the RRN model on real-world data, we extracted datasets from two well-known knowledge bases, DBpedia (Bizer et al., 2009) and Claros. The former of these represents part of the knowledge that is available in terms of natural language on Wikipedia, and the latter is a formalization of a catalog of archaeologic artifacts, which was created as part of the RDFox project (Nenov et al., 2015). Since these datasets do not naturally separate into samples, we extracted sample knowledge bases that are subgraphs of the original knowledge graphs, each of which contains a total of 200 individuals. This way, it is possible to evaluate the model on sample knowledge graphs that it has not seen as part of the training. Appendix D provides additional details about how we extracted those samples.

While Claros makes use of a total of 33 classes and 77 relations, DBpedia employs a massive vocabulary consisting of thousands of classes and relations, respectively. To make the according experiments more computationally feasible, we restricted this to the 101 most frequent classes and those 518 relation types that allow for the greatest numbers of inferences. What is interesting about both of the considered knowledge bases is that neither of them contains any specified, but only inferable class memberships, which is a very common characteristic of real-world datasets. Furthermore, the distribution of relations with respect to individuals differs quite heavily in both of them. While the branching factor in the knowledge graph specified by DBpedia varies rather smoothly, Claros contains certain clusters of individuals that share very large numbers of relations, whereas their vicinities are

[3] The term »*girl first cousin once removed*« refers to the daughter of someone's cousin.

dataset	accuracy on				F1 score on			
	spec. classes	inf. classes	spec. relations	inf. relations	spec. classes	inf. classes	spec. relations	inf. relations
family trees	1.000	1.000	1.000	0.999	1.000	1.000	1.000	0.976
countries (S1)	—	1.000	1.000	0.999	—	1.000	1.000	0.999
countries (S2)	—	1.000	0.999	0.999	—	1.000	0.997	0.929
countries (S3)	—	1.000	0.999	0.999	—	1.000	0.996	0.916
DBpedia	—	0.998	0.998	0.989	—	0.997	0.998	0.962
Claros	—	0.999	0.999	0.996	—	0.999	0.999	0.997
UMLS-reasoning	0.989	0.990	0.997	0.997	0.969	0.994	0.996	0.989

Table 2: This table summarizes our experimental results. Accuracy and F1 score are reported separately for class memberships and relations, and within each group, for those triples that describe specified knowledge, that is, facts, and those that represent inferable information.

comparably sparse. This is reflected by the fact that some sample knowledge bases consist of a few hundred specified relations only, while others contain more than 100,000.

In addition to this, we created a dataset based on the Unified Medical Language System (UMLS; McCray 2003), an ontology that has been introduced for describing concepts from the biomedical domain in a uniform way. The UMLS ontology makes use of 127 classes and 53 relations, and thus, in contrast to DBpedia and Claros, contains more classes than relations. The UMLS is just an ontology, however, and does not specify any facts at all, which is why we generated a dataset of 5000 sample knowledge bases, all of which make use of the UMLS ontology as a training set. Additional details about the used generation process are provided in Appendix D.

Interestingly, the UMLS is commonly used as a benchmark dataset for methods of knowledge base completion (Kok & Domingos, 2007). In this context, however, the UMLS is not interpreted as an ontology, but simply viewed as an ordinary knowledge graph. This is possible, as the structure of the ontology is defined in terms of triples, such as $\langle Bacterium, isa, Organism \rangle$, which may be interpreted as the facts that specify a knowledge graph, and in the benchmark dataset, some of these triples are missing and thus have to be predicted based on the remaining information. In the sequel, we use the term »*UMLS-reasoning*« to refer to our own dataset, and set it apart from the benchmark by (Kok & Domingos, 2007), which is commonly just referred to as »*UMLS*«.

Finally, notice that we partitioned each of our datasets into pairwise disjoint sets for training and testing, which ensures that the model actually learns to perform ontology reasoning as opposed to just memorizing the data. Also, we would like to point out that, unfortunately, Ebrahimi et al. (2018), who cited our work, mistakenly claimed that we trained and evaluated our models on one and the same data, which is *not* the case.

4.2 Results

Table 2 summarizes the results that have been achieved in our experiments. For reports in much greater detail, we refer the reader to Appendix F and G, which provide additional information about results on the toy datasets. The detailed results of our experiments with

DBpedia, Claros, and UMLS-reasoning are too comprehensive to be included in this paper, and are thus provided online at http://paho.at/rrn. In addition to this, Appendix A provides a description of our experimental setup.

There are a number of interesting aspects to notice across all the considered reasoning tasks. First, we see that the RRN is able to effectively encode provided facts, about both classes and relations, as these are predicted correctly from the created embeddings with an accuracy greater than 98.9%, and except for UMLS-reasoning, even with more than 99.8%. Furthermore, we observe that the model also learns to reason over classes with an equally high accuracy of 99.8% on all datasets except UMLS-reasoning, for which we find an accuracy of 99.0%. This difference is not surprising, though, as the UMLS ontology specifies by far the largest taxonomy of classes. Therefore, separate class prediction layers might be required to achieve an even higher accuracy on predicting inferable class memberships in this dataset. For reasoning over relations, we find a slightly lower accuracy of 98.9% on DBpedia, while derivable relations in other datasets are predicted correctly in at least 99.6% of all cases. Again, however, this difference is not surprising, as DBpedia's is by far the most complex among the ontologies that were used in our experiments, and might thus require a larger training dataset in order to achieve an even higher accuracy. Also, it was to be expected that the model would generally perform better in predicting inferable classes than relations, since most of these are inferences depending on single triples only.

As stated in the detailed result tables, comparing specified with inferable triples reveals that, throughout all datasets, there are imbalances with respect to positive and negative relations, that is, the majority of specified triples is positive, while most inferable ones are negative. Therefore, we would like to stress at this point that the RRN employs the same MLPs for predicting both kinds of triples, and thus cannot »*cheat*« by leveraging those imbalances in the data. Also, the great surplus of negative triples among all inferable ones explains why the reported F1 values are slightly lower than the accuracies quoted.

Earlier, we pointed out a few problems that symbolic methods of KRR commonly struggle with. To evaluate whether the RRN suffers from these issues as well, we corrupted the test data of our real-world datasets, DBpedia and Claros, and examined whether the models that have been trained as part of our experiments are able to resolve the introduced flaws. For the problem of missing information, we randomly removed one fact that could not be inferred by means of symbolic reasoning from each sample, and checked whether the model was able to reconstruct it correctly. For DBpedia, this was the case for 33.8% of all missing triples, and for Claros, 38.4% were predicted correctly. Just like this, we tested the model's ability to resolve conflicts by randomly choosing one fact in each test sample, and adding a negated version of the same as yet another fact. For DBpedia, the RRN resolved 88.4% of the introduced conflicts correctly, and for Claros, it were even 96.2%. Most importantly, however, none of the total accuracies reported earlier dropped by more than 0.9 for either of the corrupted datasets. This is remarkable, as both kinds of deficiencies pose serious problems for symbolic methods, and conflicting information, in particular, prohibits the same entirely for most of the formalisms that are used today.

All the query predictions of the RRN are solely based on the embeddings that it generated for the individuals in the respective datasets, which is why it is instructive to have a closer look at such a set of embedding vectors. Figure 4 provides a two-dimensional t-SNE projection of those embeddings for the individuals in one sample of the countries dataset

model	accuracy on inferable relation			
	family trees	countries (S1)	countries (S2)	countries (S3)
RRN	0.999	0.999	0.999	0.998
NTP	0.971	0.999	0.890	0.875

Table 3: The results of our experimental comparison of the RRN and the NTP. In this table, accuracies are stated with respect to predicting relations that are inferable relative to the used ontologies, and for the countries datasets, we considered the located-in relation only.

(S1). In this visualization, dots represent individuals, and their colors indicate whether they represent countries, regions, or subregions. Furthermore, each country and each subregion is endowed with a colored shadow that reveals the region that the respective individual is located in. Without even looking at the labels provided, one can easily detect groups of countries that belong to one and the same subregion. Furthermore, most of the clusters representing subregions that are located in the same region are neighbors in this visualization as well. Finally, observe that the individuals that represent regions and subregions, respectively, are mostly encoded as embeddings in confined regions of the embedding space.

Despite a considerable body of related work (cf. Section 5), hardly any of the introduced models can be used to address the same exact problem setting that is considered in this work, such that a direct comparison would not be fair in most cases. There is one exception, however, and this is the Neural Theorem Prover (NTP; Rocktäschel & Riedel 2017). Although it has been introduced in the context of knowledge graph completion, the NTP is based on the backward chaining algorithm that is used in Prolog, and can thus be used for logical reasoning. Since the NTP model is rather involved, we refrain from providing a description here, and refer the interested reader to the original paper by Rocktäschel & Riedel (2017) instead. There are, however, two important aspects to mention. First, it is possible for the NTP to make use of symbolic rules that are provided as part of the input, which is why the model is suitable for ontology reasoning. These rules are restricted to positive Horn rules, though, and thus cover just a fraction of the full reasoning problem that we aim to solve in this paper. Second, like most models for knowledge graph completion, the NTP is trained on the same knowledge graph that is supposed to be completed, and thus cannot make use of the RRN's training scheme described above.

As pointed out by Rocktäschel & Riedel (2017), the NTP suffers from severe performance problems, which is why we compared it to the RRN on our toy datasets only. Furthermore, due to the restricted set of rules that can be provided to an NTP, we removed all those rules from the used ontologies that are not supported by the model, and instead employed it for reasoning under the CWA. This way, it is possible to employ the NTP model for the considered reasoning problems on our toy datasets. Notice, however, that this is not the case in general. Furthermore, we made use of the (empirically) best version of the NTP model, which is referred to as NTPλ in the original paper.

Table 3 summarizes our comparison of the RRN and the NTP on the family trees and the different versions of the countries dataset. For evaluating the models against each other, we considered inferable relations for computing accuracies only, that is, we excluded facts as well as any triples about class memberships, as these provide a better picture of a model's

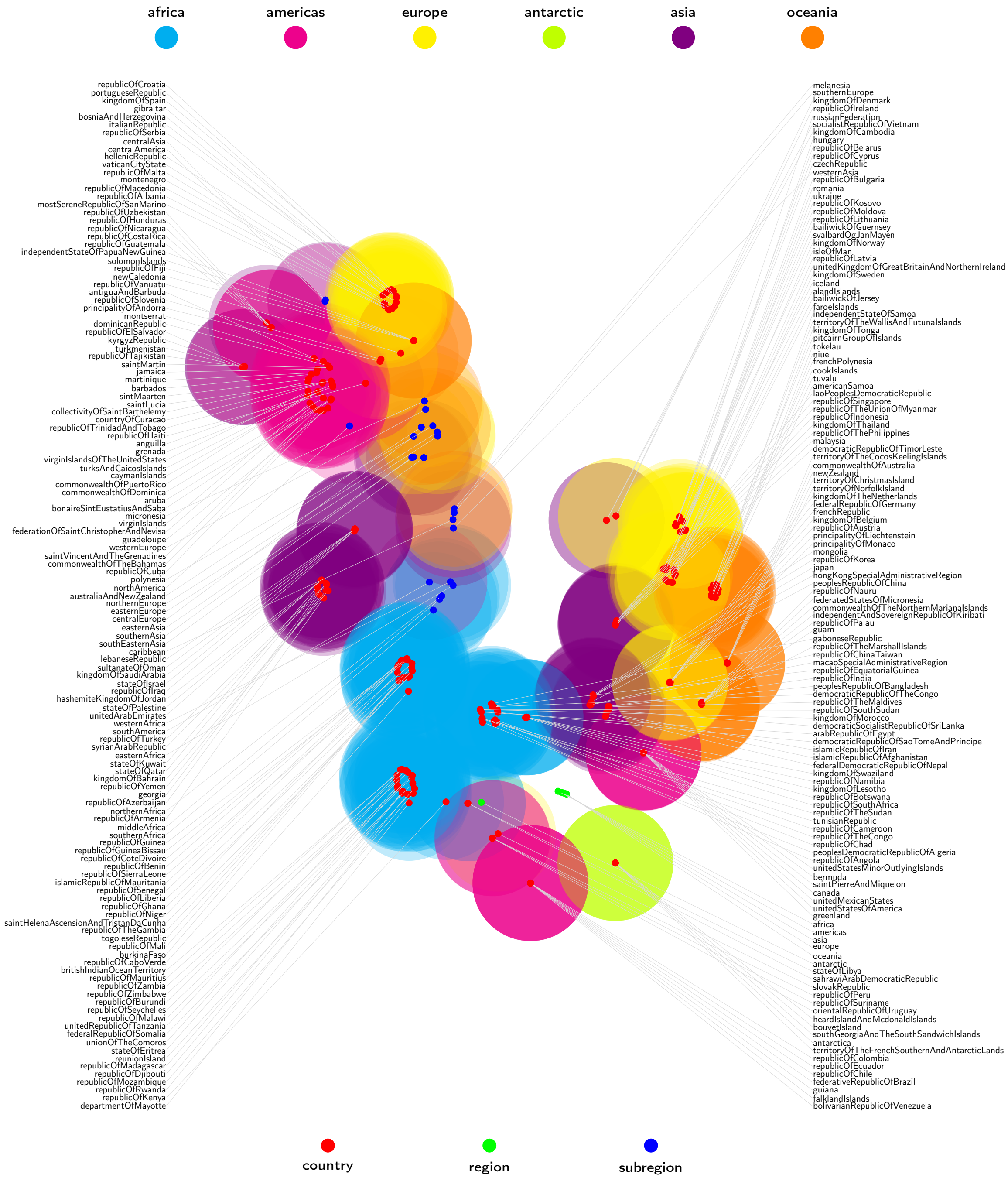


Figure 4: This is a t-SNE projection of embeddings that were generated for a single training sample from version S1 of the countries dataset. Dots represent individuals, which means for this data countries, regions, or subregions. Since each country and each subregion is located in exactly one region, their respective dots are endowed with a colored shadow such that different colors indicate different regions of belonging.

abilities. On the family trees dataset, we evaluated both models with respect to all inferable relations, but on the countries dataset, we confined ourselves to located-in relations only, as inferable relations of other relation types are more or less trivial. As stated in the table, the RRN, which achieves an accuracy of at least 99.8% on all datasets, outperforms the NTP in all the considered cases except version S1 of the countries data, for which both models perform equally well. On version S2 and S3 of the countries dataset, however, we observe a signification margin of more than 10% accuracy.

There is another interesting detail about the application of the NTP on the family trees dataset. All facts about relations in the dataset are parent-of relations, which means that none of the other predicates that appear in any inferable triples can be observed as facts. As the NTP is always trained on the same knowledge base that it is evaluated on, however, it is not able to learn anything about these relation types, and cannot even provide predictions for the same. In contrast to this, the RRN can be trained on synthetically generated training data, and thus learns to predict inferable relations almost perfectly. To allow for comparing the RRN and the NTP on the family trees dataset anyway, we thus added half of the inferable relations of each relation type as facts to the according sample knowledge bases. Notice, however, that the RRN achieved the same accuracy of 99.9% on the original version of the family trees data, as stated in Table 2.

4.3 Qualitative Evaluation

Despite the convincing results, the RRN, like most models that are based on deep learning, remains basically a black box for which it is not obvious how exactly predictions are computed. Our experimental evaluation shows, however, that the model's computations are effectively guided by the rules of the considered ontology, and we are planning to make this aspect fully transparent in future extensions of the RRN architecture. In this section, we analyze further characteristics of the RRN model, and try to understand how it depends on different design choices.

Update iterations. As described above, the number of update iterations is used to specify how often each fact in a knowledge base is considered for updating the embeddings of the individuals involved in the same (cf. Algorithm 1). Intuitively, the reason why we have to perform multiple update iterations is that update steps are local in the sense that they are based on a single fact together with the information that is encoded in the embeddings of the individuals involved. Often, however, ontologies contain rules that are based on multiple triples at the same time, for example, if inferences depend on two individuals in a knowledge graph being connected by a path of length greater than one, which is commonly referred to as multi-hop reasoning. In a case like this, multiple update iterations are necessary, since more than two embeddings have to be adjusted in order to transfer information between all individuals involved.

To verify this, consider Table 4, which provides prediction accuracies for inferable relations of different relation types in the family trees dataset. The relations described in this table range from easy 1-hop inferences up to much more elaborate 5-hop reasoning, and the according accuracies were evaluated after each of 5 update iterations in total. Now, there are two interesting aspects to observe in this table. First and foremost, we see that the number of update iterations that are necessary in order to achieve the peak accuracies with respect

update iters	**accuracy on inferable relations**				
	fatherOf (1 hop)	sisterOf (2 hops)	greatGrandsonOf (3 hops)	girlCousinOf (4 hops)	boyFirstCousinOnceRemoved (5 hops)
1	0.992	0.991	0.921	0.919	0.940
2	**1.000**	0.996	0.995	0.994	0.992
3	1.000	**0.997**	0.999	0.998	0.999
4	1.000	0.997	**1.000**	**0.999**	0.999
5	1.000	0.997	1.000	0.999	**1.000**

Table 4: The prediction accuracies for inferable relations with different predicates in the family trees dataset after different numbers of update iterations. Bold-faced values mark the lowest number of update iterations that allow for achieving the best possible accuracy for a relation type.

dataset	family trees	countries (S3)	DBpedia	Claros	UMLS-reasoning
1 pred. layer per relation	0.999	0.999	0.997	0.989	0.996
1 pred. layer for all relations	0.778	0.942	0.563	0.616	0.644

Table 5: The accuracies for inferable relations that were achieved with one prediction layer per relation type on the one hand and a single prediction layer for all relations on the other.

to the different relation types is indeed correlated with the according number of hops. Even more interestingly, however, all accuracies are greater than 0.91 after the very first update iteration already, and greater than 0.99 after the second one. This suggests that the model learns to make effective use of multiple rules in the ontology at the same time, and is not confined to drawing simple one-step inferences per update iteration.

Number of layers. A noticeable property of the RRN's model architecture is that it, for both updates and predictions, comprises one layer for each relation type that exists in the considered ontology, but just a single one for class types altogether. The intuition behind this is that inferences about class memberships tend to be »*easier*« in the sense that they frequently depend on single triples only or can be drawn based solely on the knowledge about the individual concerned. Empirically, this is confirmed by the results presented in the previous section, as the predictive accuracy on inferable classes is greater than or equal to the accuracy on inferable relations on all considered datasets. This raises the question of whether it is possible to reduce the number of layers that are used for operations related to relations in a similar manner in order to reduce the total number of trainable parameters.

It turns out, though, that this is not the case. Table 5 summarizes, for all datasets, how the accuracy of predicting inferable relations changes if we use just a single compound prediction layer, like for predicting class memberships. To that end, we observe a significant drop for each of the datasets, which becomes more severe as the number of relation type in the considered ontology grows. On DBpedia, for example, the accuracy achieved with a single prediction layer is just slightly better than random guessing.

5. Related Work

In this section, we provide a comparison to existing learning-based approaches to logical reasoning as well as to symbolic methods to logical reasoning. In this context, we also discuss some general limitations of the presented RRN model.

5.1 Learning-Based Logical Reasoning

While there has been an increasing interest in the application of learning-based methods to various kinds of logical reasoning in the last few years, ontology reasoning in particular has received just modest attention. The only published paper that we are aware of is by Makni & Hendler (2018), who developed an approach to RDFS reasoning contemporaneously with the work presented in this article. To that end, Makni & Hendler suggested an algorithm for mapping entire knowledge graphs to embedding vectors, which they refer to as graph words. Building on this, they use a BiGRU encoder-decoder architecture for translating the embedding of the knowledge graph specified by a database to the embedding of the according inference graph. Finally, the inference graph is reconstructed from the translated embedding, and predictions are extracted from the same. Makni & Hendler evaluated their approach on two different datasets, a toy dataset that makes use of the LUBM ontology (Guo et al., 2005), a very simple toy ontology, and real-world data that have been extracted from DBpedia. On the unimpaired versions of both datasets, they reported prediction accuracies of 0.98 and 0.87 for LUBM and DBpedia, respectively, which are well below the results that the RRN achieved on the comparable datasets that we used in our own experiments. Despite the fact that the RRN learns to perform ontology reasoning with a very high accuracy, it does not yet allow for providing justifications for the predicted inferences, such as via inference graphs like the model by Makni & Hendler (2018). The reason for this is that the RRN iteratively performs local updates of individual embeddings, which does not easily allow for tracking global inference paths. However, we are planning to extend the RRN architecture as part of future work in order to account for this task.

Another very recent work by Ebrahimi et al. (2018) addresses reasoning over RDF knowledge bases via an adapted version of end-to-end memory networks (MemN2N; Sukhbaatar et al., 2015). To that end, Ebrahimi et al. treat triples like text, and map the elements of any such (that is, subject, predicate, and object) to what they call normalized embeddings, which are created by first applying standard data preprocessing for logical reasoning, and then randomly mapping the names of primitives of the considered logical language (that is, variables, constants, functions, and predicates) to a predefined set of entity names. The rationale behind this is that the model should learn that names of primitives do not matter, and inferences depend on structural information about a knowledge base only. After this, the embedded triples are placed in the memory of an adapted MemN2N, which computes a prediction for a query that is also provided as an embedded triple. To evaluate their approach, Ebrahimi et al. extracted three different datasets, consisting of sample knowledge bases with 1000 triples each, from various public sources. They tested their model on a few different versions of each dataset, and reported 0.96 as the highest accuracy that has been achieved on any of them. In contrast to this, the RRN achieved an accuracy of more than 0.99 (taken over both classes and relations) in our own experiments with real-world data. Furthermore, the model introduced by Ebrahimi et al. (2018) does not compute all inferences that may be derived from a knowledge base, but provides predictions for single queries only. Differently from the approaches by Ebrahimi et al. (2018) and also Makni & Hendler (2018), the RRN is trained relative to a fixed ontology, instead of reading the same as part of the input. This is a design choice that involves a bit of trade off, since it results in reasoning models of unprecedented accuracy, as shown in our experiments, but requires us to train a new model from scratch whenever a different ontology is considered. We do believe, however, that this meets the requirements in practice, as ontologies are usually considered as fixed in most use cases.

Apart from this, a lot of previous work addresses the combination of logic-based symbolic reasoning and deep learning in some way, but is not related to ontology reasoning per se—notice that the following is not an exhaustive account. There exists a large body of previous work on neural-symbolic systems for learning and reasoning (see especially Sun & Alexandre, 2013; Hammer & Hitzler, 2007; d'Avila Garcez et al., 2012, 2015), which are classified into hybrid translational and hybrid functional systems. While the former translate and extract symbolic representations into and from neural networks, respectively, in which each symbolic sentence may be encoded in clearly localized neurons and in a distributed way over all neurons, the latter couple a symbolic and a neural representation and reasoning system (Sun & Alexandre, 2013). However, a coupling of two systems in hybrid functional systems comes with a mismatch of data learning capabilities, easy adaptability, and error tolerance between the two systems, while a localized hybrid translational system essentially just moves this mismatch into a single neural network.

Another line of research in neural-symbolic logical inference takes inspiration from the backward chaining algorithm used in logic programming technologies such as Prolog. More specifically, Rocktäschel & Riedel (2017) as well as Minervini et al. (2018) replace symbolic unification with a differentiable computation on vector representations of symbols, using radial basis function kernels. The approach learns to place representations of similar symbols in close proximity in a vector space, makes use of such similarities to prove queries, induces logical rules, and uses provided and induced logical rules for multi-step reasoning. It thus also combines logic-based reasoning with learning vector representations, but is intuitively best described as a vector simulation of backward chaining. Differently from these approaches, the RRN is in particular not able to induce additional ontological rules. Closely related are logic tensor networks (Serafini & d'Avila Garcez, 2016; Donadello et al., 2017), which are different from the approaches by Rocktäschel & Riedel (2017) and Minervini et al. (2018) in that they fully ground first-order logic rules and also support function terms. Other neural-symbolic approaches focus on first-order inference, but do not learn subsymbolic vector representations from training facts in a knowledge base, for example, CLIP++ (França et al., 2014), lifted relational neural networks (Sourek et al., 2015), Neural Prolog (Ding, 1995), SHRUTI (Shastri, 1992), and TensorLog (Cohen, 2016; Cohen et al., 2017). Further related earlier proposals for neural-symbolic networks for logical inference are limited to propositional rules, for example, C-IL 2 P (d'Avila Garcez & Zaverucha, 1999), EBL-ANN (Shavlik & Towell, 1991), and KBANN (Towell & Shavlik, 1994), including the recent ones by Bowman et al. (2014, 2015), which present successful approaches to simple propositional forms of logical reasoning. These approaches are based on recursive neural tensor networks, and consider the binary logical relationships entailment, equivalence, disjointness, partition, cover, and independence on elementary propositional events.

Another line of work (Rocktäschel et al., 2015; Demeester et al., 2016; Hu et al., 2016; Vendrov et al., 2016; Minervini, Costabello, et al., 2017; Minervini, Demeester, et al., 2017) regularizes distributed representations via domain-specific rules, but these approaches often support a restricted subset of first-order logic only. In particular, Rocktäschel et al. (2015) and Demeester et al. (2016) incorporate implication rules relative to ground terms into distributed representations for natural language processing, while Hu et al. (2016) incorporate ground instances of first-order rules into deep learning, projected to the ground terms of the learned data in each minibatch. Even with such restrictions, in experimental results in sentiment analysis and in named-entity recognition, the approach achieves (with a few intuitive rules) substantial improvements and state-of-the-art results to previous best-performing systems. Closely related are some extensions of knowledge-base completion that additionally consider non-factual symbolic knowledge to act as constraints to the learning process (Diligenti et al., 2012; Nickel et al., 2012). In particular, Diligenti et al. (2012) investigate a bridge between logic and kernel machines, and use non-factual symbolic knowledge as constraints in the second step of a two-stage process, where learning is done in the first one. Minervini, Costabello, et al. (2017) account for very simple relationships between different relation types, such as equivalence or inverse predicates, by making use of appropriate regularization terms. Closely related to this, Xu et al. (2018) introduced the semantic loss, which is another type of regularization term that allows for enforcing constraints that can be expressed by means of propositional logic. Another interesting approach by Minervini, Demeester, et al. (2017) incorporates background knowledge specified as Horn clauses into knowledge base completion by means of adversarial training (Goodfellow et al., 2014). Note that, in general, knowledge base completion (Socher et al., 2013; Trouillon et al., 2017), or link prediction in knowledge bases, which is the problem of predicting non-existing facts in a knowledge base consisting of a finite set of facts, differs from logical inference relative to a knowledge base, as it is generally missing logical knowledge beyond simple facts.

Finally, notice that within the deep-learning literature, the term »*reasoning*« is often used informally, that is, without reference to any kind of symbolic logic at all (e.g., Socher et al., 2013; Weston, Chopra, & Bordes, 2015; Henaff et al., 2017; Santoro et al., 2017).

5.2 Symbolic Methods of Logical Reasoning

One important question, which has been addressed briefly in the introduction already, is how to motivate the neural approach to ontology reasoning in the first place, and how the RRN relates to purely symbolic methods of logical reasoning.

First, the presented neural approach to ontology reasoning is a step towards answering the wide open problem of how to combine deep learning technologies with symbolic methods for logical reasoning, which is commonly regarded as a prerequisite for further substantial progress in AI. Implementing symbolic ontology reasoning with neural networks of very high accuracy in some sense bridges the gap between neural and symbolic methods, and offers new ways of providing machine learning models with reasoning capabilities that have previously been reserved for symbolic methods only, which opens up interesting new opportunities. From a machine-learning perspective, the RRN can be considered as a method of knowledge-graph embedding (Wang et al., 2017) that produces semantically meaningful embeddings of the individuals in a knowledge graph. These, in turn, may serve as input to models that are used for learning downstream tasks, and thus allows for leveraging symbolic background knowledge in learning deep neural networks (and thus for knowledge transfer and for learning from smaller amounts of data) and explainable symbolic inference in computing according predictions towards a better explainability of the learned neural systems.

Second, the neural approach to ontology reasoning is useful in its own right as an alternative to symbolic methods to logical reasoning. Even though it does not allow for fully accurate logical reasoning, it paves the way for highly scalable implementations of nearly accurate approximate logical reasoning via parallel computations on GPUs. Such implementations may not be used for safety-critical applications (such as for verifying the control of nuclear power plants), but it may be sufficiently accurate for many other applications where full accuracy is not required (such as question answering over the web).

In the same vein, one major issue that many symbolic approaches, including all reasoning formalisms rooted in classical logic, suffer from is conflicting information. In practice, however, information is frequently imperfect, which is why conflicts inevitably have to be dealt with in many use cases that symbolic reasoning is applied to. While many reasoning methods simply do not work in any such case, our experiments with the RRN suggest that the model is able to effectively resolve conflicts and thus better suited for applying logical reasoning in a real-world scenario. Even though there exist formalisms for reasoning over inconsistent knowledge bases that are also able to resolve conflicts, such as inconsistency-tolerant reasoning (Lembo et al., 2010) and paraconsistent logics (Middelburg, 2011), these approaches are generally quite limited in practice, since there is a price to be paid in terms of computational complexity. Apart from confined special cases, theoretically powerful methods are usually computationally harder than their counterparts for the consistent case (see, e.g., Lukasiewicz et al., 2015), whereas the RRN allows for reasoning in linear time, irrespective of any conflicts that may or may not exist in the considered knowledge base.

Finally, another challenge that is commonly encountered in practice is missing information, that is, details that are neither specified as facts in the considered knowledge base nor inferable from them via symbolic reasoning. Strictly speaking, recovering such missing pieces is a prediction rather than a reasoning task, and hence usually not considered in the context of symbolic reasoning. This is not in line with the requirements that are typically faced in practice, though, as we frequently seek to do both, compute predictions and perform reasoning. Again, however, our experiments indicate that this is exactly what the RRN does. To that end, we observed that the model is able to provide sensible predictions for any missing details that are compatible with the considered set of facts relative to the used ontology in many cases, and that these, in turn, also affect inferences computed by the model.

6. Conclusion

The results of this work show that the RRN model is able to learn to effectively reason over diverse ontological knowledge bases, and, in doing so, is the first one to achieve an accuracy that is very close to the yet unattainable accuracy of symbolic methods, while being distinctly more robust. Notice further that this was possible without the necessity of any kind of external memory, as it is used by many state-of-the-art models of deep learning, such as the differential neural computer (DNC; Graves et al., 2016) or memory networks (Weston, Chopra, & Bordes, 2015). This paves the way for applying comprehensive logical reasoning to a range of important problems that logic-based symbolic methods are, in general, hard to apply to. In addition to standard ontology reasoning tasks with missing or conflicting information, these include tasks such as understanding natural language, interpreting visual inputs, or even autonomous driving, which is highly demanding in terms of reasoning about an agent's surrounding. Furthermore, training an RRN relative to a suitable ontology is a simple and at the same time very powerful way to provide a model with domain knowledge, which is difficult to achieve with most of the state-of-the-art methods.

The RRN is among the very first deep-learning-based approaches to comprehensive ontology reasoning, which is why it is hard to compare to the state-of-the-art of machine learning models for reasoning, whose architectures do not allow for performing the same kind of inferences per se. Still, it is interesting to observe the performance of related approaches on learning tasks that are included as part of the reasoning problems presented in this work. For instance, the DNC was previously evaluated on the problem of determining relationships based on family trees (Graves et al., 2016), very much like the reasoning problem presented earlier. In doing so, it achieved an average accuracy of 81.8% on predicting four-step family relations, while the RRN predicted more than 99.9% of those relations correctly. Another interesting comparison is with the recently introduced Neural Theorem Prover (NTP; Rocktäschel & Riedel, 2017), which can indeed be used for simple ontology reasoning. As shown above, however, the RRN outperforms the NTP throughout all datasets, sometimes by more than 10% accuracy. The substantial margins observed do suggest that the RRN is able to grasp reasoning concepts that other state-of-the-art methods struggle with. The reason for this, we believe, is that our holistic approach to ontology reasoning adds a lot more structure to the considered data than other narrower prediction tasks, and hence makes it much easier for a model to »*understand*« the same. Furthermore, ontology reasoning, in many cases, naturally allows for breaking down complex inferences into several easier reasoning tasks, which, in some sense, provide guidance for learning to draw more elaborate conclusions.

Besides this, as discussed in Section 5, the RRN is superior to machine learning models for ontology reasoning in particular as well. This is most likely due to the fact that each RRN is trained relative to a fixed ontology, which allows for adjusting its architecture appropriately, while other existing approaches consider the specification of the same as part of the input. Hence, there seems to be a certain tradeoff between generality, and thus independence of the considered ontology, on the one hand, and reasoning accuracy, on the other. In practice, however, ontologies are usually considered to be fixed, which is why a higher accuracy is, in general, more desirable than a reasoning model that is not tied to a single ontology.

Finally, there is one subject that we raised just incidentally at the very beginning of this article. Despite the fact that the human brain served as a major source of inspiration for the development of artificial neural networks (Hassabis et al., 2017), most network architectures that are used for machine learning today lack biological plausibility. However, while many mechanics of the human neurology remain uncharted, there exist a number of aspects that are considered to be confirmed by now, and some of them allow for drawing interesting parallels to the RRN model. For instance, there is a broad consensus that reasoning in the human brain is not realized like logic-based symbolic reasoning, neither conceptually nor computationally. Instead, it is believed that our brain maintains a probabilistic cognitive model of the world (Oaksford & Chater, 2007), which provides a base for any kind of thought or action. To that end, the storage of semantic memory is organized in a distributed way, and information is encoded by strengthening synaptic connections among some neurons, while others are left to be strangers to one another (Martin & Chao, 2001). This is similar to how the RRN adjusts individual embeddings, which could be considered as groups of neurons, whenever a new piece of information arrives. Both absolute as well as relative positions of such vectors in the embedding space determine what is believed to be true about the world, which is why the adjustment of those may be interpreted as strengthening certain connections between neurons that store different pieces of information in a distributed way. Another interesting point is that the RRN, unlike other recent models, conducts logical reasoning without any kind of external memory, which is not believed to happen in the human brain either. Instead, reasoning is entangled with the generation of individual embeddings in the presented model, and thus part of encoding information in a distributed manner.

At the bottom line, the RRN embodies a surprising pairing of results, as it (i) learns to conduct highly accurate reasoning in a logic-based sense, (ii) is able to work with complex real-world knowledge bases, and (iii) is biologically plausible in a number of ways.

Despite the fact that an RRN is effectively guided by the ontology that it has been trained on, it cannot provide justifications for predictions yet. As an important future work, we are thus currently extending the presented architecture such that it allows for inducing rules and explanations, such as inference graphs, alongside computed predictions, hence making the RRN a fully interpretable neural reasoner. Other interesting topics for future research include the generalization of the presented approach to more expressive ontology languages, such as those involving existential rules like the Datalog$^{\pm}$ family (Calì et al., 2012), and to other forms of logical reasoning, such as default reasoning (Eiter & Lukasiewicz, 2000) and argumentation (Bench-Capon & Dunne, 2007).

Acknowledgments

This work was supported by the UK Engineering and Physical Sciences Research Council (EPSRC) under the grants EP/J008346/1, EP/R013667/1, EP/L012138/1, and EP/M025268/1, as well as the Alan Turing Institute under the EPSRC grant EP/N510129/1. Furthermore, Patrick is supported by the EPSRC under the grant OUCL/2016/PH and the Oxford-DeepMind Graduate Scholarship under the grant GAF1617_OGSMF-DMCS_1036172. We also acknowledge the use of the EPSRC-funded Tier 2 facility JADE (EP/P020275/1).

Appendix A. Experimental Setup

As part of the conducted experiments, we performed a grid search in order to determine an appropriate set of hyperparameters, all of which are reported in Table 6. Interestingly, the RRN seems to be broadly task-agnostic, as similar values worked well for all the considered datasets. Merely the size of the individual embeddings as well as the number of update iterations had to be adjusted based on the respective reasoning task. Furthermore, there was no need to manually create mini-batches of training data, as the single training samples

dataset	**embedding size**	**update iterations**	**initial learning rate**	**batch size**	**weight decay λ**	**MLP hidden layers**
family trees	100	7	0.001	1	10^{-6}	1
countries (S1)	100	2	0.001	1	10^{-6}	1
countries (S2)	200	5	0.001	1	10^{-6}	1
countries (S3)	200	5	0.001	1	10^{-6}	1
DBpedia	300	5	0.001	1	10^{-6}	1
Claros	200	5	0.001	1	10^{-6}	1
UMLS-reasoning	300	5	0.001	1	10^{-6}	1

Table 6: A summary of the hyperparameters that were used to achieve the results presented in this work.

contained numerous triples for each class and relation type already. All our models were trained by means of Adam (Kingma & Ba, 2015) with an initial learning rate of 0.001, β_1 set to 0.9, and β_2 set to 0.999. Furthermore, all MLPs had a single hidden layer of ReLU units and sigmoid units on the output layers. The sizes of the hidden layers were set to the average of input and output sizes for all of them. To prevent overfitting, we employed weight decay, which means that we added the term $\lambda\|\boldsymbol{\theta}\|_2$ to the computed loss, where $\boldsymbol{\theta}$ represents a vector of all model parameters, and $\lambda \in \mathbb{R}_{\geq 0}$ is a hyperparameter. The actual training loss was chosen to be a standard cross-entropy loss.

We trained our models for about two days on the toy datasets, and about seven days on the real-world datasets. However, for our experiments, we used a straightforward CPU-only implementation of the RRN model, which did not make use of any optimization or parallelization strategies. Therefore, it should be possible to reduce the duration of the model training by several orders of magnitude by making use of different ways to optimize the implementation of the model, as outlined in Section 3.8, as well as parallelizing computations over (multiple) GPUs. Generating embeddings for a single sample knowledge base took, on average, about 10 seconds for the toy datasets as well as Claros, and about 20 seconds for samples from DBpedia. Computing predictions for single queries took less than a tenth of a second for all datasets. Once again, however, it should be possible to reduce these durations significantly by optimizing the code.

Appendix B. Generation of Family-Tree Data

For the family-tree dataset, we generated training samples incrementally. For each of these samples, starting from a tree that consisted of a single person only, we randomly selected an existing person, and, again randomly, added either a child or, if possible, a parent to the current family tree. This process was repeated until either the maximum tree size, which we set to 26 people, was reached or the sample generation was stopped randomly, which happened with a probability of 0.02 whenever another person was added to a tree. As part of the data generation, we constrained sample family trees to a maximum depth of 5 as well as a maximum branching factor of 5. Furthermore, we ensured that the created dataset does not contain isomorphic family trees, such that it is not possible for a model to just overfit the training data. Finally, we computed all possible inferences for each of the created samples via symbolic reasoning.

Appendix C. Generation of Countries Data

Following the instructions in Nickel et al. (2016), we generated three datasets of varying difficulty, and endowed them with the formal ontology:

S1) This is the easiest version of the considered reasoning problem. Here, some of the countries' regions are missing, and thus have to be inferred from the knowledge about the subregions that those countries belong to, on the one hand, and the regions that the subregions are part of, on the other.

S2) In this variation of the problem, there are countries that have neither a region nor a subregion specified, and both of them have to be inferred from what is known about their neighborhoods. Notice that missing details in the test and evaluation set cannot be completed via ontology reasoning only, since countries that are situated at the border of a territory possess neighbors with differing regions and subregions, respectively.

S3) This is the hardest of the tasks considered. To that end, in addition to the problem described by version S2, the neighbors of those countries whose regions and subregions are missing do not have a region specified either. Again, problems of this kind cannot be solved solely by ontology reasoning.

While the original task considered reasoning about countries' locations only, we augmented the problem scenario with three classes that represent the types of individuals that occur, that is, countries, regions, and subregions. Irrespective of the considered version of the problem, none of these are ever provided as facts, and thus always have to be predicted as part of the reasoning problem. To that end, just like it is the case for some of the relations, classes cannot always be inferred via the ontology. This is the case, for example, if a region does not have any subregions at all, which means that we cannot leverage the transitivity of the located-in relation.

Every time we created a sample, we randomly selected 20 countries, and removed information about them, as required by the considered problem setting. In doing so, we ensured that every test country has at least one neighbor that is not part of the test set itself. For evaluating a trained model, we created two samples with such hold-out sets of countries for evaluation and testing, respectively, with two distinct sets of countries used as test individuals. To ensure that the model does not just overfit the data, we removed those 40 test countries from the knowledge base before we generated 5000 samples as training data. In addition to this, we included only those inferences in the evaluation and test sample that concerned at least one of the test individuals, again to make sure that the model cannot just overfit the training data, the only exception being regions and subregions, for which we included class predictions as well. Unlike this, training samples contain all inferences whatsoever.

Finally, in order to obtain a larger amount of test data, we generated 20 independent datasets for each of the three versions outlined above. For every one of those, we trained an RRN on the training data, and evaluated the model on the according test sample. The results reported in this work summarize the outcomes of all these evaluation runs.

Appendix D. Preparation of Real-world Data

For both considered real-world databases, DBpedia and Claros, we extracted training samples that are subgraphs of the knowledge graphs that are defined implicitly by the original data. To that end, the single training samples were created by running breadth-first search, starting from a randomly selected individual, on the knowledge graph that was specified by the facts in the respective database until a total of 200 individuals was discovered. Subsequently, all specified triples that concerned only those extracted individuals were considered as the facts of the created sample knowledge base, and their inferences were computed via symbolic reasoning. Like most of the knowledge bases that are encountered in practice, both DBpedia and Claros consist of positive facts and inferences only. Therefore, we made use of the so-called local closed-world assumption (Dong et al., 2014), and augmented the data with generated negative inferences that were created by randomly corrupting either the subject or the object of existing positive triples. This step is crucial, since the model would learn to blindly predict any queried inference to be true, otherwise. More precisely, we generated exactly one negative inference for each positive inference that exists in the data by corrupting each of these positive inferences exactly once. In doing so, we ensured, by means of symbolic reasoning, that the created triples do not introduce inconsistencies, that is, conflicts, in the created sample knowledge bases. Furthermore, whenever we generated a corrupted triple, we ensured that the same has not been added to the dataset before. For the evaluation and test data, we generated fixed sets of such negative inferences. For the training data, however, new negative triples were generated on-the-fly in each training iteration.

For the UMLS-reasoning dataset, we first created a Datalog program that implements that UMLS ontology. After this, we generated sample knowledge bases by means of an iterative procedure. For each sample, we started from an empty knowledge base that contained 75 individuals, but no facts about them. Then, we randomly sampled a candidate fact, evaluated whether it was consistent with the current state of knowledge relative to the ontology, and added it to the knowledge base, if possible. In the sampling step, we first decided randomly whether to add a class membership or a relation, both attributed a probability of 50%, and then sampled the remaining parts of the according triple. If an inconsistent fact was generated, then we created a new fact up to a total of 10 times, and if sampling a consistent fact failed 10 times in a row, then we stopped the procedure for the current sample knowledge base. Otherwise, this step was repeated until a maximum of 200 triples had been added to a sample knowledge base, and the remaining individuals that were not included in any fact got pruned again from the same. In addition to this, the generation of a sample knowledge base was stopped earlier with a probability of 0.5% every time after a fact was added. Finally, we computed all possible inferences for each sample knowledge base by means of symbolic reasoning with the created Datalog program.

Like for the family trees data, we ensured that the created datasets do not contain isomorphic sample knowledge graphs, such that it is not possible for a model to just overfit the training data.

Appendix E. Data Availability

All the datasets that have been used in our experiments are available from https://paho.at/rrn. Furthermore, the code that we used to generate our toy datasets, including the employed formal ontologies, is available as open source from https://github.com/phohenecker/family-tree-data-gen and https://github.com/phohenecker/country-data-gen, respectively.

Appendix F. Detailed Results for the Family Trees Dataset

In this section, we present detailed results of our experiments with the family-trees dataset.

Results for Specified Classes

	name	F1 score	AUC-PR	total accuracy	accuracy on positives	accuracy on negatives	# of triples (pos./neg.)
0	female	1.000	1.000	1.000	1.000	—	5732 / 0
1	male	1.000	1.000	1.000	1.000	—	5702 / 0

TOTAL	F1 score	AUC-PR	accuracy
	1.000	1.000	1.000

Results for Inferable Classes

	name	F1 score	AUC-PR	total accuracy	accuracy on positives	accuracy on negatives	# of triples (pos./neg.)
0	female	—	—	1.000	—	1.000	0 / 5702
1	male	—	—	1.000	—	1.000	0 / 5732

TOTAL	F1 score	AUC-PR	accuracy
	—	—	1.000

Results for Specified Relations

	name	F1 score	AUC-PR	total accuracy	accuracy on positives	accuracy on negatives	# of triples (pos./neg.)
23	parentOf	1.000	1.000	1.000	1.000	—	28232 / 0

TOTAL	F1 score	AUC-PR	accuracy
	1.000	1.000	1.000

Results for Inferable Relations

	name	F1 score	AUC-PR	total accuracy	accuracy on positives	accuracy on negatives	# of triples (pos./neg.)
0	auntOf	0.983	0.998	0.999	0.998	0.999	8268 / 556484
1	boyCousinOf	0.982	0.999	1.000	0.993	1.000	5806 / 558946
2	boyFirstCousinOnceRemovedOf	0.967	0.999	1.000	0.999	1.000	2554 / 562198
3	boySecondCousinOf	0.769	0.989	0.999	1.000	0.999	958 / 563794
4	brotherOf	0.908	0.995	0.996	0.999	0.996	10388 / 554364
5	daughterOf	0.991	0.999	1.000	1.000	1.000	14116 / 550636
6	fatherOf	0.992	0.999	1.000	1.000	1.000	14116 / 550636
7	girlCousinOf	0.966	0.998	0.999	0.994	0.999	5586 / 559166
8	girlFirstCousinOnceRemovedOf	0.944	0.999	0.999	0.998	0.999	2522 / 562230
9	girlSecondCousinOf	0.879	0.995	0.999	0.998	0.999	1050 / 563702
10	granddaughterOf	0.990	1.000	1.000	1.000	1.000	13374 / 551378
11	grandfatherOf	0.989	0.999	0.999	1.000	0.999	13374 / 551378
12	grandmotherOf	0.989	0.999	0.999	1.000	0.999	13374 / 551378
13	grandsonOf	0.993	0.999	1.000	0.999	1.000	13374 / 551378
14	greatAuntOf	0.979	0.999	1.000	0.998	1.000	4840 / 559912
15	greatGranddaughterOf	0.990	1.000	1.000	1.000	1.000	9692 / 555060
16	greatGrandfatherOf	0.989	0.999	1.000	1.000	1.000	9692 / 555060
17	greatGrandmotherOf	0.989	1.000	1.000	1.000	1.000	9692 / 555060
18	greatGrandsonOf	0.988	1.000	1.000	1.000	1.000	9692 / 555060
19	greatUncleOf	0.972	0.999	0.999	1.000	0.999	4922 / 559830
20	motherOf	0.989	0.999	0.999	1.000	0.999	14116 / 550636
21	nephewOf	0.972	0.999	0.999	0.999	0.999	8450 / 556302
22	nieceOf	0.976	0.998	0.999	0.997	0.999	8376 / 556376
23	parentOf	—	—	0.999	—	0.999	0 / 536520
24	secondAuntOf	0.958	0.998	1.000	0.995	1.000	2518 / 562234
25	secondUncleOf	0.946	0.999	0.999	0.997	0.999	2558 / 562194
26	sisterOf	0.922	0.996	0.997	0.999	0.997	10072 / 554680
27	sonOf	0.992	0.999	1.000	1.000	1.000	14116 / 550636
28	uncleOf	0.980	0.999	0.999	0.998	0.999	8558 / 556194

TOTAL	F1 score	AUC-PR	accuracy
	0.976	0.998	0.999

Appendix G. Detailed Results for the Countries Dataset

In this section, we present detailed results of our experiments with the different versions of the countries dataset.

G.1 Countries S1

Results for Inferable Classes

	name	F1 score	AUC-PR	total accuracy	accuracy on positives	accuracy on negatives	# of triples (pos./neg.)
0	country	1.000	1.000	1.000	1.000	1.000	400 / 580
1	region	1.000	1.000	1.000	1.000	1.000	120 / 860
2	subregion	1.000	1.000	1.000	1.000	1.000	460 / 520

TOTAL	F1 score	AUC-PR	accuracy
	1.000	1.000	1.000

Results for Specified Relations

	name	F1 score	AUC-PR	total accuracy	accuracy on positives	accuracy on negatives	# of triples (pos./neg.)
0	locatedIn	1.000	1.000	1.000	1.000	—	9160 / 0
1	neighborOf	1.000	1.000	1.000	1.000	—	10114 / 0

TOTAL	F1 score	AUC-PR	accuracy
	1.000	1.000	1.000

Results for Inferable Relations

	name	F1 score	AUC-PR	total accuracy	accuracy on positives	accuracy on negatives	# of triples (pos./neg.)
0	locatedIn	1.000	1.000	1.000	1.000	1.000	400 / 10800
1	neighborOf	—	—	0.999	—	0.999	0 / 172904

TOTAL	F1 score	AUC-PR	accuracy
	0.999	1.000	0.999

G.2 Countries S2

Results for Inferable Classes

	name	F1 score	AUC-PR	total accuracy	accuracy on positives	accuracy on negatives	# of triples (pos./neg.)
0	country	1.000	1.000	1.000	1.000	1.000	400 / 580
1	region	1.000	1.000	1.000	1.000	1.000	120 / 860
2	subregion	1.000	1.000	1.000	1.000	1.000	460 / 520

TOTAL	F1 score	AUC-PR	accuracy
	1.000	1.000	1.000

Results for Specified Relations

	name	F1 score	AUC-PR	total accuracy	accuracy on positives	accuracy on negatives	# of triples (pos./neg.)
0	locatedIn	0.996	0.999	0.995	0.995	—	8760 / 0
1	neighborOf	0.999	0.999	0.999	0.999	—	9888 / 0

TOTAL	F1 score	AUC-PR	accuracy
	0.997	0.999	0.999

Results for Inferable Relations

	name	F1 score	AUC-PR	total accuracy	accuracy on positives	accuracy on negatives	# of triples (pos./neg.)
0	locatedIn	0.987	0.991	0.999	0.986	0.999	800 / 10800
1	neighborOf	—	—	0.999	—	0.999	0 / 173072

TOTAL	F1 score	AUC-PR	accuracy
	0.929	0.991	0.999

G.3 Countries S3

Results for Inferable Classes

	name	F1 score	AUC-PR	total accuracy	accuracy on positives	accuracy on negatives	# of triples (pos./neg.)
0	country	1.000	1.000	1.000	1.000	1.000	400 / 580
1	region	1.000	1.000	1.000	1.000	1.000	120 / 860
2	subregion	1.000	1.000	1.000	1.000	1.000	460 / 520

TOTAL	F1 score	AUC-PR	accuracy
	1.000	1.000	1.000

Results for Specified Relations

	name	F1 score	AUC-PR	total accuracy	accuracy on positives	accuracy on negatives	# of triples (pos./neg.)
0	locatedIn	0.993	0.999	0.994	0.994	—	7820 / 0
1	neighborOf	1.000	1.000	1.000	1.000	—	10216 / 0

TOTAL	F1 score	AUC-PR	accuracy
	0.996	0.999	0.999

Results for Inferable Relations

	name	F1 score	AUC-PR	total accuracy	accuracy on positives	accuracy on negatives	# of triples (pos./neg.)
0	locatedIn	0.930	0.988	0.998	0.981	0.998	800 / 10800
1	neighborOf	—	—	0.999	—	0.999	0 / 172966

TOTAL	F1 score	AUC-PR	accuracy
	0.916	0.988	0.999

References

Bench-Capon, T. J. M., & Dunne, P. E. (2007). Argumentation in artificial intelligence. *Artificial Intelligence*, *171*(10-15), 619–641.

Bengio, Y. (2009). Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, *2*(1), 1–127.

Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., & Hellmann, S. (2009). DBpedia – A crystallization point for the Web of Data. *Journal of Web Semantics*, *7*(3), 154–165.

Bouchard, G., Singh, S., & Trouillon, T. (2015). On approximate reasoning capabilities of low-rank vector spaces. In *Proceedings of the 2015 AAAI Spring Symposium on Knowledge Representation and Reasoning: Integrating Symbolic and Neural Approaches* (pp. 6–9).

Bowman, S. R., Potts, C., & Manning, C. D. (2014). Recursive neural networks can learn logical semantics. *arXiv preprint arXiv:1406.1827*.

Bowman, S. R., Potts, C., & Manning, C. D. (2015). Learning distributed word representations for natural logic reasoning. In *Proceedings of the 2015 AAAI Spring Symposium on Knowledge Representation and Reasoning: Integrating Symbolic and Neural Approaches* (pp. 10–13).

Cai, C.-H., Ke, D., Xu, Y., & Su, K. (2017). Symbolic manipulation based on deep neural networks and its application to axiom discovery. In *Proceedings of the 2017 International Joint Conference on Neural Networks* (pp. 2136–2143).

Calì, A., Gottlob, G., & Lukasiewicz, T. (2012). A general Datalog-based framework for tractable query answering over ontologies. *Journal of Web Semantics*, *14*, 57–83.

Cingillioglu, N., & Russo, A. (2018). DeepLogic: End-to-end logical reasoning. *arXiv preprint arXiv:1805.07433*.

Cohen, W. W. (2016). TensorLog: A differentiable deductive database. *arXiv preprint arXiv:1605.06523*.

Cohen, W. W., Yang, F., & Rivard Mazaitis, K. (2017). TensorLog: Deep learning meets probabilistic DBs. *arXiv preprint arXiv:1707.05390*.

Dai, W.-Z., Xu, Q.-L., Yu, Y., & Zhou, Z.-H. (2018). Tunneling neural perception and logic reasoning through abductive learning. *arXiv preprint arXiv:1802.01173*.

d'Avila Garcez, A. S., Besold, T. R., De Raedt, L., Földiak, P., Hitzler, P., Icard, T., … Silver, D. L. (2015). Neural-symbolic learning and reasoning: Contributions and challenges. In *Proceedings of the 2015 AAAI Spring Symposium on Knowledge Representation and Reasoning: Integrating Symbolic and Neural Approaches* (pp. 18–21).

d'Avila Garcez, A. S., Broda, K. B., & Gabbay, D. M. (2012). *Neural-Symbolic Learning Systems: Foundations and Applications*. Springer Science & Business Media.

d'Avila Garcez, A. S., & Zaverucha, G. (1999). The connectionist inductive learning and logic programming system. *Applied Intelligence*, *11*, 59–77.

Demeester, T., Rocktäschel, T., & Riedel, S. (2016). Lifted rule injection for relation embeddings. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing* (pp. 1389–1399).

Diligenti, M., Gori, M., Maggini, M., & Rigutini, L. (2012). Bridging logic and kernel machines. *Machine Learning*, *86*, 57–88.

Ding, L. (1995). Neural Prolog—The concepts, construction and mechanism. In *Proceedings of the 1995 IEEE International Conference on Systems, Man and Cybernetics. Intelligent Systems for the 21st Century* (Vol. 4, pp. 3603–3608).

Donadello, I., Serafini, L., & d'Avila Garcez, A. (2017). Logic tensor networks for semantic image interpretation. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence* (pp. 1596–1602).

Dong, X. L., Gabrilovich, E., Heitz, G., Horn, W., Lao, N., Murphy, K., … Zhang, W. (2014). Knowledge Vault: A Web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 601–610).

Ebrahimi, M., Sarker, M. K., Bianchi, F., Xie, N., Doran, D., & Hitzler, P. (2018). Reasoning over RDF knowledge bases using deep learning. *arXiv preprint arXiv:1811.04132*.

Eiter, T., & Lukasiewicz, T. (2000). Default reasoning from conditional knowledge bases: Complexity and tractable cases. *Artificial Intelligence*, *124*(2), 169–241.

Evans, R., Saxton, D., Amos, D., Kohli, P., & Grefenstette, E. (2018). Can neural networks understand logical entailment? In *Proceedings of the 6th International Conference on Learning Representations.*

França, M. V. M., Zaverucha, G., & d'Avila Garcez, A. S. (2014). Fast relational learning using bottom clause propositionalization with artificial neural networks. *Machine Learning*, *94*(1), 81–104.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., … Bengio, Y. (2014). Generative adversarial nets. In *Advances in Neural Information Processing Systems 27.*

Graves, A., Wayne, G., Reynolds, M., Harley, T., Danihelka, I., Grabska-Barwińska, A., … Hassabis, D. (2016). Hybrid computing using a neural network with dynamic external memory. *Nature*, *538*, 471–476.

Guo, Y., Pan, Z., & Heflin, J. (2005). LUBM: A benchmark for OWL knowledge base systems. *Journal of Web Semantics*, *3*(2/3), 158–182.

Hammer, B., & Hitzler, P. (2007). *Perspectives of Neural-Symbolic Integration*. Springer.

Hassabis, D., Kumaran, D., Summerfield, C., & Botvinick, M. (2017). Neuroscience-inspired artificial intelligence. *Neuron*, *95*(2), 245–258.

Henaff, M., Weston, J., Szlam, A., Bordes, A., & LeCun, Y. (2017). Tracking the world state with recurrent entity networks. *arXiv preprint arXiv:1612.03969*.

Hu, Z., Ma, X., Liu, Z., Hovy, E., & Xing, E. (2016). Harnessing deep neural networks with logic rules. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 2410–2420).

Kingma, D. P., & Ba, J. L. (2015). Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*.

Kok, S., & Domingos, P. (2007). Statistical predicate invention. In *Proceedings of the 24th International Conference on Machine Learning* (pp. 433–440).

Lembo, D., Lenzerini, M., Rosati, R., Ruzzi, M., & Savo, D. F. (2010). Inconsistency-tolerant semantics for description logics. In *Proceedings of the 4th International Conference on Web Reasoning and Rule Systems* (pp. 103–117).

Lukasiewicz, T., Martinez, M. V., Pieris, A., & Simari, G. I. (2015). From classical to consistent query answering under existential rules. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence* (pp. 1546–1552).

Makni, B., & Hendler, J. (2018). Deep learning for noise-tolerant RDFS reasoning. *Semantic Web*, *10*(5), 823–862.

Manhaeve, R., Dumančić, S., Kimmig, A., Demeester, T., & De Raedt, L. (2018). DeepProbLog: Neural probabilistic logic programming. *arXiv preprint arXiv:1805.10872*.

Martin, A., & Chao, L. L. (2001). Semantic memory and the brain: Structure and processes. *Current Opinion in Neurobiology*, *11*(2), 194–201.

McCray, A. T. (2003). An upper-level ontology for the biomedical domain. *Comparative and Functional Genomics*, *4*, 80–84.

Middelburg, C. A. (2011). A survey of paraconsistent logics. *arXiv preprint arXiv:1103.4324*.

Mikolov, T., Corrado, G., Chen, K., & Dean, J. (2013). Efficient estimation of word representations in vector space. In *Proceedings of the 1st International Conference on Learning Representations*.

Minervini, P., Bosnjak, M., Rocktäschel, T., & Riedel, S. (2018). Towards neural theorem proving at scale. *arXiv preprint arXiv:1807.08204*.

Minervini, P., Costabello, L., Muñoz, E., Nováček, V., & Vandebussche, P.-Y. (2017). Regularizing knowledge graph embeddings via equivalence and inversion axioms. In *Machine Learning and Knowledge Discovery in Databases* (pp. 668–683).

Minervini, P., Demeester, T., Rocktäschel, T., & Riedel, S. (2017). Adversarial sets for regularising neural link predictors. *arXiv preprint arXiv:1707.07596*.

Nenov, Y., Piro, R., Motik, B., Horrocks, I., Wu, Z., & Banerjee, J. (2015). RDFox: A highly-scalable RDF store. In *Proceedings of the 14th International Semantic Web Conference* (pp. 3–20).

Nickel, M., Rosasco, L., & Poggio, T. (2016). Holographic embeddings of knowledge graphs. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence* (pp. 1955–1961).

Nickel, M., Tresp, V., & Kriegel, H.-P. (2012). Factorizing YAGO: Scalable machine learning for Linked Data. In *Proceedings of the 21st World Wide Web Conference* (pp. 271–280).

Oaksford, M., & Chater, N. (2007). *Bayesian Rationality: The Probabilistic Approach to Human Reasoning*. Oxford University Press.

Pollack, J. B. (1990). Recursive distributed representations. *Artificial Intelligence*, *46*(1/2), 77–105.

Rocktäschel, T., & Riedel, S. (2017). End-to-end differentiable proving. In *Advances in Neural Information Processing Systems 30* (pp. 3788–3800).

Rocktäschel, T., Singh, S., & Riedel, S. (2015). Injecting logical background knowledge into embeddings for relation extraction. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 1119–1129).

Santoro, A., Raposo, D., Barrett, D. G., Malinowski, M., Pascanu, R., Battaglia, P., & Lillicrap, T. (2017). A simple neural network module for relational reasoning. In *Advances in neural information processing systems 30*.

Serafini, L., & d'Avila Garcez, A. (2016). Logic tensor networks: Deep learning and logical reasoning from data and knowledge. *arXiv preprint arXiv:1606.04422*.

Shastri, L. (1992). Neurally motivated constraints on the working memory capacity of a production system for parallel processing: Implications of a connectionist model based on temporal synchrony. In *Proceedings of the 14th Annual Conference of the Cognitive Science Society* (Vol. 29, pp. 159–164).

Shavlik, J. W., & Towell, G. G. (1991). An approach to combining explanation-based and neural learning algorithms. In *Applications of Learning and Planning Methods* (pp. 231–253). World Scientific.

Socher, R., Chen, D., Manning, C. D., & Ng, A. Y. (2013). Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems 26* (pp. 926–934).

Sourek, G., Aschenbrenner, V., Zelezny, F., & Kuzelka, O. (2015). Lifted relational neural networks. *arXiv preprint arXiv:1508.05128*.

Sukhbaatar, S., Szlam, A., Weston, J., & Fergus, R. (2015). End-to-end memory networks. In *Advances in Neural Information Processing Systems 28* (pp. 2440–2448).

Sun, R., & Alexandre, F. (2013). *Connectionist-Symbolic Integration: From Unified to Hybrid Approaches.* Psychology Press.

Towell, G. G., & Shavlik, J. W. (1994). Knowledge-based artificial neural networks. *Artificial Intelligence*, *70*(1/2), 119–165.

Trouillon, T., Dance, C. R., Gaussier, É., Welbl, J., Riedel, S., & Bouchard, G. (2017). Knowledge graph completion via complex tensor factorization. *Journal of Machine Learning Research*, *18*(130), 1–38.

Vendrov, I., Kiros, R., Fidler, S., & Urtasun, R. (2016). Order-embeddings of images and language. In *Proceedings of the 4th International Conference on Learning Representations.*

Wang, Q., Mao, Z., Wang, B., & Guo, L. (2017). Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, *29*(12), 2724–2743.

Weston, J., Bordes, A., Chopra, S., & Mikolov, T. (2015). Towards AI-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*.

Weston, J., Chopra, S., & Bordes, A. (2015). Memory networks. In *Proceedings of the 3rd International Conference on Learning Representations.*

Xu, J., Zhang, Z., Friedman, T., Liang, Y., & Van den Broeck, G. (2018). A semantic loss function for deep learning with symbolic knowledge. In *Proceedings of the 35th International Conference on Machine Learning* (pp. 5498–5507).