**Noname manuscript No.**
(will be inserted by the editor)

# Towards Bridging the Neuro-Symbolic Gap: Deep Deductive Reasoners

**Monireh Ebrahimi · Aaron Eberhart · Federico Bianchi · Pascal Hitzler**

Received: date / Accepted: date

**Abstract** Symbolic knowledge representation and reasoning and deep learning are fundamentally different approaches to artificial intelligence with complementary capabilities. The former are transparent and data-efficient, but they are sensitive to noise and cannot be applied to non-symbolic domains where the data is ambiguous. The latter can learn complex tasks from examples, are robust to noise, but are black boxes; require large amounts of –not necessarily easily obtained– data, and are slow to learn and prone to adversarial examples. Either paradigm excels at certain types of problems where the other paradigm performs poorly. In order to develop stronger AI systems, integrated neuro-symbolic systems that combine artificial neural networks and symbolic reasoning are being sought. In this context, one of the fundamental open problems is how to perform logic-based deductive reasoning over knowledge bases by means of trainable artificial neural networks.

This paper provides a brief summary of the authors' recent efforts to bridge the neural and symbolic divide in the context of deep deductive reasoners. Throughout the paper we will discuss strengths and limitations of models in term of accuracy, scalability, transferability, generalizabiliy, speed, and interpretability, and finally will talk about possible modifications to enhance desirable capabilities. More specifically, in terms of architectures, we are looking at Memory-augmented networks, Logic Tensor Networks, and compositions of LSTM models to explore their capabilities and limitations in conducting deductive reasoning. We are ap-

M. Ebrahimi
Data Semantics Laboratory
Kansas State University, USA E-mail: monireh@ksu.edu

A. Eberhart
Data Semantics Laboratory
Kansas State University, USA E-mail: aaroneberhart@ksu.edu

F. Bianchi
Bocconi University, Italy E-mail: f.bianchi@unibocconi.it

P. Hitzler
Data Semantics Laboratory
Kansas State University, USA E-mail: hitzler@ksu.edu

plying these models on Resource Description Framework (RDF), first-order logic, and the description logic $\mathcal{EL}^+$respectively.

**Keywords** Neuro-symbolic reasoning · Deep deductive reasoners · Memory-augmented networks · Logic tensor networks · LSTM · Logic

## 1 Introduction & Motivation

Approaches in Artificial Intelligence (AI) based on machine learning, and in particular those employing artificial neural networks, differ fundamentally from approaches that perform logical deduction and reasoning on knowledge bases. The first are *connectionist* or *subsymbolic* AI systems that are able to solve complex tasks over unstructured data using supervised or unsupervised learning, including problems which cannot reasonably be hand-coded by humans. Subsymbolic methods are generally robust against noise in training or input data. And recently, in the wake of deep learning, they have been shown to exceed human performance in tasks involving video, audio, and text processing. *Symbolic* systems, by contrast, thrive in tasks that use highly structured data, including agent planning, constraint solving, data management, integration and querying, and other traditional application areas of expert systems and formal semantics. Classical rule-based systems, ontologies, and knowledge graphs that power search and information retrieval across the Web are also types of symbolic AI systems.

Symbolic and subsymbolic systems are almost entirely complementary to each other. For example, the key strengths of subsymbolic systems are weaknesses of symbolic ones, and vice versa. Symbolic systems are *brittle*; they are susceptible to data noise or minor flaws in the logical encoding of a problem, which stands in contrast to the robustness of connectionist approaches. But subsymbolic systems are generally *black boxes* in the sense that the systems cannot be inspected in ways that provide insight into their decisions (despite some recent progress on this in the *explainable AI* effort) while symbolic knowledge bases can in principle be inspected to interpret how a decision follows from input. Most importantly, symbolic and subsymbolic systems differ in the types of problems and data they excel at solving. Scene recognition from images appears to be a problem that lies generally outside the capabilities of symbolic systems, for example, while complex planning scenarios appear to be outside the scope of current deep learning approaches.[1]

The complementary nature of these methods has drawn a stark divide in the rich field of AI. The split is technical in nature; symbol manipulation as captured by deductive reasoning cannot be sufficiently performed using current subsymbolic systems. Moreover, the training to study subsymbolic systems (involving probability theory, statistics, linear algebra, and optimization) differs from symbolic systems (involving logic and propositional calculus, set theory, recursion, and computability) so strongly that AI researchers tend to find a side of the divide based on their intellectual interests and background. There is even a cultural aspect to he schism, pitting mindsets and prior beliefs of communities against one

[1] The topic is being investigated, of course, with some recent progress being made. For example, [1] report on an application of deep learning to planning, and explicitly frame it as work towards bridging the "subsymbolic-symbolic boundary."

another, that in the past could sometimes split the academic AI research community by provoking (heated) fundamental discussions. Even geography has an effect: researchers working on symbolic approaches are more prevalent in the European Union than in the United States.

We are interested in answering fundamental problems needed to build a technical bridge between the symbolic and subsymbolic sides of the divide. The promises of successfully bridging of the technological divide are plenty [29,33,7,17]. In the abstract, one could hope for best-of-both-world systems, which combines the transparency and reasoning-ability of symbolic systems with the robustness and learning-capabilities of subsymbolic ones. Integrated symbolic-subsymbolic systems may be able to address the knowledge acquisition bottleneck faced by symbolic systems, learn to perform advanced logical or symbolic reasoning tasks even in the presence of noisy or uncertain facts, and even yield self-explanatory subsymbolic models. More abstractly, bridging the two may also shed insights into how natural (human) neural networks can perform symbolic tasks as witnessed by people doing mathematics, formal logic, and other pursuits that we, introspectively, see as symbolic in nature. This is a basic research problem for Cognitive Science.

This paper provides the brief summary of the authors' recent efforts toward bridging the neural and symbolic approaches divide. Indeed, this work is a merged and expanded version of the authors' recent publications [26,9,8,25] at conferences and symposia.[2] Throughout the paper we will discuss strengths and limitations of models in term of the accuracy, scalability, transferability, generalizabiliy, speed, and interpretability capability and finally will talk about possible modifications to enhance such desirable capabilities. In terms of architectures, we are looking at Memory-augmented networks, Logic Tensor Networks (LTNs), and compositions of LSTM models to explore their capabilities and limitations in conducting deductive reasoning. We are applying these models to RDF, first-order logic, and the description logic $\mathcal{EL}^+$ respectively.

The paper is organized as follows: in section 2 we summarize related work for our line of research. Section 3 provides a summary of our work in the context of techniques, logics, and logical embeddings that have been used. In section 4 we first outline the experimental results of our memory network based RDF deductive reasoning system with focus on transferability and generalization. Next we explore LTNs in the context of deductive reasoning tasks, highlighting the properties, weaknesses, and strengths of these models. We also show that integrating subsymbolic commonsense representations in the form of pre-trained embeddings improves the performance of LTNs for reasoning tasks. Finally, we give an overview of our work on conducting reasoning for the more complex description logic $\mathcal{EL}^+$. We give concluding remarks and ideas for future work in Section 5.

## 2 Related Work

The research into how subsymbolic systems can perform deductive reasoning is often referred to as the study of neuro-symbolic integration. It can be traced

[2] [26] is under review at AAAI-MAKE 2021 symposium at the time of submitting this journal paper.

back at least to a landmark 1942 article by McCulloch and Pitts [53] showing how propositional logic formulas can be represented using simple neural network models with threshold activation functions. A comprehensive and recent state of the art survey can be found in [7], and hence we will only mention essentials for understanding the context of our work.

Most of the body of work on neuro-sybmolic integration concerns propositional logic only (see, for example, [28]), and relationships both theoretical and practical in nature between propositional logics and subsymbolic systems are relatively easy to come by, an observation to which John McCarthy refered as the "propositional fixation" of artificial neural networks [52]. Some of these include Knowledge-Based Artificial Neural Networks [74] and the closely related propositional Core method [42,36]. Early attempts to go beyond propositional logic included the SHRUTI system [70,71] which, however, uses a non-standard connectionist architecture and thus had severe limitations as far as learning was concerned. Approaches that use standard artificial neural network architectures with proven learning capabilities for first-order predicate logic [32] or first-order logic programming [5,4] were by their very design unable to scale beyond toy examples.

In the past few years, however, deep learning as a subsymbolic machine learning paradigm has surpassed expectations in machine-learning based problem solving, and it is a reasonable assumption that these developments have not yet met their natural limit. Consequently, they are being looked upon as promising for trying to overcome the symbolic-subsymbolic divide [1,23,67,68,51,41,64] – this list is not exhaustive. Even more work exists on inductive logical inference, for example [64, 24,59], but this is not what we are investigating in our work.[3] Recently, neural theorem provers [64] have shown exciting capabilities [57,56] in link prediction tasks.

On the issue of logical reasoning using deep networks we mention some selected contributions. Tensor-based approaches for reasoning have been proposed [23,67,68,64], following [72,31], but present models remain restricted in terms of logical expressivity and/or to toy examples and limited evaluations. [51] performs knowledge graph reasoning using RDF(S) [39,19] based on knowledge graph embeddings. However evaluation and training is done on the same knowledge graph, that is, there is no learning of the general logical deduction calculus, and consequently no transfer thereof to new data. Likewise, recent years have seen some progress in zero-shot relation learning in the subsymbolic domain [58,66]. Zero-shot learning refers to the ability of the model to infer new unseen relationships between pairs of entities. This generalization capability is still quite limited and fundamentally different from our work in terms of both methodology and purpose. [41] moves away from RDFS to consider OWL RL reasoning [39,38], however again no general deduction calculus is acquired during training.

There are different approaches from Statistical Relational Learning that do not integrate neural networks with logic, but rather tackle the problem in a symbolic manner by also using statistical information. Examples from this category are ProbLog [21], which is a probabilistic logic programming language, and Markov Logic Networks (MLNs) are a statistical relational learning model that are effective

[3] Induction like in Inductive Logic Programming or Relational Learning has statistical aspects and is much closer in nature to a machine learning task, and thus arguably easier to tackle using machine learning approaches.

on a large variety of tasks [62,54]. The intuition behind MLNs and LTNs is similar since they both base their approach on logical languages. MLNs define weights for formulas and interpret the world from a probabilistic point of view, while LTNs use fuzzy logic and a neural architectures to generate their inferences.

Finally, in the context of description logic reasoning there are additional unique challenges for the neuro-symbolic integration task. Not only are variable terms *implicit*, or not stated, in expressions, but also the open world assumption means that there is no fixed set of constants to use in training like in logic programming. There is promising work that attempts to use neural networks to reason over description logic profiles that have monotonic reasoning behavior using completion rules. For $\mathcal{EL}^+$and $\mathcal{EL}^{++}$reasoning, for example, some attempt to embed $\mathcal{EL}^{++}$ with translational embedding (TransE) using a novel concept of $n$-balls, though it currently does not consider the RBox as well as certain $\mathcal{EL}^{++}$ axioms that do not translate into the embedding [46,14].

## 3 Summary of our Work

Subsymbolic systems are trained to produce an output given some input, which may be a label (classification) or a numerical value (regression). For our RDF reasoning and some experiments for first-order logic reasoning, we re-frame the task as a classification problem. Any[4] given logic $\mathcal{L}$ comes with an entailment relation $\models_{\mathcal{L}} \subseteq T_{\mathcal{L}} \times F_{\mathcal{L}}$, where $F_{\mathcal{L}}$ is a subset of the set of all logical formulas (or axioms) over $\mathcal{L}$, and $T_{\mathcal{L}}$ is the set of all theories (or sets of logical formulas) over $\mathcal{L}$. If $T \models F$, then we say that $F$ is *entailed* by $T$. For a classification task we can ask whether a given pair $(T, F) \in T_{\mathcal{L}} \times F_{\mathcal{L}}$ should be classified as a valid entailment (i.e., $T \models_{\mathcal{L}} F$ holds), or as the opposite (i.e., $T \not\models_{\mathcal{L}} F$). We seek to train a DNN over the sets of examples $(T, F)$ embedded into a vector space amenable for DNN processing, such that the DNN learns to correctly classify examples as valid or invalid inferences. Of course, we would have to restrict our attention to finite theories, which is usually done in computational logic anyway.

Another way to re-frame the deductive reasoning problem (used in our $\mathcal{EL}^+$ reasoning task) is by considering, for each theory $T \in T_{\mathcal{L}}$, the set $c(T) = \{F \in F_{\mathcal{L}} \mid T \models_{\mathcal{L}} F\}$ of all formulas entailed by $T$; we call $c(T)$ the *completion* of $T$. We can then attempt to train a DNN to produce $c(T)$ for any given $T \in \mathcal{L}$, i.e., we would use pairs $(T, c(T))$ as input-output training pairs. In this case, restricting to finite $T$ may not be entirely sufficient because even for finite $T$ it is possible that $c(T)$ may be infinite. In such cases, we will have to restrict our training data to large but finite subsets of $c(T)$.[5]

Figure 1 illustrates the general setting within which the studies in this paper reside. Under a three dimensional investigative space, whereby the logic under consideration, the logical embedding, and a DNN model type, we examine capabilities and limits of different DNN architectures to perform deductive reasoning and to transfer their learning to unseen knowledge bases encoded in the same logic. Transferability means that a DNN demonstrates reasoning capabilities that

[4] *Any* may be too grandiose a statement, but these are the ones we are looking at.

[5] Attempting to find finite representations for infinite sets – in the cases where this would even be reasonably possible – would add another layer of complication which we are currently not considering.

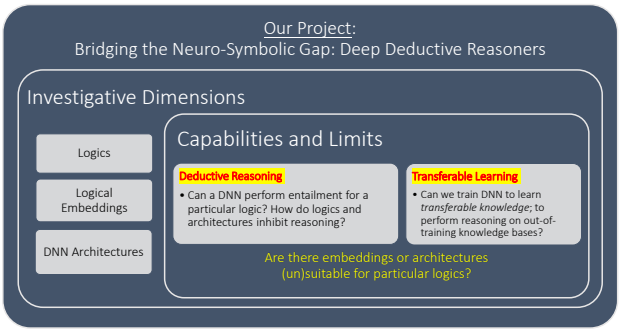


**Fig. 1** Our work explores the capabilities and limits of subsymbolic systems (DNNs) to perform transferrable deductive reasoning. Experiments are carried out along three investigative dimensions: type of logic, embeddings of logic to a quantitative space, and DNN architectures.

require acquisition of *principles* (or "inference rules," if you like) which underlie the logic and not simply specific answers. If we were to train the DNN such that it learns only to reason over *one* theory, then this could hardly be demonstrated. Theoretical perspectives are derived by studying model performance under particular constraints in the knowledge bases, for example, the degree or number of reasoning steps needed to determine an entailment.

Our general line of research can be understood based upon a selection of (i) candidate logic; (ii) logical embedding method; and (iii) DNN architecture. To provide context, here, fist we discuss each investigative dimension in more detail:

*Logics* We have so far looked at three logics of different complexity, as listed below. Two of them, RDFS and $\mathcal{EL}^+$ come from the context of Semantic Web [39] research, and have direct bearing on current data management practice [34]. The Semantic Web field indeed provides ample opportunity to instigate neuro-symbolic integration approaches [35].

(1) *RDFS (Resource Description Framework Schema).* The Resource Description Framework RDF, which includes RDF Schema (RDFS) [19,39] is an established and widely used W3C standard for expressing knowledge graphs. The standard comes with a formal semantics[6] that define an entailment relation. An RDFS knowledge base (KB) is a collection of statements stored as *triples* $(e1, r, e2)$ where $e1$ and $e2$ are called *subject* and *object*, respectively, while $r$ is a binary relation between $e1$ and $e2$.

As a logic, RDFS is of very low expressivity and reasoning algorithms are very straightforward. In fact, there is a small set of thirteen entailment rules [18], fixed

[6] In fact, it comes with three different ones, but we have only considered the most comprehensive one, the RDFS Semantics.

$$(x, \text{rdf:subClassOf}, y) \wedge (y, \text{rdfs:subClassOf}, z) \vdash (x, \text{rdfs:subClassOf}, z) \quad (1)$$
$$(x, \text{rdfs:subPropertyOf}, y) \wedge (y, \text{rdfs:subPropertyOf}, z) \vdash (x, \text{rdfs:subPropertyOf}, z) \quad (2)$$
$$(x, \text{rdfs:subClassOf}, y) \wedge (z, \text{rdf:type}, x) \vdash (z, \text{rdf:type}, y) \quad (3)$$
$$(a, \text{rdfs:domain}, x) \wedge (y, a, z) \vdash (y, \text{rdf:type}, x) \quad (4)$$
$$(a, \text{rdfs:range}, x) \wedge (y, a, z) \vdash (z, \text{rdf:type}, x) \quad (5)$$

**Fig. 2** Some RDFS entailment rules. Explanations can be found in the main text.

across all knowledge graphs, which are expressible using Datalog.[7] These thirteen rules can be used to entail new facts.

Figure 2 shows examples for some of these entailment rules. The identifiers $x, y, z, a$ are variables. The remaining elements of the triples are pre-fixed with the rdfs or rdf namespace and carry a specific meaning in the formal semantics of RDFS. E.g., rdfs:subClassOf indicates a sub-class (or sub-set) relationship, i.e. Rule 1 states transitivity of the rdf:subClassOf binary relation. Likewise, in Rule 2, $(x, \text{rdfs:subPropertyOf}, y)$ indicates that $x, y$ are to be understood as binary relations, where $x$ is a restriction (called a *subproperty*) of $y$. In Rule 3, the triple $(z, \text{rdf:type}, x)$ indicates that $z$ is a member of the class (or set) $x$. In Rules 4 and 5, rdfs:domain and rdfs:range indicate domain respectively range of $a$, which is to be interpreted as a binary relation.

(2) *Real Logic/First-Order Fuzzy Logic.* First-order fuzzy logic [45] is a generalization of first-order logic in which binary truth values are replaced with continuous real values in the range of $[0, 1]$. In [69] the authors introduce Real Logic, the logic that will be used to define Logic Tensor Networks in the next sections. This logic allows us to express the degree of truth of a given axiom that is not as crisp as binary logic. However, moving to continuous values also requires changing the behaviour of standard logical connectives such as conjunction: new operations have to be considered that can accommodate the real values of the logic. For example, the standard conjunction can be replaced with a t-norm; and different implementations of t-norms exist, like the Gödel t-norm which, given two values $a$ and $b$ equals $min(a, b)$. More details about how the connectives are treated can be found in [69].

(3) $\mathcal{EL}^+$. $\mathcal{EL}^+$ is a lightweight and highly tractable description logic [39]. A typical reasoning task in $\mathcal{EL}^+$ is a sequential process with a fixed endpoint, making it a perfect candidate for sequence learning. Unlike RDF, which reasons over links between instance data in triple format, $\mathcal{EL}^+$ reasoning occurs on the predicate level. Thus reasoning requires training the system to actually learn reasoning patterns and logical structure of $\mathcal{EL}^+$ directly from encoded knowledge bases.

The signature $\Sigma$ for $\mathcal{EL}^+$ is defined as $\Sigma = \langle N_I, N_C, N_R \rangle$ with $N_I, N_C, N_R$ pairwise disjoint. $N_I$ is a set of individual names, $N_C$ is a set of concept names that includes $\top$, and $N_R$ is a set of role names.An $\mathcal{EL}^+$ knowledge base consists of a finite set of statements of the form $\mathbf{C} \sqsubseteq \mathbf{C}$, $\mathbf{R} \sqsubseteq \mathbf{R}$ and $\mathbf{R} \circ \ldots \circ \mathbf{R} \sqsubseteq \mathbf{R}$, where $\mathbf{C}$ and $\mathbf{R}$ are defined by the following grammar.

$$\mathbf{R} ::= N_R$$
$$\mathbf{C} ::= N_C \mid \mathbf{C} \sqcap \mathbf{C} \mid \exists \mathbf{R}.\mathbf{C}$$

[7] Datalog is equivalent to function-free definite logic programming [40].

**Table 1** $\mathcal{EL}^+$ Semantics

| Description | Expression | Semantics |
|---|---|---|
| Individual | $a$ | $a \in \Delta^{\mathcal{I}}$ |
| Top | $\top$ | $\Delta^{\mathcal{I}}$ |
| Bottom | $\bot$ | $\emptyset$ |
| Concept | $C$ | $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ |
| Role | $R$ | $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ |
| Conjunction | $C \sqcap D$ | $C^{\mathcal{I}} \cap D^{\mathcal{I}}$ |
| Existential Restriction | $\exists R.C$ | $\{\ a \mid$ there is $b \in \Delta^{\mathcal{I}}$ such that $(a,b) \in R^{\mathcal{I}}$ and $b \in C^{\mathcal{I}}\ \}$ |
| Concept Subsumption | $C \sqsubseteq D$ | $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ |
| Role Subsumption | $R \sqsubseteq S$ | $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$ |
| Role Chain | $R_1 \circ \cdots \circ R_n \sqsubseteq R$ | $R_1^{\mathcal{I}} \circ \cdots \circ R_n^{\mathcal{I}} \subseteq R^{\mathcal{I}}$ |

with $\circ$ signifying standard binary composition

The semantics of $\mathcal{EL}^+$ is defined by means of interpretations $I = (\Delta^I, \cdot^I)$ which map $N_I, N_C, N_R$ to elements, sets, and relations in the domain of interpretation $\Delta^I$. For an interpretation $I$ and $C(i), R(i) \in \Sigma$, the function $\cdot^I$ is shown in Table 1.

Both RDFS and $\mathcal{EL}^+$ can be expressed in first-order predicate logic.

*Logical embeddings* It is essential that symbolic expressions of any logic can be transformed into a continuous space that is amenable for subsymbolic processing. Such a transformation must map discrete logical expressions into some continuous space $\mathbf{R}^n$ of fixed dimension as an $n-$dimensional vector. The mapping from discrete data to a continuous space is often called an *embedding*. Embeddings have been studied for many types of data,[8] including pieces of text (words, sentences, paragraphs) [48,13,60,55,44], and structured objects like graphs [16], knowledge graphs represented by a set of RDF triples [15,78,49,81,75],[9] and types of logical formulas [23].

There are two key ingredients that we need to consider for logical embeddings. First, note that logical expressions are highly structured, symbolic, and discrete. Structure refers to the ordering of logical operations, the type of each symbol in the expression, arity of predicates, variable bindings, etc. In fact, it is essentially the structure that deductive reasoning operates over.[10] An embedding should thus **(1) consider structure**, in the sense that similarly structured logical statements should be located "close" to each other within the embedding. Furthermore, the actual symbols representing entities in a knowledge base (for example the string name of an entity) representing variables, constants, functions, and predicates are insubstantial for deductive reasoning in the sense that a consistent renaming across a logical theory does not change the set of entailed formulas (under the same renaming). An embedding should thus **(2) disregard all textual features of a logical statement**: an embedding based on textural features may cause a DNN to learn to reason based on common text-based patterns in logical statements, rather than by its logical semantics. This may cause a DNN to overfit to knowledge

[8] `https://github.com/thunlp/KRLPapers` has an extensive listing of existing work on knowledge embeddings.

[9] See [11,76] for a recent survey.

[10] Some deductive entailment algorithms can even be understood as simply a type of syntax rewriting systems.

specific to a single knowledge base and, more importantly, would train the DNN to make decisions in ways that are not representative of how a symbolic system processes information.

In determining adaptations of existing embeddings (or when devising new embeddings) for our tasks, we have considered the interplay of the embedding approach with the structure of statements encoded in a particular logic. A simple translation of existing embeddings may not be fruitful, as the overwhelming use case explored for knowledge graph embeddings (knowledge graphs are commonly expressed in RDF(S)) is not deductive in nature, but concerns the problem of the discovery or suggestion of new edges in a knowledge graph. In this edge discovery setting, the actual labels for nodes or edges in the graph and their meaning matter, as reflected in most embedding methods, and this is contradictory to key ingredient (2) discussed above. Generic graph embeddings [16] also appear to be insufficient since structural aspects like the importance of specific node or edge labels from the RDF(S) namespaces to the deductive reasoning process would not get sufficient attention. We further do not anticipate a "one-embedding-fits-all" phenomenon to emerge, instead we expect different embedding methods to be necessary for different logics.

In looking at embeddings that consider the structure of logic, we can turn to inspiration from past work demonstrating how simple logic operations can be simulated over vectors in a real space [30]. The approach is able to model quantifiers in a logic language and thus many of its characteristics could be generalized to other logics. Nevertheless, this representation is completely symbolic: a vector representation of a logical entity and relation is just the one-hot encoding, and so little information about similarity is preserved. For embeddings in RDF reasoning, RDF2Vec [63] is an intriguing algorithm that maps RDF entites and relationships into vector space by a virtual document that contains lexicalized graph walks. A natural language embedding algorithm is then applied to the documents based on token co-occurrences. RDF2Vec ignores language semantics, but could be used to study the distributional properties of RDF and to build pre-trained embeddings for use in DNN architectures. The fundamental method that RDF2Vec employs should be extendable to other types of logics as well. Another inspiring approach to be taken into consideration is an embedding of facts and relations into matrices and tensors [77] found by a matrix factorization derived from proof graphs. Proof graphs may be used to describe the relationships of statements encoded in any logic, and hence might be a starting point for logics such as Datalog, $\mathcal{ACL}$ and $\mathcal{SROIQ}$ [39].

To incorporate the second ingredient where textural features of a logical statement should not be considered by an embedding, we explore *normalizations* of statements before embedding in our RDF reasoning work. Normalizations that we explore have two different types. In the first case, normalization done before invoking logical reasoning algorithms will usually control the structural complexity of the formulas and theories producing entailments. Secondly, we explore *name label normalization*, by which we mean a renaming of the primitives from a logical language (variables, constants, functions, predicates) to a set of pre-defined entity names which will be used across different theories. While simple, such a normalization would not only play the role of "forgetting" irrelevant label names, but also make it possible to transfer learning from one knowledge graph to the other. Moreover, a deep network will be limited to learning only over the *structural*

information within the theories, and not on the actual names of the primitives, which would be insubstantial for an entailment task.

In our work with Logic Tensor Networks (LTNs), we focus on the Real Logic that we defined in the previous sections. We follow [69] in the definitions of LTNs. In LTNs, constants are grounded to vectors in $\mathbb{R}^n$ and predicates are grounded to neural network operations which output values in $[0, 1]$. The neural network learns to define the truth value of an atom $P(c_1, \ldots, c_n)$ as a function of the grounding of the terms $c_1, \ldots, c_n$ [69]. For a predicate of arity $m$ and for which $\mathbf{v}_1, \ldots, \mathbf{v}_m \in \mathbb{R}^n$ are the groundings of $m$ terms, the grounding $\mathcal{G}$ of the predicate $P$ is defined as

$$\mathcal{G}(P)(\mathbf{v}) = \sigma(u_P^T(tanh(\mathbf{v}^T W_P^{[1:k]}\mathbf{v} + V_P\mathbf{v} + B_P))), \tag{6}$$

where $\mathbf{v} = \langle \mathbf{v}_1, \ldots, \mathbf{v}_m \rangle$ is the concatenation of the vectors $\mathbf{v}_i$, $\sigma$ is the sigmoid function, $W$, $V$, $B$ and $u$ are all parameters learned by the network during training, and $k$ is the tensor layer size.

Learning in LTNs comes from interpreting the problem as the one of finding the weights in such a way that they better satisfy the formulas. Thus, the task is to find values for all the weights in such a way that the satisfiability of the formulas in the knowledge base is maximized. We make this more clear with an example: Suppose we have an atom like $mammal(cat)$, learning to satisfy this atom means that the network has to update the representation of the parameters in such a way that the parameters of the tensor network $mammal$, given in input the vector representation of the constant $cat$ returns a value that is as close as possible to 1. In this way, given multiple atoms, LTNs can learn the best weights to satisfy them. As long as the atoms are combined through the use of fuzzy connectives, the optimization works similarly: given $mammal(cat) \rightarrow animal(cat)$, the values of the two atoms are obtained separately and then combined using the fuzzy interpretation of $\rightarrow$. Again, this value can be maximized during the optimization.

LTNs also support learning of quantified formulas. For example, universally quantified axioms (for example, $\forall x\ mammal(x) \rightarrow animal(x)$); are computed by using an aggregation operation [69] defined over a subset $T$ of the domain space $\mathbb{R}^n$; different aggregation operations, such as min, average or harmonic average can be considered. Hence, the optimization comes from aggregating the truth value of each formula, instantiated with the values in the domain, and then the maximization of this value, as we want $\forall x\ mammal(x) \rightarrow animal(x)$ to be true for all $x \in T$.

Another important quality of LTNs is that they can be used to do after-training reasoning over combinations of axioms on which they were not trained: we can ask the truth values of queries like $\forall x\ \neg mammal(x) \rightarrow species(x)$; this is important because it allows us to explore the space of the results using logical axioms.

To train LTNs we need to define fuzzy connectives (e.g., Gödel, Lukasiewicz, etc.) for the logic, the dimension of constant embeddings, the size of the tensor layer, and the aggregation function used for the $\forall$ formulas. LTNs can be trained until they reach a certain level of satisfiability (ideally 1) or for a given number of epochs.

Finally, in the case of our $\mathcal{EL}^+$ reasoning, we use the following encoding scheme: The maximum number of role and concept names in knowledge bases are used to scale all of the integer values for names into a range of $[-1, 1]$. To enforce disjointedness of concepts and role names, we map all concepts to $(0, 1]$, all roles

**Table 2** Translation Rules

| KB statement | | Vectorization |
|---|---|---|
| $\mathrm{CX} \sqsubseteq \mathrm{CY}$ | $\rightarrow$ | $[\ 0.0,\ \frac{X}{c}, \frac{Y}{c},\ 0.0\ ]$ |
| $\mathrm{CX} \sqcap \mathrm{CY} \sqsubseteq \mathrm{CZ}$ | $\rightarrow$ | $[\ \frac{X}{c},\ \frac{Y}{c},\ \frac{Z}{c},\ 0.0\ ]$ |
| $\mathrm{CX} \sqsubseteq \exists \mathrm{RY.CZ}$ | $\rightarrow$ | $[0.0,\ \frac{X}{c},\ \frac{-Y}{r},\ \frac{Z}{c}]$ |
| $\exists \mathrm{RX.CY} \sqsubseteq \mathrm{CZ}$ | $\rightarrow$ | $[\frac{-X}{r},\ \frac{Y}{c},\ \frac{Z}{c},\ 0.0]$ |
| $\mathrm{RX} \sqsubseteq \mathrm{RY}$ | $\rightarrow$ | $[0.0, \frac{-X}{r}, \frac{-Y}{r}, 0.0]$ |
| $\mathrm{RX} \circ \mathrm{RY} \sqsubseteq \mathrm{RZ}$ | $\rightarrow$ | $[\frac{-X}{r}, \frac{-Y}{r}, \frac{-Z}{r}, 0.0]$ |

c = Number of Possible Concept Names
r = Number of Possible Role Names

to $[-1, 0)$. Each of the six possible normalized axiom forms is encoded into a 4-tuple based on the logical encodings defined in Table 2. Tuples are concatenated end-to-end for each axiom in a knowledge base or reasoning step then duplicated across a new dimension to match the desired tensor shape where ragged tensors are padded with zeros. We refer to this as an *encoding* rather than embedding because, except for noise in the conversion back and forth from integer to floating point number, it produces a faithful logical encoding of arbitrary integer names without needing to embed structure. The correct structure of normalized $\mathcal{EL}^+$ expressions is known, not inferred, so the system enforces an approximate representation of this without the added assistance of moving similar predicate names closer together in an embedding.

### 3.1 Investigative dimension 3: DNN architectures

The design space of DNN architectures is vast. Rather than taking a "walking in the dark" strategy where we consider arbitrary constructions of multi-layered perceptrons, convolutional architectures, recurrent architectures, and architectures with combinations thereof, we will focus our exploration on: (i) models that can *recall* previously consumed information; and (ii) variants of models already shown in the literature to achieve some level of subsymbolic processing. In the former case, we are motivated by the basic idea that deducing new facts from existing ones requires consideration of the entire set of facts consumed thus far. In the latter case, we seek to build on the shoulders of past researchers who have also tried to bridge the neuro-symbolic divide, but we of course have no expectation that a simple port or copy of such reported DNN architectures will be completely (or even barely) capable of deductive reasoning. This is especially true because our investigations utilize multiple types of logic and logical embeddings.

**Models with recollection.** A common type of DNN architecture able to recall previous consumed information is memory networks [79]. Memory networks denote a family of models consisting of 'memory cells', which are defined essentially as embeddings over the set of training data. Multiple memory cells can be chained together to model multiple "memory lookups", where the embedding of a subsequent

cell in a chain can be thought of as representing a view of the memory during a lookup conditioned on the previous lookup. The idea of memory lookups naturally extends to a deductive reasoning task: whether or not a hypothesis is entailed by facts in a knowledge base is logically determined by deducing if the hypothesis is a consequence of all known facts and their entailments. This suggests a strategy of multiple 'lookups' of known facts conditioned on previous facts that have been evaluated thus far. We have selected memory networks since we believe that they are a good candidate for performing deductive logical entailment. Their sequential nature corresponds, conceptually, to the sequential process underlying some deductive reasoning algorithms. The attention modeling corresponds to pulling only relevant information (logical axioms) necessary for the next reasoning step. And their success in natural language inferencing is also promising: while natural language inferencing does not follow a formal logical semantics, logical deductive entailment is nevertheless akin to some aspects of natural language reasoning. Besides, as attention can be traced over the run of a memory network, we will furthermore glean insights into the "reasoning" underlying the network output, as we will be able to see which pieces of the memory (for example, the input knowledge graph) are taken into account at each step.

Some limitations of memory networks need to be overcome to make them applicable for deductive reasoning. The most crucial limitation will be how most memory networks rely on some word-level embedding with a fixed size lookup table over a vocabulary to represent memory cells. They are thus known to have difficulties dealing with out-of-vocabulary terms as a word lookup table cannot provide a representation for the unseen, and thus cannot be applied to natural language inference over new sets of words [6], and for us this will pose a challenge in the transfer to new knowledge bases. Additionally, learning good representations for rare words is challenging as these models require a high frequency of each word to generalize well. One option may be to pursue variants of the copy mechanism and pointer networks [27,61] to refer to the unknown words in the memory in generating the responses. Another option is utilizing character-level embeddings [50] to compose the representation of characters into words. Despite the success of these mentioned methods in handling few unknown words absent during training, transferability and the ability of these models to generalize to a completely new set of vocabulary is still an open research question. Similarly, using character-level embeddings may prove to be an inelegant solution in our case, since one of our hypothesized key ingredients of an embedding is independence of the strings used to represent logical statements.

**Building on proven models.** Besides exploring the design space of memory networks, we have also identified models from the literature that have shown some ability to perform symbolic tasks. This includes Logic Tensor Networks (LTN) [67, 68,23] which are based on Tensor Networks [72]. In the LTN setting, first-order fuzzy predicate logic primitives are embedded as tensors (an $n$-dimensional array of reals), and complex predicates and formulas are built by applying tensors as functions of other tensors. LTNs have been shown to handle deductive reasoning, but only under small toy examples and simple inferences [67,68]. The scalability of LTNs has not been addressed other than in qualitative arguments which would need quantitative evaluation. Our work has explored LTNs in greater detail re-

garding their performance, scalability, and reasoning ability over different logic types.

**Reasoning Structure Emulation** In a logic-based system there is transparency at any stage in the reasoning. We cannot, of course, expect this in most neural networks. With a network that aims to emulate reasoner behavior rather than output, however, we can impose a degree of intermediate structure. This intermediate structure allows us to inspect a network part-way through and perform a sort of "debugging" since we know exactly what it should have learned at that point. This is a crucial break from current thinking that advocates more and deeper opaque hidden layers in networks that improve accuracy but detract from explainability. Inspection of intermediate answers could indicate whether a proposed architecture is actually learning what we intend, which should aid development of more correct systems.

We will look at this for the simple logic $\mathcal{EL}^+$, reason over knowledge bases in that logic, and then extract supports from the reasoning steps, mapping the reasoner supports back to sets of the original knowledge base axioms. The support thus consists of the set of original axioms from which a reasoning step conclusion can be drawn. This allows us to encode the input data in terms of only knowledge base statements. It also provides an intermediate answer that might improve results when provided to the system. This logic data is fed into three different LSTM architectures with identical input and output dimensionalities. One architecture, which we call "Deep", does not train with support data but has a hidden layer the same size as the supports we have defined. Another architecture, called "Piecewise", trains two separate half-systems, the first takes knowledge bases and learns supports, and the second takes correct supports provided by the reasoner and learns reasoner answers. The last system, called "Flat", simply trains to map knowledge base inputs directly to reasoner outputs for each reasoning step.

## 4 Summary of our Experimental Settings & Evaluations

### 4.1 RDFS Reasoning with Memory Networks

#### *4.1.1 Problem Setting*

We begin with reasoning for the simplest logic under consideration: RDFS. A plethora of embeddings for RDFS have been proposed [15,78,49,81,75], but we were not aware of an embedding that has the two key ingredients (considering logical structure and ignoring entity strings) we consider necessary for the deductive reasoning task. We thus first consider a hand-coded embedding where all RDFS URIs not in the RDF or RDFS namespaces are mapped to a random integer from a pre-defined set $\{a_1, \ldots, a_n\}$, were $n$ is the upper bound of the size of any knowledge base to be considered. URIs in the RDF/RDFS namespace are not renamed as the 'types' of entities and relationships are important to the deductive reasoning process using the RDFS deduction rules as exemplified in Figure 2. This normalization not only plays the role of "forgetting" irrelevant label names, but also makes it possible to transfer learning from one knowledge graph to the other.

After normalization, we must map a knowledge graph of RDFS triples to a numerical multi-dimensional tensor. This could be done in several ways. Here, we

map each normalized URI to a vector in $R^d$, where $d$ is the embedding size. Then we can map each element in the RDF triple to a corresponding ($d$-dimensional vector, and finally the full knowledge graph into a $(d \times k)$-tensor, where $k$ is the number of triples in the knowledge graph. We use an end-to-end memory network architecture (MemN2N) [73] where the network learns memory cell embeddings at the same time as the rest of the model weights, including attention mechanisms.

For this and for other approaches, we conjecture that reasoning depth acquired by the network will correspond to both: (i) the number of layers of the DNN model; and (ii) the ratio of deep versus shallow reasoning required to perform the deductive reasoning. This is because forward-chaining reasoners (which are standard for RDF(S), $\mathcal{EL}^+$, and Datalog) iteratively apply inference rules in order to derive new entailed facts. In subsequent iterations, the previously derived facts need to be taken into account. The number of sequential applications of the inference rules required to obtain a given logical consequence can be understood as a measure of the "depth" of the deductive entailment. Typically, one expects the number of entailed facts over the number of inference rule applications to follow a long-tail distribution, which means that in training data, "deep" entailments would be underrepresented, and this may cause a network to not actually acquire deep inference skills. Thus, we have conducted experiments with different training sets, possibly overrepresenting "deep" entailments, to counter this problem. Furthermore, a naive expectation on the trained network would be that each layer performs something equivalent to an inference rule application. If so, then the number of layers would limit the entailment depth the network could acquire, but we have yet to assess this assumption experimentally.

In terms of scalability, we have put a global limit on the size of knowledge graphs a trained system will be able to handle, as required training time can be expected to grow super-linearly in the size of the knowledge graphs. A practical solution to this problem may be to use a clustering or path ranking algorithm [47] that filters away irrelevant triplets or extracts sets of all paths that connect query-entity pairs. This way we may be able to decrease the memory size substantially and attempt to train on knowledge graphs with tens of thousands of triples. The contribution of our work, however, is on the fundamental capabilities of deep learning approaches to perform deductive reasoning, so we do not yet report on scalability aspects.

#### *4.1.2 Evaluations*

We now present and discuss our evaluation and results. We obtain training and test data from Linked Data Cloud website[11] and LOD laundromat[12]. The candidate entailments are composed of true entailments inferred by Jena[13] and false entailments generated by replacing a random element of a present or entailed triple with another random element of the same rdf:type. The specifics of the datasets, memory network architecture and training hyper-parameters are detailed in [26]. Our evaluation metrics are average of precision and recall and f-score for all the KGs in the test dataset, obtained for both inferred and non-inferred sets of triples.

[11] https://lod-cloud.net/
[12] http://lodlaundromat.org/
[13] `https://jena.apache.org`

| Training Dataset | Test Dataset | Valid Triples Class | | | Invalid Triples Class | | | Accuracy |
|---|---|---|---|---|---|---|---|---|
| | | Precision | Recall /Sensitivity | F-measure | Precision | Recall /Specificity | F-measure | |
| OWL-Centric Dataset | Linked Data | 93 | 98 | 96 | 98 | 93 | 95 | **96** |
| OWL-Centric Dataset (90%) | OWL-Centric Dataset (10%) | 88 | 91 | 89 | 90 | 88 | 89 | **90** |
| OWL-Centric Dataset | OWL-Centric Test Set [b] | 79 | 62 | 68 | 70 | 84 | 76 | **69** |
| OWL-Centric Dataset | Synthetic Data | 65 | 49 | 40 | 52 | 54 | 42 | **52** |
| OWL-Centric Dataset | Linked Data [a] | 54 | 98 | 70 | 91 | 16 | 27 | 86 |
| OWL-Centric Dataset [a] | Linked Data [a] | 62 | 72 | 67 | 67 | 56 | 61 | 91 |
| OWL-Centric Dataset(90%) [a] | OWL-Centric Dataset(10%) [a] | 79 | 72 | 75 | 74 | 81 | 77 | 80 |
| OWL-Centric Dataset | OWL-Centric Test Set [ab] | 58 | 68 | 62 | 62 | 50 | 54 | 58 |
| OWL-Centric Dataset [a] | OWL-Centric Test Set [ab] | 77 | 57 | 65 | 66 | 82 | 73 | 73 |
| OWL-Centric Dataset | Synthetic Data [a] | 70 | 51 | 40 | 47 | 52 | 38 | 51 |
| OWL-Centric Dataset [a] | Synthetic Data [a] | 67 | 23 | 25 | 52 | 80 | 62 | 50 |
| **Baseline** | | | | | | | | |
| OWL-Centric Dataset | Linked Data | 73 | 98 | 83 | 94 | 46 | 61 | 43 |
| OWL-Centric Dataset (90%) | OWL-Centric Dataset (10%) | 84 | 83 | 84 | 84 | 84 | 84 | 82 |
| OWL-Centric Dataset | OWL-Centric Test Set [b] | 62 | 84 | 70 | 80 | 40 | 48 | 61 |
| OWL-Centric Dataset | Synthetic Data | 35 | 41 | 32 | 48 | 55 | 45 | 48 |

[a] More Tricky Nos & Balanced Dataset [b] Completely Different Domain.

**Table 3** Experimental results of proposed model

We also report the recall for the class of negatives (specificity) by calculating the number of true negatives. Besides, we have done zero-padding to the batches of 100 queries. Thus we need to introduce another class label for zero paddings in the training and test sets. We have not considered the zero-padding class in the calculation of precision, recall and f-measure. Through our evaluations, however, we have observed some missclassifications from/to this class. Thus, we report accuracy as well to show the impact of any such mistakes.

To the best of our knowledge there is no architecture capable of performing deductive reasoning over unseen RDFS KGs. Hence, we have used a non-normalized embedding version of our memory network as a baseline. Our technique outperforms the baseline as depicted in Table 3.

A further, even more prominent advantage of utilizing our normalization model is its training time duration. Indeed, this huge time complexity difference is because of the notable size difference of embedding matrices in the original and normalized cases. For example, the size of embedding matrices for the normalized OWL-Centric dataset is $3,033 \times 20$ compared to $811,261 \times 20$ for the non-normalized one (and $1,974,062 \times 20$ for Linked Data which is prohibitively big). This causes a significant decrease in training time and space complexity and hence has helped improve the scalability of our memory networks. In the OWL-Centric dataset, for instance, the space required for saving the normalized model is 80 times less than the intact model ($\approx 4G$ after compression). Nevertheless, the normalized model is almost 40 times faster to train than the non-normalized one for this dataset. Our normalized model trained for just a day on OWL-Centric data but achieves better accuracy, whereas it trained on the same non-normalized dataset more than a week on a 12-core machine.

To further investigate the performance of our approach on different datasets, we have run our approach on multiple datasets with various characteristics. The performance across all variations are reported in Table 3. As the table shows, beside our strikingly good performance compared to the baseline, there are a number of other interesting findings: Our model shows even better performance on the Linked Data task while it has trained on the OWL-Centric dataset. We believe that this may be because of a generally simpler structure of Linked Data, but validating this will need further investigation. The majority of our few false negative instances relates to the inability of our approach to learn reflexivity, that is, to infer that any class is a subclass of itself.

| Training Dataset | Test Dataset | Valid Triples Class | | | Invalid Triples Class | | | Accuracy |
|---|---|---|---|---|---|---|---|---|
| | | Precision | Recall | F-measure | Precision | Recall | F-measure | |
| OWL-Centric Dataset | Linked Data | 94 | 97 | 95 | 97 | 93 | 95 | 28 |
| OWL-Centric Dataset (90%) | OWL-Centric Dataset (10%) | 85 | 92 | 88 | 92 | 83 | 87 | 76 |
| OWL-Centric Dataset | OWL-Centric Test Set [a] | 73 | 80 | 75 | 80 | 67 | 71 | 61 |
| OWL-Centric Dataset | Synthetic Data | 52 | 43 | 46 | 51 | 60 | 54 | 51 |

[a] Completely Different Domain.

**Table 4** Ablation Study: No Positional Encoding

| Dataset | Hop 1 | | Hop 2 | | Hop 3 | | Hop 4 | | Hop 5 | | Hop 6 | | Hop 7 | | Hop 8 | | Hop 9 | | Hop 10 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | F% | D% | F% | D% | F% | D% | F% | D% | F% | D% | F% | D% | F% | D% | F% | D% | F% | D% | F% | D% |
| **OWL-Centric**[a] | - | **8** | - | **67** | - | **24** | - | **1** | - | - | - | - | - | - | - | - | - | - | - | - |
| OWL-Centric[b] | 42 | 5 | 78 | 64 | 44 | 30 | 6 | 1 | - | - | - | - | - | - | - | - | - | - | - | - |
| Linked Data[c] | 88 | 31 | 93 | 50 | 86 | 19 | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Linked Data [d] | 86 | 34 | 93 | 46 | 88 | 20 | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Synthetic | 38 | 0.03 | 44 | 1.42 | 32 | 1 | 33 | 1.56 | 33 | 3.09 | 33 | 6.03 | 33 | 11.46 | 31 | 20.48 | 31 | 31.25 | 28 | 23.65% |

[a] ***Training set*** [b] Completely different domain [c] LemonUby Ontology [d] Agrovoc Ontology

**Table 5** F-measure and Data Distribution over each reasoning hop

Our algorithm shows poor performance when it has trained on the OWL-Centric dataset and tested on the tricky Linked Data. In that case our model has classified a majority of the triples to the "yes" class and this has caused the low specificity (recall for "no" class) of 16%. This seems inevitable since the negative triples are very similar to the positives ones, making differentiation more complicated. However, training the model on the tricky OWL-Centric dataset has improved that by a large margin (more than three times).

For our particularly challenging synthetic data, performance is not as good, and this may be the result of the unique nature of this dataset that includes much longer reasoning chains compared to non-synthetic data. We have trained our model only on real-world datasets; it may be interesting to investigate the results of training on synthetic data, but that was out of scope of our work.

It appears natural to analyze the reasoning depth acquired by our network. We hypothesize that the reasoning depth acquired by the network will correlate with both the network depth, and the ratio of deep versus shallow steps required to conduct the deductive reasoning. Forward-chaining reasoners iteratively apply inference rules in order to derive new entailed facts. In subsequent iterations, the previously derived facts need to be taken into consideration. To gain an intuition of what our model has learned in this respect, we have emulated this symbolic reasoner behavior in creating our test set. We first started from our input KG $K_0$ in hop 0. We then produced, subsequently, KGs of $K_1$,..., $K_n$ until no new triples are added (i.e. $K_{n+1}$ is empty) by applying the RDFS inference rules from the specification: The hop 0 dataset contains the original KG's triples in the inferred axioms, hop 1 contains the RDFS axiomatic triples. The real inference steps start with $K_n$ where $n >= 2$. Table 5 summarizes our results in this setup.

Unsurprisingly, the result for synthetic data is poor. This may be because of the huge gap between the distribution of our training data over reasoning hops and the synthetic data reasoning hop length distribution as shown in the first row of Table 5. From that, one can see how the distribution of our training set affects the learning capability of our model. Apart from our observations, previous studies [20,65,82] also acknowledged that the reasoning chain length in real-world KGs is limited to 3 or 4. Hence, a synthetic training toy set would have to be built as part of follow-up work, to further analyze the reasoning depth issue.

Furthermore, a naive expectation would be that each network layer would perform processing equivalent to an inference rule application. If this is the case, then the number of layers would limit the entailment depth the network could

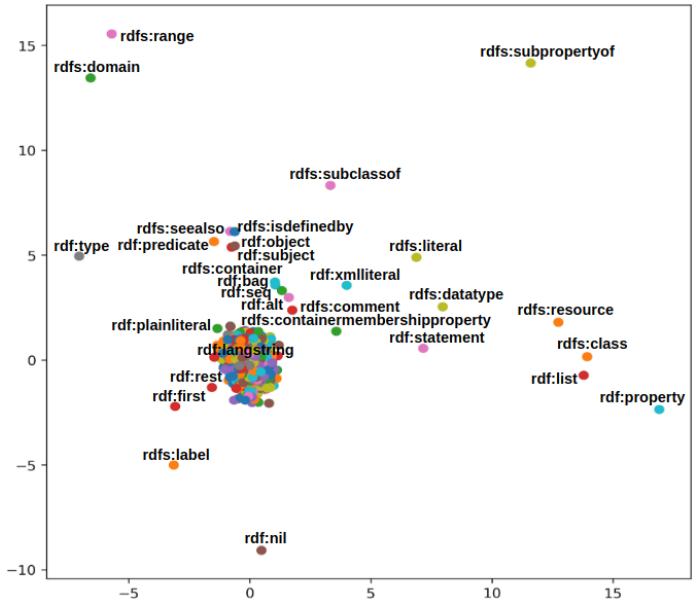


**Fig. 3** PCA projection of embeddings for the vocabulary

acquire. We assessed this assumption experimentally. For this purpose, we have done 10 experiments (K=1 to 10) to assess the effect of changing the number of computational hops on our results, over the OWL-Centric Dataset. Interestingly, our results suggest that our model is able to get almost the same performance with K=1, and furthermore the F-measure remains almost constant when increasing K stepwise from 1 to 10. This shows us that multi-hop reasoning can already be done using one-hop attention of memory networks over our training set, and that increasing the of number of hops does not hurt performance. This suggests that each attention hop of our network is able to conduct more than one naive deductive reasoning step. At the same time, this also demonstrates robustness of our method against change of its structure.

*General Embeddings Visualization* The PCA projections of the embedding learned in the first memory cell of the network are shown in Figure 3. The PCA projection reveals how the network learns to differentiate RDF/RDFS namespace relations from the random strings assigned to entity and relational names, and that it learns meaningful similarities between RDF/RDFS relations expected when performing deductive reasoning.

*Ablation Study* We further performed an ablation study where we remove positional encoding from embeddings and compare the results to assess their impact. The idea behind positional encoding is keeping order of elements in the triples into account. Here instead we are using bag of words and do not take ordering of elements in each triple into account. The results have been listed in Table 4. As anticipated, removing the positional encoding results in a performance decrease for all of our experiments in terms of accuracy. Indeed, through more detailed analysis

of the result for our first model, we found that it classifies all zero-paddings to the negative class. That is the explanation for the huge gap of accuracy and f-measure in that model. Nevertheless, removing positional encoding does not decrease the performance for some of our experiments substantially. Indeed, this is not practically surprising in light of the fact that orderless representations have always shown tremendous success in the NLP domain even when order matters.

## 4.2 Deductive Reasoning Capabilities of Logic Tensor Networks

### *4.2.1 Problem Setting*

Logic Tensor Networks (LTNs) are meant to provide a logical framework grounded in a vector space. However, learning in machine learning happens by reducing errors, and the logic learned in the space might not be as perfect and consistent as expected. Nevertheless, being able to logically query the vector space is an important asset of LTNs that makes the model very useful under an interpretability point of view.

In the next section we are going to evaluate LTNs on two different tasks: the first one, **deductive reasoning** is meant to show how effective LTNs are as a deductive reasoning tool. We will train LTNs and use them to infer facts about an unseen predicate, checking how well LTNs can learn using rules. In the second task, **reasoning with pre-trained entity embeddings** we will show that that it is possible to combine LTNs with pre-trained entity embeddings [10] to account for both logical reasoning and a more general similarity based reasoning. Eventually, we are going to show a few more details about LTN scalability.

## 4.3 Evaluations

**Deductive Reasoning.** The first task we want to test for LTNs is deductive reasoning: given some data in the form of instantiated axioms and a set of rules, how well can LTNs combine these two to infer new knowledge? The experiment we describe here has been obtained after searching for the best parameters, as described in [9]. The results we show use the harmonic mean as aggregator, 10 dimensional embeddings and 10 neural tensor layer. All the experiments described in this section can be replicated using code, data and parameters we release and describe in the online repository.[14]

The setting we are going to consider is described in Figure 4, where we show nodes that represent people and edges that represent parental relationships (for example, $P$ is parent of $S$ and thus $parent(P, S)$).

Our objective in this context is another predicate, *ancestor*. What we want to test is the ability of LTNs in generalizing towards this new predicate, without having access to data that describe it: can LTNs, from the set of all *parent* instantiated atoms and with the aid of some transitivity axioms, deduce the ancestor relations? For example, can LTNs, after training, correctly deduce $ancestor(C, S)$ and $ancestor(D, N)$? We will refer to this set of *ancestor* instantiated axioms as

[14] `https://github.com/vinid/ltns-experiments`

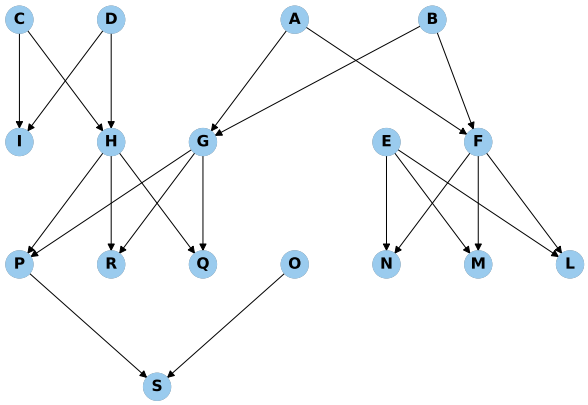


**Fig. 4** Representation of the parental relationships in the P dataset

$KB^a$. We also test how the model performs over the set of ancestor formulas that require multi-hop inferences (that is, those that cannot be directly inferred from $\forall a, b \in P : parent(a, b) \rightarrow ancestor(a, b)$), and those ancestors pairs for which the parent pair is false (for example, $ancestor(C, S)$). We will refer to this set of axioms as $KB^a_{filtered}$. This operation is done to check if LTNs are able to pass information from the *parent* predicate to the *ancestor* predicate and whether this is enough to give to the network the ability to make even more complex inferences that are related to chains of ancestors.

The representation for the *ancestor* predicate should be generated from knowledge in the axioms, since no data about it is provided.

LTNs will thus be trained on all *parent* atoms and to the following universally quantified rules. These rules are enough to complete the knowledge base in first-order logic, and we want to see if we can do the same with LTNs.

- $\forall a, b \in P : parent(a, b) \rightarrow ancestor(a, b)$
- $\forall a, b, c \in P : (ancestor(a, b) \wedge parent(b, c)) \rightarrow ancestor(a, c)$
- $\forall a \in P : \neg parent(a, a)$
- $\forall a \in P : \neg ancestor(a, a)$
- $\forall a, b \in P : parent(a, b) \rightarrow \neg parent(b, a)$
- $\forall a, b \in P : ancestor(a, b) \rightarrow \neg ancestor(b, a)$

After training, LTNs on $KB^a$ have an F1 score of 0.77. However, if we only consider $KB^a_{filtered}$, the model correctly infers 22 ancestors while generating 25 false positives, thus generating an F1 that is equal to 0.62. The network seems to be able to fit the data well, but multi-hop inferences are still difficult to predict.

To provide a better understanding of this experiment we decided to add two novel axioms to the previous set of axioms. The two axioms we add explicitly describe the relationships between the two predicates:

- $\forall a, b, c \in P : (ancestor(a, b) \wedge ancestor(b, c)) \rightarrow ancestor(a, c)$
- $\forall a, b, c \in P : (parent(a, b) \wedge parent(b, c)) \rightarrow ancestor(a, c)$

**Table 6** Ancestor completion task with different number of axioms. Value out of the parentheses are computed over the complete ancestor knowledge base, $KB^a$, while those within the parenthesis are computed only on those axiom that require require multi-hop inferences, $KB^a_{filtered}$.

| Type | F1 | Precision | Recall |
|---|---|---|---|
| Six Rules | 0.77 (0.62) | 0.64 (0.47) | **0.96** (0.92) |
| Eight Rules | **0.85** (0.72) | **0.80** (0.66) | 0.89 (0.79) |

Table 6 shows what happens when we extend the previous set of rules (*Six Rules*) with the two novel rules (*Eight Rules*), tested again on the ancestor dataset. We use F1 measure, precision and recall as evaluation metrics. Results show that the added rules are beneficial to the learning process. This result offers an interesting point of view: adding more rules that under a logical point of view are redundant (they can be inferred from the six rules), is helpful to a model that is trained in a machine learning context. This is because the model will see the examples more times, given that there are now more axioms.

**Reasoning with Pre-trained Entity Embeddings.** When we use logics it is difficult to encode knowledge represented by a more general, commonsense understanding of the world. For example, realizing that *cats* are similar to *tigers* might help in inferring something about *cats*. This fact allows for extended reasoning: if we know that *tigers* are *mammals*, then even though we know nothing about *cats* we can use its similarity to infer that also *cats* are mammals. This similarity can be captured by pre-trained embeddings [55] commonly used to give a vector representation to each word; in these embeddings words that occur in similar contexts will have similar vectors.

However, we are interested in the embedding of logical constants. Here, we make use of Knowledge Graph Distributional Entity Embeddings described in [10], where Entity Embeddings (EEs) are generated from entity-to-entity co-occurrence in text that has been annotated with an entity linker. Since both *dbr:tiger*[15] and *dbr:cat* appear in similar contexts, they will have similar vectors. These embeddings are 100-dimensional and the entities come from DBpedia.[16]

The question that is left to answer is: "how do we combine these embeddings with LTNs?" LTNs treat constants as vectors and thus it is possible to use pre-trained embeddings in place of those vectors. If we freeze these representations, we can use LTNs in a zero-shot fashion: at test time, novel entities for which we have an embedding can be used to test the system, even though they were never seen in training. This makes LTNs more general and more interesting to use. All the experiments described in this section can be replicated using code, data and parameters we release and describe in the online repository.[17]

We follow this procedure to generate our reference knowledge base to test the combination between LTNs and entity embeddings: we query the DBpedia SPARQL endpoint for entities of the following classes: Mammal (0.38%), Fungus (0.17%), Bacteria (0.03%), Plant (0.42%). Note that some classes are much more frequent than others. We generate the transitive closure with respect to the pred-

[15] We will use the prefix dbr: to refer to DBpedia entities.

[16] `https://wiki.dbpedia.org/`

[17] `https://github.com/vinid/logical_commonsense`

icates Animal, Eukaryote, and Species, collecting the class memberships for each entity (for example, $\neg mammal$(dbr:cat), $eukaryote$(dbr:cat), $species$(dbr:cat) . Indeed, we also generate all the negative instantiated atoms like, $\neg fungus$(dbr:cat). Considering positive and negative instantiated atoms, the knowledge base used in these experiments contains 35,133 elements.

We now describe the three settings that we have defined to evaluate our proposal. The idea behind these three settings is to show three different aspects of how inference can be supported by the combination of LTNs and embedded representations.

- **S1**. In training, we have 1,400 positive atoms. In the test phase we ask the models to find all the other 7,077 atoms that are exclusively related to the entities found in the 1,400 atoms. Models have to infer something about the instance "dbr:cat" even if the only atom that was present in the training set was $Species(dbr : cat)$. **Objective:** evaluate the performance of the algorithms in a task in which the models have only access to positive atoms and not negative ones.
- **S2**. In training, we have 7,026 atoms both positive and negatives. In the test phase we ask the models to find all the other 20,890 atoms (positive and negative). **Objective:** evaluate the performance in a task in which the models can access to both positive and negative atoms. Also, note that each entity in the test set appears also in the training set.
- **S3**. In training, we have 1,756 atoms. The models are now asked to infer the value of the rest of the entire dataset 33,377 atoms (positive and negative). **Objective:** evaluate the performance in a task in which both positive and negative atoms are given, but the test set will also contain atoms of entities that were not present in the training set. The models will need to rely on the pre-trained embeddings to infer the values of predicates with respect to unseen entities.

To support logic models, we define 22 universally quantified rules that are used to support inference, here is a sample of the rules we give to the model.

$$\begin{aligned}
\forall x(plant(x) &\rightarrow eukaryote(x)) \\
\forall x(mammal(x) &\rightarrow animal(x)) \\
\forall x : (mammal(x) &\rightarrow Animal(x)) \\
\forall x(plant(x) &\rightarrow \neg mammal(x)) \\
\forall x : (mammal(x) &\rightarrow \neg plant(x)) \\
\forall x : (mammal(x) &\rightarrow \neg fungus(x)) \\
\forall x : (mammal(x) &\rightarrow \neg bacteria(x)) \\
\forall x : (bacteria(x) &\rightarrow \neg Animal(x) \\
\forall x(fungus(x) &\rightarrow \neg animal(x))
\end{aligned}$$

We are going to refer to our proposed approach, the LTNs model initialized with entity embeddings, as $LTN_{EE}$.

**Baseline.** We consider the following three algorithms as alternative methods on the setting we have defined:

- Simple LTN architecture not initialized with pre-trained embeddings.
- Probabilistic Soft Logic [2] will be trained on both atoms and universally quantified formulas. We use the tool provided by the authors with default parameters.[18]
- A simple deep neural network (DNN) that takes as input the entity embeddings and it is trained to assign 0 or 1 to instantiated atoms, we explored several architectures often obtaining similar results. The DNN separately embeds the pre-trained representations of entities and a one-hot representation of predicates in 20 dimensions, concatenates them and applies another transformation to 1 dimension plus a sigmoid function to predict a binary score. We use 20% of the input dataset for validation and it is used to early stop the training with a patience of 10. The DNN is not able to use the domain theory and will need to rely on the data.

**Table 7** $F1$ score per tested class.

| **S1** | $A_{F1}$ | $F_{F1}$ | $M_{F1}$ | $P_{F1}$ | $B_{F1}$ | $E_{F1}$ | $S_{F1}$ |
|---|---|---|---|---|---|---|---|
| $LTN_{EE}$ | **0.81** | **0.74** | **0.84** | **0.66** | **0.52** | **0.97** | **1.00** |
| $LTN$ | 0.40 | 0.14 | 0.12 | 0.10 | 0.03 | 0.93 | **1.00** |
| $PSL$ | 0.54 | 0.19 | 0.15 | 0.14 | 0.07 | 0.93 | **1.00** |
| **S2** | $A_{F1}$ | $F_{F1}$ | $M_{F1}$ | $P_{F1}$ | $B_{F1}$ | $E_{F1}$ | $S_{F1}$ |
| $LTN_{EE}$ | 0.91 | **0.86** | 0.91 | 0.86 | **0.63** | **0.99** | **1.00** |
| $DNN$ | **0.93** | 0.82 | **0.93** | **0.87** | 0.54 | **0.99** | 1.00 |
| $PSL$ | 0.56 | 0.20 | 0.20 | 0.17 | 0.10 | 0.88 | 0.98 |
| **S3** | $A_{F1}$ | $F_{F1}$ | $M_{F1}$ | $P_{F1}$ | $B_{F1}$ | $E_{F1}$ | $S_{F1}$ |
| $LTN_{EE}$ | **0.88** | **0.80** | **0.89** | **0.82** | **0.60** | **0.99** | **1.00** |
| $DNN$ | 0.87 | 0.64 | 0.85 | 0.77 | 0.47 | 0.98 | 1.00 |

In Table 7 we report the results of the various models using the F1 measure. To give a better understanding of the results, the F1 is reported on a predicate level.

**Experiments on S1** In this setting we compare $LTN_{EE}$ with $LTN$ and $PSL$.[19] The $LTN_{EE}$ approach is the best performing one. Interestingly, while $PSL$ seems to have better results than $LTN$, their difference is not huge. Also note that as simple rule-based baseline model that can use the 22 axioms we previously defined would have been able to infer only 45% of the atoms correctly (with a 100% precision)

**Experiments on S2** In this setting, the models are trained on both positive and negative atoms and the test set contains atoms of seen entities. $PSL$ performance is close to the one seen in the setting S1, and it is not at the level of the other two models used. The performances of $LTN_{EE}$ and $DNN$ are comparable. Thus, in this settings the domain theory does not seem to provide increases in performance. However, with LTNs we can now logically query the model, showing better explainablity.

[18] `https://psl.linqs.org/`

[19] DNNs cannot be used because the training consists of just positive instantiated atoms, the network would eventually just learn to output 1 for every input.

**Experiments on S3** In this setting, the models are trained on both positive and negative atoms, however the test set might contain atoms of entities that have never been seen in the test set. The only element that can be used for inference is the pre-trained embedding representation. $LTN_{EE}$ generalizes slightly better than the DNN. It is interesting to see that the $LTN_{EE}$ model shows better performances for the classes Fungus and Bacteria, even if the general F1 is lower than the one shown in the previous setting.

**After-training Inference in LTNs** One last experiment we ran was meant to show a different perspective on this kind of exploration, including axioms that are more relevant. Table 8 reports some examples. In the first part of the table we show results related to the task of the previous section. It is possible to see that the model is able to effectively learn that some species are not overlapping.

**Table 8** The truth values of novel axioms.

| Axiom | Truth |
|---|---|
| $\forall x(species(x) \rightarrow animal(x))$ | 0 |
| $\forall x(eukaryote(x) \rightarrow \neg bacteria(x))$ | 0.73 |
| $\exists x(eukaryote(x) \wedge \neg plant(x))$ | 1 |
| $\forall x, y, z(nationality(x, y) \wedge locatedIn(y, z) \rightarrow bornIn(x, z))$ | 0.33 |
| $\exists x(nationality(x, Canada) \wedge bornIn(x, Montreal))$ | 1 |
| $\forall x(bornIn(x, New\ York) \rightarrow nationality(x, United\ States))$ | 0.88 |

We additionally extended this experiment by considering KG triples from DBpedia of the following types: *nationality(Person, Country)*, *bornIn(Person, City)* and *locatedIn(City, Country)*. In total, we collected 200 training examples and we defined some simple axioms like

$$\forall x, \forall y, \forall z(bornIn(x, y) \wedge locatedIn(y, z) \rightarrow nationality(x, z))$$

to be used during training (the ones shown in the Table are not present in this set). Even with a few training samples, it is interesting to look at the results: LTNs can learn to reason on non-trivial properties of the data. For example, if you are born in New York, you are American. Though this small experiment is constrained by current implementation limitations of LTNs [8], it also shows the promising quality of this model. These results show that the combination of subsymbolic pre-trained information, entity embeddings, and logical reasoning capabilities provided by LTNs are an effective way to solve completion tasks even in contexts in which there is missing information.

## 4.4 Scalability

This last experiment is meant to show how long training with different combinations of predicates and arguments can take with LTNs. To do this, we generate different universally quantified rules, with a variable number of arguments and with different predicate arity and test how long LTNs needs to do do the training epochs. In detail, we consider $n$ predicates with arity that goes from 1 to 3:

– $\forall x : pred_n(x)$
– $\forall x, y : pred_n(x, y)$
– $\forall x, y, z : pred_n(x, y, z)$

Also, we introduce $k$ different constants that will be the domain of the $\forall pred_n()$. In our setting $k$ and $n$ will take the following values [4, 8, 12, 20, 30]. This means that in the setting with $k = 4$ constants and $n = 8$, for predicates of arity 3 we introduce 4 constants $(a, b, c, d)$ in the model and 8 predicates $(pred_1, pred_2, \ldots, pred_8)$ and each predicate is universally quantified (e.g., $\forall x, y, z : pred_1(x, y, z)$). We run the model for 5000 epochs with an embedding size equal to 10.

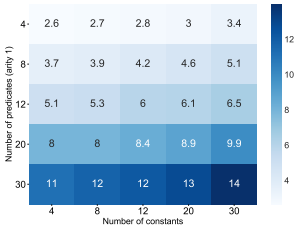


**Fig. 5** Computational times in seconds for predicates of arity one.

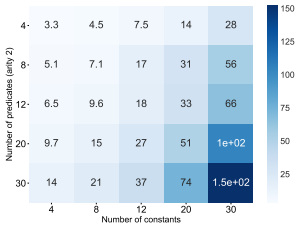


**Fig. 6** Computational times in seconds for predicates of arity two.

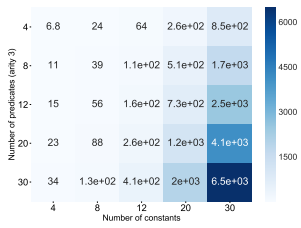


**Fig. 7** Computational times in seconds for predicates of arity three.

Figures 5, 6, 7 show the seconds needed to complete the learning phase for each setting. These results clearly show that what impacts the most in the models is the arity of the predicate: this requires the model to create multiple combinations of the inputs to pass to the network, slowing the entire training procedure.

## 4.5 Reasoning Emulation for the Description Logic $\mathcal{EL}^+$

A major roadblock to progress for neuro-symbolic reasoning is that solutions and evaluations which work well for logic, or for deep learning and machine learning, do not work well in the opposite, and likely do not further the goal of integration. In this section we demonstrate an approach that embraces the liminality of integrating deep learning and deductive logic. Our approach is by its very nature ill-suited to either neural networks or logic alone. But it tries to avoid the pitfalls of unintentionally favoring one paradigm over the other, aiming instead to grasp at something new in the space between.

### *4.5.1 Problem Setting*

In a deductive reasoning system the semantics of the data is known *explicitly*. Why then would we want to blindly allow such a system to teach itself what is important when we already know what and how it should learn? Possibly we don't know the best ways to guide a complex network to the correct conclusions, but surely more, not less, transparency is needed for integrating logic and deep learning. Transparency often becomes difficult when we use extremely deep or complex

**Table 9** $\mathcal{EL}^+$ Completion Rules

| | | | | |
|---|---|---|---|---|
| (1) | $A \sqsubseteq C$ | $C \sqsubseteq D$ | | $\models A \sqsubseteq D$ |
| (2) | $A \sqsubseteq C_1$ | $A \sqsubseteq C_2$ | $C_1 \sqcap C_2 \sqsubseteq D$ | $\models A \sqsubseteq D$ |
| (3) | $A \sqsubseteq C$ | $C \sqsubseteq \exists R.D$ | | $\models A \sqsubseteq \exists R.D$ |
| (4) | $A \sqsubseteq \exists R.B$ | $B \sqsubseteq C$ | $\exists R.C \sqsubseteq D$ | $\models A \sqsubseteq D$ |
| (5) | $A \sqsubseteq \exists S.D$ | $S \sqsubseteq R$ | | $\models A \sqsubseteq \exists R.D$ |
| (6) | $A \sqsubseteq \exists R_1.C$ | $C \sqsubseteq \exists R_2.D$ | $R_1 \circ R_2 \sqsubseteq R$ | $\models A \sqsubseteq \exists R.D$ |

**Table 10** Support Generation

| | New Fact | Rule | Support |
|---|---|---|---|
| Step 1 | C1 $\sqsubseteq$ C3 | (1) | C1 $\sqsubseteq$ C2,C2 $\sqsubseteq$ C3 |
| | C1 $\sqsubseteq$ C4 | (4) | C1 $\sqsubseteq$ C2,C1 $\sqsubseteq$ $\exists$R1.C1,$\exists$R1.C2 $\sqsubseteq$ C4 |
| | C1 $\sqsubseteq$ $\exists$R1.C3 | (3) | C1 $\sqsubseteq$ C2,C2 $\sqsubseteq$ $\exists$R1.C3 |
| | C1 $\sqsubseteq$ $\exists$R2.C1 | (5) | C1 $\sqsubseteq$ $\exists$R1.C1,R1 $\sqsubseteq$ R2 |
| | C1 $\sqsubseteq$ $\exists$R4.C4 | (6) | C1 $\sqsubseteq$ $\exists$R1.C1,R1 $\circ$ R3 $\sqsubseteq$ R4,C1 $\sqsubseteq$ $\exists$R3.C4 |
| Step 2 | C1 $\sqsubseteq$ C5 | (2) | C3 $\sqcap$ C4 $\sqsubseteq$ C5,C1 $\sqsubseteq$ C2,C2 $\sqsubseteq$ C3,C1 $\sqsubseteq$ C2,C1 $\sqsubseteq$ $\exists$R1.C1,$\exists$R1.C2 $\sqsubseteq$ C4 |

networks that cannot be reduced to components. It also makes things difficult when we pre-process our data to improve results by training the system to learn embeddings. When we do this we struggle to tell if it was the embedding or the system itself or one of a dozen other things that might have caused an improvement. In response to these and other concerns we have performed an evaluation that tests whether a neural network is in fact capable of learning the structure, and not just the output, of a $\mathcal{EL}^+$ reasoning task without assistance.

It is a well established result that any $\mathcal{EL}^+$ knowledge base has a least fixed point that can be determined by repeatedly applying a finite set of completion rules that produce all entailments of a desired type [7,43]. In other words, we can say that reasoning in $\mathcal{EL}^+$ often amounts to an interconnected sequence of applications of a set of pattern-matching rules. One such set of rules, the set we have used in our experiment, is given in Table 9. The reasoning reaches *completion* when there are no new conclusions to be made. Because people are usually interested most in concept inclusions and restrictions, those are the types of statements we choose to include in our reasoning.

After reasoning finishes we are able to recursively define supports for each conclusion the reasoner reaches. The first step, of course, only has supports from the knowledge base. After this step supports are determined by effectively running the reasoner in reverse, and replacing each statement that is not in the original knowledge base with a superset that is, as you can see by the colored substitutions in Table 10. When the reasoner proved the last statement it did not consider all of the supports, since it had already proved them. It used the new facts it had learned in the last iteration. But we have drawn their supports back out so that we can define a fixed set of inputs exclusively from the knowledge base.

To provide sufficient training input to our system we use a synthetic generation procedure that combines a structured forced-lower-bound reasoning sequence with a connected randomized knowledge base. This allows us to rapidly generate many normal semi-random $\mathcal{EL}^+$ knowledge bases of arbitrary reasoning difficulty that

Initial Axioms:

| | | | |
|---|---|---|---|
| $C1 \sqsubseteq C2$ | $C1 \sqsubseteq \exists R1.C1$ | $C2 \sqsubseteq \exists R1.C3$ | $\exists R1.C2 \sqsubseteq C4$ |
| $C2 \sqsubseteq C3$ | $C1 \sqsubseteq \exists R2.C3$ | $C1 \sqsubseteq \exists R3.C4$ | $R1 \sqsubseteq R2$ |
| $C3 \sqcap C4 \sqsubseteq C5$ | $C2 \sqsubseteq \exists R2.C3$ | $C2 \sqsubseteq \exists R1.C3$ | $R1 \circ R3 \sqsubseteq R4$ |

Entailments Step 1:

| | | |
|---|---|---|
| $C1 \sqsubseteq C3$ | $C1 \sqsubseteq \exists R1.C3$ | $C1 \sqsubseteq \exists R2.C1$ |
| $C1 \sqsubseteq C4$ | | $C1 \sqsubseteq \exists R4.C4$ |

Entailments Step 2:

$\text{seed}_1 = C1 \sqsubseteq C5$

**Fig. 8** First Iteration of Sequence in an Example

always use all of the $\mathcal{EL}^+$ completion rules. An example of one iteration of the two-part sequence is provided in Figure 8. We also import data from the SNOMED 2012 ontology and sample connected subsets with a minimum of reasoning activity to ensure that our method is applicable to non-synthetic data. SNOMED is a widely-used, publicly available, ontology of medical terms and relationships [22]. SNOMED 2012 has 39392 logical axioms, some of which are complex, but this can be normalized in constant time to a logically equivalent set of 124,428 normal form axioms. We require that the samples be connected because any normal knowledge base is connected, and it improves the chances that the statements will have entailments. The reasoning task for SNOMED is more unbalanced than for the synthetic data. It is trivial for a reasoner to solve any $\mathcal{EL}^+$ knowledge base type. However, we observe that random connected sampling tends to favor application of rules 3, 5, and 6 (see Table 9) much more heavily than others, so the neural system will have a more difficult time learning the overall reasoning patterns. This imbalance is likely an artifact from SNOMED because it seems to recur in different sample sizes with different settings, though we acknowledge that it could be correlated somehow with the sampling procedure.

### *4.5.2 Evaluation*

Our system attempts to learn the structure of a reasoning task rather than reasoning answers. This is not to say we do not care about reasoning answers, or that they do not matter. Those values are reported for our system. However if reasoning structure is learned well enough then a system should emulate the same behavior and correct answers should follow.

If we examine the example output from the synthetic data inputs in Table 11, it is clear that it is getting very close to many correct answers. When it misses it still appears to be learning the shape, and this makes us optimistic about its future potential. The fact that most answers are close but not exact fits with our strategy of training to learn structure rather than answers. The SNOMED predictions are much more dense and do not fit well into a table, but we have included a few good examples with the original data labels translated into English sentences.

**Table 11** Example Synthetic Output

| | Correct Answer | Predicted Answer |
|---|---|---|
| Step 0 | C9 ⊑ C11 | C8 ⊑ C9 |
| | C2 ⊑ C10 | C1 ⊑ C9 |
| | C9 ⊑ C12 | C8 ⊑ C9 |
| | C7 ⊑ C6 | C8 ⊑ ∃R4.C9 |
| | C9 ⊑ ∃R4.C11 | C1 ⊑ ∃R5.C9 |
| | C2 ⊑ ∃R4.C9 | C8 ⊑ ∃R4.C9 |
| | C9 ⊑ ∃R5.C9 | C9 ⊑ ∃R5.C9 |
| | C2 ⊑ ∃R5.C11 | |
| | C9 ⊑ ∃R7.C12 | |
| | C2 ⊑ ∃R6.C12 | |
| Step 1 | C9 ⊑ C13 | C8 ⊑ C12 |
| | C2 ⊑ C11 | C2 ⊑ C10 |
| | C2 ⊑ C12 | C1 ⊑ C11 |
| | C2 ⊑ ∃R4.C11 | C1 ⊑ ∃R3.C12 |
| | C2 ⊑ ∃R5.C9 | C1 ⊑ ∃R4.C8 |
| | C2 ⊑ ∃R7.C12 | |
| Step 2 | C2 ⊑ C13 | C1 ⊑ C12 |

**Table 12** Example SNOMED Outputs

| | |
|---|---|
| Correct Answer | C6 ⊑ ∃R4.C1 |
| Meaning | if something is a zone of superficial fascia, then there is a subcutaneous tissue that it is PartOf |
| Prediction | C6 ⊑ ∃R4.C3 |
| Meaning | if something is a zone of superficial fascia, then there is a subcutaneous tissue of palmar area of little finger that it is PartOf |
| Correct Answer | C8 ⊑ ∃R3.C2 |
| Meaning | if something is a infrapubic region of pelvis, then there is a perineum that it is PartOf |
| Prediction | C9 ⊑ ∃R3.C2 |
| Meaning | if something is a zone of integument, then there is a perineum that it is PartOf |

For our evaluations we use three unique edit-distance measurements. Edit distance is used because it captures the degree to which each predicted statement misses what it should have been better than a simple accuracy number. We have a naive "Character" Levenshtein distance function that takes two unaltered knowledge base statement strings and computes their edit distance [80]. However, because some names in the namespace are one digit numbers and other names are two digit numbers, we include a modified version of this function, called "Atomic", that uniformly substitutes all two digit numbers in the strings with symbols that do not occur in either. Since there cannot be more than eight unique numbers in two decoded strings there are no issues with finding enough new symbols. By doing the substitutions we can see the impact that the number digits were having on the edits from the Atomic Levenshtein distance. Finally we devise a distance function that is based on our encoding scheme. The "Predicate" Distance method

**Table 13** Average Statement Edit Distances with Reasoner

| | Atomic Levenshtein Distance | | | Character Levenshtein Distance | | | Predicate Distance | | |
|---|---|---|---|---|---|---|---|---|---|
| | From | To | Average | From | To | Average | From | To | Average |
| | Synthetic Data | | | | | | | | |
| Piecewise Prediction | 1.336599 | 1.687640 | 1.512119 | 1.533115 | 1.812006 | 1.672560 | 2.633427 | 4.587382 | 3.610404 |
| Deep Prediction | **1.256940** | **1.507150** | **1.382045** | **1.454787** | **1.559751** | **1.507269** | **2.504496** | **3.552074** | **3.028285** |
| Flat Prediction | 1.344946 | 1.584674 | 1.464810 | 1.586281 | 1.660409 | 1.623345 | 2.517655 | 3.739770 | 3.128713 |
| Random Prediction | 1.598016 | 1.906369 | 1.752192 | 1.970604 | 1.289533 | 1.630068 | 5.467918 | 10.57324 | 8.020583 |
| | SNOMED Data | | | | | | | | |
| Piecewise Prediction | 1.704931 | **2.686562** | **2.195746** | 2.016249 | 2.862737 | 2.439493 | 6.556592 | **5.857769** | 6.207181 |
| Deep Prediction | 1.759633 | 3.052080 | 2.405857 | 2.027190 | 3.328850 | 2.678020 | **4.577427** | 6.179389 | **5.378408** |
| Flat Prediction | **1.691738** | 2.769542 | 2.230640 | **1.948757** | 2.991328 | 2.470042 | 5.548226 | 6.665659 | 6.106942 |
| Random Prediction | 1.814656 | 3.599629 | 2.707143 | 2.094682 | **1.621700** | **1.858191** | 5.169093 | 12.392325 | 8.780709 |

**Table 14** Average Precision Recall and F1-score For each Distance Evaluation

| | Atomic Levenshtein Distance | | | Character Levenshtein Distance | | | Predicate Distance | | |
|---|---|---|---|---|---|---|---|---|---|
| | Precision | Recall | F1-score | Precision | Recall | F1-score | Precision | Recall | F1-score |
| | Synthetic Data | | | | | | | | |
| Piecewise Prediction | 0.138663 | 0.142208 | 0.140412 | 0.138663 | 0.142208 | 0.140412 | 0.138646 | 0.141923 | 0.140264 |
| Deep Prediction | **0.154398** | **0.156056** | **0.155222** | **0.154398** | **0.156056** | **0.155222** | **0.154258** | **0.155736** | **0.154993** |
| Flat Prediction | 0.140410 | 0.142976 | 0.141681 | 0.140410 | 0.142976 | 0.141681 | 0.140375 | 0.142687 | 0.141521 |
| Random Prediction | 0.010951 | 0.0200518 | 0.014166 | 0.006833 | 0.012401 | 0.008811 | 0.004352 | 0.007908 | 0.007908 |
| | SNOMED Data | | | | | | | | |
| Piecewise Prediction | 0.010530 | 0.013554 | 0.011845 | 0.010530 | 0.013554 | 0.011845 | 0.010521 | 0.013554 | 0.011839 |
| Deep Prediction | **0.015983** | 0.0172811 | **0.016595** | **0.015983** | 0.017281 | **0.016595** | **0.015614** | 0.017281 | **0.016396** |
| Flat Prediction | 0.014414 | **0.018300** | 0.016112 | 0.0144140 | **0.018300** | 0.016112 | 0.013495 | **0.018300** | 0.015525 |
| Random Prediction | 0.002807 | 0.006803 | 0.003975 | 0.001433 | 0.003444 | 0.002023 | 0.001769 | 0.004281 | 0.002504 |

disassembles each string into only its predicates. Then, for each position in the 4-tuple, a distance is calculated that yields zero for perfect matches, absolute value of (guessed number - actual number) for correct Class and Role guesses, and (guessed number + actual number) for incorrect class and role matches. So, for instance, guessing C1 when the answer is C2 will yield a Predicate Distance of 1, while a guess of R2 for a correct answer of C15 will yield 17. Though this method is specific to our unique encoding, we believe it detects good and bad results quite well because perfect hits are 0, close misses are penalized a little, and large misses are penalized a lot.

For each method we take every statement in a knowledge base completion and compare it with the best match in the reasoner answer and random answers. While we compute these distances we are able to obtain precision, recall, and F1-score by counting the the number of times the distance returns zero and treating the statement predictions as classifications. Each time the system runs it can make any number of predictions, from zero to the maximum size of the output tensor. This means that, although the predictions and reasoner are usually around the same size, we have to generate random data to compare against that is as big as could conceivably be needed by the system. Any artificial shaping decisions we made to compensate for the variations between runs would invariably introduce their own bias in how we selected them. Thus the need to use the biggest possible random data to compare against means the precision, recall, and F1-score for random are low.

Our system is trained using randomized 10-fold cross validation at a learning rate of 0.0001 to 20000 epochs on the deep and flat systems and 10000 epochs each for the parts of the piecewise system. The data in Table 14 shows the edit distances calculated for the predictions against the correct answers, and 13 show the precision, recall, and F1-score numbers that result from those distance calculations.

It is interesting to note that by comparing Table 14 with Table 13 we can see that on the much harder SNOMED data the deep system *appears* to have a better result because of the higher F1 score, but the average edit distance, which is our preferred alternative measure for evaluation, is not obviously correlated with the F1-score. This is reflective of the shift in focus from purely accuracy optimizing systems and a more semantic type of structural learning. The "best" result will depend on which criteria is preferred.

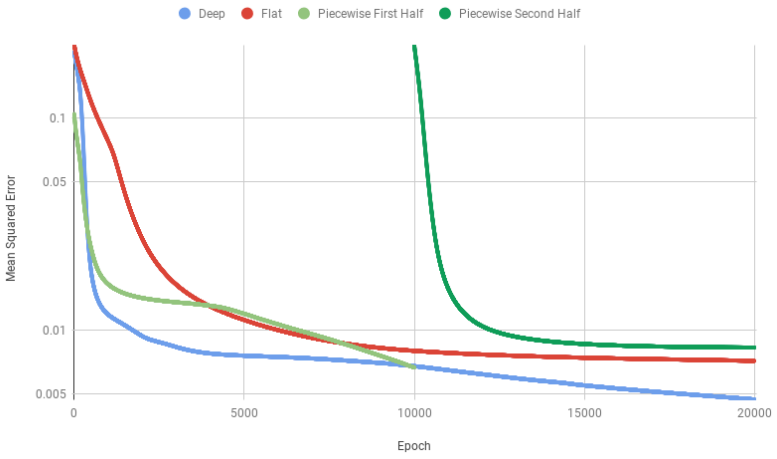


**Fig. 9** Synthetic Training

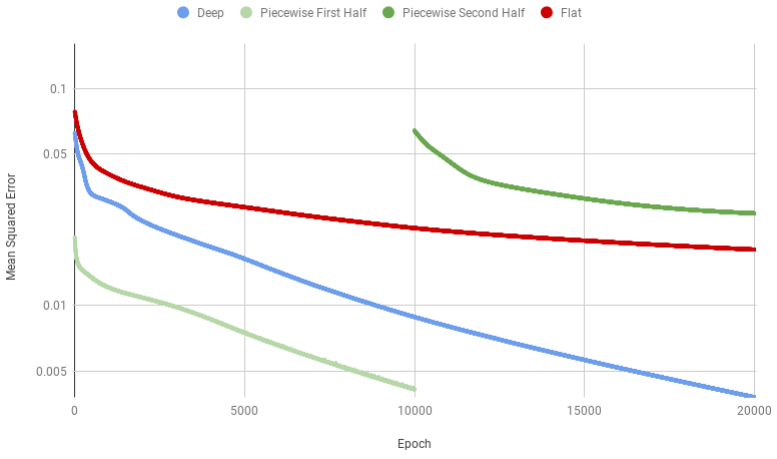


**Fig. 10** SNOMED Training

A cause for this difference may be the higher training difficulty for reaching the completion versus reaching the supports in the SNOMED data, which you can see in Figures 9 and 10. We can speculate on the degree to which various factors are contributing to this by comparing the piecewise architecture we have designed with the more traditional flat and deep systems. Forcing the network to conform to transparency-improving strategies like the piecewise network involves many trade-offs, some of which likely sacrifice a degree of accuracy, but for a highly structured task like neuro-symbolic reasoning, the ability to stop halfway and inspect answers has great potential for improving integration.

Source code and experiment data is available on GitHub.[20] Additional details can be found in the original publication [25].

## 5 Conclusions

This paper summarizes the authors' contributions in the neuro-symbolic integration research direction. First in the paper we examined the capability of memory-augmented networks in performing the RDF entailment for cross-knowledge graph deductive reasoning. Then we evaluated the deductive reasoning capability of LTNs over first-order fuzzy logic. Finally, we examined the strengths and weaknesses of variations of LSTM networks for $\mathcal{EL}^+$ reasoning. We aim to better understand the effectiveness of each model and the desirable properties expected with respect to three different logics (RDF, first-order fuzzy, and $\mathcal{EL}^+$). Such understanding would help pave the way for future efforts in this research direction.

Indeed, the results reported herein, while providing advances on the topic of neuro-symbolic deductive reasoning, also expose significant gaps in both the foundational methods used, and in terms of issues to be solved before practical applications can be built. For the RDFS approach, precision and recall values are good, but scalability is far beneath practical requirements. For the LTN approach, scalability is limited to mostly toy examples. For the $\mathcal{EL}^+$ approach, precision and recall values are good enough to stimulate further investigation, but are still way below any application needs.

There is a plethora of different deep learning approaches that could be investigated for neuro-symbolic dedcutive reasoning. The verdict is still open, though whether deductive reasoning, at reasonable scale and precision, is a problem class that can indeed be tackled using deep learning. On a fundamental level, deductive reasoning is known to be formally akin to topological dynamical systems (a.k.a., chaotic systems) [12,3,37], and thus pose a particularly hard challenge.

**Acknowledgements** This work was supported by the Air Force Office of Scientific Research under award number FA9550-18-1-0386 and by the National Science Foundation (NSF) under award OIA-2033521 "KnowWhereGraph: Enriching and Linking Cross-Domain Knowledge Graphs using Spatially-Explicit AI Technologies."

## References

1. Asai, M., Fukunaga, A.: Classical planning in deep latent space: Bridging the subsymbolic-symbolic boundary. In: S.A. McIlraith, K.Q. Weinberger (eds.) Proceedings of the Thirty-

[20] `https://github.com/aaronEberhart/ERCompletionReasoningLSTM`

Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018. AAAI Press (2018)
2. Bach, S.H., Broecheler, M., Huang, B., Getoor, L.: Hinge-loss markov random fields and probabilistic soft logic. Journal of Machine Learning Research **18**, 1–67 (2017)
3. Bader, S., Hitzler, P.: Logic programs, iterated function systems, and recurrent radial basis function networks. J. Appl. Log. **2**(3), 273–300 (2004). URL `https://doi.org/10.1016/j.jal.2004.03.003`
4. Bader, S., Hitzler, P., Hölldobler, S.: Connectionist model generation: A first-order approach. Neurocomputing **71**(13-15), 2420–2432 (2008)
5. Bader, S., Hitzler, P., Hölldobler, S., Witzel, A.: A fully connectionist model generator for covered first-order logic programs. In: M.M. Veloso (ed.) IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007, pp. 666–671 (2007)
6. Bahdanau, D., Bosc, T., Jastrzebski, S., Grefenstette, E., Vincent, P., Bengio, Y.: Learning to compute word embeddings on the fly. CoRR **abs/1706.00286** (2017). URL `http://arxiv.org/abs/1706.00286`
7. Besold, T.R., d'Avila Garcez, A.S., Bader, S., Bowman, H., Domingos, P.M., Hitzler, P., Kühnberger, K., Lamb, L.C., Lowd, D., Lima, P.M.V., de Penning, L., Pinkas, G., Poon, H., Zaverucha, G.: Neural-symbolic learning and reasoning: A survey and interpretation. CoRR **abs/1711.03902** (2017). URL `http://arxiv.org/abs/1711.03902`
8. Bianchi, F., Hitzler, P.: On the capabilities of logic tensor networks for deductive reasoning. In: AAAI Spring Symposium: Combining Machine Learning with Knowledge Engineering (2019)
9. Bianchi, F., Palmonari, M., Hitzler, P., Serafini, L.: Complementing logical reasoning with sub-symbolic commonsense. In: International Joint Conference on Rules and Reasoning, pp. 161–170. Springer (2019)
10. Bianchi, F., Palmonari, M., Nozza, D.: Towards encoding time in text-based entity embeddings. In: International Semantic Web Conference, pp. 56–71. Springer (2018)
11. Bianchi, F., Rossiello, G., Costabello, L., Palmonari, M., Minervini, P.: Knowledge graph embeddings and explainable ai. arXiv preprint arXiv:2004.14843 (2020)
12. Blair, H.A., Chidella, J., Dushin, F., Ferry, A., Humenn, P.R.: A continuum of discrete systems. Ann. Math. Artif. Intell. **21**(2-4), 153–186 (1997). URL `https://doi.org/10.1023/A:1018913302060`
13. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics **5**, 135–146 (2017)
14. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: Advances in neural information processing systems, pp. 2787–2795 (2013)
15. Bordes, A., Usunier, N., García-Durán, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: C.J.C. Burges, L. Bottou, Z. Ghahramani, K.Q. Weinberger (eds.) Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States., pp. 2787–2795 (2013)
16. Cai, H., Zheng, V.W., Chang, K.: A comprehensive survey of graph embedding: problems, techniques and applications. IEEE Transactions on Knowledge and Data Engineering (2018)
17. Confalonieri, R., Besold, T.R., Weyde, T., Creel, K., Lombrozo, T., Mueller, S.T., Shafto, P.: What makes a good explanation? cognitive dimensions of explaining intelligent machines. In: A.K. Goel, C.M. Seifert, C. Freksa (eds.) Proceedings of the 41th Annual Meeting of the Cognitive Science Society, CogSci 2019: Creativity + Cognition + Computation, Montreal, Canada, July 24-27, 2019, pp. 25–26. cognitivesciencesociety.org (2019). URL `https://mindmodeling.org/cogsci2019/papers/0013/index.html`
18. Consortium, W.W.W., et al.: Rdf 1.1 semantics. empty (2014)
19. Cyganiak, R., Wood, D., Lanthaler, M. (eds.): RDF 1.1 Concepts and Abstract Syntax. W3C Recommendation 25 February 2014 (2014). Available from http://www.w3.org/TR/rdf11-concepts/
20. Das, R., Dhuliawala, S., Zaheer, M., Vilnis, L., Durugkar, I., Krishnamurthy, A., Smola, A., McCallum, A.: Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. In: International Conference on Learning Representations (2018)

21. De Raedt, L., Kimmig, A.: Probabilistic (logic) programming concepts. Machine Learning **100**(1), 5–47 (2015)
22. De Silva, T.S., MacDonald, D., Paterson, G., Sikdar, K.C., Cochrane, B.: Systematized nomenclature of medicine clinical terms (snomed ct) to represent computed tomography procedures. Comput. Methods Prog. Biomed. **101**(3), 324–329 (2011). DOI 10.1016/j.cmpb.2011.01.002
23. Donadello, I., Serafini, L., d'Avila Garcez, A.S.: Logic tensor networks for semantic image interpretation. In: C. Sierra (ed.) Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017, pp. 1596–1602. ijcai.org (2017)
24. Dong, H., Mao, J., Lin, T., Wang, C., Li, L., Zhou, D.: Neural logic machines. In: International Conference on Learning Representations (2018)
25. Eberhart, A., Ebrahimi, M., Zhou, L., Shimizu, C., Hitzler, P.: Completion reasoning emulation for the description logic EL+. In: A. Martin, K. Hinkelmann, H. Fill, A. Gerber, D. Lenat, R. Stolle, F. van Harmelen (eds.) Proceedings of the AAAI 2020 Spring Symposium on Combining Machine Learning and Knowledge Engineering in Practice, AAAI-MAKE 2020, Palo Alto, CA, USA, March 23-25, 2020, Volume I, *CEUR Workshop Proceedings*, vol. 2600. CEUR-WS.org (2020). URL `http://ceur-ws.org/Vol-2600/paper5.pdf`
26. Ebrahimi, M., Sarker, M.K., Bianchi, F., Xie, N., Doran, D., Hitzler, P.: Reasoning over rdf knowledge bases using deep learning. arXiv preprint arXiv:1811.04132 (2018)
27. Fung, P., Wu, C., Madotto, A.: Mem2seq: Effectively incorporating knowledge bases into end-to-end task-oriented dialog systems. In: I. Gurevych, Y. Miyao (eds.) Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers, pp. 1468–1478. Association for Computational Linguistics (2018)
28. d'Avila Garcez, A., Lamb, L., Gabbay, D.M.: Neural-Symbolic Cognitive Reasoning. Springer, Heidelberg (2009)
29. d'Avila Garcez, A.S., Besold, T.R., Raedt, L.D., Földiák, P., Hitzler, P., Icard, T., Kühnberger, K., Lamb, L.C., Miikkulainen, R., Silver, D.L.: Neural-symbolic learning and reasoning: Contributions and challenges. In: 2015 AAAI Spring Symposia, Stanford University, Palo Alto, California, USA, March 22-25, 2015. AAAI Press (2015). URL `http://www.aaai.org/ocs/index.php/SSS/SSS15/paper/view/10281`
30. Grefenstette, E.: Towards a formal distributional semantics: Simulating logical calculi with tensors. In: Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity, pp. 1–10 (2013)
31. Grefenstette, E., Hermann, K.M., Suleyman, M., Blunsom, P.: Learning to transduce with unbounded memory. In: C. Cortes, N.D. Lawrence, D.D. Lee, M. Sugiyama, R. Garnett (eds.) Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada, pp. 1828–1836 (2015)
32. Gust, H., Kühnberger, K., Geibel, P.: Learning models of predicate logical theories with neural networks based on topos theory. In: B. Hammer, P. Hitzler (eds.) Perspectives of Neural-Symbolic Integration, *Studies in Computational Intelligence*, vol. 77, pp. 233–264. Springer (2007)
33. Hammer, B., Hitzler, P. (eds.): Perspectives of Neural-Symbolic Integration, *Studies in Computational Intelligence*, vol. 77. Springer (2007)
34. Hitzler, P.: Semantic Web: A review of the field. Communications of the ACM (2021). To appear
35. Hitzler, P., Bianchi, F., Ebrahimi, M., Sarker, M.K.: Neural-symbolic integration and the semantic web. Semantic Web (Preprint) pp. 1–9 (2019)
36. Hitzler, P., Hölldobler, S., Seda, A.K.: Logic programs and connectionist networks. J. Applied Logic **2**(3), 245–272 (2004)
37. Hitzler, P., Hölldobler, S., Seda, A.K.: Logic programs and connectionist networks. J. Appl. Log. **2**(3), 245–272 (2004). URL `https://doi.org/10.1016/j.jal.2004.03.002`
38. Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P.F., Rudolph, S. (eds.): OWL 2 Web Ontology Language: Primer (Second Edition). W3C Recommendation 11 December 2012 (2012). Available from http://www.w3.org/TR/owl2-primer/
39. Hitzler, P., Krötzsch, M., Rudolph, S.: Foundations of Semantic Web Technologies. Chapman & Hall/CRC (2010)

40. Hitzler, P., Seda, A.K.: Mathematical Aspects of Logic Programming Semantics. Chapman and Hall / CRC studies in informatics series. CRC Press (2011)
41. Hohenecker, P., Lukasiewicz, T.: Ontology reasoning with deep neural networks. Journal of Artificial Intelligence Research **68**, 503–540 (2020)
42. Hölldobler, S., Kalinke, Y.: Ein massiv paralleles modell für die logikprogrammierung. In: WLP, pp. 89–92 (1994)
43. Kazakov, Y., Krötzsch, M., Simančík, F.: Elk: a reasoner for owl el ontologies. System Description (2012)
44. Kiros, R., Zhu, Y., Salakhutdinov, R.R., Zemel, R., Urtasun, R., Torralba, A., Fidler, S.: Skip-thought vectors. In: C. Cortes, N.D. Lawrence, D.D. Lee, M. Sugiyama, R. Garnett (eds.) Advances in Neural Information Processing Systems 28, pp. 3294–3302. Curran Associates, Inc. (2015)
45. Klir, G., Yuan, B.: Fuzzy sets and fuzzy logic, vol. 4. Prentice hall New Jersey (1995)
46. Kulmanov, M., Liu-Wei, W., Yan, Y., Hoehndorf, R.: El embeddings: geometric construction of models for the description logic el++. In: Proceedings of the 28th International Joint Conference on Artificial Intelligence, pp. 6103–6109. AAAI Press (2019)
47. Lao, N., Mitchell, T., Cohen, W.W.: Random walk inference and learning in a large scale knowledge base. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, pp. 529–539. Association for Computational Linguistics (2011)
48. Le, Q., Mikolov, T.: Distributed representations of sentences and documents. In: International Conference on Machine Learning, pp. 1188–1196 (2014)
49. Lin, Y., Liu, Z., Sun, M., Liu, Y., Zhu, X.: Learning entity and relation embeddings for knowledge graph completion. In: B. Bonet, S. Koenig (eds.) Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA., pp. 2181–2187. AAAI Press (2015)
50. Ling, W., Dyer, C., Black, A.W., Trancoso, I., Fermandez, R., Amir, S., Marujo, L., Luís, T.: Finding function in form: Compositional character models for open vocabulary word representation. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 1520–1530 (2015)
51. Makni, B., Hendler, J.A.: Deep learning for noise-tolerant RDFS reasoning. Semantic Web **10**(5), 823–862 (2019). DOI 10.3233/SW-190363
52. McCarthy, J.: Epistemological challenges for connectionism. Behavioral and Brain Sciences p. 44 (1988)
53. McCulloch, W.S., Pitts, W.: A logical calculus of the ideas immanent in nervous activity. The Bulletin of Mathematical Biophysics **5**(4), 115–133 (1943)
54. Meza-Ruiz, I., Riedel, S.: Jointly identifying predicates, arguments and senses using markov logic. In: Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, pp. 155–163 (2009)
55. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems, pp. 3111–3119 (2013)
56. Minervini, P., Bošnjak, M., Rocktäschel, T., Riedel, S., Grefenstette, E.: Differentiable reasoning on large knowledge bases and natural language. Proceedings of the AAAI Conference on Artificial Intelligence **34**(04), 5182–5190 (2020). DOI 10.1609/aaai.v34i04.5962
57. Minervini, P., Riedel, S., Stenetorp, P., Grefenstette, E., Rocktäschel, T.: Learning reasoning strategies in end-to-end differentiable proving. In: ICML (2020)
58. Neelakantan, A., Roth, B., McCallum, A.: Compositional vector space models for knowledge base completion. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pp. 156–166 (2015)
59. Nguyen, D.Q., Nguyen, D.Q., Nguyen, T.D., Phung, D.: A convolutional neural network-based model for knowledge base completion and its application to search personalization. Semantic Web **10**(5), 947–960 (2019)
60. Pennington, J., Socher, R., Manning, C.: Glove: Global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pp. 1532–1543 (2014)
61. Raghu, D., Gupta, N., Mausam: Hierarchical pointer memory network for task oriented dialogue. CoRR **abs/1805.01216** (2018). URL `http://arxiv.org/abs/1805.01216`
62. Richardson, M., Domingos, P.: Markov logic networks. Machine learning **62**(1-2), 107–136 (2006)

63. Ristoski, P., Paulheim, H.: Rdf2vec: Rdf graph embeddings for data mining. In: International Semantic Web Conference, pp. 498–514. Springer (2016)
64. Rocktäschel, T., Riedel, S.: End-to-end differentiable proving. In: I. Guyon, U. von Luxburg, S. Bengio, H.M. Wallach, R. Fergus, S.V.N. Vishwanathan, R. Garnett (eds.) Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA, pp. 3791–3803 (2017)
65. Rocktäschel, T., Riedel, S.: End-to-end differentiable proving. In: Advances in Neural Information Processing Systems, pp. 3788–3800 (2017)
66. Rocktäschel, T., Singh, S., Riedel, S.: 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 (2015)
67. Serafini, L., d'Avila Garcez, A.S.: Learning and reasoning with logic tensor networks. In: G. Adorni, S. Cagnoni, M. Gori, M. Maratea (eds.) AI⋆A 2016: Advances in Artificial Intelligence – XVth International Conference of the Italian Association for Artificial Intelligence, Genova, Italy, November 29 – December 1, 2016, Proceedings, *Lecture Notes in Computer Science*, vol. 10037, pp. 334–348. Springer (2016)
68. Serafini, L., d'Avila Garcez, A.S.: Logic tensor networks: Deep learning and logical reasoning from data and knowledge. In: T.R. Besold, L.C. Lamb, L. Serafini, W. Tabor (eds.) Proceedings of the 11th International Workshop on Neural-Symbolic Learning and Reasoning (NeSy'16) co-located with the Joint Multi-Conference on Human-Level Artificial Intelligence (HLAI 2016), New York City, NY, USA, July 16-17, 2016., *CEUR Workshop Proceedings*, vol. 1768. CEUR-WS.org (2016)
69. Serafini, L., d'Avila Garcez, A.S.: Logic tensor networks: Deep learning and logical reasoning from data and knowledge. In: T.R. Besold, L.C. Lamb, L. Serafini, W. Tabor (eds.) Proceedings of the 11th International Workshop on Neural-Symbolic Learning and Reasoning (NeSy'16) co-located with the Joint Multi-Conference on Human-Level Artificial Intelligence (HLAI 2016), New York City, NY, USA, July 16-17, 2016, *CEUR Workshop Proceedings*, vol. 1768. CEUR-WS.org (2016). URL `http://ceur-ws.org/Vol-1768/NESY16\_paper3.pdf`
70. Shastri, L.: Advances in SHRUTI-A neurally motivated model of relational knowledge representation and rapid inference using temporal synchrony. Appl. Intell. **11**(1), 79–108 (1999)
71. Shastri, L.: SHRUTI: A neurally motivated architecture for rapid, scalable inference. In: B. Hammer, P. Hitzler (eds.) Perspectives of Neural-Symbolic Integration, *Studies in Computational Intelligence*, vol. 77, pp. 183–203. Springer (2007)
72. Socher, R., Chen, D., Manning, C.D., Ng, A.Y.: Reasoning with neural tensor networks for knowledge base completion. In: C.J.C. Burges, L. Bottou, Z. Ghahramani, K.Q. Weinberger (eds.) Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States., pp. 926–934 (2013)
73. Sukhbaatar, S., Szlam, A., Weston, J., Fergus, R.: End-to-end memory networks. In: C. Cortes, N.D. Lawrence, D.D. Lee, M. Sugiyama, R. Garnett (eds.) Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada, pp. 2440–2448 (2015)
74. Towell, G.G., Shavlik, J.W.: Knowledge-based artificial neural networks. Artif. Intell. **70**(1-2), 119–165 (1994)
75. Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., Bouchard, G.: Complex embeddings for simple link prediction. In: M. Balcan, K.Q. Weinberger (eds.) Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016, *JMLR Workshop and Conference Proceedings*, vol. 48, pp. 2071–2080. JMLR.org (2016)
76. Wang, Q., Mao, Z., Wang, B., Guo, L.: Knowledge graph embedding: A survey of approaches and applications. IEEE Trans. Knowl. Data Eng. **29**(12), 2724–2743 (2017)
77. Wang, W.Y., Cohen, W.W.: Learning first-order logic embeddings via matrix factorization. In: IJCAI, pp. 2132–2138 (2016)
78. Wang, Z., Zhang, J., Feng, J., Chen, Z.: Knowledge graph embedding by translating on hyperplanes. In: C.E. Brodley, P. Stone (eds.) Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada., pp. 1112–1119. AAAI Press (2014)

79. Weston, J., Chopra, S., Bordes, A.: Memory networks. In: Y. Bengio, Y. LeCun (eds.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (2015). URL http://arxiv.org/abs/1410.3916
80. Wikibooks contributors: Algorithm implementation/strings/levenshtein distance (2019 (accessed November 19, 2019)). URL https://en.wikibooks.org/wiki/Algorithm_Implementation/Strings/Levenshtein_distance#Python
81. Yang, B., Yih, W., He, X., Gao, J., Deng, L.: Embedding entities and relations for learning and inference in knowledge bases. In: Y. Bengio, Y. LeCun (eds.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (2015). URL http://arxiv.org/abs/1412.6575
82. Yang, F., Yang, Z., Cohen, W.W.: Differentiable learning of logical rules for knowledge base reasoning. In: Advances in Neural Information Processing Systems, pp. 2319–2328 (2017)