



Towards bridging the neuro-symbolic gap: deep deductive reasoners

Monireh Ebrahimi¹ · Aaron Eberhart¹ · Federico Bianchi² · Pascal Hitzler¹

Accepted: 18 December 2020 / Published online: 6 February 2021
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC part of Springer Nature 2021

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 terms of accuracy, scalability, transferability, generalizability, 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 applying 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

This article belongs to the Topical Collection: *30th Anniversary Special Issue*

✉ Monireh Ebrahimi
monireh@ksu.edu

Aaron Eberhart
aaroneberhart@ksu.edu

Federico Bianchi
f.bianchi@unibocconi.it

Pascal Hitzler
hitzler@ksu.edu

¹ Data Semantics Laboratory, Kansas State University, Manhattan, KS, USA

² Bocconi University, Milan, Italy

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 the schism, pitting mindsets and prior beliefs of communities against one 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 [6, 14, 25, 29]. In the abstract, one could hope for best-of-both-world systems, which combines the transparency and

¹The 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.”

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 [7, 8, 21, 22] at conferences and symposia.² Throughout the paper we will discuss strengths and limitations of models in terms of the accuracy, scalability, transferability, generalizability, 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

²Evrhimi et al [22] is under review at AAAI-MAKE 2021 symposium at the time of submitting this journal paper.

neuro-symbolic integration. It can be traced back at least to a landmark 1942 article by McCulloch and Pitts [47] 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 [6], and hence we will only mention essentials for understanding the context of our work.

Most of the body of work on neuro-symbolic integration concerns propositional logic only (see, for example, [24]), 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 referred as the “propositional fixation” of artificial neural networks [46]. Some of these include Knowledge-Based Artificial Neural Networks [68] and the closely related propositional Core method [31, 36]. Early attempts to go beyond propositional logic included the SHRUTI system [64, 65] 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 [28] or first-order logic programming [3, 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, 19, 35, 45, 58, 62, 63] – this list is not exhaustive. Even more work exists on inductive logical inference, for example [20, 53, 58], but this is not what we are investigating in our work.³ Recently, neural theorem provers [58] have shown exciting capabilities [50, 51] 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 [19, 58, 62, 63], following [27, 66], but present models remain restricted in terms of logical expressivity and/or to toy examples and limited evaluations. [45] performs knowledge graph reasoning using RDF(S) [15, 33] 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 [52, 60]. Zero-shot learning refers to

³Induction 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.

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. [35] moves away from RDFS to consider OWL RL reasoning [32, 33], 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 [17], which is a probabilistic logic programming language, and Markov Logic Networks (MLNs) are a statistical relational learning model that are effective on a large variety of tasks [48, 56]. 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.

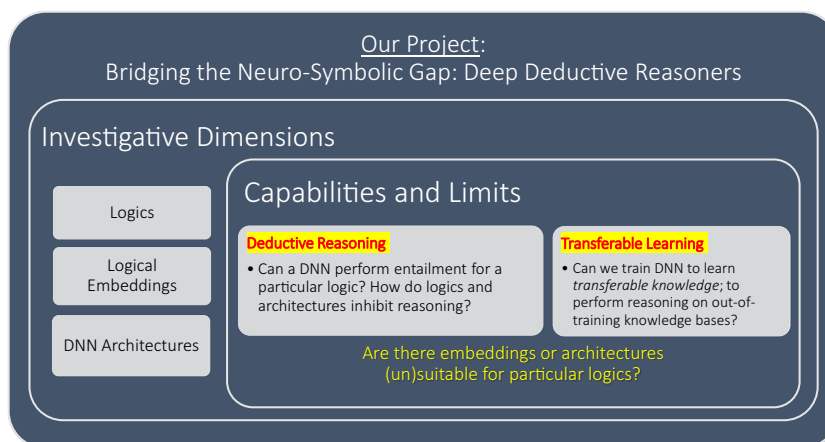
Finally, in the context of description logic reasoning there are additional unique challenges for the neuro-symbolic integration task. Not only are variable terms *implicit*, or not stated, in expressions, but also the open world assumption means that there is no fixed set of constants to use in training 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 [12, 40].

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 reframe the task as a classification problem. Any⁴ 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

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

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



(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 $T \in \mathcal{T}_{\mathcal{L}}$, 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 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.

⁵Attempting 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.

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, first 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 [33] research, and have direct bearing on current data management practice [30].

1. **RDFS (Resource Description Framework Schema).** The Resource Description Framework RDF, which includes RDF Schema (RDFS) [15, 33] is an established and widely used W3C standard for expressing knowledge graphs. The standard comes with formal semantics⁶ that define an entailment relation. 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, fixed across all knowledge graphs, which are expressible using Datalog.⁷ These thirteen rules can be used to entail new facts. The completion of a knowledge graph is in general infinite because, by definition, the set of entailed facts is infinite (related to RDF-encodings of lists); however the standard documents detail how compliant systems deal with this to produce finite completions.
2. **Real Logic/First-Order Fuzzy Logic.** First-order fuzzy logic [39] 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 [61] the authors introduce Real Logic, the logic that will be used to define Logic Tensor Networks in the next sections. This

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

⁷Datalog is equivalent to function-free definite logic programming [34].

Table 1 \mathcal{EL}^+ semantics

Description	Expression	Semantics
Individual	a	$a \in \Delta^{\mathcal{I}}$
Top	\top	$\Delta^{\mathcal{I}}$
Bottom	\perp	\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	$\exists R.C$	$\{ a \mid \text{there is } b \in \Delta^{\mathcal{I}} \text{ such that } (a, b) \in R^{\mathcal{I}} \text{ and } b \in C^{\mathcal{I}} \}$
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	$R_1 \circ \dots \circ R_n \sqsubseteq R$	$R_1^{\mathcal{I}} \circ \dots \circ R_n^{\mathcal{I}} \subseteq R^{\mathcal{I}}$

with \circ signifying standard binary composition

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 [61].

3. \mathcal{EL}^+ . \mathcal{EL}^+ is a lightweight and highly tractable description logic. 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 Σ and grammar for \mathcal{EL}^+ are 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 .

N_R is a set of Role names.

$\mathbf{R} ::= N_R$

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

The semantics of \mathcal{EL}^+ is defined by an interpretation I where $I = (\Delta^I, \cdot^I)$ which maps N_I, N_C, N_R to elements, sets, and relations in Δ^I with function \cdot^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) [11, 38, 42, 49, 54], and structured objects like graphs [13], knowledge graphs represented by a set of RDF triples [12, 43, 69, 72, 75],⁹ and types of logical formulas [19].

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

⁸<https://github.com/thunlp/KRLPapers> has an extensive listing of existing work on knowledge embeddings.

⁹See [10, 70] for a recent survey.

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

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 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 [13] 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 [26]. 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 [57] is an intriguing algorithm that maps RDF entities 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 [71] 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, *ACL* and *SRIOQ* [33].

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 [61] 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 $P(c_1, \dots, c_n)$ as a function of the grounding of the terms c_1, \dots, c_n [61]. For a predicate of arity m and for which $\mathbf{v}_1, \dots, \mathbf{v}_m \in \mathbb{R}^n$ are the groundings of m terms, the grounding of the predicate is defined as

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

where $\mathbf{v} = \langle \mathbf{v}_1, \dots, \mathbf{v}_m \rangle$ is the concatenation of the vectors \mathbf{v}_i , σ 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, $\forall x \text{mammal}(x) \rightarrow \text{animal}(x)$); are computed by using an aggregation operation [61] 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 $\forall x \text{mammal}(x) \rightarrow \text{animal}(x)$ 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 $\forall x \neg \text{mammal}(x) \rightarrow \text{species}(x)$; 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 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 [73]. Memory networks denote a family of models consisting of ‘memory cells’, which are defined as essentially 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

Table 2 Translation rules

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

c = Number of Possible Concept Names

r = Number of Possible Role Names

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 [5], 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 [23, 55] to refer to the unknown words in the memory in generating the responses. Another option is utilizing character-level embeddings [44] 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. We are particularly keen on examining Logic Tensor Networks (LTN) [19, 62, 63] which are based on Tensor Networks [66]. 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 [62, 63]. The scalability of LTNs have not been addressed other than in qualitative arguments which would need quantitative evaluation. Our work have explored LTNs in greater detail regarding 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 take a very 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. 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 settings

We begin with reasoning for the simplest logic under consideration: RDFS. A plethora of embeddings for RDFS have been proposed [12, 43, 69, 72, 75], but we are not aware of an embedding that has the two key ingredients (considering logical structure and ignoring entity strings) we hypothesize are necessary for the deductive reasoning task. We will 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 $\{a_1, \dots, a_n\}$, where 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. 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) [67] 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/RDFS, OWL EL and RL, 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 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 [41] that filters away irrelevant triplets or extracts sets of all paths that connect query-entity pairs. This way we can 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 will not focus heavily on scalability.

4.1.2 Evaluations

We now present and discuss our evaluation and results. We obtain training and test data from Linked Data Cloud website¹¹ and LOD laundromat.¹² The candidate entailments are composed of true entailments inferred by Jena¹³ 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 hyperparameters are detailed in [22]. 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. 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

¹¹<https://lod-cloud.net/>

¹²<http://lodlaundromat.org/>

¹³<https://jena.apache.org>

Table 3 Experimental results of proposed model

Training Dataset	Test Dataset	Valid Triples Class				Invalid Triples Class				Accuracy
		Precision	Recall /Sensitivity	F-measure	Precision	Recall /Specificity	F-measure	Accuracy		
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 ^{a b}	58	68	62	62	50	54	58		
OWL-Centric Dataset ^a	OWL-Centric Test Set ^{a b}	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		

^aMore tricky nos & balanced dataset^bCompletely different domain

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 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. 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 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, \dots, 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 $n \geq 2$. Table 4 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 4. 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 [16, 59, 76] 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 acquire. We assessed this assumption experimentally. For this purpose, we have done 10

Table 4 F-measure and data distribution over each reasoning hop

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%

^aTraining set

^bCompletely different domain

^cLemonUby Ontology

^dAgrovoc Ontology

Table 5 Ablation study: no positional encoding

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

^aCompletely different domain

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 Fig. 2. The PCA project 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 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 triplet into account. The results have been listed in Table 5. As anticipated, removing the positional encoding shows 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 negative class. That is the explanation for huge gap of the 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 NLP domain even when order matters.

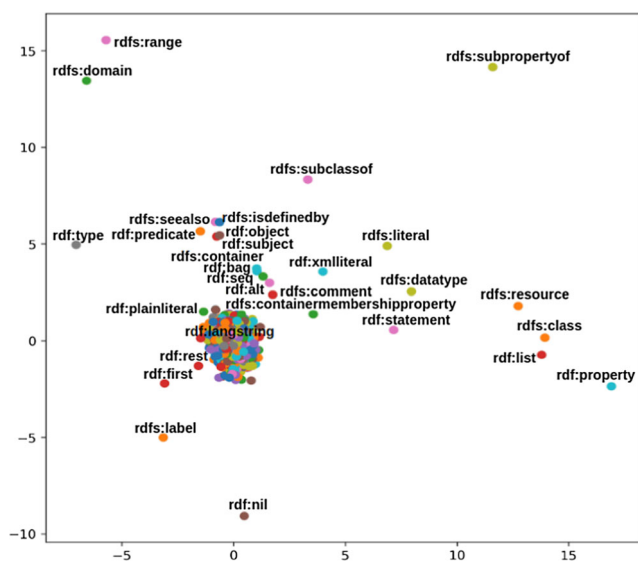


Fig. 2 PCA projection of embeddings for the vocabulary

4.2 Deductive reasoning capabilities of logic tensor networks

4.2.1 Problem settings

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

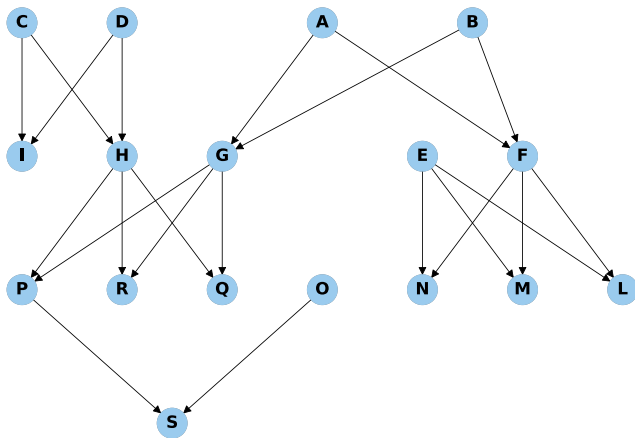


Fig. 3 Representation of the parental relationships in the P dataset

rules. In the second task, **reasoning with pre-trained entity embeddings** we will show that it is possible to combine LTNs with pre-trained entity embeddings [9] to account for both logical reasoning and a more general similarity based reasoning.

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 [8]. 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 Fig. 3, 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 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 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.

Table 6 Ancestor completion task with different number of axioms

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)

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 multi-hop inferences, $KB^a_{filtered}$. Bold entries highlight the most favorable score (depending on context, the highest or the lowest) for each relevant category

¹⁴<https://github.com/vinid/ltns-experiments>

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 [49] 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 that are not as ambiguous as words. Thus, we make use of Knowledge Graph Distributional Entity Embeddings described in [9], where Entity Embeddings (EEs) are generated from entity-to-entity co-occurrence in text. Since both *dbr:tiger*¹⁵ and *dbr:cat* appear in similar contexts, they will have similar vectors. These embeddings are 100-dimensional and the entity come from DBpedia.

The question that is left to answer is: “how do we combine these embeddings with LTNs?” LTNs treat constant 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.¹⁶

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 predicates 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 (for example, $\neg fungus(dbr:cat)$). 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.

- $\forall x(plant(x) \rightarrow eukaryote(x))$
- $\forall x(mammal(x) \rightarrow animal(x))$
- $\forall x : (mammal(x) \rightarrow Animal(x))$
- $\forall x(plant(x) \rightarrow \neg mammal(x))$
- $\forall x : (mammal(x) \rightarrow \neg plant(x))$
- $\forall x : (mammal(x) \rightarrow \neg fungus(x))$
- $\forall x : (mammal(x) \rightarrow \neg bacteria(x))$
- $\forall x : (bacteria(x) \rightarrow \neg Animal(x))$
- $\forall x(fungus(x) \rightarrow \neg animal(x))$

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:

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

¹⁶https://github.com/vinid/logical_commonsense

Table 7 F1 score per tested class

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

Bold entries highlight the most favorable score (depending on context, the highest or the lowest) for each relevant category

- Simple LTNs 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.¹⁷
- A simple deep neural network (DNN) that takes in 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, concatenate them and apply 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.

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*.¹⁸ 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 explainability.

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 run 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.

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

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

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

¹⁷<https://psl.linqs.org/>

¹⁸DNN cannot be used because the training consists of just positive instantiated atoms, the network would eventually just learn to output 1 for every input.

Table 8 The truth values of novel axioms

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

are born in New York, you are American. Though this small experiment is constrained by current implementation limitations of LTNs [7], 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 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 work we demonstrate a new 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.

4.4.1 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 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 axioms that produce all entailments of a desired type [6, 37]. In other words, we can say that reasoning in \mathcal{EL}^+ often amounts to a 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

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)	$C3 \sqcap C4 \sqsubseteq C5, C1 \sqsubseteq C2, C2 \sqsubseteq C3, C1 \sqsubseteq C2, C1 \sqsubseteq \exists R1.C1, \exists R1.C2 \sqsubseteq C4$

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 always use all of the \mathcal{EL}^+ completion rules. An example of one iteration of the two-part sequence is provided in Fig. 4. 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 [18]. 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 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 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.

4.4.2 Evaluation

Our system attempts to learn the structure of a reasoning task rather than reasoning answers. This is not to say we don't care about reasoning answers, or that they don't 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 (Table 12).

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.

Fig. 4 First iteration of sequence

Sequence:

$$\begin{aligned} \text{seed}_0 = & C1 \sqsubseteq C2 \\ & C2 \sqsubseteq C3 \\ & C3 \sqcap C4 \sqsubseteq C5 \end{aligned}$$

$$\begin{aligned} & C1 \sqsubseteq \exists R1.C1 \\ & C1 \sqsubseteq \exists R2.C3 \\ & C2 \sqsubseteq \exists R2.C3 \end{aligned}$$

$$\begin{aligned} & C2 \sqsubseteq \exists R1.C3 \\ & C1 \sqsubseteq \exists R3.C4 \\ & C2 \sqsubseteq \exists R1.C3 \end{aligned}$$

$$\begin{aligned} & \exists R1.C2 \sqsubseteq C4 \\ & R1 \sqsubseteq R2 \\ & R1 \circ R3 \sqsubseteq R4 \end{aligned}$$

Entailments Step 1:

$$\begin{aligned} & C1 \sqsubseteq C3 \\ & C1 \sqsubseteq C4 \end{aligned}$$

$$C1 \sqsubseteq \exists R1.C3$$

$$\begin{aligned} & C1 \sqsubseteq \exists R2.C1 \\ & C1 \sqsubseteq \exists R4.C4 \end{aligned}$$

Entailments Step 2:

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

Table 11 Example synthetic output

	Correct Answer	Predicted Answer
Step 0	$C9 \sqsubseteq C11$	$C8 \sqsubseteq C9$
	$C2 \sqsubseteq C10$	$C1 \sqsubseteq C9$
	$C9 \sqsubseteq C12$	$C8 \sqsubseteq C9$
	$C7 \sqsubseteq C6$	$C8 \sqsubseteq \exists R4.C9$
	$C9 \sqsubseteq \exists R4.C11$	$C1 \sqsubseteq \exists R5.C9$
	$C2 \sqsubseteq \exists R4.C9$	$C8 \sqsubseteq \exists R4.C9$
	$C9 \sqsubseteq \exists R5.C9$	$C9 \sqsubseteq \exists R5.C9$
	$C2 \sqsubseteq \exists R5.C11$	
	$C9 \sqsubseteq \exists R7.C12$	
Step 1	$C9 \sqsubseteq C13$	$C8 \sqsubseteq C12$
	$C2 \sqsubseteq C11$	$C2 \sqsubseteq C10$
	$C2 \sqsubseteq C12$	$C1 \sqsubseteq C11$
	$C2 \sqsubseteq \exists R4.C11$	$C1 \sqsubseteq \exists R3.C12$
	$C2 \sqsubseteq \exists R5.C9$	$C1 \sqsubseteq \exists R4.C8$
	$C2 \sqsubseteq \exists R7.C12$	
Step 2	$C2 \sqsubseteq C13$	$C1 \sqsubseteq C12$

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.[74] 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

Table 12 Example SNOMED outputs

Correct Answer	$C6 \sqsubseteq \exists R4.C1$
Meaning	if something is a zone of superficial fascia, then there is a subcutaneous tissue that it is PartOf
Prediction	$C6 \sqsubseteq \exists R4.C3$
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	$C8 \sqsubseteq \exists R3.C2$
Meaning	if something is an infrapubic region of pelvis, then there is a perineum that it is PartOf
Prediction	$C9 \sqsubseteq \exists R3.C2$
Meaning	if something is a zone of integument, then there is a perineum that it is PartOf

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 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.

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

Bold entries highlight the most favorable score (depending on context, the highest or the lowest) for each relevant category

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

Bold entries highlight the most favorable score (depending on context, the highest or the lowest) for each relevant category

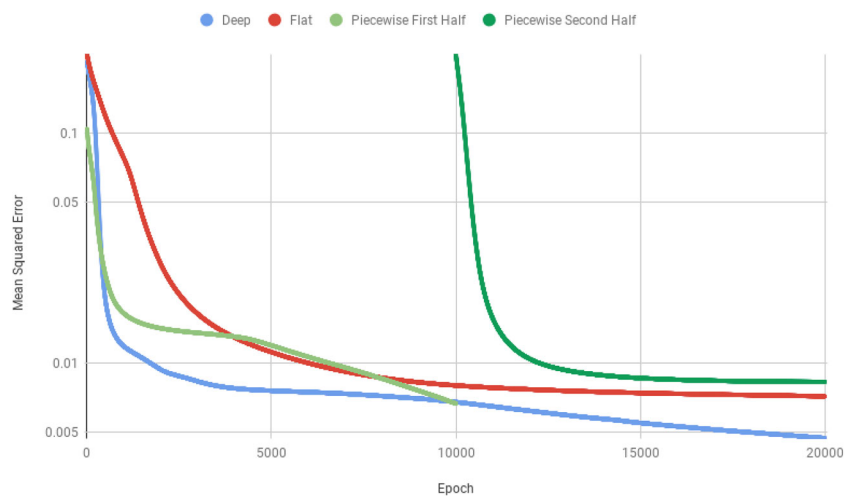
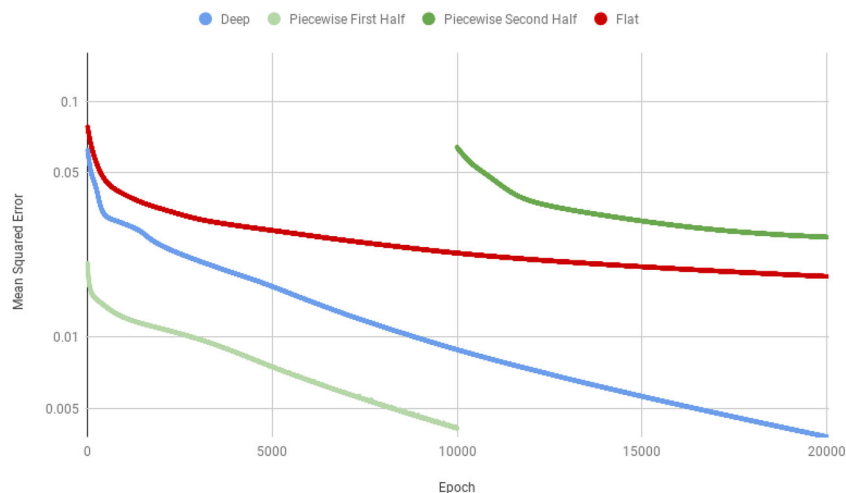
Fig. 5 Synthetic training

Fig. 6 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 Figs. 5 and 6. 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.¹⁹ Additional details can be found in the original publication [21].

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.

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

- Asai M, Fukunaga A (2018) Classical planning in deep latent space: Bridging the subsymbolic-symbolic boundary. In: McIlraith SA, Weinberger KQ (eds) Proceedings of the Thirty-Second AAAI Conference on artificial intelligence, New Orleans, Louisiana, USA, February 2-7, 2018, AAAI Press
- Bach SH, Broecheler M, Huang B, Getoor L (2017) Hinge-loss Markov random fields and probabilistic soft logic. *J Mach Learn Res* 18:1–67
- Bader S, Hitzler P, Hölldobler S (2008) Connectionist model generation: a first-order approach. *Neurocomputing* 71(13–15):2420–2432
- Bader S, Hitzler P, Hölldobler S, Witzel A (2007) A fully connectionist model generator for covered first-order logic programs. In: Veloso MM (ed) IJCAI 2007, Proceedings of the 20th international joint conference on artificial intelligence, Hyderabad, India, January 6–12, 2007, pp 666–671
- Bahdanau D, Bosc T, Jastrzebski S, Grefenstette E, Vincent P, Bengio Y (2017) Learning to compute word embeddings on the fly. arXiv:1706.00286
- Besold TR, d'Avila Garcez A, Bader S, Bowman H, Domingos P, Hitzler P, Kühnberger K, Lamb L, Lowd D, Lima PMV, de Penning L, Pinkas G, Poon H, Zaverucha G (2017) Neural-symbolic learning and reasoning: A survey and interpretation. arXiv:1711.03902
- Bianchi F, Hitzler P (2019) On the capabilities of logic tensor networks for deductive reasoning. In: AAAI Spring symposium: combining machine learning with knowledge engineering
- Bianchi F, Palmonari M, Hitzler P, Serafini L (2019) Complementing logical reasoning with sub-symbolic commonsense. In: International joint conference on rules and reasoning, Springer, pp 161–170
- Bianchi F, Palmonari M, Nozza D (2018) Towards encoding time in text-based entity embeddings. In: International semantic web conference, Springer, pp 56–71

¹⁹<https://github.com/aaronEberhart/ERCompletionReasoningLSTM>

10. Bianchi F, Rossiello G, Costabello L, Palmonari M, Minervini P (2020) Knowledge graph embeddings and explainable AI arXiv:2004.14843
11. Bojanowski P, Grave E, Joulin A, Mikolov T (2017) Enriching word vectors with subword information. *Trans Assoc Comput Linguist* 5:135–146
12. Bordes A, Usunier N, García-Durán A, Weston J, Yakhnenko O (2013) Translating embeddings for modeling multi-relational data. In: Burges CJC, Bottou L, Ghahramani Z, Weinberger KQ (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
13. Cai H, Zheng VW, Chang K (2018) A comprehensive survey of graph embedding: problems, techniques and applications. *IEEE Trans Knowl Data Eng*
14. Confalonieri R, Besold TR, Weyde T, Creel K, Lombrozo T, Mueller ST, Shafto P (2019) What makes a good explanation? cognitive dimensions of explaining intelligent machines. In: Goel AK, Seifert CM, Freksa C. (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. https://mindmodeling.org/cogsci2019/papers/0013/index.html](https://mindmodeling.org/cogsci2019/papers/0013/index.html)
15. 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/>
16. Das R, Dhuliawala S, Zaheer M, Vilnis L, Durugkar I, Krishnamurthy A, Smola A, McCallum A (2018) Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. In: *International conference on learning representations*
17. De Raedt L, Kimmig A (2015) Probabilistic (logic) programming concepts. *Mach Learn* 100(1):5–47
18. De Silva TS, MacDonald D, Paterson G, Sikdar KC, Cochrane B (2011) Systematized nomenclature of medicine clinical terms (SNOMED CT) to represent computed tomography procedures. *Comput Methods Prog Biomed* 101(3):324–329. <https://doi.org/10.1016/j.cmpb.2011.01.002>
19. Donadello I, Serafini L, d'Avila Garcez A (2017) Logic tensor networks for semantic image interpretation. In: Sierra C (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
20. Dong H, Mao J, Lin T, Wang C, Li L, Zhou D (2018) Neural logic machines. In: *International conference on learning representations*
21. Eberhart A, Ebrahimi M, Zhou L, Shimizu C, Hitzler P (2020) Completion reasoning emulation for the description logic EL+. In: Martin A, Hinkelmann K, Fill H, Gerber A, Lenat D, Stolle R, van Harmelen F (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. <http://ceur-ws.org/Vol-2600/paper5.pdf>
22. Ebrahimi M, Sarker MK, Bianchi F, Xie N, Doran D, Hitzler P (2018) Reasoning over RDF knowledge bases using deep learning arXiv:1811.04132
23. Fung P, Wu C, Madotto A (2018) Mem2seq: Effectively incorporating knowledge bases into end-to-end task-oriented dialog systems. In: Gurevych I, Miyao Y (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
24. d'Avila Garcez A, Lamb L, Gabbay DM (2009) *Neural-symbolic cognitive reasoning*. Springer, Heidelberg
25. d'Avila Garcez A, Besold TR, Raedt LD, Földiák P, Hitzler P, Icard T, Kühnberger K, Lamb L, Miikkulainen R, Silver DL (2015) 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. <http://www.aaai.org/ocs/index.php/SSS/SSS15/paper/view/10281>
26. Grefenstette E (2013) 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
27. Grefenstette E, Hermann KM, Suleyman M, Blunsom P (2015) Learning to transduce with unbounded memory. In: Cortes C, Lawrence ND, Lee DD, Sugiyama M, Garnett R (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
28. Gust H, Kühnberger K, Geibel P (2007) Learning models of predicate logical theories with neural networks based on topos theory. In: Hammer B, Hitzler P (eds) *Perspectives of neural-symbolic integration, studies in computational intelligence*, vol 77. Springer, pp 233–264
29. Hammer B, Hitzler P (eds) (2007) *Perspectives of neural-symbolic integration, studies in computational intelligence*, vol 77. Springer, Berlin
30. Hitzler P (2021) *Semantic Web: A review of the field*. Communications of the ACM. To appear
31. Hitzler P, Hölldobler S, Seda AK (2004) Logic programs and connectionist networks. *J Appl Logic* 2(3):245–272
32. Hitzler P, Krötzsch M, Parsia B, Patel-Schneider PF, Rudolph S *OWL 2 Web Ontology Language: primer (Second Edition)*. W3C Recommendation 11 December 2012 (2012). Available from <http://www.w3.org/TR/owl2-primer/>
33. Hitzler P, Krötzsch M, Rudolph S (2010) *Foundations of semantic web technologies*. Chapman & Hall/CRC, Boca Raton
34. Hitzler P, Seda AK (2011) *Mathematical aspects of logic programming semantics*. Chapman and Hall / CRC studies in informatics series. CRC Press, Boca Raton
35. Hohenecker P, Lukasiewicz T (2020) Ontology reasoning with deep neural networks. *J Artif Intell Res* 68:503–540
36. Hölldobler S, Kalinke Y (1994) Ein massiv paralleles Modell für die Logikprogrammierung. In: *WLP*, pp 89–92
37. Kazakov Y, Krötzsch M, Simančík F (2012) *Elk: a reasoner for OWL EL ontologies*. System Description
38. Kiros R, Zhu Y, Salakhutdinov RR, Zemel R, Urtasun R, Torralba A, Fidler S (2015) Skip-thought vectors. In: Cortes C, Lawrence ND, Lee DD, Sugiyama M, Garnett R (eds) *Advances in neural information processing systems 28*, pp 3294–3302. Curran Associates, Inc.
39. Klir G, Yuan B (1995) *Fuzzy sets and fuzzy logic*, vol 4, Prentice hall, New Jersey
40. Kulmanov M, Liu-Wei W, Yan Y, Hoehndorf R (2019) EL embeddings: geometric construction of models for the description logic EL++. In: *Proceedings of the 28th International joint conference on artificial intelligence, AAAI Press*, pp 6103–6109
41. Lao N, Mitchell T, Cohen WW (2011) 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
42. Le Q, Mikolov T (2014) Distributed representations of sentences and documents. In: *International conference on machine learning*, pp 1188–1196

43. Lin Y, Liu Z, Sun M, Liu Y, Zhu X (2015) Learning entity and relation embeddings for knowledge graph completion. In: Bonet B, Koenig S (eds) Proceedings of the Twenty-Ninth AAAI conference on artificial intelligence, January 25–30, 2015, Austin, Texas, USA, AAAI Press, pp 2181–2187
44. Ling W, Dyer C, Black AW, Trancoso I, Fernandez R, Amir S, Marujo L, Lufs T (2015) 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
45. Makni B, Hendler J (2019) Deep learning for noise-tolerant RDFS reasoning. *Semantic Web* 10(5):823–862
46. McCarthy J (1988) Epistemological challenges for connectionism. *Behav Brain Sci* 44
47. McCulloch WS, Pitts W (1943) A logical calculus of the ideas immanent in nervous activity. *Bull Math Biophys* 5(4):115–133
48. Meza-Ruiz I, Riedel S (2009) 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
49. Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J (2013) Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems, pp 3111–3119
50. Minervini P, Bošnjak M, Rocktäschel T, Riedel S, Grefenstette E (2020) Differentiable reasoning on large knowledge bases and natural language. In: Proceedings of the AAAI conference on artificial intelligence, vol 34, pp 5182–5190. <https://doi.org/10.1609/aaai.v34i04.5962>
51. Minervini P, Riedel S, Stenetorp P, Grefenstette E, Rocktäschel T (2020) Learning reasoning strategies in end-to-end differentiable proving. In: ICML
52. Neelakantan A, Roth B, McCallum A (2015) 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
53. Nguyen DQ, Nguyen DQ, Nguyen TD, Phung D (2019) A convolutional neural network-based model for knowledge base completion and its application to search personalization. *Semantic Web* 10(5):947–960
54. Pennington J, Socher R, Manning C (2014) Glove: Global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pp 1532–1543
55. Raghu D, Gupta N (2018) Mausam: Hierarchical pointer memory network for task oriented dialogue. arXiv:1805.01216
56. Richardson M, Domingos P (2006) Markov logic networks. *Mach Learn* 62(1–2):107–136
57. Ristoski P, Paulheim H (2016) RDF2Vec: RDF graph embeddings for data mining. In: International semantic web conference. Springer, pp 498–514
58. Rocktäschel T, Riedel S (2017) End-to-end differentiable proving. In: Guyon I, von Luxburg U, Bengio S, Wallach HM, Fergus R, Vishwanathan SVN, Garnett R (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
59. Rocktäschel T, Riedel S (2017) End-to-end differentiable proving. In: Advances in neural information processing systems, pp 3788–3800
60. Rocktäschel T, Singh S, Riedel S (2015) Injecting logical background knowledge into embeddings for relation extraction. In: Proceedings of the 2015 conference of the north american chapter of the association for computational linguistics: human language technologies, pp 1119–1129
61. Serafini L, d’Avila Garcez A (2016) Logic tensor networks: Deep learning and logical reasoning from data and knowledge. arXiv:1606
62. Serafini L, d’Avila Garcez A (2016) Learning and reasoning with logic tensor networks. In: Adorni G, Cagnoni S, Gori M, Maratea M (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. Springer, pp 334–348
63. Serafini L, d’Avila Garcez A (2016) Logic tensor networks: Deep learning and logical reasoning from data and knowledge. In: Besold TR, Lamb LC, Serafini L, Tabor W (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
64. Shastri L (1999) Advances in SHRUTI-A neurally motivated model of relational knowledge representation and rapid inference using temporal synchrony. *Appl Intell* 11(1):79–108
65. Shastri L (2007) SHRUTI: A neurally motivated architecture for rapid, scalable inference. In: Hammer B, Hitzler P (eds) Perspectives of neural-symbolic integration, studies in computational intelligence, vol 77. Springer, pp 183–203
66. Socher R, Chen D, Manning CD, Ng AY (2013) Reasoning with neural tensor networks for knowledge base completion. In: Burges CJC, Bottou L, Ghahramani Z, Weinberger KQ (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
67. Sukhbaatar S, Szlam A, Weston J, Fergus R (2015) End-to-end memory networks. In: Cortes C, Lawrence ND, Lee DD, Sugiyama M, Garnett R (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
68. Towell GG, Shavlik JW (1994) Knowledge-based artificial neural networks. *Artif Intell* 70(1–2):119–165
69. Trouillon T, Welbl J, Riedel S, Gaussier E, Bouchard G (2016) Complex embeddings for simple link prediction. In: Balcan M, Weinberger KQ (eds) Proceedings of the 33rd 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
70. Wang Q, Mao Z, Wang B, Guo L (2017) Knowledge graph embedding: A survey of approaches and applications. *IEEE Trans Knowl Data Eng* 29(12):2724–2743
71. Wang WY, Cohen WW (2016) Learning first-order logic embeddings via matrix factorization. In: IJCAI, pp 2132–2138
72. Wang Z, Zhang J, Feng J, Chen Z (2014) Knowledge graph embedding by translating on hyperplanes. In: Brodley CE, Stone P (eds) Proceedings of the Twenty-Eighth AAAI Conference on artificial intelligence, July 27–31, 2014, Québec City, Québec, Canada., AAAI Press, pp 1112–1119
73. Weston J, Chopra S, Bordes A (2014) Memory networks. arXiv:1410.3916. Published as ICLR 2015 conference paper
74. Wikibooks contributors: Algorithm implementation/strings/levenshtein distance (2019 (accessed November 19, 2019)). https://en.wikibooks.org/wiki/Algorithm_Implementation/Strings/Levenshtein_distance#Python

75. Yang B, Yih W, He X, Gao J, Deng L (2014) Embedding entities and relations for learning and inference in knowledge bases. arXiv:1412.6575
76. Yang F, Yang Z, Cohen WW (2017) Differentiable learning of logical rules for knowledge base reasoning. In: Advances in neural information processing systems, pp 2319–2328

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Monireh Ebrahimi is a PhD candidate at the Data Semantics (DaSe) Lab, Department of Computer Science at Kansas State University and Senior Cognitive Software Developer at Center for Open Source Data and AI Technologies (CODAIT) at IBM Watson in San Francisco.



Aaron Eberhart is a PhD Candidate in Computer Science at the Data Semantics (DaSe) Lab, Kansas State University.



Federico Bianchi is a Post-doctoral Researcher at Bocconi University and a member of the Bocconi Institute for Data Science and Analytics (BIDSA). He obtained his PhD from the University of Milan-Bicocca in 2020.



Pascal Hitzler is Lloyd T. Smith Creativity in Engineering Chair, Director of the Center for Artificial Intelligence and Data Science (CAIDS), and Director of the Data Semantics (DaSe) Lab, Department of Computer Science, Kansas State University.