Towards generalizable neuro-symbolic reasoners

by

Monireh Ebrahimi

B.S., Shiraz University, 2011

M.S., University Technology Malaysia, 2013

AN ABSTRACT OF A DISSERTATION

submitted in partial fulfillment of the
requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Computer Science
Carl R. Ice College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

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.

Over the course of this dissertation, we provide a brief summary of our recent efforts to bridge the neural and symbolic divide in the context of deep deductive reasoners. More specifically, We designed a novel way of conducting neuro-symbolic through pointing to the input elements. More importantly we showed that the proposed approach is generalizable across new domain and vocabulary demonstrating symbol-invariant zero-shot reasoning capability. Furthermore, We have demonstrated that a deep learning architecture based on memory networks and pre-embedding normalization is capable of learning how to perform deductive reason over previously unseen RDF KGs with high accuracy. We are applying these models on Resource Description Framework (RDF), first-order logic, and the description logic $\mathcal{EL}^+$respectively. Throughout this dissertation we will discuss strengths and limitations of these models particularly in term of accuracy, scalability, transferability, and generalizabiliy. Based on our experimental results, pointer networks perform remarkably well across multiple reasoning tasks while outperforming the previously reported state of the art by a significant

margin. We observe that the Pointer Networks preserve their performance even when challenged with knowledge graphs of the domain/vocabulary it has never encountered before. To our knowledge, this work is the first attempt to reveal the impressive power of pointer networks for conducting deductive reasoning. Similarly, we show that memory networks can be trained to perform deductive RDFS reasoning with high precision and recall. The trained memory network's capabilities in fact transfer to previously unseen knowledge bases. Finally will talk about possible modifications to enhance desirable capabilities. Altogether, these research topics, resulted in a methodology for symbol-invariant neuro-symbolic reasoning.

Towards generalizable neuro-symbolic reasoners

by

Monireh Ebrahimi

B.S., Shiraz University, 2011

M.S., University Technology Malaysia, 2013

A DISSERTATION

submitted in partial fulfillment of the
requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Computer Science
Carl R. Ice College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2021

Approved by:

Major Professor
Pascal Hitzler

Copyright

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.

Over the course of this dissertation, we provide a brief summary of our recent efforts to bridge the neural and symbolic divide in the context of deep deductive reasoners. More specifically, We designed a novel way of conducting neuro-symbolic through pointing to the input elements. More importantly we showed that the proposed approach is generalizable across new domain and vocabulary demonstrating symbol-invariant zero-shot reasoning capability. Furthermore, We have demonstrated that a deep learning architecture based on memory networks and pre-embedding normalization is capable of learning how to perform deductive reason over previously unseen RDF KGs with high accuracy. We are applying these models on Resource Description Framework (RDF), first-order logic, and the description logic $\mathcal{EL}^+$respectively. Throughout this dissertation we will discuss strengths and limitations of these models particularly in term of accuracy, scalability, transferability, and generalizabiliy. Based on our experimental results, pointer networks perform remarkably well across multiple reasoning tasks while outperforming the previously reported state of the art by a significant

margin. We observe that the Pointer Networks preserve their performance even when challenged with knowledge graphs of the domain/vocabulary it has never encountered before. To our knowledge, this work is the first attempt to reveal the impressive power of pointer networks for conducting deductive reasoning. Similarly, we show that memory networks can be trained to perform deductive RDFS reasoning with high precision and recall. The trained memory network's capabilities in fact transfer to previously unseen knowledge bases. Finally will talk about possible modifications to enhance desirable capabilities. Altogether, these research topics, resulted in a methodology for symbol-invariant neuro-symbolic reasoning.

Table of Contents

List of Figures

Acknowledgments

During this journey, I have had the fortune to be supported by a number of truly incredible people. The first and foremost, I would like to express my sincere appreciation and heartfelt gratitude to my advisor, Dr. Pascal Hitzler, for the continuous support of my research, for his patience, encouragement, and guidance. He believed in me when I doubted myself and supported me during my difficult research time and academic journey. I could not have imagined having a better advisor for this long journey. Without his encouragement and continuous guidance especially during my transfer from my previous advisor, I could not have finished this dissertation. I can never thank him enough. I would also like to thank Rr. Mateen M. Rizki and Dr. Michelle Cheatham for helping me out during this journey especially when I was switching my advisor.

Besides, I would like to thank all my colleagues and friends for their support and the cherished time we spent together. Also, to the several mentors I have had whom pushed my boundaries in the new domains while doing internship under their supervision. I express my gratitude to all the friends and collaborators, especially at Data Semantics (DaSe) Lab, my colleagues and mentors at IBM, Bosch Research & Technology Center, and The Hive. I also have to thank the members of my PhD committee, Professors Doina Caragea, Michael Raymer, William Hsu, and Punit Prakash for their helpful advice.

Last but not least, a special thanks to my husband, for supporting me in every step to see this through to the end. To my parents in Iran that I miss so much, and to my brothers and sister who supported me spiritually along the way.

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

Dedication

To Mom and Dad, who have always been great source of inspiration and support for me.

Chapter 1

Introduction and Background

1.1 Neuro-Symbolic Deductive Reasoning

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

Symbolic and subsymbolic systems are almost entirely complementary to each other. For example, the key strengths of subsymbolic systems are weaknesses of symbolic ones, and vice versa. Symbolic systems are *brittle*; they are susceptible to data noise or minor flaws in the

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

The complementary nature of these methods has drawn a stark divide in the rich field of AI. The split is technical in nature; symbol manipulation as captured by deductive reasoning cannot be sufficiently performed using current subsymbolic systems. Moreover, the training to study subsymbolic systems (involving probability theory, statistics, linear algebra, and optimization) differs from symbolic systems (involving logic and propositional calculus, set theory, recursion, and computability) so strongly that AI researchers tend to find a side of the divide based on their intellectual interests and background. There is even a cultural aspect to he schism, pitting mindsets and prior beliefs of communities against one 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 [19, 31, 6, 15]. In the abstract, one could hope for best-of-both-world systems, which combines the transparency and reasoning-ability of symbolic systems with the robustness and learning-capabilities of subsymbolic ones. Integrated symbolic-subsymbolic systems may be able to address the knowledge acquisition bottleneck faced by symbolic systems, learn to perform advanced logical or symbolic reason-

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

ing 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.

Many of the earlier lines of research on neuro-symbolic integration, discussed primarily from a cognitive science perspective, can be found in [8]. Of particular interest is the integration of deep learning with logics that are not propositional in nature, since propositional logic is of limited applicability to knowledge representation and reasoning tasks. In the wake of deep learning breakthroughs, fundamental issues around neuro-symbolic integration have recently received increased attention with some progress being made as new approaches emerge. In particular, there has been progress in developing neural networks that can learn to reason. These include the Neural Theorem Prover (NTP) and its variations [57, 56, 47, 46], Logic Tensor Networks (LTN) [61, 9, 3], and the application of memory networks and LSTMs [22] and others [45, 38]. Yet, there is still much work to do in terms of new model development and investigation of inductive bias in existing architectures and their reasoning capability.

This thesis describes the efforts made to investigate our methodologies toward bridging the neural and symbolic approaches divide, and focused on the following primary tasks.

1. Examining the reasoning capability of pointer networks [68] in emulating deductive reasoning

2. Investigating the emulation of deductive symbolic reasoning using memory networks

3. Designing a framework for Neuro-Symbolic Deductive Reasoning for Cross-Knowledge Graph Entailment

4. Examining the generalization power and transfer learning capability of proposed approaches

The outline of the particular topics researched to accomplish these is provided in the next section.

1.2 Outline

This thesis is a cumulative dissertation that details the foundational research towards neuro-symbolic techniques in general and their generalization capability in particular. As mentioned in the above introduction, the methods introduced here can be employed for conducting deductive reasoning using neural networks accurately even when challenged with knowledge graphs of the domain/vocabulary it has never encountered before.

The remainder of this dissertation is outlined as follows:

Chapter 2 provides a background and motivation of our work in the context of techniques, logics, and logical embeddings that have been used and we summarize related work for our line of research. In chapter 3 we explore Pointer Networks in the context of deductive reasoning tasks, highlighting the properties, weaknesses, and strengths of these models. In chapter 4we outline the experimental results of our memory network based RDF deductive reasoning system with focus on transferability and generalization. We give concluding remarks and ideas for future work in Section 5.

The primary contributions referenced in this dissertation are:

- [24] Monireh Ebrahimi, Aaron Eberhart, and Pascal Hitzler. On the capabilities of pointer networks for deep deductive reasoning. *arXiv preprint arXiv:2106.09225*, 2021

  A pointer attention based neuro-symbolic approach for conducting the symbol-invariant deductive reasoning over RDF and $\mathcal{EL}^+$knowledge bases by formulating the reasoning task as a pointing problem [24]

- [23] Monireh Ebrahimi, Aaron Eberhart, Federico Bianchi, and Pascal Hitzler. Towards bridging the neuro-symbolic gap: Deep deductive reasoners. *Applied Intelligence*, pages 1–23, 2021

- [26] Monireh Ebrahimi, Md. Kamruzzaman Sarker, Federico Bianchi, Ning Xie, Aaron Eberhart, Derek Doran, HyeongSik Kim, and Pascal Hitzler. Neuro-symbolic deductive reasoning for cross-knowledge graph entailment. In Andreas Martin, Knut Hinkelmann, Hans-Georg Fill, Aurona Gerber, Doug Lenat, Reinhard Stolle, and Frank van Harmelen, editors, *Proceedings of the AAAI 2021 Spring Symposium on Combining Machine Learning and Knowledge Engineering (AAAI-MAKE 2021), Stanford University, Palo Alto, California, USA, March 22-24, 2021*, volume 2846 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2021

- [25] Monireh Ebrahimi, Md Kamruzzaman Sarker, Federico Bianchi, Ning Xie, Aaron Eberhart, Derek Doran, and Pascal Hitzler. On the generalization capability of memory networks for reasoning. *ICML 2019 Workshop on Understanding and Improving Generalization in Deep Learning*, 2019

  A differentiable end-to-end deep memory based neuro-symbolic approach for conducting the symbol-invariant deductive reasoning over RDF knowledge bases by formulating the reasoning task as a memory access problem [23, 26, 25]

Chapter 3 strives to answer two main research questions: "Can Pointer Networks perform logical deductive reasoning using pointer attention?", and more generally, "Can other attention-based sequence-to-sequence models like self-attention based popular Transformer architectures successfully perform the same task?","How well do pointer network reasoners perform on completely new knowledge graphs?", and finally, "How robust is our model to noise?". To answer these questions we conduct a set of experiments by applying pointer networks and transformers to RDFS and $\mathcal{EL}^+$ reasoning tasks. We believe, the answer to our third question is particularly very important since it a very big step toward developing accurate yet symbol-invariant deep deductive reasoners which generalize very well on unseen knowledge bases of differing domain or vocabulary.

Our contributions are as follows:

- [24] Monireh Ebrahimi, Aaron Eberhart, and Pascal Hitzler. On the capabilities of pointer networks for deep deductive reasoning. *arXiv preprint arXiv:2106.09225*, 2021

- *A novel paradigm for viewing a symbolic reasoning problem as a pointing problem.*
- *Pointer Networks are used to neurally resolve symbolic reasoning for the first time.*
- *The proposed approach is able to transfer its reasoning ability to new domain/vocabulary knowledge graph of same logic.*
- *We report the state-of-the-art performance of the $\mathcal{EL}^+$ and RDF reasoning*

Chapter 4 demonstrates that a deep learning architecture based on memory networks and pre-embedding normalization is capable of learning how to perform deductive reason over previously unseen RDF knowledge graphs with high accuracy.

Our contributions are as follows:

- [23] Monireh Ebrahimi, Aaron Eberhart, Federico Bianchi, and Pascal Hitzler. Towards bridging the neuro-symbolic gap: Deep deductive reasoners. *Applied Intelligence*, pages 1–23, 2021
- [26] Monireh Ebrahimi, Md. Kamruzzaman Sarker, Federico Bianchi, Ning Xie, Aaron Eberhart, Derek Doran, HyeongSik Kim, and Pascal Hitzler. Neuro-symbolic deductive reasoning for cross-knowledge graph entailment. In Andreas Martin, Knut Hinkelmann, Hans-Georg Fill, Aurona Gerber, Doug Lenat, Reinhard Stolle, and Frank van Harmelen, editors, *Proceedings of the AAAI 2021 Spring Symposium on Combining Machine Learning and Knowledge Engineering (AAAI-MAKE 2021), Stanford University, Palo Alto, California, USA, March 22-24, 2021*, volume 2846 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2021
- [25] Monireh Ebrahimi, Md Kamruzzaman Sarker, Federico Bianchi, Ning Xie, Aaron Eberhart, Derek Doran, and Pascal Hitzler. On the generalization capability of memory networks for reasoning. *ICML 2019 Workshop on Understanding and Improving Generalization in Deep Learning*, 2019
- *We present the construction of memory networks for emulating the symbolic deductive reasoning.*

- *We propose an optimization to this architecture using normalization approach to enhance their transfer capability. We show that in an unnormalized setting, they fail to perform well across knowledge graphs.*

- *We examine the efficacy of our model for cross-domain and cross-KG deductive reasoning. We also show the scalability of our model (in terms of reduced time and space complexity) for large datasets.*

Chapter 5 presents concluding remarks through a brief summary that highlights the overall contributions. Besides, we provide an outlook on future work.

Chapter 2

Background

2.1 Deep Deductive Reasoning

Training artificial neural networks to learn deductive reasoning is a hard machine learning task that was out of reach before the advent of deep learning. In the last few years, several publications have shown that deep deductive reasoning – using deep learning methods – is possible. We will briefly review the core body of existing work. As we will see, it remains a hard task, even for deep learning.

Before we do so, though, let us point out that our work is different from what is usually called knowledge graph completion, or the study of knowledge graph embeddings, although we deal with logics relevant for knowledge graphs [34]: Knowledge graph completion (sometimes called link prediction or knowledge graph refinement) [50] refers to enriching a knowledge graph with additional relationships that are statistically induced, sometimes using machine learning methods. In contrast to this, we are studying *deductive* reasoning, which is not based on statistics or likelihood, but based on a mathematical, logical calculus that derives additional statements which were already implicit – in a mathematically precisely defined sense – in the statements already made. Deductive inference tasks are usually hard computationally (e.g., for propositional logic, it is NP-complete), and are traditionally addressed using complex but provably correct algorithms – correctness in this sense is in

relation to the underlying mathematical definitions that determine what is, and what is not, a deductive logical consequence. The study of knowledge graph embeddings [55], in isolation, is about the learning of representations of knowledge graphs in multi-dimensional Euclidean space. While embeddings are often a component of deep deductive reasoning systems, our goal is the overall functionality of deep deductive reasoning, and not just knowledge graph embeddings in isolation.

A good overview of existing deep deductive reasoning work is [22]. It appears to be appropriate to distinguish between the different logics that are addressed in the literature, the reasonable assumption being that less complex logics are easier to learn, and this resonates with the as yet limited body of work. We refer to [36] for background on all the mentioned logics. We know about investigations of RDFs [26, 45], of $\mathcal{EL}^+$[21], of OWL RL [38], and of first-order predicate logic (FOL) [9].

paper	logic	transfer	generative	scale	performance
[45]	RDFS	no	yes	low	high
[21]	$\mathcal{EL}^+$	yes	yes	moderate	low
[38]	OWL RL	no*	no	low	high
[9]	FOL	no	yes	very low	high

Table 2.1: *Overview of published deep deductive reasoning work. See the main text for details on the columns. no* indicates that the paper claims that transfer is possible in principle, but it was not demonstrated or evaluated to what extent transfer really happens.*

paper	logic	transfer	generative	scale	performance
[26]	RDFS	yes	no	moderate	high
[24]	RDFS & $\mathcal{EL}^+$	yes	yes	moderate	high

Table 2.2: *Overview of this dissertation's contribution to deep deductive reasoning work. See the main text for details on the columns.*

We give an overview of the key aspects of each of these in Table 2.1 – we admit that some interpretations in this table may be somewhat subjective. The column "transfer" indicates whether the system was demonstrated to have a good transfer capability to previously unknown and very different knowledge bases. The column "generative" indicates whether the system generates all (under certain finiteness constraints) deductive inferences in one run – if not, then it would usually be query-based, i.e. it would be able to tell whether a

given logical expression is a logical consequence of the knowledge base. The column "scale" indicates how large the input knowledge bases were in the experiments, ranging from a few logical statements as in the FOL case to RDF graphs with 1,000 triples in [26]. The column "performance" indicates how well the system learned to reason; "high" indicates 70% or more in terms of f-measure, while "low" indicates values just a bit better than random guessing.

For a deep deductive reasoner, we would ideally like to have it on an expressive logic, with transfer, generative, at massive scale, and with high performance. For all the referenced works, except the FOL one, the scale aspect has not been systematically explored yet; for the FOL case, it does look rather unfavorable as discussed in [9]. Otherwise, it is important to note that only one of the works is both generative and able to transfer, however this was also the system with very low performance. As we will see, our new approach we report on in this paper is able to do transfer and is generative, with high performance. This is the key contribution of this paper.

2.1.1 RDF Reasoning

The Resource Description Framework RDF, which includes RDF Schema (RDFS) [17, 36] is an established and widely used W3C standard for expressing knowledge graphs. The standard comes with a formal semantics[1] that defines an entailment relation. An RDFS knowledge base (KB) is a collection of statements stored as *triples* $(e1, r, e2)$ where $e1$ and $e2$ are called *subject* and *object*, respectively, while r is a binary relation between $e1$ and $e2$. In the context of RDF/RDFS, the triple notation $(e1, r, e2)$ is more common than a notation like $r(e1, e2)$ as it is suggestive of a node-edge-node piece of a labelled graph, and so we will use the triple notation.

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 [16], fixed across all knowledge

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

graphs, which are expressible using Datalog.[2] These thirteen rules can be used to entail new facts.

Table 2.3: *Selected RDFS Completion Rules*

$$(x, \text{rdfs:subClassOf}, y), (y, \text{rdfs:subClassOf}, z) \models (x, \text{rdfs:subClassOf}, z) \quad (2.1)$$
$$(x, \text{rdfs:subPropertyOf}, y), (y, \text{rdfs:subPropertyOf}, z) \models (x, \text{rdfs:subPropertyOf}, z) \quad (2.2)$$
$$(x, \text{rdfs:subClassOf}, y), (z, \text{rdf:type}, x) \models (z, \text{rdf:type}, y) \quad (2.3)$$
$$(a, \text{rdfs:domain}, x), (y, a, z) \models (y, \text{rdf:type}, x) \quad (2.4)$$
$$(a, \text{rdfs:range}, x), (y, a, z) \models (z, \text{rdf:type}, x) \quad (2.5)$$

Table 2.3 shows examples for some of these entailment rules. The identifiers x, y, z, a are variables. The remaining elements of the triples are pre-fixed with the rdfs or rdf namespace (a concept borrowed from XML) and carry a specific meaning in the formal semantics of RDFS. E.g., rdfs:subClassOf indicates a sub-class (or sub-set) relationship, i.e. Rule 2.1 states transitivity of the rdf:subClassOf binary relation. Likewise, in Rule 2.2, $(x, \text{rdfs:subPropertyOf}, y)$ indicates that x, y are to be understood as binary relations, where x is a restriction (called a *subproperty*) of y. In Rule 2.3, the triple $(z, \text{rdf:type}, x)$ indicates that z is a member of the class (or set) x. In Rules 2.4 and 2.5, rdfs:domain and rdfs:range indicate domain respectively range of a, which is to be interpreted as a binary relation. The rules are applied exhaustively on an input RDF knowledge base, i.e. inferred triples are added and then rule execution continues taking the new triples also into account.

2.1.2 $\mathcal{EL}^+$easoning

The standard reasoning task over $\mathcal{EL}^+$ is called *classification* and can be understood as the computation of all formulas of the form $\forall x(p(x) \rightarrow q(x))$ entailed by the given theory, and the set of all these formulas, which is called the *completion* of the input theory, is finite if the input theory is finite.

Formally, let N_C be a set of atomic classes (or concepts, or class names), let N_R be a set

[2]Datalog is equivalent to function-free definite logic programming [37].

of roles (or properties), and let N_I be a set of individuals. Complex class expressions (or simply complex classes or classes) in the description logic $\mathcal{EL}^+$ are defined by the grammar

$$C ::= A | C_1 \sqcap C_2 | \exists R.C,$$

where $A \in N_C$, $R \in N_R$, and C_1, C_2, and C are complex class expressions. A TBox in $\mathcal{EL}^+$ is a set of general class inclusion axioms (or TBox statements) of the form $C \sqsubseteq D$, where C, D are (complex) classes. We use $C \equiv D$ as abbreviation for the two statements $C \sqsubseteq D$ and $D \sqsubseteq C$. An RBox in $\mathcal{EL}^+$ is a set of general role inclusion axioms (or RBox statements) of the form $R_1 \circ \cdots \circ R_n \sqsubseteq R$, where R, $R_i \in N_R$ (for all i). An $\mathcal{EL}^+$ knowledge base (or ontology) is a set of TBox and RBox statements.

$\mathcal{EL}^+$ is in fact a fragment of first-order predicate logic: all statements can be translated into it, and the inherited semantics is exactly the first-order predicte logic semantics – details can be found in [36]. Classification is known to be P-complete.

An $\mathcal{EL}^+$ *normal form* knowledge base contains only axioms of the following forms.

$$C \sqsubseteq D \qquad C_1 \sqcap C_2 \sqsubseteq D \qquad C \sqsubseteq \exists R.D$$

$$\exists R.C \sqsubseteq D \qquad R_1 \sqsubseteq R \qquad R_1 \circ R_2 \sqsubseteq R$$

As usual, every $\mathcal{EL}^+$ knowledge base can be cast into normal form in polynomial time, and such that it suffices to perform classification over the normal form knowledge base.

Given an $\mathcal{EL}^+$ knowledge base K in normal form, the completion $\mathrm{comp}(K)$ of K can for example be obtained from K by exhaustively applying the completion rules from Table 2.4. There are of course different ways to perform classification using reasoning algorithms, e.g. reasoning can also be encoded using a larger number of Datalog rules which remain fixed across input theories [40]. So on the surface this seems similar to RDFS reasoning. However $\mathcal{EL}^+$ as a logic has a different look and feel: RDFS reasoning focuses on the derivation of new facts from old facts, while $\mathcal{EL}^+$ is about the processing of schema knowledge, in particular subclass relationships, in the presence of existential quantification (which is completely

absent from RDFS). The transformation into Datalog reasoning is also more complicated than for RDFS.

Table 2.4: $\mathcal{EL}^+$ *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)	$A \sqsubseteq \exists R_1.C$	$C \sqsubseteq \exists R_2.D$	$R_1 \circ R_2 \sqsubseteq R$	$\models$	$A \sqsubseteq \exists R.D$

Chapter 3

On the Generalization Capability of Pointer Networks for RDF and $\mathcal{EL}^+$ Reasoning

3.1 Overview

In the wake of deep learning breakthroughs, fundamental issues around neuro-symbolic integration have recently received increased attention with some progress being made as new approaches emerge. In particular, there has been progress in developing neural networks that can learn to reason. These include the Neural Theorem Prover (NTP) and its variations [57, 56, 47, 46], Logic Tensor Networks (LTN) [61, 9, 3], and the application of memory networks and LSTMs [22] and others [45, 38]. Yet, there is still much work to do in terms of new model development and investigation of inductive bias in existing architectures and their reasoning capability.

This dissertation tries to fill this gap by examining the reasoning capability of pointer networks [68] in emulating deductive reasoning. Pointer Networks and their variations have been applied successfully to a variety of sophisticated tasks including theoretical computer science problems (i.e., NP-hard Travelling Salesman Problem (TSP), Delaunay Triangula-

tion, and Convex Hull) [68] and practical problems like abstractive [60] and extractive [39] text summarization, code completion [41], and dependency parsing [29, 28, 44]. Nevertheless, almost nothing is known about their potential for conducting logical deductive reasoning accurately. In fact, they have been mainly used for solving discrete combinatorial optimization problems because of their variable-size output vocabulary and for resolving the rare or out-of-vocabulary problem in Natural Language Processing. We hypothesize that using the pointer attention to decide what elements of the input knowledge base should be chosen as the output, and in which order, will work well for deductive reasoning tasks. Indeed, using pointer networks we can mimic human reasoning behaviour where one can learn to choose a set of symbols in different locations and copy these symbols to suitable locations to generate new logical consequences based on a set of predefined logical entailment rules. To verify this, here, we explore the capabilities and limitations of pointer networks for performing deductive reasoning on Resource Description Framework (RDF) [13] and $\mathcal{EL}^+$ [2] knowledge bases in terms of accuracy, and generalizability.Based on our experimental results, pointer networks perform remarkably well across multiple reasoning tasks while outperforming the previously reported state of the art by a significant margin. We observe that the Pointer Networks preserve their performance even when challenged with knowledge graphs of the domain/vocabulary it has never encountered before. To our knowledge, this work is the first attempt to reveal the impressive power of pointer networks for conducting deductive reasoning.

In terms of the logic, in this dissertation we are looking at two logics with different expressivity, power, and reasoning difficulty. The Resource Description Framework Schema (RDFS) [13] is non-trivial (and non-propositional), yet one of the simplest widely used logics: It is a mature W3C Semantic Web standard that is commonly used to express knowledge graphs and linked data [34], and many corresponding data sets are freely available on the World Wide Web [54]. The standard carries a model-theoretic semantics which defines deductive entailment [32, Section 9.2], and reasoning over RDFS is usually done using rule-based reasoning engines. The second logic is the description logic $\mathcal{EL}^+$ (or $\mathcal{ER}$) [2] that is the basis for the W3C standard OWL EL [35]. It is considered to be a rather inexpressive

but practically useful logic [59] and generally used for expressing ontologies and knowledge graph schemas [36] particularly in medical domain ontologies.

3.2 Contribution

[24] Monireh Ebrahimi, Aaron Eberhart, and Pascal Hitzler. On the capabilities of pointer networks for deep deductive reasoning. *arXiv preprint arXiv:2106.09225*, 2021

In the publication, *On the capabilities of pointer networks for deep deductive reasoning* [24], we investigate the benefits and limitations of encoder-decoder architectures in general and pointer networks in particular for developing accurate and generalizable neuro-symbolic reasoners. In short this paper strives to answer two main research questions: "Can Pointer Networks perform logical deductive reasoning using pointer attention?", and more generally, "Can other attention-based sequence-to-sequence models like self-attention based popular Transformer architectures successfully perform the same task?","How well do pointer network reasoners perform on completely new knowledge graphs?", and finally, "How robust is our model to noise?". To answer these questions we conduct a set of experiments by applying pointer networks and transformers to RDFS and $\mathcal{EL}^+$ reasoning tasks. We believe, the answer to our third question is particularly very important since it a very big step toward developing accurate yet symbol-invariant deep deductive reasoners which generalize very well on unseen knowledge bases of differing domain or vocabulary. The contributions of this work are fourfold:

1. A novel paradigm for viewing a symbolic reasoning problem as a pointing problem.
2. Pointer Networks are used to neurally resolve symbolic reasoning for the first time.
3. The proposed approach is able to transfer its reasoning ability to new domain/vocabulary knowledge graph of same logic.
4. We report the state-of-the-art performance of the $\mathcal{EL}^+$ and RDF reasoning.

RDF and $\mathcal{EL}^+$ Reasoning using Pointer Networks: We re-frame our entailment prob-

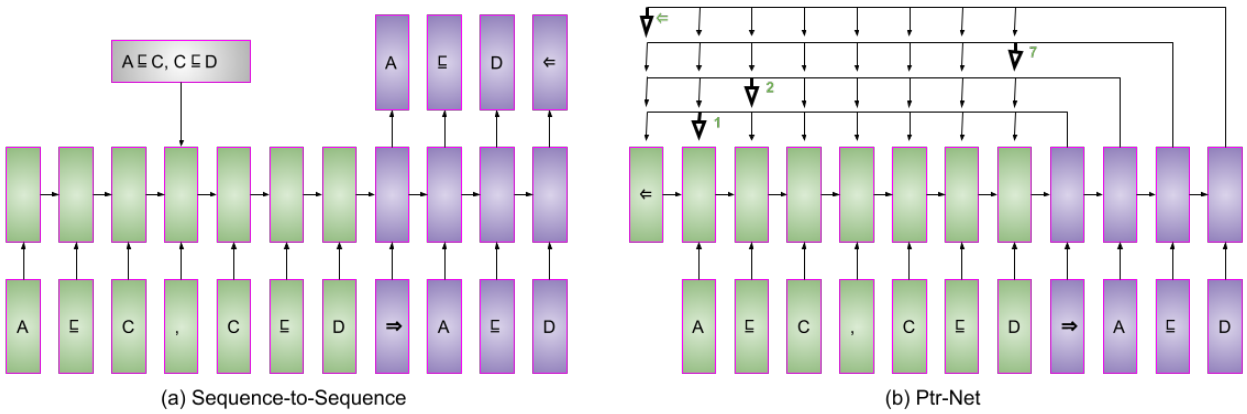

Figure 3.1: *$\mathcal{EL}^+$ Completion Rule (1): "(a) Sequence-to-Sequence - An RNN (green) processes the input sequence to create a code vector that is used to generate the output sequence (purple) using the probability chain rule and another RNN. (b) Ptr-Net - An encoding RNN converts the input sequence to a code (green) that is fed to the generating network (purple). At each step, the generating network produces a vector that modulates a content-based attention mechanism over inputs. The output of the attention mechanism is a softmax distribution with dictionary size equal to the length of the input."*

lem as an input-output mapping task: Given some logic $\mathcal{L}$, for each theory T over $\mathcal{L}$, the set $c(T) = \{F \mid T \models_{\mathcal{L}} F\}$ of all formulas over $\mathcal{L}$ that are entailed by T; we call $c(T)$ the *completion* of T. We can then attempt to train a neural network to produce $c(T)$ for any given T over $\mathcal{L}$, i.e., we would use pairs $(T, c(T))$ as input-output training pairs for a generative deep deductive reasoner.

We have used pointer networks to copy the symbols from the knowledge base via pointing to generate logical consequences.

In our corpus, each knowledge graph and its completion denoted as $(T, c(T))$ comprises the sequence of symbols. Given the $(T, c(T))$ pair, we are feeding the pointer network with $(\mathcal{T}, \mathcal{C}(\mathcal{T})')$ where $\mathcal{T} = \{T_1, ..., T_n\}$ is a sequence of n symbols each refer to an element in our input knowledge graph and $\mathcal{C}(\mathcal{T})' = \{c(T_1)', ..., c(T_m)'\}$ is a sequence of m indices each between 1 and n. The sequence-to-sequence model then computes the conditional probability $p(c(T_i)'|c(T_1)', ..., c(T_{i-1})', \mathcal{T})$ changing the Bahdanau [5] attention to the pointer attention

as follows:

$$u_j^i = v^T \tanh(W_1 e_j + W_2 d_i) \qquad \text{for } j \in (1, ..., n) \qquad \text{and} \tag{3.1}$$

$$p(c(T)'_i | c(T)'_1, ..., c(T)'_i - 1, \mathcal{T}) = \text{Softmax}(u^i), \tag{3.2}$$

where $(e_1, ..., e_n)$ and $(d_1, ..., d_m)$ denote the encoder and decoder hidden states respectively, and v, W_1, and W_2 are learnable parameters of the output model. The softmax

$$\text{Softmax}(u_i) = \frac{e^{(u_i)}}{\sum_j e^{(u_j)}}$$

normalizes the vector u^i of length n to be an output distribution over the dictionary of inputs. Indeed the model uses u_j^i as pointers to the input symbols.

See A for the details on datasets, training process, the baseline models, and the performance of our proposed model in terms of correctness, and generalizability in comparison to existing baselines. Based on our experimental results, pointer networks performs remarkably well across multiple reasoning tasks while outperforming the previously reported state of the art by a significant margin. We observe that the Pointer Networks preserve their performance even when challenged with knowledge graphs of the domain/vocabulary it has never encountered before.

Chapter 4

On the Generalization Capability of Memory Networks for RDF Reasoning

4.1 Overview

With the recent revival of interest in artificial neural networks, they have been applied vastly for the completion of KBs. These methods [49, 53, 63, 14, 73, 66, 67] heavily rely on the subsymbolic representation of entities and relations learned through maximization of a scoring objective function over valid factual triples. Thus, the current success of such models hinges primarily on the power of those subsymbolic continuous real-valued representations in encoding the similarity/relatedness of entities and relations. Recent attempts have focused on neural multi-hop reasoners [48, 51, 18, 70, 62] to equip the model to deal with more complex reasoning. More recently, a Neural Theorem Prover [57] has been proposed in an attempt to take advantage of both symbolic and sub-symbolic reasoning.

Despite their success, the main restriction common to neural reasoners is that they are unable to generalize to new domains. This inherent limitation follows from both the representation functions used and the learning process. The major issue comes from the mere

reliance of these models on the representation of entities learned during the training or in the pre-training phase stored in a lookup table. Consequently, these models have difficulty to deal with out-of-vocabulary(OOV) entities. Although the small-scale OOV problem has been addressed in part in the natural language processing (NLP) domain by taking advantage of character-level embedding [43], learning embeddings on the fly by leveraging text descriptions or spelling [4], copy mechanism [27] or pointer networks [52], still these solutions are insufficient for transferring purposes. [65] shows the success of natural language inference (NLI) methods is heavily benchmark specific. An even greater source of concern is that reasoning in most of the above sub-symbolic approaches hinges more on the notion of similarity and geometric-based proximity of real-valued vectors (induction) as opposed to performing transitive reasoning (deduction) over them. In short, to the best of our knowledge, to date, there is no sub-symbolic reasoning work, which is able to transfer the learning capability from one KB to unseen one. In fact, since previous works have focused to conduct reasoning on the unseen part of the same KB, they have tried to gain generalization ability through induction and robustness to missing edges[30] as opposed to deduction. Likewise, recent years have seen some progress in zero-shot relation learning in sub-symbolic reasoning domain[48, 72, 58]. Zero-shot learning refers to the ability of the model to infer new relations where that relation has not been seen before in training set[12]. This generalization capability is still quite limited and fundamentally different from our work in terms of both methodology and purpose.

4.2 Contributions

[22] Monireh Ebrahimi, Aaron Eberhart, Federico Bianchi, and Pascal Hitzler. Towards bridging the neuro-symbolic gap: Deep deductive reasoners. *Applied Intelligence*, 2021. to appear [26] Monireh Ebrahimi, Md. Kamruzzaman Sarker, Federico Bianchi, Ning Xie, Aaron Eberhart, Derek Doran, HyeongSik Kim, and Pascal Hitzler. Neuro-symbolic deductive reasoning for cross-knowledge graph entailment. In Andreas Martin, Knut Hinkelmann, Hans-Georg Fill, Aurona Gerber, Doug Lenat, Reinhard Stolle, and Frank

van Harmelen, editors, *Proceedings of the AAAI 2021 Spring Symposium on Combining Machine Learning and Knowledge Engineering (AAAI-MAKE 2021), Stanford University, Palo Alto, California, USA, March 22-24, 2021*, volume 2846 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2021 [25] Monireh Ebrahimi, Md Kamruzzaman Sarker, Federico Bianchi, Ning Xie, Aaron Eberhart, Derek Doran, and Pascal Hitzler. On the generalization capability of memory networks for reasoning. *ICML 2019 Workshop on Understanding and Improving Generalization in Deep Learning*, 2019

In the publications, *Neuro-Symbolic Deductive Reasoning for Cross-Knowledge Graph Entailment* [26] and *Towards Bridging the Neuro-Symbolic Gap: Deep Deductive Reasoners* [22], we investigate the emulation of deductive symbolic reasoning using memory networks. Memory networks [71] are a class of learning models capable of conducting multiple computational steps over an explicit memory component before returning an answer. They have been recently applied successfully to a range of NLP tasks such as question answering [64, 33], language modeling [64], and dialogue tasks [10, 20]. End-to-end memory networks (MemN2N) [64] are a less-supervised, more general version of these networks, applicable to the settings where labeled supporting memories are not available. We have selected such networks since we believe that they are a primary candidate to perform well for 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 NLI is also promising: while NLI 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 get insights into the "reasoning" underlying the network output, as we will be able to see which pieces of the input KB are taken into account at each step.

The main contribution of this dissertation, however, is a recipe involving a simple but effective KB triple normalization before learning their representation within a MemN2N. To perform logical inference in more abstract level, and thereby facilitating the transfer of reasoning expertise from one KB to another, the normalization maps entities and predicates

in a knowledge to a generic vocabulary. Facts in additional KBs are normalized using the same vocabulary, so that the network does not learn to overfit its learning to entity and predicate names in a specific KB. This emulates symbolic reasoning by neural embeddings as the actual names (as strings) of entities from the underlying logic such as variables, constants, functions, and predicates are insubstantial for logical entailment in the sense that a consistent renaming across a theory does not change the set of entailed formulas (under the same renaming). Thanks to the term-agnostic feature of our representation, we are able to create a reasoning system capable of performing reasoning over an unseen set of vocabularies in the test phase.

RDF Reasoning using Memory Networks: We wish to train a neural model that will

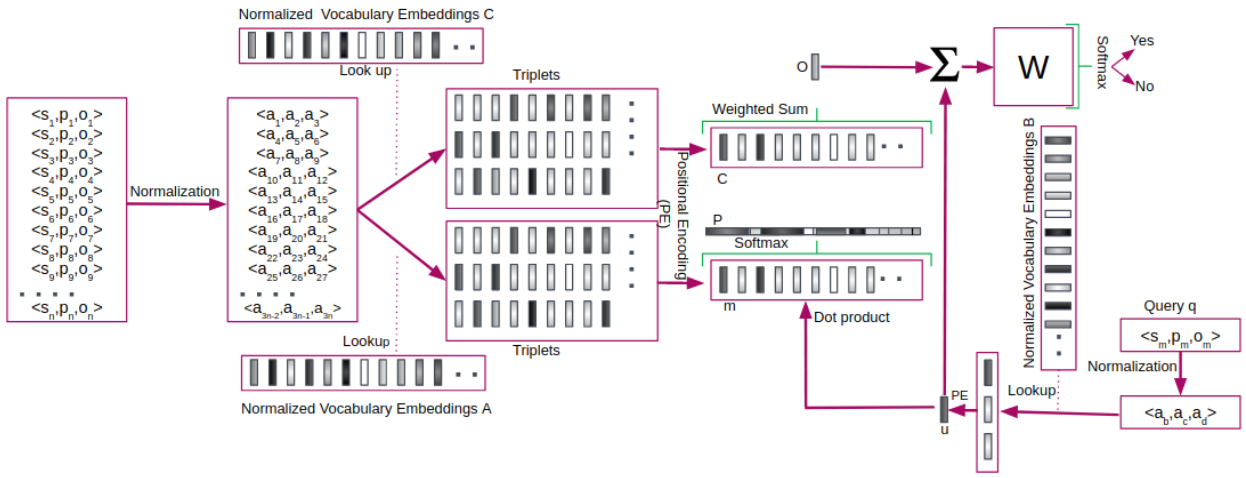

Figure 4.1: *Diagram of the proposed model, for K=1*

learn to reason over one set of theories, and can then transfer that learning to new theories over the same logic. One of the key obstacles we face with our task is to understand how to represent training and test data. To use standard neural approaches, symbols will have to be represented over the real coordinate space R as vectors (points), matrices or tensors. Many embeddings for KBs have been proposed [11, 42, 67, 69], but we are not aware of an existing embedding that captures what seems important for the deductive reasoning scenario. Indeed, the prominent use case explored for KB embeddings is not deductive in nature; rather, it concerns the problem of the discovery or suggestion of additional links or edges in the graph,

together with appropriate edge labels. In this link discovery setting, the actual labels for nodes or edges in the graph, and as such their commonsense meanings, are likely important, and most existing embeddings reflect this. However, for deductive reasoning the names of entities are insubstantial and should not be captured by an embedding. Another inherent problem in the use of such representations across KBs is the OOV problem. While a word lookup table can be initialized with vectors in an unsupervised task or during training of the reasoner, it still cannot generate vector representations for unseen terms. It is further impractical to store the vectors of all words when vocabulary size is huge [43]. Similarly, memory networks usually rely on word-level embedding lookup tables, i.e., learned with the underlying rationale that words that occur in similar supervised scenarios should be represented by similar vectors. That is why they are known to have difficulties dealing with OOV, as a word lookup table cannot provide a representation for the unseen, and thus has difficulty to do NLI over new words [4], and for us this would pose a challenge in the transfer to new KBs.

We thus need representations that are agnostic to the terms used as primitives in the KB. To build such a representation, we use syntactic normalization: a renaming of primitives from the logical symbols to a set of predefined entity names that are used across different normalized theories. By randomly assigning the mapping for the renaming, the network's learning will be based on the structural information within the theories, and not on the actual names of the primitives. Note that this normalization not only plays the role of "forgetting" irrelevant label names, but also makes it possible to transfer learning from one KB to the other. Indeed, the network can be trained with many KBs, and then subsequently tested on completely new ones.

Model Architecture We consider a model architecture that adapts the MemN2N with fundamental alterations necessary for abstract reasoning. A high-level view of our model is shown in Figure 4.1. It takes a discrete set G of normalized RDFS statements (called *triples*) $t_1, ..., t_n$ that are stored in memory, a query q, and outputs a "yes" or "no" answer to determine if q is entailed by G. Each of the normalized t_i and q contains symbols coming

from a general dictionary with V normalized words shared among all of the normalized RDFS theories in both training and test sets. The model writes all triples to the memory and then calculates a continuous embedding for G and q. Through multiple hop attention over those continuous representations, the model then classifies the query. The model is trained by back-propagation of error from output to the input through multiple memory accesses. More Specifically, the model is augmented with an external memory that stores the embeddings of the normalized triples in our KB. This memory is defined as an $n \times d$ tensor where n denotes the number of triples in the KB and d is the dimensionality of the embeddings. The KB is stored in the memory vectors from two continuous representations of m_i and c_i obtained from two input and output embedding matrices of A and C with size $d \times V$ where V is the size of vocabulary. Similarly, the query q is embedded via a matrix B to obtain an internal state u. In each reasoning step, those memory slots useful for finding the correct answers should have their contents retrieved. To enable this, we use an attention mechanism for q over memory input representations by taking an internal product followed by a softmax:

$$p_i = \text{Softmax}(u^T(m_i)) \tag{4.1}$$

Equation (4.1) calculates a probability vector p over the memory inputs, the output vector o is computed as the weighted sum of the transformed memory contents c_i with respect to their corresponding probabilities p_i by $o = \sum_i p_i c_i$. This describes the computation within a single hop. The internal state of the query vector updates for the next hop as $u^{k+1} = u^k + o^k$. The process repeats K times where K is the number of computational hops. The output of the K^{th} hop is used to predict the label $\hat{a}$ by passing o^K and u^K through a weight matrix of size $V \times d$ and a softmax:

$$\hat{a} = \text{Softmax}(W(u^{K+1})) = \text{Softmax}(W(u^k + o^k)).$$

Figure 4.1 shows the model for $K = 1$. The parameters to be learned by backpropagation are A, B, C, and W matrices.

Memory Content An RDFS KB is a collection of statements stored as triples $(e1, r, e2)$ where $e1$ and $e2$ are called *subject* and *object*, respectively, while r is a relation binding $e1$ and $e2$ together. Every entity in an RDFS KB is represented by a unique Universal Resource Identifier (URI). We normalize these triples by systematically renaming all URIs which are not in the RDF or RDFS namespaces as discussed previously. Each such URI is mapped to a set of arbitrary strings in a predefined set $\mathcal{A} = \{a_1, ..., a_n\}$, where n is taken as a training hyper-parameter giving an upper bound for the largest number of entities in a KB the system will be able to handle. Note that URIs in the RDF/RDFS namespaces are not renamed, as they are important for the deductive reasoning according to the RDFS model-theoretic semantics. Consequently, each normalized RDFS KB will be a collection of facts stored as triples $\{(a_i, a_j, a_k)\}$.

It is important to note that each symbol is mapped into an element of $\mathcal{A}$ regardless of its position in the triple. Yet the position of an element within a triple is an important feature to consider. Thus we employ a positional encoding (PE) [64] to encode the position of each element within the triple. Each memory slot thus represents the positional-weighted summation of each triplet. The PE ensures that the order of the elements now affects the encoding of each memory slot m_i.

See A for the details on datasets, training process, the baseline models, and the performance of our proposed model in terms of correctness, and generalizability in comparison to existing baselines. Our experimental results show that a deep learning architecture based on memory networks and pre-embedding normalization is capable of learning how to perform deductive reasoning over previously unseen RDFS KBs with high accuracy. We believe that we have thus provided the first deep learning approach that is capable of high accuracy RDFS deductive reasoning over previously unseen KBs. Normalization appears to be a critical component for high performance of our system. This obviates the need for supervised retraining over the task of interest or unsupervised pretraining over the external source of

data for learning the representations when encountered with a new KB.

Chapter 5

Conclusion

5.1 Summary

This thesis summarizes the author's contributions in the neuro-symbolic integration research direction. First in the thesis we evaluated the deductive reasoning capability of Pointer Networks over RDS and $\mathcal{EL}^+$logical reasoning. Then we examined the capability of memory-augmented networks in performing the RDF entailment for cross-knowledge graph deductive reasoning. We aimed to better understand the effectiveness of each model and the desirable properties expected with respect to two different logics (RDF, and $\mathcal{EL}^+$). Such understanding would help pave the way for future efforts in this research direction.

Over the course of this dissertation, we have outlined a number of contributions that have improved the process to what we see today.

First, we significantly enhanced the state-of-the-art model for RDFS and $\mathcal{EL}^+$ reasoning by developing a deep learning architecture based on pointer networks. We designed a novel way of conducting neuro-symbolic through pointing to the input elements. More importantly we showed that the proposed approach is generalizable across new domain and vocabulary demonstrating symbol-invariant zero-shot reasoning capability.

Next, We have demonstrated that a deep learning architecture based on memory networks and pre-embedding normalization is capable of learning how to perform deductive reasoning

over previously unseen RDFS KBs with high accuracy. We believe that we have thus provided the first deep learning approach that is capable of high accuracy RDFS deductive reasoning over previously unseen KBs. Normalization appears to be a critical component for high performance of our system. This obviates the need for supervised retraining over the task of interest or unsupervised pretraining over the external source of data for learning the representations when encountered with a new KB. It also provides insights into representation learning for rare or OOV words, transfer learning, zero-shot learning, and domain adaptation in the reasoning domain.

Altogether, these research topics, resulted in first neural deductive reasoners which are capable of high accuracy RDFS and $\mathcal{EL}^+$ deductive reasoning over previously unseen knowledge bases.

5.2 Future Work

There is an immediate next step to address.

1. How can the models introduced in this research be leveraged to different logics such as W3C Standard OWL RL, Datalog, $\mathcal{ALC}$, and $\mathcal{SROIQ}$ for conducting deductive neuro-symbolic reasoning?

   With respect to conducting the reasoning over the other logics, we envision that employing the Pointer Networks, we are able to perform the reasoning and expanding our findings on capabilities and limitations of such networks by gaining insights from their behaviour when encountered with the new logical reasoning tasks.

2. We plan to properly investigate scalability of our approaches and their robustness to the noise.

3. There is a plethora of different deep learning approaches that could be investigated for neuro-symbolic dedcutive reasoning. We furthermore intend to investigate the capability and limitations of other neural networks model such as Hierarchical Pointer Memory

Networks, Mem2Seq, Graph Neural Networks, and TPR-Transformer for conducting logical reasoning in terms of correctness, generalizability, scalability, and robustness to the noise.

Results from this dissertation have a high potential for impact way beyond the specific deep learning field. Ways to overcome the symbolic-subsymbolic divide are considered to be of central interest in Cognitive Science [7]. Ways to overcome this divide furthermore promise to provide pathways for best-of-both-world scenarios in AI applications, combining the speed and robustness of deep learning approaches with the transparency of logic-based systems. As such, outcomes of this dissertation also have an impact on studies of trust and explainability of deep learning approaches. Broader impact of the results thus lies potentially in all application areas of deep learning and logic-based expert systems – the former are currently heavily being investigated in research and industry, and the latter are well-established in applications, with known limitations which bridging the symbolic-subsymbolic divide promises to overcome.

We hope our impressive results on these reasoning problems will encourage broader exploration of pointer networks and other neural network architectures' capabilities for conducting reasoning over more complex logics and for other neuro-symbolic problems especially in terms of their generalization power.

Bibliography

[1] Masataro Asai and Alex Fukunaga. Classical planning in deep latent space: Bridging the subsymbolic-symbolic boundary. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018*. AAAI Press, 2018.

[2] Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the EL envelope. In Leslie Pack Kaelbling and Alessandro Saffiotti, editors, *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30 - August 5, 2005*, pages 364–369. Professional Book Center, 2005.

[3] Samy Badreddine, Artur d'Avila Garcez, Luciano Serafini, and Michael Spranger. Logic tensor networks. *CoRR*, abs/2012.13635, 2020.

[4] Dzmitry Bahdanau, Tom Bosc, Stanislaw Jastrzebski, Edward Grefenstette, Pascal Vincent, and Yoshua Bengio. Learning to compute word embeddings on the fly. *CoRR*, abs/1706.00286, 2017.

[5] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[6] Tarek R. Besold, Artur S. d'Avila Garcez, Sebastian Bader, Howard Bowman, Pedro M. Domingos, Pascal Hitzler, Kai-Uwe Kühnberger, Luís C. Lamb, Daniel Lowd, Priscila Machado Vieira Lima, Leo de Penning, Gadi Pinkas, Hoifung Poon, and Gerson Zaverucha. Neural-symbolic learning and reasoning: A survey and interpretation. *CoRR*, abs/1711.03902, 2017.

[7] Tarek R. Besold, Artur S. d'Avila Garcez, Sebastian Bader, Howard Bowman, Pedro M. Domingos, Pascal Hitzler, Kai-Uwe Kühnberger, Luís C. Lamb, Daniel Lowd, Priscila Machado Vieira Lima, Leo de Penning, Gadi Pinkas, Hoifung Poon, and Gerson Zaverucha. Neural-symbolic learning and reasoning: A survey and interpretation. *CoRR*, abs/1711.03902, 2017.

[8] Tarek R Besold, Artur d'Avila Garcez, Sebastian Bader, Howard Bowman, Pedro Domingos, Pascal Hitzler, Kai-Uwe Kühnberger, Luis C Lamb, Daniel Lowd, Priscila Machado Vieira Lima, et al. Neural-symbolic learning and reasoning: A survey and interpretation. *arXiv preprint arXiv:1711.03902*, 2017.

[9] Federico Bianchi and Pascal Hitzler. On the capabilities of logic tensor networks for deductive reasoning. In Andreas Martin, Knut Hinkelmann, Aurona Gerber, Doug Lenat, Frank van Harmelen, and Peter Clark, editors, *Proceedings of the AAAI 2019 Spring Symposium on Combining Machine Learning with Knowledge Engineering (AAAI-MAKE 2019) Stanford University, Palo Alto, California, USA, March 25-27, 2019., Stanford University, Palo Alto, California, USA, March 25-27, 2019*, volume 2350 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2019.

[10] Antoine Bordes, Y-Lan Boureau, and Jason Weston. Learning end-to-end goal-oriented dialog. *arXiv preprint arXiv:1605.07683*, 2016.

[11] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795, 2013.

[12] Antoine Bordes, Jason Weston, Ronan Collobert, Yoshua Bengio, et al. Learning structured embeddings of knowledge bases. In *AAAI*, volume 6, page 6, 2011.

[13] Dan Brickley and R.V. Guha, editors. *RDF Schema 1.1.* "", W3C Recommendation 25 February 2014. Available from http://www.w3.org/TR/rdf-schema/.

[14] Kai-Wei Chang, Scott Wen-tau Yih, Bishan Yang, and Chris Meek. Typed tensor decomposition of knowledge bases for relation extraction. 2014.

[15] Roberto Confalonieri, Tarek R. Besold, Tillman Weyde, Kathleen Creel, Tania Lombrozo, Shane T. Mueller, and Patrick Shafto. What makes a good explanation? cognitive dimensions of explaining intelligent machines. In Ashok K. Goel, Colleen M. Seifert, and Christian Freksa, editors, *Proceedings of the 41th Annual Meeting of the Cognitive Science Society, CogSci 2019: Creativity + Cognition + Computation, Montreal, Canada, July 24-27, 2019*, pages 25–26. cognitivesciencesociety.org, 2019.

[16] World Wide Web Consortium et al. Rdf 1.1 semantics. *empty*, 2014.

[17] Richard Cyganiak, David Wood, and Markus Lanthaler, editors. *RDF 1.1 Concepts and Abstract Syntax*. W3C Recommendation 25 February 2014, 2014. Available from http://www.w3.org/TR/rdf11-concepts/.

[18] Rajarshi Das, Arvind Neelakantan, David Belanger, and Andrew McCallum. Chains of reasoning over entities, relations, and text using recurrent neural networks. *arXiv preprint arXiv:1607.01426*, 2016.

[19] Artur S. d'Avila Garcez, Tarek R. Besold, Luc De Raedt, Peter Földiák, Pascal Hitzler, Thomas Icard, Kai-Uwe Kühnberger, Luís C. Lamb, Risto Miikkulainen, and Daniel L. Silver. Neural-symbolic learning and reasoning: Contributions and challenges. In *2015 AAAI Spring Symposia, Stanford University, Palo Alto, California, USA, March 22-25, 2015*. AAAI Press, 2015.

[20] Jesse Dodge, Andreea Gane, Xiang Zhang, Antoine Bordes, Sumit Chopra, Alexander Miller, Arthur Szlam, and Jason Weston. Evaluating prerequisite qualities for learning end-to-end dialog systems. *arXiv preprint arXiv:1511.06931*, 2015.

[21] Aaron Eberhart, Monireh Ebrahimi, Lu Zhou, Cogan Shimizu, and Pascal Hitzler. Completion reasoning emulation for the description logic EL+. In Andreas Martin, Knut Hinkelmann, Hans-Georg Fill, Aurona Gerber, Doug Lenat, Reinhard Stolle, and Frank

van Harmelen, editors, *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*, volume 2600 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2020.

[22] Monireh Ebrahimi, Aaron Eberhart, Federico Bianchi, and Pascal Hitzler. Towards bridging the neuro-symbolic gap: Deep deductive reasoners. *Applied Intelligence*, 2021. to appear.

[23] Monireh Ebrahimi, Aaron Eberhart, Federico Bianchi, and Pascal Hitzler. Towards bridging the neuro-symbolic gap: Deep deductive reasoners. *Applied Intelligence*, pages 1–23, 2021.

[24] Monireh Ebrahimi, Aaron Eberhart, and Pascal Hitzler. On the capabilities of pointer networks for deep deductive reasoning. *arXiv preprint arXiv:2106.09225*, 2021.

[25] Monireh Ebrahimi, Md Kamruzzaman Sarker, Federico Bianchi, Ning Xie, Aaron Eberhart, Derek Doran, and Pascal Hitzler. On the generalization capability of memory networks for reasoning. *ICML 2019 Workshop on Understanding and Improving Generalization in Deep Learning*, 2019.

[26] Monireh Ebrahimi, Md. Kamruzzaman Sarker, Federico Bianchi, Ning Xie, Aaron Eberhart, Derek Doran, HyeongSik Kim, and Pascal Hitzler. Neuro-symbolic deductive reasoning for cross-knowledge graph entailment. In Andreas Martin, Knut Hinkelmann, Hans-Georg Fill, Aurona Gerber, Doug Lenat, Reinhard Stolle, and Frank van Harmelen, editors, *Proceedings of the AAAI 2021 Spring Symposium on Combining Machine Learning and Knowledge Engineering (AAAI-MAKE 2021), Stanford University, Palo Alto, California, USA, March 22-24, 2021*, volume 2846 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2021.

[27] Mihail Eric and Christopher D Manning. A copy-augmented sequence-to-sequence architecture gives good performance on task-oriented dialogue. *arXiv preprint arXiv:1701.04024*, 2017.

[28] Daniel Fernández-González and Carlos Gómez-Rodríguez. Left-to-right dependency parsing with pointer networks. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 710–716. Association for Computational Linguistics, 2019.

[29] Daniel Fernández-González and Carlos Gómez-Rodríguez. Transition-based semantic dependency parsing with pointer networks. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7035–7046. Association for Computational Linguistics, 2020.

[30] Kelvin Guu, John Miller, and Percy Liang. Traversing knowledge graphs in vector space. *arXiv preprint arXiv:1506.01094*, 2015.

[31] Barbara Hammer and Pascal Hitzler, editors. *Perspectives of Neural-Symbolic Integration*, volume 77 of *Studies in Computational Intelligence*. Springer, 2007.

[32] Patrick J. Hayes and Peter F. Patel-Schneider, editors. *RDF 1.1 Semantics*. "", W3C Recommendation 25 February 2014. Available from http://www.w3.org/TR/rdf11-mt/.

[33] Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. The goldilocks principle: Reading children's books with explicit memory representations. *arXiv preprint arXiv:1511.02301*, 2015.

[34] Pascal Hitzler. A review of the semantic web field. *Communications of the ACM*, 64(2):76–83, 2021.

[35] Pascal Hitzler, Markus Krötzsch, Bijan Parsia, Peter F. Patel-Schneider, and Sebastian Rudolph, editors. *OWL 2 Web Ontology Language: Primer (Second Edition)*. W3C Recommendation 11 December 2012, 2012. Available from http://www.w3.org/TR/owl2-primer/.

[36] Pascal Hitzler, Markus Krötzsch, and Sebastian Rudolph. *Foundations of Semantic Web Technologies*. Chapman & Hall/CRC, 2010.

[37] Pascal Hitzler and Anthony Karel Seda. *Mathematical Aspects of Logic Programming Semantics*. Chapman and Hall / CRC studies in informatics series. CRC Press, 2011.

[38] Patrick Hohenecker and Thomas Lukasiewicz. Ontology reasoning with deep neural networks. *J. Artif. Intell. Res.*, 68:503–540, 2020.

[39] Aishwarya Jadhav and Vaibhav Rajan. Extractive summarization with SWAP-NET: sentences and words from alternating pointer networks. In Iryna Gurevych and Yusuke Miyao, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 142–151. Association for Computational Linguistics, 2018.

[40] Markus Krötzsch. Efficient rule-based inferencing for OWL EL. In Toby Walsh, editor, *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, pages 2668–2673. IJCAI/AAAI, 2011.

[41] Jian Li, Yue Wang, Michael R. Lyu, and Irwin King. Code completion with neural attention and pointer networks. In Jérôme Lang, editor, *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 4159–4165. ijcai.org, 2018.

[42] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, volume 15, pages 2181–2187, 2015.

[43] Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramón Fermandez, Silvio Amir, Luis Marujo, and Tiago Luís. 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*, pages 1520–1530, 2015.

[44] Xuezhe Ma, Zecong Hu, Jingzhou Liu, Nanyun Peng, Graham Neubig, and Eduard H. Hovy. Stack-pointer networks for dependency parsing. In Iryna Gurevych and Yusuke Miyao, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 1403–1414. Association for Computational Linguistics, 2018.

[45] Bassem Makni and James A. Hendler. Deep learning for noise-tolerant RDFS reasoning. *Semantic Web*, 10(5):823–862, 2019.

[46] Pasquale Minervini, Matko Bosnjak, Tim Rocktäschel, Sebastian Riedel, and Edward Grefenstette. Differentiable reasoning on large knowledge bases and natural language. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 5182–5190. AAAI Press, 2020.

[47] Pasquale Minervini, Sebastian Riedel, Pontus Stenetorp, Edward Grefenstette, and Tim Rocktäschel. Learning reasoning strategies in end-to-end differentiable proving. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 6938–6949. PMLR, 2020.

[48] Arvind Neelakantan, Benjamin Roth, and Andrew McCallum. 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)*, pages 156–166, 2015.

[49] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. Factorizing yago: scalable machine learning for linked data. In *Proceedings of the 21st international conference on World Wide Web*, pages 271–280. ACM, 2012.

[50] Heiko Paulheim. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic Web*, 8(3):489–508, 2017.

[51] Baolin Peng, Zhengdong Lu, Hang Li, and Kam-Fai Wong. Towards neural network-based reasoning. *arXiv preprint arXiv:1508.05508*, 2015.

[52] Dinesh Raghu, Nikhil Gupta, et al. Hierarchical pointer memory network for task oriented dialogue. *arXiv preprint arXiv:1805.01216*, 2018.

[53] Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. Relation extraction with matrix factorization and universal schemas. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 74–84, 2013.

[54] Laurens Rietveld, Wouter Beek, Rinke Hoekstra, and Stefan Schlobach. Meta-data for a lot of LOD. *Semantic Web*, 8(6):1067–1080, 2017.

[55] Petar Ristoski, Jessica Rosati, Tommaso Di Noia, Renato De Leone, and Heiko Paulheim. Rdf2vec: RDF graph embeddings and their applications. *Semantic Web*, 10(4):721–752, 2019.

[56] Tim Rocktäschel and Sebastian Riedel. Learning knowledge base inference with neural theorem provers. In Jay Pujara, Tim Rocktäschel, Danqi Chen, and Sameer Singh, editors, *Proceedings of the 5th Workshop on Automated Knowledge Base Construction, AKBC@NAACL-HLT 2016, San Diego, CA, USA, June 17, 2016*, pages 45–50. The Association for Computer Linguistics, 2016.

[57] Tim Rocktäschel and Sebastian Riedel. End-to-end differentiable proving. In *Advances in Neural Information Processing Systems*, pages 3788–3800, 2017.

[58] Tim Rocktäschel, Sameer Singh, and Sebastian Riedel. 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*, pages 1119–1129, 2015.

[59] Stefan Schulz, Boontawee Suntisrivaraporn, Franz Baader, and Martin Boeker. SNOMED reaching its adolescence: Ontologists' and logicians' health check. *I. J. Medical Informatics*, 78(Supplement-1):S86–S94, 2009.

[60] Abigail See, Peter J. Liu, and Christopher D. Manning. Get to the point: Summarization with pointer-generator networks. In Regina Barzilay and Min-Yen Kan, editors, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1073–1083. Association for Computational Linguistics, 2017.

[61] Luciano Serafini and Artur S d'Avila Garcez. Learning and reasoning with logic tensor networks. In *Conference of the Italian Association for Artificial Intelligence*, pages 334–348. Springer, 2016.

[62] Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. Reasonet: Learning to stop reading in machine comprehension. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1047–1055. ACM, 2017.

[63] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. Reasoning with neural tensor networks for knowledge base completion. In *Advances in neural information processing systems*, pages 926–934, 2013.

[64] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448, 2015.

[65] Aarne Talman and Stergios Chatzikyriakidis. Testing the generalization power of neural network models across nli benchmarks. *arXiv preprint arXiv:1810.09774*, 2018.

[66] Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. Representing text for joint embedding of text and knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1509, 2015.

[67] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *International Conference on Machine Learning*, pages 2071–2080, 2016.

[68] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2692–2700, 2015.

[69] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *AAAI*, volume 14, pages 1112–1119, 2014.

[70] Dirk Weissenborn. Separating answers from queries for neural reading comprehension. *arXiv preprint arXiv:1607.03316*, 2016.

[71] Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. corr abs/1410.3916, 2014.

[72] Wenhan Xiong, Thien Hoang, and William Yang Wang. Deeppath: A reinforcement learning method for knowledge graph reasoning. *arXiv preprint arXiv:1707.06690*, 2017.

[73] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*, 2014.

Appendix A

Contributions

In the following pages, all contributions made for this dissertation are listed in order of appearance in this document.

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

On the Capabilities of Pointer Networks for Deep Deductive Reasoning

Monireh Ebrahimi [*1] · **Aaron Eberhart**[1] · **Pascal Hitzler**[1]

Received: date / Accepted: date

Abstract The importance of building neural networks that can learn to reason has been well recognized in the neuro-symbolic community. In this paper, we apply neural pointer networks for conducting reasoning over symbolic knowledge bases. In doing so, we explore the benefits and limitations of encoder-decoder architectures in general and pointer networks in particular for developing accurate, generalizable and robust neuro-symbolic reasoners. Based on our experimental results, pointer networks performs remarkably well across multiple reasoning tasks while outperforming the previously reported state of the art by a significant margin. We observe that the Pointer Networks preserve their performance even when challenged with knowledge graphs of the domain/vocabulary it has never encountered before. To the best of our knowledge, this is the first study on neuro-symbolic reasoning using Pointer Networks. We hope our impressive results on these reasoning problems will encourage broader exploration of pointer networks' capabilities for reasoning over more complex logics and for other neuro-symbolic problems.

Keywords Neuro-symbolic reasoning · Pointer networks · Transformers · RDF reasoning · $\mathcal{EL}^+$ reasoning

Monireh Ebrahimi
E-mail: monireh@ksu.edu

Aaron Eberhart
E-mail: aaroneberhart@ksu.edu

Pascal Hitzler
E-mail: hitzler@ksu.edu

[*]Corresponding Author
[1]Department of Computer Science at Kansas State University

1 Introduction

The study of architectures and methods for artificial neural networks so that they can learn and perform tasks from the realm of logic-based knowledge representation and reasoning has a long-standing tradition [6]. This research area is sometimes referred to as "neuro-symbolic integration" (or "neural-symbolic integration") and there are at least two primary rationales that can be found in the literature on the subject. The first is the desire to arrive at systems that combine the robustness and trainability of artificial neural networks with the transparency and interpretability of knowledge-based systems, while at the same time making use of structured background knowledge. The second rationale is more prevalent in cognitive science and lies in addressing the fundamental gap between symbolic and subsymbolic representation and processing, based on the observation that humans perceive much of their own thinking, introspectively, as symbolic, while the physical structure of the brain gives rise to artificial neural networks as a mathematical and computational abstraction.

Many of the earlier lines of research on neuro-symbolic integration, discussed primarily from a cognitive science perspective, can be found in [6]. Of particular interest is the integration of deep learning with logics that are not propositional in nature, since propositional logic is of limited applicability to knowledge representation and reasoning tasks. In the wake of deep learning breakthroughs, fundamental issues around neuro-symbolic integration have recently received increased attention with some progress being made as new approaches emerge. In particular, there has been progress in developing neural networks that can learn to reason. These include the Neural Theorem Prover (NTP) and its variations [43,42,36,35], Logic Tensor Networks (LTN) [47,7, 3], and the application of memory networks and LSTMs [13] and others [34, 27]. Yet, there is still much work to do in terms of new model development and investigation of inductive bias in existing architectures and their reasoning capability.

This paper tries to fill this gap by examining the reasoning capability of pointer networks [49] in emulating deductive reasoning. Pointer Networks and their variations have been applied successfully to a variety of sophisticated tasks including theoretical computer science problems (i.e., NP-hard Travelling Salesman Problem (TSP), Delaunay Triangulation, and Convex Hull) [49] and practical problems like abstractive [45] and extractive [28] text summarization, code completion [31], and dependency parsing [17,16,33]. Nevertheless, almost nothing is known about their potential for conducting logical deductive reasoning accurately. In fact, they have been mainly used for solving discrete combinatorial optimization problems because of their variable-size output vocabulary and for resolving the rare or out-of-vocabulary problem in Natural Language Processing. We hypothesize that using the pointer attention to decide what elements of the input knowledge base should be chosen as the output, and in which order, will work well for deductive reasoning tasks. Indeed, using pointer networks we can mimic human reasoning behaviour where

one can learn to choose a set of symbols in different locations and copy these symbols to suitable locations to generate new logical consequences based on a set of predefined logical entailment rules. To verify this, here, we explore the capabilities and limitations of pointer networks for performing deductive reasoning on Resource Description Framework (RDF) [8] and $\mathcal{EL}^+$ [2] knowledge bases in terms of accuracy, and generalizability.Based on our experimental results, pointer networks perform remarkably well across multiple reasoning tasks while outperforming the previously reported state of the art by a significant margin. We observe that the Pointer Networks preserve their performance even when challenged with knowledge graphs of the domain/vocabulary it has never encountered before. To our knowledge, this work is the first attempt to reveal the impressive power of pointer networks for conducting deductive reasoning.

In terms of the logic, in this paper we are looking at two logics with different expressivity, power, and reasoning difficulty. The Resource Description Framework Schema (RDFS) [8] is non-trivial (and non-propositional), yet one of the simplest widely used logics: It is a mature W3C Semantic Web standard that is commonly used to express knowledge graphs and linked data [23], and many corresponding data sets are freely available on the World Wide Web [40]. The standard carries a model-theoretic semantics which defines deductive entailment [22, Section 9.2], and reasoning over RDFS is usually done using rule-based reasoning engines. The second logic is the description logic $\mathcal{EL}^+$ (or $\mathcal{ER}$) [2] that is the basis for the W3C standard OWL EL [24]. It is considered to be a rather inexpressive but practically useful logic [44] and generally used for expressing ontologies and knowledge graph schemas [25] particularly in medical domain ontologies.

In short this paper strives to answer two main research questions: "Can Pointer Networks perform logical deductive reasoning using pointer attention?", and more generally, "Can other attention-based sequence-to-sequence models like self-attention based popular Transformer architectures successfully perform the same task?","How well do pointer network reasoners perform on completely new knowledge graphs?", and finally, "How robust is our model to noise?". To answer these questions we conduct a set of experiments by applying pointer networks and transformers to RDFS and $\mathcal{EL}^+$ reasoning tasks. We believe, the answer to our third question is particularly very important since it a very big step toward developing accurate yet symbol-invariant deep deductive reasoners which generalize very well on unseen knowledge bases of differing domain or vocabulary. The contributions of this work are fourfold:

1. A novel paradigm for viewing a symbolic reasoning problem as a pointing problem.
2. Pointer Networks are used to neurally resolve symbolic reasoning for the first time.
3. The proposed approach is able to transfer its reasoning ability to new domain/vocabulary knowledge graph of same logic.
4. We report the state-of-the-art performance of the $\mathcal{EL}^+$ and RDF reasoning.

The remainder of the paper is organized as follows. In Section 2 we discuss related research efforts, more precisely an overview of recent work on deep deductive reasoning over RDFS, $\mathcal{EL}^+$, and other logics, followed by a list of various tasks where pointer networks have been effectively applied. In Section 3, we concretely present the deep learning architecture and the logics we used. In Sections 4, we present an experimental evaluation of our approach and discuss our findings. We conclude and discuss future work in Section 5.

2 Related Work

2.1 Deep Deductive Reasoning

Training artificial neural networks to learn deductive reasoning is a hard machine learning task that was out of reach before the advent of deep learning. In the last few years, several publications have shown that deep deductive reasoning – using deep learning methods – is possible. We will briefly review the core body of existing work. As we will see, it remains a hard task, even for deep learning.

Before we do so, though, let us point out that our work is different from what is usually called knowledge graph completion, or the study of knowledge graph embeddings, although we deal with logics relevant for knowledge graphs [23]: Knowledge graph completion (sometimes called link prediction or knowledge graph refinement) [39] refers to enriching a knowledge graph with additional relationships that are statistically induced, sometimes using machine learning methods. In contrast to this, we are studying *deductive* reasoning, which is not based on statistics or likelihood, but based on a mathematical, logical calculus that derives additional statements which were already implicit – in a mathematically precisely defined sense – in the statements already made. Deductive inference tasks are usually hard computationally (e.g., for propositional logic, it is NP-complete), and are traditionally addressed using complex but provably correct algorithms – correctness in this sense is in relation to the underlying mathematical definitions that determine what is, and what is not, a deductive logical consequence. The study of knowledge graph embeddings [41], in isolation, is about the learning of representations of knowledge graphs in multi-dimensional Euclidean space. While embeddings are often a component of deep deductive reasoning systems, our goal is the overall functionality of deep deductive reasoning, and not just knowledge graph embeddings in isolation.

A good overview of existing deep deductive reasoning work is [13]. It appears to be appropriate to distinguish between the different logics that are addressed in the literature, the reasonable assumption being that less complex logics are easier to learn, and this resonates with the as yet limited body of work. We refer to [25] for background on all the mentioned logics. We know about investigations of RDFs [14,34], of $\mathcal{EL}^+$[12], of OWL RL [27], and of first-order predicate logic (FOL) [7].

paper	logic	transfer	generative	scale	performance
[14]	RDFS	yes	no	moderate	high
[34]	RDFS	no	yes	low	high
[12]	$\mathcal{EL}^+$	yes	yes	moderate	low
[27]	OWL RL	no*	no	low	high
[7]	FOL	no	yes	very low	high

Table 1 Overview of published deep deductive reasoning work. See the main text for details on the columns. no* indicates that the paper claims that transfer is possible in principle, but it was not demonstrated or evaluated to what extent transfer really happens.

We give an overview of the key aspects of each of these in Table 1 – we admit that some interpretations in this table may be somewhat subjective. The column "transfer" indicates whether the system was demonstrated to have a good transfer capability to previously unknown and very different knowledge bases. The column "generative" indicates whether the system generates all (under certain finiteness constraints) deductive inferences in one run – if not, then it would usually be query-based, i.e. it would be able to tell whether a given logical expression is a logical consequence of the knowledge base. The column "scale" indicates how large the input knowledge bases were in the experiments, ranging from a few logical statements as in the FOL case to RDF graphs with 1,000 triples in [14]. The column "performance" indicates how well the system learned to reason; "high" indicates 70% or more in terms of f-measure, while "low" indicates values just a bit better than random guessing.

For a deep deductive reasoner, we would ideally like to have it on an expressive logic, with transfer, generative, at massive scale, and with high performance. For all the referenced works, except the FOL one, the scale aspect has not been systematically explored yet; for the FOL case, it does look rather unfavorable as discussed in [7]. Otherwise, it is important to note that only one of the works is both generative and able to transfer, however this was also the system with very low performance. As we will see, our new approach we report on in this paper is able to do transfer and is generative, with high performance. This is the key contribution of this paper.

2.2 Pointer Networks

Pointer Networks and their variations have been applied successfully to a variety of sophisticated tasks including theoretical computer science problems (i.e., NP-hard Travelling Salesman Problem (TSP), Delaunay Triangulation, and Convex Hull [49] as well as 0–1 Knapsack problem [19]) and practical problems like abstractive [45] and extractive [28] text summarization, code completion [31], dependency parsing [17, 16, 33], named entity boundary detection [32], conversation disentanglement[50], anaphora resolution [30], paragraph ordering [38], paraphrase generation for data augmentation [21], entity linking [5], and airline itinerary prediction [37]. Nevertheless, almost nothing is known about their possible application and ability for conducting logical

deductive reasoning accurately. In fact, they have been mainly used for solving discrete combinatorial optimization problems because of their variable-size output vocabulary and for resolving the rare or out-of-vocabulary problem in Natural Language Processing.

3 Methodology

In order to explain more formally what we are setting out to do, let us first reframe our entailment problem as an input-output mapping task: Given some logic $\mathcal{L}$, for each theory T over $\mathcal{L}$, the set $c(T) = \{F \mid T \models_{\mathcal{L}} F\}$ of all formulas over $\mathcal{L}$ that are entailed by T; we call $c(T)$ the *completion* of T. We can then attempt to train a neural network to produce $c(T)$ for any given T over $\mathcal{L}$, i.e., we would use pairs $(T, c(T))$ as input-output training pairs for a generative deep deductive reasoner.

3.1 Logics

RDF The Resource Description Framework RDF, which includes RDF Schema (RDFS) [11, 25] is an established and widely used W3C standard for expressing knowledge graphs. The standard comes with a formal semantics[1] that defines an entailment relation. An RDFS knowledge base (KB) is a collection of statements stored as *triples* $(e1, r, e2)$ where $e1$ and $e2$ are called *subject* and *object*, respectively, while r is a binary relation between $e1$ and $e2$. In the context of RDF/RDFS, the triple notation $(e1, r, e2)$ is more common than a notation like $r(e1, e2)$ as it is suggestive of a node-edge-node piece of a labelled graph, and so we will use the triple notation.

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 [10], fixed across all knowledge graphs, which are expressible using Datalog.[2] These thirteen rules can be used to entail new facts.

Table 2 Selected RDFS Completion Rules

$$(x, \text{rdfs:subClassOf}, y), (y, \text{rdfs:subClassOf}, z) \models (x, \text{rdfs:subClassOf}, z) \quad (1)$$

$$(x, \text{rdfs:subPropertyOf}, y), (y, \text{rdfs:subPropertyOf}, z) \models (x, \text{rdfs:subPropertyOf}, z) \quad (2)$$

$$(x, \text{rdfs:subClassOf}, y), (z, \text{rdf:type}, x) \models (z, \text{rdf:type}, y) \quad (3)$$

$$(a, \text{rdfs:domain}, x), (y, a, z) \models (y, \text{rdf:type}, x) \quad (4)$$

$$(a, \text{rdfs:range}, x), (y, a, z) \models (z, \text{rdf:type}, x) \quad (5)$$

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

[2] Datalog is equivalent to function-free definite logic programming [26].

Table 2 shows examples for some of these entailment rules. The identifiers x, y, z, a are variables. The remaining elements of the triples are pre-fixed with the rdfs or rdf namespace (a concept borrowed from XML) and carry a specific meaning in the formal semantics of RDFS. E.g., rdfs:subClassOf indicates a sub-class (or sub-set) relationship, i.e. Rule 1 states transitivity of the rdf:subClassOf binary relation. Likewise, in Rule 2, $(x, \text{rdfs:subPropertyOf}, y)$ indicates that x, y are to be understood as binary relations, where x is a restriction (called a *subproperty*) of y. In Rule 3, the triple $(z, \text{rdf:type}, x)$ indicates that z is a member of the class (or set) x. In Rules 4 and 5, rdfs:domain and rdfs:range indicate domain respectively range of a, which is to be interpreted as a binary relation. The rules are applied exhaustively on an input RDF knowledge base, i.e. inferred triples are added and then rule execution continues taking the new triples also into account.

$\mathcal{EL}^+$ The standard reasoning task over $\mathcal{EL}^+$ is called *classification* and can be understood as the computation of all formulas of the form $\forall x(p(x) \rightarrow q(x))$ entailed by the given theory, and the set of all these formulas, which is called the *completion* of the input theory, is finite if the input theory is finite.

Formally, let N_C be a set of atomic classes (or concepts, or class names), let N_R be a set of roles (or properties), and let N_I be a set of individuals. Complex class expressions (or simply complex classes or classes) in the description logic $\mathcal{EL}^+$ are defined by the grammar

$$C ::= A | C_1 \sqcap C_2 | \exists R.C,$$

where $A \in N_C$, $R \in N_R$, and C_1, C_2, and C are complex class expressions. A TBox in $\mathcal{EL}^+$ is a set of general class inclusion axioms (or TBox statements) of the form $C \sqsubseteq D$, where C, D are (complex) classes. We use $C \equiv D$ as abbreviation for the two statements $C \sqsubseteq D$ and $D \sqsubseteq C$. An RBox in $\mathcal{EL}^+$ is a set of general role inclusion axioms (or RBox statements) of the form $R_1 \circ \cdots \circ R_n \sqsubseteq R$, where R, $R_i \in N_R$ (for all i). An $\mathcal{EL}^+$ knowledge base (or ontology) is a set of TBox and RBox statements.

$\mathcal{EL}^+$ is in fact a fragment of first-order predicate logic: all statements can be translated into it, and the inherited semantics is exactly the first-order predicte logic semantics – details can be found in [25]. Classification is known to be P-complete.

An $\mathcal{EL}^+$ *normal form* knowledge base contains only axioms of the following forms.

$$\begin{array}{ccc} C \sqsubseteq D & C_1 \sqcap C_2 \sqsubseteq D & C \sqsubseteq \exists R.D \\ \exists R.C \sqsubseteq D & R_1 \sqsubseteq R & R_1 \circ R_2 \sqsubseteq R \end{array}$$

As usual, every $\mathcal{EL}^+$ knowledge base can be cast into normal form in polynomial time, and such that it suffices to perform classification over the normal form knowledge base.

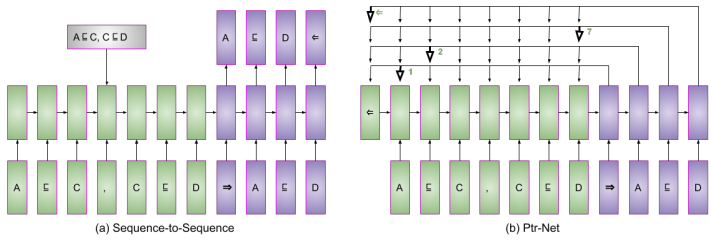

Fig. 1 $\mathcal{EL}^+$ Completion Rule (1): "(a) Sequence-to-Sequence - An RNN (green) processes the input sequence to create a code vector that is used to generate the output sequence (purple) using the probability chain rule and another RNN. (b) Ptr-Net - An encoding RNN converts the input sequence to a code (green) that is fed to the generating network (purple). At each step, the generating network produces a vector that modulates a content-based attention mechanism over inputs. The output of the attention mechanism is a softmax distribution with dictionary size equal to the length of the input."

Given an $\mathcal{EL}^+$ knowledge base K in normal form, the completion comp(K) of K can for example be obtained from K by exhaustively applying the completion rules from Table 3. There are of course different ways to perform classification using reasoning algorithms, e.g. reasoning can also be encoded using a larger number of Datalog rules which remain fixed across input theories [29]. So on the surface this seems similar to RDFS reasoning. However $\mathcal{EL}^+$ as a logic has a different look and feel: RDFS reasoning focuses on the derivation of new facts from old facts, while $\mathcal{EL}^+$ is about the processing of schema knowledge, in particular subclass relationships, in the presence of existential quantification (which is completely absent from RDFS). The transformation into Datalog reasoning is also more complicated than for RDFS.

Table 3 $\mathcal{EL}^+$ 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)	$A \sqsubseteq \exists R_1.C$	$C \sqsubseteq \exists R_2.D$	$R_1 \circ R_2 \sqsubseteq R$	$\models$	$A \sqsubseteq \exists R.D$

3.2 Pointer Networks

Pointer networks is an encoder-decoder architecture based model which uses attention as a pointer to choose an element of the input in each decoding time step. The remarkable main advantage of this model compared to other sequence-to-sequence models like Transfromers is that the learned models generalize beyond the maximum lengths that they were trained on. Thus, they

were initially proposed to generate a correct variable size output sequence, given an input sequence consisting of a variable size combinatorial optimization problem. Amazingly, it has even outperformed the fixed input size problem baseline, showing its potential to be used in wider applications.

Inspired by the above-mentioned advantages, here we have used pointer networks to copy the symbols from the knowledge base via pointing to generate logical consequences.

In our corpus, each knowledge graph and its completion denoted as $(T, c(T))$ comprises the sequence of symbols. Given the $(T, c(T))$ pair, we are feeding the pointer network with $(\mathcal{T}, \mathcal{C}(\mathcal{T})')$ where $\mathcal{T} = \{T_1, ..., T_n\}$ is a sequence of n symbols each refer to an element in our input knowledge graph and $\mathcal{C}(\mathcal{T})' = \{c(T_1)', ..., c(T_m)'\}$ is a sequence of m indices each between 1 and n. The sequence-to-sequence model then computes the conditional probability $p(c(T_i)'|c(T_1)', ..., c(T_{i-1})', \mathcal{T})$ changing the Bahdanau [4] attention to the pointer attention as follows:

$$u_j^i = v^T \tanh(W_1 e_j + W_2 d_i) \qquad \text{for } j \in (1, ..., n) \qquad \text{and} \tag{6}$$

$$p(c(T)_i'|c(T)_1', ..., c(T)_i' - 1, \mathcal{T}) = \text{Softmax}(u^i), \tag{7}$$

where $(e_1, ..., e_n)$ and $(d_1, ..., d_m)$ denote the encoder and decoder hidden states respectively, and v, W_1, and W_2 are learnable parameters of the output model. The softmax

$$\text{Softmax}(u_i) = \frac{e^{(u_i)}}{\sum_j e^{(u_j)}}$$

normalizes the vector u^i of length n to be an output distribution over the dictionary of inputs. Indeed the model uses u_j^i as pointers to the input symbols.

4 Experimental Setup

In this section, we describe the detail of datasets, training process, the baseline models, and the performance of our proposed model in terms of correctness, and generalizability in comparison to existing baselines.

4.1 Datasets

We evaluate different approaches on two benchmarked datasets:

$\mathcal{EL}^+$ Dataset To provide sufficient training input to our network we followed the same synthetic generation procedure as proposed in [12] 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. For this experiment we choose the knowledge bases of size 40, 50, and 120 statements with a moderate difficulty setting so that it can compare with nonsynthetic

$C0 \sqsubseteq C0$	$C2 \sqsubseteq C10$
$C1 \sqsubseteq C1$	$C2 \sqsubseteq C11$
$C2 \sqsubseteq C2$	$C2 \sqsubseteq C12$
$C3 \sqsubseteq C3$	$C2 \sqsubseteq C13$
$C4 \sqsubseteq C4$	$C4 \sqsubseteq C9$
$C5 \sqsubseteq C5$	$C4 \sqsubseteq C10$
$C6 \sqsubseteq C6$	$C4 \sqsubseteq C11$
$C7 \sqsubseteq C7$	$C4 \sqsubseteq C12$
$C8 \sqsubseteq C8$	$C4 \sqsubseteq C13$
$C9 \sqsubseteq C9$	$C9 \sqsubseteq C11$
$C10 \sqsubseteq C10$	$C9 \sqsubseteq C12$
$C11 \sqsubseteq C11$	$C9 \sqsubseteq C13$
$C12 \sqsubseteq C12$	$C2 \sqsubseteq \exists R4.C9$
$C13 \sqsubseteq C13$	$C2 \sqsubseteq \exists R4.C11$
$C2 \sqsubseteq C9$	$C2 \sqsubseteq \exists R5.C9$
$C4 \sqsubseteq C2$	$C2 \sqsubseteq \exists R5.C11$
$C8 \sqsubseteq C0$	$C2 \sqsubseteq \exists R6.C12$
$C9 \sqsubseteq C10$	$C2 \sqsubseteq \exists R7.C12$
$C10 \sqsubseteq C11$	$C4 \sqsubseteq \exists R4.C9$
$C0 \sqcap C1 \sqsubseteq C7$	$C4 \sqsubseteq \exists R4.C11$
$C3 \sqcap C6 \sqsubseteq C8$	$C4 \sqsubseteq \exists R5.C9$
$C11 \sqcap C12 \sqsubseteq C13$	$C4 \sqsubseteq \exists R5.C11$
$C1 \sqsubseteq \exists R1.C9$	$C4 \sqsubseteq \exists R6.C12$
$C3 \sqsubseteq \exists R3.C6$	$C4 \sqsubseteq \exists R7.C12$
$C9 \sqsubseteq \exists R4.C9$	$C9 \sqsubseteq \exists R4.C11$
$C9 \sqsubseteq \exists R5.C11$	$C9 \sqsubseteq \exists R5.C9$
$C9 \sqsubseteq \exists R6.C12$	$C9 \sqsubseteq \exists R7.C12$
$C9 \sqsubseteq \exists R6.C12$	
$C10 \sqsubseteq \exists R4.C11$	
$C10 \sqsubseteq \exists R5.C11$	
$\exists R2.C3 \sqsubseteq C0$	
$\exists R3.C7 \sqsubseteq C4$	
$\exists R4.C10 \sqsubseteq C12$	
$R4 \sqsubseteq R5$	
$R2 \sqsubseteq R0$	
$R4 \circ R6 \sqsubseteq R7$	
$R3 \circ R0 \sqsubseteq R2$	

Table 4 $\mathcal{EL}^+$ Knowledge Graph & Inference Knowledge Graph

data. To ensure that the randomized statements do not shortcut this pattern, the random statements are generated in a nearly disjoint space and connected only to the initial seed term. This ensures that at least one element of the random space will also produce random entailments for the duration of the sequence, possibly longer. Our procedure also guarantees that each completion rule will be used at least once every iteration of the sequence so that all reasoning patterns can potentially be learned by the system. An example of a graph and its corresponding inference in our $\mathcal{EL}^+$dataset is demonstrated in Table 4.

RDF Dataset For testing the capability of our model in conducting RDF reasoning we are using the same two datasets used in [34] namely a synthetic dataset from "Lehigh University Benchmark (LUBM) [20] and a real-world

Scientist dataset from DBpedia. Essentially, the mission for our deep reasoner is to learn the mapping between input RDF graphs and their inference graphs generated using Apache Jena API [9] -a state of the art tool for RDF and OWL reasoning -. The first dataset is created on top of LUBM ontology developed for benchmarking Semantic Web knowledge base systems with respect to use in large OWL applications including deductive reasoning. It conceptualizes 42 classes from the academic domain with 28 properties relating these classes. Using Univ-Bench Artificial Data Generator (UBA) [3], they yielded LUBM1 containing one hundred thousand triples with 17189 subject resources within 15 classes. For each resource r in the set of these subject-resources, a graph g (graph description of the resource r) is created by executing the following SPARQL Query:

```
DESCRIBE <r>
```

Listing 1 SPARQL query to retrieve each resource's description and creating a knowledge graph describing that resource.

For each knowledge graph g then they have obtained an inference graph i based on the LUBM ontology using Apache Jena API for applying the RDF inference rules covered partially in Table 2.

The second dataset namely "Scientists" is a real-world dataset including $\simeq 5.5 million$ triples describing 25760 URIs of scientists obtained by applying following SPARQL query against DBpedia [1] endpoint:

```
prefix dbo: <http://dbpedia.org/ontology/>
select distinct ?scientist
where {
?scientist a dbo:Scientist .
}
```

Listing 2 SPARQL query for retrieving the scientist from the DBpedia.

The dataset also includes a few other classes related to the Scientist concept in DBpedia i.e., University and Award related based on set of relationships. For the sake of our evaluations we will conduct our evaluations in each of these classes datasets separately. For a more detailed description and statistics of these two datasets please see [34].

We split each dataset in a ratio of $80\% - 10\% - 10\%$ for training, validation, and testing.

4.2 Training Details

The core of our experiments comprises training a sequence-to-sequence based model trained on large set of knowledge bases and completions pairs $(T, c(T))$. We use two single layer LSTMs of 128 hidden units each: an LSTM encoder

[3] http://swat.cse.lehigh.edu/projects/lubm/

for encoding the knowledge graph and and the Pointer LSTM for generating the completion via pointing to the input knowledge base symbols. It has been trained with stochastic gradient descent, batch size of 100, random uniform weight initialization from -0.08 to 0.08, and L2 gradient clipping of 2.0. The Adam optimizer was used with an initial learning rate of 0.1. Depending on the maximum knowledge base and completion sizes in our dataset various maximum input sequence lengths and the maximum output lengths have been enforced for each of our experiments.

4.3 Input Representation Details

Tokenization For the tokenization of the text we experiment with both Whitespace tokenizers and SubWordText tokenizers [46]. Tokenization of the text is the process of splitting the text into meaningful chunks called tokens. We believe that experimenting with different types of tokenization not only change the accuracy of our results but also gives us better understanding about the nature of reasoning and generalization ability of our network. SubWordText tokeinzer, which works based on variant of byte pair encoding segmentation algorithm [18], translates rare words into smaller units than words. As an example, SubWordText tokenizer tokenizes the triples below into {"http", "www", "department2", "university0", ... } while the Whitespace tokenizer splits each triple into subject, predicate, and object.

```
<http://www.Department2.University0.edu/GraduateStudent1>
<http://swat.cse.lehigh.edu/onto/univ-bench.owl#takesCourse>
<http://www.Department2.University0.edu/GraduateCourse0> .
```

Listing 3 Sample RDF Triple

It is worth noting that since the symbols in our synthetic $\mathcal{EL}^+$ dataset follow the $[A-Z]\backslash d^+$ regular expression both Whitespace and SubWordText tokenizers will lead to the almost same tokens splitting for $\mathcal{EL}^+$ and hence will not change the results for $\mathcal{EL}^+$.

Normalization To analyze the deductive reasoning capability of our network as opposed to the inductive reasoning capability usually obtained by learning a good representation of entities during the training or in the pre-training phase here we use the normalization[14, 15]. Unlike inductive reasoning, in the deductive reasoning the names of entities are insubstantial and should not be leveraged by the reasoner. In $\mathcal{EL}^+$the logical operators $(\sqsubseteq, \exists, \circ, \sqcap, .)$ are the only elements of the language in each knowledge base that have consistent implicit semantics across knowledge bases. In this sense, two entailments "$A \sqsubseteq C,\ C \sqsubseteq D \models A \sqsubseteq D$" and "$P \sqsubseteq Q,\ Q \sqsubseteq R \models P \sqsubseteq R$" should be treated as equivalent by the ideal reasoner. Similarly for RDF reasoning, the actual names (as strings) of entities from the underlying logic such as variables, constants, functions, and predicates are insubstantial and should not ideally

be captured by model. The only elements of the language in each knowledge base that have consistent implicit semantics across the knowledge bases here are the RDF and RDFS controlled vocabulary. Hence, two entailments "$(a, \text{rdfs:domain}, x), (y, a, z) \models (y, \text{rdf:type}, x)$" and "$(b, \text{rdfs:domain}, p), (q, b, r) \models (q, \text{rdf:type}, p)$" should be ideally considered as equivalent for logical entailment. Therefore, consistent renaming across a theory should not change the set of entailed formulas (under the same renaming). To encourage models to capture this invariance, we should either provide the term-agnostic input to our model or implement a term-agnostic strategy for the reasoning. To implement the former we use syntactic normalization: a renaming of primitives from the logical symbols to a set of predefined entity names that are used across different normalized theories. By randomly assigning the mapping for the renaming, the network's learning will be based on the structural information within the theories, and not on the actual names of the primitives. Note that this normalization not only plays the role of "forgetting" irrelevant label names, but also makes it possible to transfer learning from one KB to the other. Indeed, the network can be trained with many KBs, and then subsequently tested on completely new ones. To do so, for RDFS reasoning, we normalize all the triples within the knowledge graph by systematically renaming all URIs which are not in the RDF or RDFS namespaces. Each such URI is mapped to a set of arbitrary strings in a predefined set $A = \{a_1, ..., a_n\}$, where n is number of entities in our largest KB. Note that URIs in the RDF/RDFS namespaces are not renamed, as they are important for the deductive reasoning according to the RDFS model-theoretic semantics. Consequently, each normalized RDFS KB will be a collection of facts stored as set of triples $\{(a_i, a_j, a_k)\}$. Similarly, for the $\mathcal{EL}^+$ we have generated our syntactic dataset randomly such that only logical operators have consistent semantic meaning across the knowledge graphs. For the latter, later in Section 4.6 we show that - unlike most of the deep learning architectures which mostly rely on learning the symbols' representations - Pointer Network is inherently symbol-invariant and hence we do not need to apply such normalization to the input for Pointer Networks.

4.4 Baselines

Transformers With the recent shift towards using Transformer methods in a variety of tasks and their tremendous success, achieving the state-of-the-art in tasks such as language modeling and machine translation, we believe that it is important to assess their capability in conducting reasoning over the $\mathcal{EL}^+$ and RDFS deductive reasoning tasks. As such, for our baseline model, we use a standard vanilla encoder-decoder Transformers as proposed in [48]. The transformer architecture is merely based on self-attention mechanism and is very parallelizable. It follows with the Encoder-Decoder framework in that, given an input sequence, the network obtain a continuous representation of it based on the context and decode that context-based representation into the output sequence. It replaces the LSTMs with Self-Attention layer and encodes

the order using the sinusoidal Positional Encodings. Our network stacks 2 encoder blocks on top of each other where each block consists of 2 sub-layers, a multi-head self-attention mechanism and a position-wise dense feed-forward network. Around each sub-layer a residual connection is employed followed a layer normalization. The 2 decoder blocks has the same structure except it contains an additional multi-head attention layer that applied on the output of an encoder block. The multi-head attention that works on output representations masks all subsequent positions and the output embeddings are shifted right by one position so that a prediction for the current step depends only on previously predicted known outputs.

All embedding layers and all sub-layers in the model produce outputs of size 512. Our Transformer trainer uses the teacher-forcing strategy where the target output gets passed to the next time step regardless of model's prediction at the current time step. The Rectified Linear Unit (ReLU) has been used as our activation function. The batch size has been set to 64 and the multi-head attention consist of 8 heads. The dropout with rate $P_{drop} = 0.1$ has been applied to the output of each sub-layer and also to the embeddings summation and the positional encodings in both the encoder and decoder stacks.

Graph Words Translation defines layering RDF graphs for each of the relations in the ontology and encoding them in the form of 3D adjacency matrices where each layer layout forms a graph word. Each input graph and its entailments are then represented as sequences of graph words, and RDFS inference can be formulated as sequence-to-sequences problem, solved using neural machine translation techinques. In an effort to understand the benefits and drawbacks of our method compared to Graph words Translation [34] -current state-of-the-art method in RDF Reasoning-; here we report our results on the same dataset. Our results show Pointer Networks outperform Graph Words Translation.

Piece-Wise LSTM A deductive reasoning involves the learners being given the general rule of entailment in the language, which is then applied to specific knowledge base iteratively. It involves a set of intermediate results added at each step to the original knowledge base until we cannot generate any new statement. The Piece-Wise LSTM and its variants proposed in [12] strive to emulate this reasoning steps by mapping them to each time step in an LSTM learner. To our knowledge, this is the only work has been done for emulating deductive reasoning for $\mathcal{EL}^+$logic. As such, here we use the same procedure for generating our data and compare our result. Our finding show Pointer Networks outperform Piece-Wise LSTM and its variants by a huge margin.

LSTM Decoder As an ablation study, we replace the Pointer LSTM decoder in our encoder-decoder architecture with vanilla LSTM and evaluate its performance. This gives us a clear understanding on the contribution of the Pointer attention in our proposed model.

Table 5 Exact Match Accuracy Results

Logic	KG Size	Pointer Networks		Transformer			LSTM
		SubWordText	Tokenizer	Normalized	Not-Normalized		
					SubWordText	Tokenizer	
RDF	3 - 735	87%	**99%**	5%	25%	4%	0.17%
ER	40	73%	**73%**	8%	8%	0.4 %	0%
	50	68%	**68%**	11%	11%	0.3%	0%
	120	49%	**49%**	15%	NA	NA	0%

4.5 Correctness

In order to reflect how well our Pointer Networks have learned to conduct the reasoning task accurately; here we report the exact matching accuracy for this model and compared that to the above baselines as shown in 5. As we can see from the table, our Pointer network model has performed very well (99% accuracy) in conducting the RDF reasoning outperforming the state-of-the-art results obtained by Graph Words Translation [34](98% accuracy). Later, we show another advantage of Pointer Networks over Graph Words Translation namely its generalization capability to unseen domains. There is also an added benefit that Pointer Networks can be easily applied to any reasoning problems by defining that as "input knowledge base to inferred knowledge base mapping problem". In Graph Words Translation case, however, the network is specifically designed and tailored for RDF reasoning. Compared to the Transformer, our method has shown extraordinarily better accuracy showing clear significant performance gain of Pointer attention over self-attention in conducting the neuro-symbolic reasoning. The very poor performance of our vanilla encoder-decoder LSTM network further corroborate the benefits of using Pointer attentions. Finally, it is worth noting that the random guess is only accurate 2.8e-07% of the time for RDF reasoning over LUBM dataset demonstrating how difficult this task is.

Similarly, for $\mathcal{EL}^+$ reasoning task, our proposed model performs extraordinarily well across all the dataset, and achieves much better results achieving 73% accuracy as opposed to 0.16% accuracy reported in [12]. Similarly to our RDF reasoning experiments, here, we found our Pointer attention based method has shown extraordinarily better accuracy compared to the self-attention based Transformers and the vanilla encoder-decoder LSTM network.

4.6 Generalizability: Zero-Shot Reasoning

Using the pointer networks for simply copying from the input knowledge base to the completed one might seem simple. Despite their simple nature, the generalizability that they can provide is far more intricate and in our interest. Indeed, the main goal of this paper is to demonstrate the general symbol/naming-invariant reasoning learning capability of Pointer Networks when they encounter with the knowledge graph of the new domain/vocabulary

Table 6 Exact Match Accuracy Results for Transfer Learning/Representation: SubWord-Text Tokenization Encoding

Train \ Test	LUBM	Awards	University
LUBM	*	75%	78%
Awards	79%	*	77%
University	81%	82%	*

Table 7 Exact Match Accuracy Results for Transfer Learning/ Representation: Whitespace Tokenization Encoding

Train \ Test	LUBM	Awards	University
LUBM	*	61%	47%
Awards	96%	*	84%
University	82%	88%	*

in the testing phase. This ensures the model has gained the deep understanding of the logical semantics and reasoning as opposed to merely working based on the representation learning and induction. As such, we measured exact matching accuracy of the results yielded by Pointer Networks when trained on one domain and tested on another without fine-tuning i.e., zero-shot . The results for its transfer capability for RDFS reasoning is shown in Tables 6 and 7, while for the $\mathcal{EL}^+$ the results in Table 5 already shown this capability. Based on the table, Pointer Network gives surprisingly good and consistent empirical results when it comes to transfer learning. Indeed, Pointer Networks have several desirable inherent characteristics leading into this transfer learning behaviour. They are capable of dealing with dynamic vocabulary length as opposed to fixed vocabulary output, dealing with rare or out-of-vocabulary word, and heavy-tailed vocabulary distribution.

Additionally, to further understand the nature of how Transformers learn to reason?, we have applied normalization and various tokenization on our RDF dataset and examined the change in the accuracy. Not surprisingly, unlike Pointer Networks, Transformers are very sensitive to the changes of tokenization and the normalization. This is mainly because Transformers heavily rely on the subsymbolic representation of entities and relations learned by the network. Indeed the power of Transformers mainly comes from their self/intra-attention module primarily used to learn the representation of the tokens in the input based on their relations with other tokens. This explains why our baseline Transformer model tends to obtain its highest accuracy when trained on not-normalized SubWordText encoded RDF knowledge base. This way, the network can learn much better representation for the symbols in the knowledge base which leads to better reasoning accuracy. Unsurprisingly, the normalization decreases the accuracy of the Transformer showing poor symbol-invariant reasoning capability, as indicated in the *Normalized* column in Table 5.

5 Conclusion & Future Works

We have shown that a deep learning architecture based on pointer networks is capable of learning how to perform deductive reason over RDFS and $\mathcal{EL}^+$ KBs with high accuracy. We designed a novel way of conducting neuro-symbolic through pointing to the input elements. More importantly we showed that the proposed approach is generalizable across new domain and vocabulary demonstrating symbol-invariant zero-shot reasoning capability. We plan to properly investigate scalability of our approach and to adapt it to other, more complex logics. We furthermore intend to investigate the added values which should arise out of adding subsymbolic deductive reasoning components to more traditional deep learning scenarios, in particular in the areas of knowledge graph inference and natural-language-based commonsense reasoning.

Acknowledgements This work was supported by the National Science Foundation (NSF) under award OIA-2033521 *KnowWhereGraph: Enriching and Linking Cross-Domain Knowledge Graphs using Spatially-Explicit AI Technologies.*

Conflict of interest

The authors declare that they have no conflict of interest.

References

1. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.G.: Dbpedia: A nucleus for a web of open data. In: ISWC/ASWC, *Lecture Notes in Computer Science*, vol. 4825, pp. 722–735. Springer (2007)
2. Baader, F., Brandt, S., Lutz, C.: Pushing the EL envelope. In: L.P. Kaelbling, A. Saffiotti (eds.) IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30 - August 5, 2005, pp. 364–369. Professional Book Center (2005)
3. Badreddine, S., d'Avila Garcez, A., Serafini, L., Spranger, M.: Logic tensor networks. CoRR **abs/2012.13635** (2020). URL https://arxiv.org/abs/2012.13635
4. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. In: Y. Bengio, Y. LeCun (eds.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (2015). URL http://arxiv.org/abs/1409.0473
5. Banerjee, D., Chaudhuri, D., Dubey, M., Lehmann, J.: PNEL: pointer network based end-to-end entity linking over knowledge graphs. In: J.Z. Pan, V.A.M. Tamma, C. d'Amato, K. Janowicz, B. Fu, A. Polleres, O. Seneviratne, L. Kagal (eds.) The Semantic Web - ISWC 2020 - 19th International Semantic Web Conference, Athens, Greece, November 2-6, 2020, Proceedings, Part I, *Lecture Notes in Computer Science*, vol. 12506, pp. 21–38. Springer (2020). DOI 10.1007/978-3-030-62419-4_2. URL https://doi.org/10.1007/978-3-030-62419-4_2
6. Besold, T.R., Garcez, A.d., Bader, S., Bowman, H., Domingos, P., Hitzler, P., Kühnberger, K.U., Lamb, L.C., Lowd, D., Lima, P.M.V., et al.: Neural-symbolic learning and reasoning: A survey and interpretation. arXiv preprint arXiv:1711.03902 (2017)
7. Bianchi, F., Hitzler, P.: On the capabilities of logic tensor networks for deductive reasoning. In: A. Martin, K. Hinkelmann, A. Gerber, D. Lenat, F. van Harmelen, P. Clark

(eds.) Proceedings of the AAAI 2019 Spring Symposium on Combining Machine Learning with Knowledge Engineering (AAAI-MAKE 2019) Stanford University, Palo Alto, California, USA, March 25-27, 2019., Stanford University, Palo Alto, California, USA, March 25-27, 2019, *CEUR Workshop Proceedings*, vol. 2350. CEUR-WS.org (2019). URL http://ceur-ws.org/Vol-2350/paper22.pdf
8. Brickley, D., Guha, R. (eds.): RDF Schema 1.1. "" (W3C Recommendation 25 February 2014). Available from http://www.w3.org/TR/rdf-schema/
9. Carroll, J.J., Dickinson, I., Dollin, C., Reynolds, D., Seaborne, A., Wilkinson, K.: Jena: implementing the semantic web recommendations. In: WWW (Alternate Track Papers & Posters), pp. 74–83. ACM (2004)
10. Consortium, W.W.W., et al.: Rdf 1.1 semantics. empty (2014)
11. 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/
12. Eberhart, A., Ebrahimi, M., Zhou, L., Shimizu, C., Hitzler, P.: Completion reasoning emulation for the description logic EL+. In: A. Martin, K. Hinkelmann, H. Fill, A. Gerber, D. Lenat, R. Stolle, F. van Harmelen (eds.) Proceedings of the AAAI 2020 Spring Symposium on Combining Machine Learning and Knowledge Engineering in Practice, AAAI-MAKE 2020, Palo Alto, CA, USA, March 23-25, 2020, Volume I, *CEUR Workshop Proceedings*, vol. 2600. CEUR-WS.org (2020). URL http://ceur-ws.org/Vol-2600/paper5.pdf
13. Ebrahimi, M., Eberhart, A., Bianchi, F., Hitzler, P.: Towards bridging the neuro-symbolic gap: Deep deductive reasoners. Applied Intelligence (2021). To appear
14. Ebrahimi, M., Sarker, M.K., Bianchi, F., Xie, N., Eberhart, A., Doran, D., Kim, H., Hitzler, P.: Neuro-symbolic deductive reasoning for cross-knowledge graph entailment. In: A. Martin, K. Hinkelmann, H. Fill, A. Gerber, D. Lenat, R. Stolle, F. van Harmelen (eds.) Proceedings of the AAAI 2021 Spring Symposium on Combining Machine Learning and Knowledge Engineering (AAAI-MAKE 2021), Stanford University, Palo Alto, California, USA, March 22-24, 2021, *CEUR Workshop Proceedings*, vol. 2846. CEUR-WS.org (2021). URL http://ceur-ws.org/Vol-2846/paper8.pdf
15. Evans, R., Saxton, D., Amos, D., Kohli, P., Grefenstette, E.: Can neural networks understand logical entailment? In: ICLR (Poster). OpenReview.net (2018)
16. Fernández-González, D., Gómez-Rodríguez, C.: Left-to-right dependency parsing with pointer networks. In: J. Burstein, C. Doran, T. Solorio (eds.) Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers), pp. 710–716. Association for Computational Linguistics (2019). DOI 10.18653/v1/n19-1076. URL https://doi.org/10.18653/v1/n19-1076
17. Fernández-González, D., Gómez-Rodríguez, C.: Transition-based semantic dependency parsing with pointer networks. In: D. Jurafsky, J. Chai, N. Schluter, J.R. Tetreault (eds.) Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020, pp. 7035–7046. Association for Computational Linguistics (2020). DOI 10.18653/v1/2020.acl-main.629. URL https://doi.org/10.18653/v1/2020.acl-main.629
18. Gage, P.: A new algorithm for data compression. C Users Journal **12**(2), 23–38 (1994)
19. Gu, S., Hao, T.: A pointer network based deep learning algorithm for 0-1 knapsack problem. In: ICACI, pp. 473–477. IEEE (2018)
20. Guo, Y., Pan, Z., Heflin, J.: LUBM: A benchmark for OWL knowledge base systems. J. Web Semant. **3**(2-3), 158–182 (2005)
21. Gupta, V., Krzyzak, A.: An empirical evaluation of attention and pointer networks for paraphrase generation. In: ICCS (3), *Lecture Notes in Computer Science*, vol. 12139, pp. 399–413. Springer (2020)
22. Hayes, P.J., Patel-Schneider, P.F. (eds.): RDF 1.1 Semantics. "" (W3C Recommendation 25 February 2014). Available from http://www.w3.org/TR/rdf11-mt/
23. Hitzler, P.: A review of the semantic web field. Communications of the ACM **64**(2), 76–83 (2021)

24. Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P.F., Rudolph, S. (eds.): OWL 2 Web Ontology Language: Primer (Second Edition). W3C Recommendation 11 December 2012 (2012). Available from http://www.w3.org/TR/owl2-primer/
25. Hitzler, P., Krötzsch, M., Rudolph, S.: Foundations of Semantic Web Technologies. Chapman & Hall/CRC (2010)
26. Hitzler, P., Seda, A.K.: Mathematical Aspects of Logic Programming Semantics. Chapman and Hall / CRC studies in informatics series. CRC Press (2011)
27. Hohenecker, P., Lukasiewicz, T.: Ontology reasoning with deep neural networks. J. Artif. Intell. Res. **68**, 503–540 (2020). DOI 10.1613/jair.1.11661. URL https://doi.org/10.1613/jair.1.11661
28. Jadhav, A., Rajan, V.: Extractive summarization with SWAP-NET: sentences and words from alternating pointer networks. In: I. Gurevych, Y. Miyao (eds.) Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers, pp. 142–151. Association for Computational Linguistics (2018). DOI 10.18653/v1/P18-1014. URL https://www.aclweb.org/anthology/P18-1014/
29. Krötzsch, M.: Efficient rule-based inferencing for OWL EL. In: T. Walsh (ed.) IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011, pp. 2668–2673. IJCAI/AAAI (2011). DOI 10.5591/978-1-57735-516-8/IJCAI11-444. URL https://doi.org/10.5591/978-1-57735-516-8/IJCAI11-444
30. Lee, C., Jung, S., Park, C.: Anaphora resolution with pointer networks. Pattern Recognit. Lett. **95**, 1–7 (2017)
31. Li, J., Wang, Y., Lyu, M.R., King, I.: Code completion with neural attention and pointer networks. In: J. Lang (ed.) Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden, pp. 4159–4165. ijcai.org (2018). DOI 10.24963/ijcai.2018/578. URL https://doi.org/10.24963/ijcai.2018/578
32. Li, J., Ye, D., Shang, S.: Adversarial transfer for named entity boundary detection with pointer networks. In: IJCAI, pp. 5053–5059. ijcai.org (2019)
33. Ma, X., Hu, Z., Liu, J., Peng, N., Neubig, G., Hovy, E.H.: Stack-pointer networks for dependency parsing. In: I. Gurevych, Y. Miyao (eds.) Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers, pp. 1403–1414. Association for Computational Linguistics (2018). DOI 10.18653/v1/P18-1130. URL https://www.aclweb.org/anthology/P18-1130/
34. Makni, B., Hendler, J.A.: Deep learning for noise-tolerant RDFS reasoning. Semantic Web **10**(5), 823–862 (2019). DOI 10.3233/SW-190363. URL https://doi.org/10.3233/SW-190363
35. Minervini, P., Bosnjak, M., Rocktäschel, T., Riedel, S., Grefenstette, E.: Differentiable reasoning on large knowledge bases and natural language. In: The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020, pp. 5182–5190. AAAI Press (2020). URL https://aaai.org/ojs/index.php/AAAI/article/view/5962
36. Minervini, P., Riedel, S., Stenetorp, P., Grefenstette, E., Rocktäschel, T.: Learning reasoning strategies in end-to-end differentiable proving. In: Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event, *Proceedings of Machine Learning Research*, vol. 119, pp. 6938–6949. PMLR (2020). URL http://proceedings.mlr.press/v119/minervini20a.html
37. Mottini, A., Acuna-Agost, R.: Deep choice model using pointer networks for airline itinerary prediction. In: KDD, pp. 1575–1583. ACM (2017)
38. Pandey, D., Chowdary, C.R.: Modeling coherence by ordering paragraphs using pointer networks. Neural Networks **126**, 36–41 (2020)
39. Paulheim, H.: Knowledge graph refinement: A survey of approaches and evaluation methods. Semantic Web **8**(3), 489–508 (2017). DOI 10.3233/SW-160218. URL https://doi.org/10.3233/SW-160218

40. Rietveld, L., Beek, W., Hoekstra, R., Schlobach, S.: Meta-data for a lot of LOD. Semantic Web **8**(6), 1067–1080 (2017)
41. Ristoski, P., Rosati, J., Noia, T.D., Leone, R.D., Paulheim, H.: Rdf2vec: RDF graph embeddings and their applications. Semantic Web **10**(4), 721–752 (2019). DOI 10.3233/SW-180317. URL https://doi.org/10.3233/SW-180317
42. Rocktäschel, T., Riedel, S.: Learning knowledge base inference with neural theorem provers. In: J. Pujara, T. Rocktäschel, D. Chen, S. Singh (eds.) Proceedings of the 5th Workshop on Automated Knowledge Base Construction, AKBC@NAACL-HLT 2016, San Diego, CA, USA, June 17, 2016, pp. 45–50. The Association for Computer Linguistics (2016). DOI 10.18653/v1/w16-1309. URL https://doi.org/10.18653/v1/w16-1309
43. Rocktäschel, T., Riedel, S.: End-to-end differentiable proving. In: Advances in Neural Information Processing Systems, pp. 3788–3800 (2017)
44. Schulz, S., Suntisrivaraporn, B., Baader, F., Boeker, M.: SNOMED reaching its adolescence: Ontologists' and logicians' health check. I. J. Medical Informatics **78**(Supplement-1), S86–S94 (2009)
45. See, A., Liu, P.J., Manning, C.D.: Get to the point: Summarization with pointer-generator networks. In: R. Barzilay, M. Kan (eds.) Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers, pp. 1073–1083. Association for Computational Linguistics (2017). DOI 10.18653/v1/P17-1099. URL https://doi.org/10.18653/v1/P17-1099
46. Sennrich, R., Haddow, B., Birch, A.: Neural machine translation of rare words with subword units. In: ACL (1). The Association for Computer Linguistics (2016)
47. Serafini, L., Garcez, A.S.d.: Learning and reasoning with logic tensor networks. In: Conference of the Italian Association for Artificial Intelligence, pp. 334–348. Springer (2016)
48. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: NIPS, pp. 5998–6008 (2017)
49. Vinyals, O., Fortunato, M., Jaitly, N.: Pointer networks. In: C. Cortes, N.D. Lawrence, D.D. Lee, M. Sugiyama, R. Garnett (eds.) Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada, pp. 2692–2700 (2015). URL http://papers.nips.cc/paper/5866-pointer-networks
50. Yu, T., Joty, S.R.: Online conversation disentanglement with pointer networks. In: EMNLP (1), pp. 6321–6330. Association for Computational Linguistics (2020)

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

Towards Bridging the Neuro-Symbolic Gap: Deep Deductive Reasoners

Monireh Ebrahimi · Aaron Eberhart · Federico Bianchi · Pascal Hitzler

Received: date / Accepted: date

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

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

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

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

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

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

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

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

1 Introduction & Motivation

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

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

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

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

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

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

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

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

2 Related Work

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

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

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

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

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

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

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

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

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

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

3 Summary of our Work

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

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

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

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

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

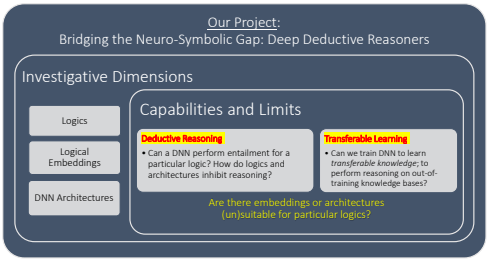

Fig. 1 Our work explores the capabilities and limits of subsymbolic systems (DNNs) to perform transferrable deductive reasoning. Experiments are carried out along three investigative dimensions: type of logic, embeddings of logic to a quantitative space, and DNN architectures.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

with $\circ$ signifying standard binary composition

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

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

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

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

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

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

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

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

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

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

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

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

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

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

where $\mathbf{v} = \langle \mathbf{v}_1, \ldots, \mathbf{v}_m \rangle$ is the concatenation of the vectors $\mathbf{v}_i$, 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, $\forall x\ mammal(x) \rightarrow animal(x)$); are computed by using an aggregation operation [69] defined over a subset T of the domain space $\mathbb{R}^n$; different aggregation operations, such as min, average or harmonic average can be considered. Hence, the optimization comes from aggregating the truth value of each formula, instantiated with the values in the domain, and then the maximization of this value, as we want $\forall x\ mammal(x) \rightarrow animal(x)$ to be true for all $x \in T$.

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

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

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

Table 2 Translation Rules

KB statement		Vectorization
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

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

3.1 Investigative dimension 3: DNN architectures

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

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

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

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

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

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

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

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

4 Summary of our Experimental Settings & Evaluations

4.1 RDFS Reasoning with Memory Networks

4.1.1 Problem Setting

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

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

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

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

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

4.1.2 Evaluations

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

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

Training Dataset	Test Dataset	Valid Triples Class			Invalid Triples Class			Accuracy
		Precision	Recall /Sensitivity	F-measure	Precision	Recall /Specificity	F-measure	
OWL-Centric Dataset	Linked Data	93	98	96	98	93	95	**96**
OWL-Centric Dataset (90%)	OWL-Centric Dataset (10%)	88	91	89	90	88	89	**90**
OWL-Centric Dataset	OWL-Centric Test Set [b]	79	62	68	70	84	76	**69**
OWL-Centric Dataset	Synthetic Data	65	49	40	52	54	42	**52**
OWL-Centric Dataset	Linked Data [a]	54	98	70	91	16	27	86
OWL-Centric Dataset [a]	Linked Data [a]	62	72	67	67	56	61	91
OWL-Centric Dataset(90%) [a]	OWL-Centric Dataset(10%) [a]	79	72	75	74	81	77	80
OWL-Centric Dataset	OWL-Centric Test Set [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

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

Table 3 Experimental results of proposed model

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

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

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

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

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

[a] Completely Different Domain.

Table 4 Ablation Study: No Positional Encoding

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

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

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

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

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

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

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

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

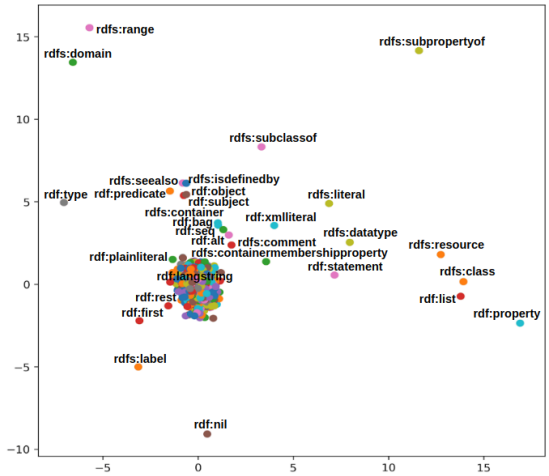

Fig. 3 PCA projection of embeddings for the vocabulary

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

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

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

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

4.2 Deductive Reasoning Capabilities of Logic Tensor Networks

4.2.1 Problem Setting

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

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

4.3 Evaluations

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

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

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

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

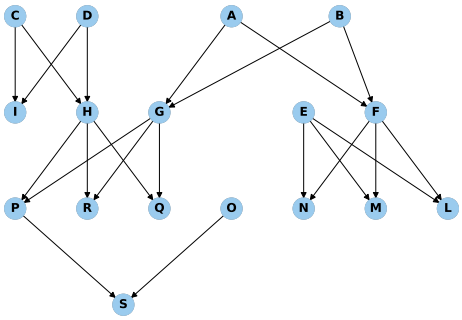

Fig. 4 Representation of the parental relationships in the P dataset

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Table 7 $F1$ score per tested class.

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

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

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

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

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

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

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

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

Table 8 The truth values of novel axioms.

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

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

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

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

4.4 Scalability

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

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

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

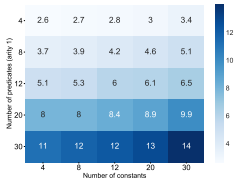

Fig. 5 Computational times in seconds for predicates of arity one.

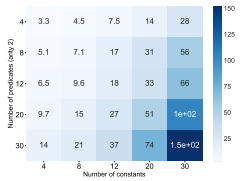

Fig. 6 Computational times in seconds for predicates of arity two.

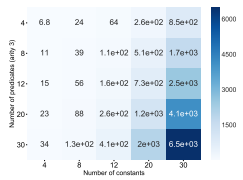

Fig. 7 Computational times in seconds for predicates of arity three.

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

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

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

4.5.1 Problem Setting

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

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

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

Table 10 Support Generation

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

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

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

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

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

Initial Axioms:

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

Entailments Step 1:

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

Entailments Step 2:

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

Fig. 8 First Iteration of Sequence in an Example

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

4.5.2 Evaluation

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

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

Table 11 Example Synthetic Output

	Correct Answer	Predicted Answer
Step 0	C9 $\sqsubseteq$ 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	
	C2 $\sqsubseteq$ $\exists$R6.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

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 a 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

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

Table 13 Average Statement Edit Distances with Reasoner

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

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

	Atomic Levenshtein Distance			Character Levenshtein Distance			Predicate Distance		
	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score
	Synthetic Data								
Piecewise Prediction	0.138663	0.142208	0.140412	0.138663	0.14220S	0.140412	0.138646	0.141923	0.140264
Deep Prediction	**0.154398**	**0.156056**	**0.155222**	**0.154398**	**0.156056**	**0.155222**	**0.154258**	**0.155736**	**0.154993**
Flat Prediction	0.140410	0.142976	0.141681	0.140410	0.142976	0.141681	0.140375	0.142687	0.141521
Random Prediction	0.010951	0.0200518	0.014166	0.006833	0.012401	0.008811	0.004352	0.007908	0.007908
	SNOMED Data								
Piecewise Prediction	0.010530	0.013554	0.011845	0.010530	0.013554	0.011845	0.010521	0.013554	0.011839
Deep Prediction	**0.015983**	0.0172811	**0.016595**	**0.015983**	0.017281	**0.016595**	**0.015614**	0.017281	**0.016396**
Flat Prediction	0.014414	**0.018300**	0.016112	0.0144140	**0.018300**	0.016112	0.013495	**0.018300**	0.015525
Random Prediction	0.002807	0.006803	0.003975	0.001433	0.003444	0.002023	0.001769	0.004281	0.002504

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

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

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

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

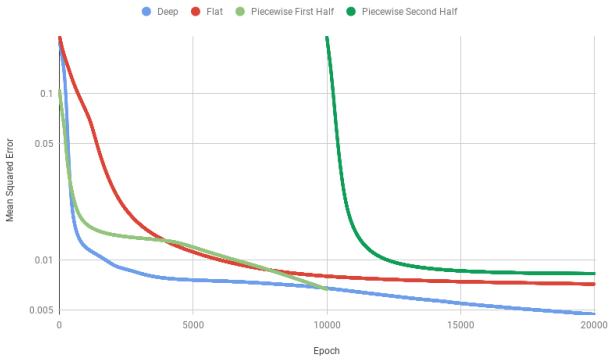

Fig. 9 Synthetic Training

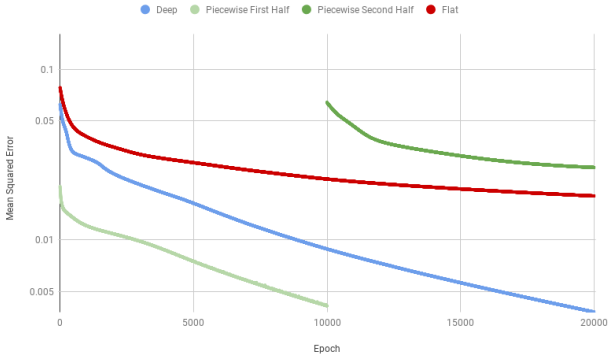

Fig. 10 SNOMED Training

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

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

5 Conclusions

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

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

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

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

References

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

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

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

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

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

63. Ristoski, P., Paulheim, H.: Rdf2vec: Rdf graph embeddings for data mining. In: International Semantic Web Conference, pp. 498–514. Springer (2016)
64. Rocktäschel, T., Riedel, S.: End-to-end differentiable proving. In: I. Guyon, U. von Luxburg, S. Bengio, H.M. Wallach, R. Fergus, S.V.N. Vishwanathan, R. Garnett (eds.) Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA, pp. 3791–3803 (2017)
65. Rocktäschel, T., Riedel, S.: End-to-end differentiable proving. In: Advances in Neural Information Processing Systems, pp. 3788–3800 (2017)
66. Rocktäschel, T., Singh, S., Riedel, S.: Injecting logical background knowledge into embeddings for relation extraction. In: Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 1119–1129 (2015)
67. Serafini, L., d'Avila Garcez, A.S.: Learning and reasoning with logic tensor networks. In: G. Adorni, S. Cagnoni, M. Gori, M. Maratea (eds.) AI*IA 2016: Advances in Artificial Intelligence – XVth International Conference of the Italian Association for Artificial Intelligence, Genova, Italy, November 29 – December 1, 2016, Proceedings, *Lecture Notes in Computer Science*, vol. 10037, pp. 334–348. Springer (2016)
68. Serafini, L., d'Avila Garcez, A.S.: Logic tensor networks: Deep learning and logical reasoning from data and knowledge. In: T.R. Besold, L.C. Lamb, L. Serafini, W. Tabor (eds.) Proceedings of the 11th International Workshop on Neural-Symbolic Learning and Reasoning (NeSy'16) co-located with the Joint Multi-Conference on Human-Level Artificial Intelligence (HLAI 2016), New York City, NY, USA, July 16-17, 2016., *CEUR Workshop Proceedings*, vol. 1768. CEUR-WS.org (2016)
69. Serafini, L., d'Avila Garcez, A.S.: Logic tensor networks: Deep learning and logical reasoning from data and knowledge. In: T.R. Besold, L.C. Lamb, L. Serafini, W. Tabor (eds.) Proceedings of the 11th International Workshop on Neural-Symbolic Learning and Reasoning (NeSy'16) co-located with the Joint Multi-Conference on Human-Level Artificial Intelligence (HLAI 2016), New York City, NY, USA, July 16-17, 2016, *CEUR Workshop Proceedings*, vol. 1768. CEUR-WS.org (2016). URL `http://ceur-ws.org/Vol-1768/NESY16\_paper3.pdf`
70. Shastri, L.: Advances in SHRUTI-A neurally motivated model of relational knowledge representation and rapid inference using temporal synchrony. Appl. Intell. **11**(1), 79–108 (1999)
71. Shastri, L.: SHRUTI: A neurally motivated architecture for rapid, scalable inference. In: B. Hammer, P. Hitzler (eds.) Perspectives of Neural-Symbolic Integration, *Studies in Computational Intelligence*, vol. 77, pp. 183–203. Springer (2007)
72. Socher, R., Chen, D., Manning, C.D., Ng, A.Y.: Reasoning with neural tensor networks for knowledge base completion. In: C.J.C. Burges, L. Bottou, Z. Ghahramani, K.Q. Weinberger (eds.) Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States., pp. 926–934 (2013)
73. Sukhbaatar, S., Szlam, A., Weston, J., Fergus, R.: End-to-end memory networks. In: C. Cortes, N.D. Lawrence, D.D. Lee, M. Sugiyama, R. Garnett (eds.) Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada, pp. 2440–2448 (2015)
74. Towell, G.G., Shavlik, J.W.: Knowledge-based artificial neural networks. Artif. Intell. **70**(1-2), 119–165 (1994)
75. Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., Bouchard, G.: Complex embeddings for simple link prediction. In: M. Balcan, K.Q. Weinberger (eds.) Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016, *JMLR Workshop and Conference Proceedings*, vol. 48, pp. 2071–2080. JMLR.org (2016)
76. Wang, Q., Mao, Z., Wang, B., Guo, L.: Knowledge graph embedding: A survey of approaches and applications. IEEE Trans. Knowl. Data Eng. **29**(12), 2724–2743 (2017)
77. Wang, W.Y., Cohen, W.W.: Learning first-order logic embeddings via matrix factorization. In: IJCAI, pp. 2132–2138 (2016)
78. Wang, Z., Zhang, J., Feng, J., Chen, Z.: Knowledge graph embedding by translating on hyperplanes. In: C.E. Brodley, P. Stone (eds.) Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada., pp. 1112–1119. AAAI Press (2014)

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

Neuro-Symbolic Deductive Reasoning for Cross-Knowledge Graph Entailment

Monireh Ebrahimi[a], Md Kamruzzaman Sarker[a], Federico Bianchi[b], Ning Xie[c], Aaron Eberhart[a], Derek Doran[c], HyeongSik Kim[d] and Pascal Hitzler[a]

[a]Department of Computer Science & Engineering, Kansas State University
[b]Bocconi University
[c]Department of Computer Science & Engineering, Wright State University
[d]Bosch Research & Technology Center, North America

Abstract

A significant and recent development in neural-symbolic learning are deep neural networks that can reason over symbolic knowledge graphs (KGs). A particular task of interest is *KG entailment*, which is to infer the set of all facts that are a logical consequence of current and potential facts of a KG. Initial neural-symbolic systems that can deduce the entailment of a KG have been presented, but they are limited: current systems learn fact relations and entailment patterns specific to a particular KG and hence do not truly generalize, and must be retrained for each KG they are tasked with entailing. We propose a neural-symbolic system to address this limitation in this paper. It is designed as a differentiable end-to-end deep memory network that learns over abstract, generic symbols to discover entailment patterns common to any reasoning task. A key component of the system is a simple but highly effective normalization process for continuous representation learning of KG entities within memory networks. Our results show how the model, trained over a set of KGs, can effectively entail facts from KGs excluded from the training, even when the vocabulary or the domain of test KGs is completely different from the training KGs.

Keywords

deep learning, deductive reasoning, knowledge graph entailment, neuro-symbolic

1. Introduction

For many years, reasoning has been tackled as the task of building systems capable of inferring new crisp symbolic logical rules. However, those traditional methods are too brittle to be applied to noisy automatically created KGs. [1] provides a taxonomy of noise types in web KGs with respect to its effects on reasoning and shows the detrimental impact of noises on the result of the traditional reasoners. With the recent revival of interest in artificial neural

In A. Martin, K. Hinkelmann, H.-G. Fill, A. Gerber, D. Lenat, R. Stolle, F. van Harmelen (Eds.), Proceedings of the AAAI 2021 Spring Symposium on Combining Machine Learning and Knowledge Engineering (AAAI-MAKE 2021) - Stanford University, Palo Alto, California, USA, March 22-24, 2021.

monireh@ksu.edu (M. Ebrahimi); mdkamruzzamansarker@ksu.edu (M.K. Sarker); f.bianchi@unibocconi.it (F. Bianchi); xie.25@wright.edu (N. Xie); aaroneberhart@ksu.edu (A. Eberhart); derek.doran@wright.edu (D. Doran); Hyeongsik.Kim@us.bosch.com (H. Kim); hitzler@ksu.edu (P. Hitzler)

networks, the more robust neural link prediction models have been applied vastly for the completion of KGs. These methods [2, 3, 4, 5, 6] heavily rely on the subsymbolic representation of entities and relations learned through maximization of a scoring objective function over valid factual triples. Thus, the success of such models hinges primarily on the power of those subsymbolic representations in encoding the similarity/relatedness of entities and relations. Recent attempts have focused on neural multi-hop reasoners [7, 8] to equip the model to deal with more complex reasoning where multi-hop inference is required. More recently, a Neural Theorem Prover [9] has been proposed in an attempt to take advantage of both symbolic and sub-symbolic reasoning.

Despite their success, the main restriction common to machine learning-based reasoners is that they are unable to recognize and generalize to different domains or tasks. This inherent limitation follows from both the representations used and the learning process. The major issue comes from the mere reliance of these models on the representation of entities learned during the training or in the pre-training phase stored in a lookup table. Consequently, these models usually have difficulty to deal with out-of-vocabulary (OOV) entities. Although the OOV problem has been addressed in part in the natural language processing (NLP) domain by taking advantage of character-level embedding [10], subword units [11], Byte-Pair-Encoding [12], learning embeddings on the fly by leveraging text descriptions or spelling [13], copy mechanism [14] or pointer networks [15], still these solutions are insufficient for transferring purposes for reasoning. [16] shows that the success of natural language inference (NLI) methods is heavily benchmark specific. An even greater source of concern is that reasoning in most of the above sub-symbolic approaches hinges more on the notion of similarity and geometric-based proximity of real-valued vectors (induction) as opposed to performing transitive reasoning (deduction) over them. Nevertheless, recent years have seen some progress in zero-shot relation learning in sub-symbolic reasoning domain[17]. Zero-shot learning refers to the ability of the model to infer new relations where that relation has not been seen before in training set[18]. This generalization capability is still quite limited and fundamentally different from our work in terms of both methodology and purpose.

Inspired by these observations, we take a different approach in this work by investigating the emulation of deductive symbolic reasoning using memory networks. Memory networks [19] are a class of learning models capable of conducting multiple computational steps over an explicit memory component before returning an answer. 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. Besides, as attention can be traced over the run of a memory network, we will furthermore get insights into the "reasoning" underlying the network output.

This paper contributes a recipe involving a simple but effective KG triple normalization before learning their representation within an end-to-end memory network. To perform logical inference in more abstract level, and thereby facilitating the transfer of reasoning expertise from one KG to another, the normalization maps entities and predicates in a KG to a generic vocabulary. Facts in additional KGs are normalized using the same vocabulary, so that the network does not learn to overfit its learning to entity and predicate names in a specific KG. This emulates symbolic reasoning by neural embeddings as the actual names (as strings) of entities from the underlying logic such as variables, constants, functions, and predicates are insub-

stantial for logical entailment in the sense that a consistent renaming across a theory does not change the set of entailed formulas (under the same renaming). Thanks to the term-agnostic feature of our representation, we are able to create a reasoning system capable of performing reasoning over an unseen set of vocabularies in the test phase.

Our contributions are threefold: (i) We present the construction of memory networks for emulating the symbolic deductive reasoning. (ii) We propose an optimization to this architecture using normalization approach to enhance their transfer capability. We show that in an unnormalized setting, they fail to perform well across KGs. (iii) We examine the efficacy of our model for cross-domain and cross-KG deductive reasoning. We also show the scalability of our model (in terms of reduced time and space complexity) for large datasets.

Related Work On the issue of doing logical reasoning using deep networks, we mention the following selected recent contributions: Tensor-based approaches have been used [20, 9, 21], following [3]. However, approaches are restricted in terms of logical expressivity and/or to toy examples [22, 20]. [1] perform Resource Description Framework (RDF) reasoning based on KG embeddings. [23] considers OWL RL reasoning [24]. There is a fundamental difference between these contributions and our approach, though: We train our model once, and then the model transfers to all other RDF KGs with good performance. In the above mentioned publications, training is either done on (a part of the) KG which is also used for evaluation, or training is explicitly done on similar KGs, in terms of topic. More precisely, in case of [23], it requires re-training for obtaining embeddings for new vocabularies.

2. Problem Formulation

To explain what we are setting out to do, let us first re-frame the deductive reasoning problem as a classification task. Any 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 $T \models F_{\mathcal{L}}$, then we say that F is *entailed* by T. Re-framed as a classification task, we can ask whether a given pair $(T, F) \in T_{\mathcal{L}} \times F_{\mathcal{L}}$ should be classified as a valid entailment (i.e., $T \models_{\mathcal{L}} F$) holds, or as the opposite (i.e., $T \not\models_{\mathcal{L}} F$). We would like to train a model on sets of examples (T, F), such that it learns to correctly classify them as valid or invalid inferences.

We wish to train a neural model that will learn to reason over one set of theories, and can then transfer that learning to new theories over the same logic. This way, our results will demonstrate that the reasoning principles (entailment under the model-theoretic semantics) that underlie the logic have been learned. If we were to train a model such that it learns only to reason over one theory, or a few very similar theories, then that could hardly be demonstrated. One of the key obstacles we face with our task is to understand how to represent training and test data so that they can be used in deep learning settings. To use standard deep learning approaches, formulas – or even theories – will have to be represented over the real coordinate space R as vectors, matrices or tensors. Many embeddings for RDF (i.e., KGs) have been proposed [25, 6, 26], but we are not aware of an existing embedding that captures what seems important for the deductive reasoning scenario. Indeed, the prominent use case explored

for KG embeddings is not deductive in nature; rather, it concerns the problem of the discovery or suggestion of additional links or edges in the graph, together with appropriate edge labels. In this link discovery setting, the actual labels for nodes or edges in the graph, and as such their commonsense meanings, are likely important, and most existing embeddings reflect this. However, for deductive reasoning the names of entities are insubstantial and should not be captured by an embedding. Another inherent problem in the use of such representations across KGs is the OOV problem. While a word lookup table can be initialized with vectors in an unsupervised task or during training of the reasoner, it still cannot generate vector representations for unseen terms. It is further impractical to store the vectors of all words when vocabulary size is huge [10]. Similarly, memory networks usually rely on word-level embedding lookup tables, i.e., learned with the underlying rationale that words that occur in similar supervised scenarios should be represented by similar vectors in the real coordinate space. That is why they are known to have difficulties dealing with OOV, as a word lookup table cannot provide a representation for the unseen, and thus cannot be applied to NLI over new words [13], and for us this would pose a challenge in the transfer to new KGs.

We thus need embeddings that are agnostic to the terms (i.e., strings) used as primitives in the KG. To build such an embedding, we use syntactic normalization: a renaming of primitives from the logical language (variables, constants, functions, predicates) to a set of predefined entity names that are used across different normalized theories. By randomly assigning the mapping for the renaming, the network's learning will be based on the structural information within the theories, and not on the actual names of the primitives. Note that this normalization does not only play the role of "forgetting" irrelevant label names, but also makes it possible to transfer learning from one KG to the other. Indeed, for the approach to work, the network should be trained with many KGs, and then subsequently tested on completely new ones which had not been encountered during training. Our results show that our simple but very effective normalization yields a word-agnostic system capable of deductive reasoning over previously-unseen RDF KGs containing new vocabulary.

3. Model Architecture

We consider a model architecture that adapts the end-to-end memory network proposed by [19] with fundamental alterations necessary for abstract reasoning. A high-level view of our model is shown in Figure 1. It takes a discrete set G of normalized RDF statements (called *triples*) $t_1, ..., t_n$ that are stored in memory, a query q, and outputs a "yes" or "no" answer to determine if q is entailed by G. Each of the normalized t_i and q contains symbols coming from a general dictionary with V normalized words shared among all of the normalized RDF theories in both training and test sets. The model writes all triples to the memory and then calculates a continuous embedding for G and q. Through multiple hop attention over those continuous representations, the model then classifies the query. The model is trained by back-propagation of error from output to the input through multiple memory accesses. We discuss components of the architecture in more detail below.

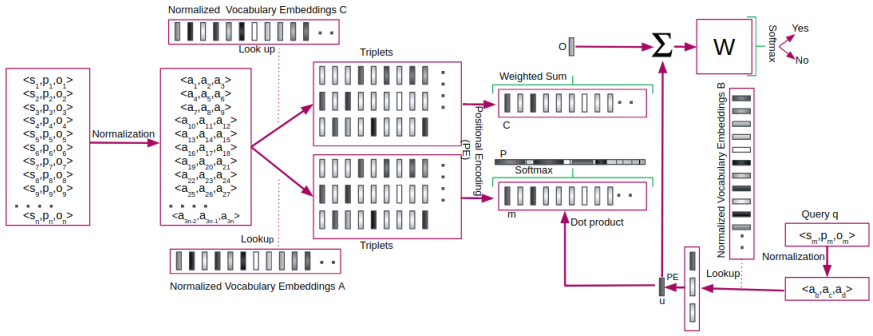

Figure 1: Diagram of the proposed model, for K=1

Model Description The model is augmented with an external memory component that stores the embeddings of the normalized triples in our KG. This memory is defined as an $n \times d$ tensor where n denotes the number of triples in the KG and d is the dimensionality of the embeddings. The KG is stored in the memory vectors from two continuous representations of m_i and c_i obtained from two input and output embedding matrices of A and C with size $d \times V$ where V is the size of vocabulary. Similarly, the query q is embedded via a matrix B to obtain an internal state u. In each reasoning step, those memory slots useful for finding the correct answers should have their contents retrieved. To enable this, we use an attention mechanism for q over memory input representations by taking an internal product followed by a softmax:

$$p_i = \text{Softmax}(u^T(m_i)) \tag{1}$$

$$\text{where} \qquad \text{Softmax}(a_i) = \frac{e^{(a_i)}}{\sum_j e^{(a_j)}}.$$

Equation (1) calculates a probability vector p over the memory inputs, the output vector o is then computed as the weighted sum of the transformed memory contents c_i with respect to their corresponding probabilities p_i by $o = \sum_i p_i c_i$. This describes the computation within a single hop. The internal state of the query vector updates for the next hop as $u^{k+1} = u^k + o^k$. The process repeats K times where K is the number of computational hops. The output of the K^{th} hop is used to predict the label $\hat{a}$ by passing o^K and u^K through a weight matrix of size $V \times d$ and a softmax:

$$\hat{a} = \text{Softmax}(W(u^{K+1})) = \text{Softmax}(W(u^k + o^k)).$$

Figure 1 shows the model for $K = 1$ (1 hop). The learning parameters are the matrices A, B, C, and W.

Memory Content There is a plethora of logics which could be used for our investigation. Here we use RDF. The RDF [27] is an established and widely used W3C standard for expressing

KGs. An RDF KG is a collection of statements stored as triples $(e1, r, e2)$ where $e1$ and $e2$ are called *subject* and *object*, respectively, while r is a relation binding $e1$ and $e2$ together. Statements can constitute base facts (logically speaking, in this case $e1$ and $e2$ would be constants, and r a binary predicate) or simple logical axioms (e.g., $e1$ and $e2$ could identify unary prediates or *classes*, and r would be class subsumption or material implication). Every entity in an RDF KG is represented by a unique Universal Resource Identifier (URI). We normalize these triples by systematically renaming all URIs which are not in the RDF/RDFS (Schema) namespaces as discussed previously. Each such URI is mapped to a set of arbitrary strings in a predefined set $\mathcal{A} = \{a_1, ..., a_n\}$, where n is taken as a training hyper-parameter giving an upper bound for the largest number of entities in a KG the system will be able to handle. Note that URIs in the RDF/RDFS namespaces are not renamed, as they are important for the deductive reasoning according to the RDF model-theoretic semantics. Consequently, each normalized RDF KG will be a collection of facts stored as triples $\{(a_i, a_j, a_k)\}$.

It is important to note that each symbol is mapped into an element of $\mathcal{A}$ regardless of its position in the triple, and whether it is a subject or an object or a predicate. Yet the position of an element within a triple is an important feature to consider. Thus we employ a positional encoding(PE) [19] to encode the position of each element within the triple. Let jth element of ith triple be $t_{i,j}$. This gives us memory vector representation of each triple as $m_i = \sum_j l_j \circ t_{i,j}$, where $\circ$ is the Hadamard (element-wise) product and l_j is a column vector with the structure $l_{k,j} = (1 - j/3) - (k(1 - 2j/3)/d)$ (assuming 1-based indexing), where d is the size of the embedding vector in the memory embedding matrix and the 3 in the denominator corresponds to the number of elements in an RDF triplet. Each memory slot thus represents the positional-weighted summation of each triplet. The positional encoding ensures that the order of the elements now affects the encoding of each memory slot.

4. Evaluation

The RDF semantics standard specification [28] describes a prodecural semantics based on 13 completion rules, which can be used to algorithmically compute logical consequences. The completion of an RDF KG is in general infinite because, by definition, there is an infinite set of facts (related to RDF-encodings of lists) which are always entailed – however for practical reasons, and as recommended in the standard specification, only certain finite subsets are computed as completions of RDF KGs, and we do the same.

Dataset There are many RDF KGs available on the World Wide Web that can be used to create our own dataset. For this purpose, we have collected RDF datasets from the Linked Data Cloud[1] and the Data Hub[2] to create our datasets.[3] Our training set (which by coincidence was based on RDF data conforming also to the OWL standard [24] and which we call an "OWL-centric" dataset) is comprised of a set of RDF KGs each of size 1,000 triples, sampled from populating around 20 OWL ontologies with different data. In order to test our model's ability

[1] https://lod-cloud.net/
[2] https://datahub.io/
[3] https://github.com/Monireh2/kg-deductive-reasoner

to generalize to completely different datasets, we have collected another dataset which we call the OWL-Centric Test Set. Furthermore, to assure our evaluation represents real-world RDF data completely independent of the training data, we have used almost all RDF KGs listed in a recent RDF quality survey [29]; we call this the Linked Data test set. Further, to test the limitations of our model on artificially difficult data, we have created a small synthetic dataset which requires long reasoning chains if done with a symbolic reasoner.

For each KG we have created the finite set of inferred triples using the Apache Jena[4] API. These inferred triples comprise our positive class instances. For generating invalid instances we used the following two methods. In the first, we generated non-inferred triples by random permutation of triple entities and removing those triples which were entailed. In the second scenario, which serves as our final quality check for not including trivially invalid triples in our dataset, we created invalid instances using the *rdf:type* predicate. More specifically, for each valid triple in the dataset, we replaced one of the elements (chosen randomly), with another random element which qualifies for being placed in that triple based on its *rdf:type* relationships. The datasets created by this strategy are marked with superscript "a" in Table 1.

Training Details Trainings were done over 10 epochs using the Adam optimizer with a learning rate of $\eta = 0.005$, a learning rate decay of $\eta/2$, and a batch size of 100 over triples. For the final batches of queries for each KG, we have used zero-padding to the maximum batch size of 100. The capacity of the external memory is 1,000 which is also the maximum size of our KGs. We used a linear starting of 1 epoch where we have removed the softmax from each memory layer except for the final layer. L2 norm clipping of max 40 was applied to the gradient. The memory input/output embeddings are vectors of size 20. The embedding matrices of A, B, and C therefore are of size $|V| \times d = 3,033 \times 20$, where 3,033 is the size of the normalized generic vocabulary plus RDF(S) namespace vocabulary. Unless otherwise mentioned, we have used $K = 10$. Adjacent weight sharing was used where the output embedding of one layer is the input embedding of the next one, as in $A^{k+1} = C$. Similarly, the answer prediction weight matrix W gets copied to the final output embedding C^K and query embedding is equal to the first layer input embedding as in $B = A^1$. All the weights are initialized by a Gaussian distribution with $\mu = 0$ and $\sigma = 0.1$. We would like to emphasize again that one and the same trained model was used in the evaluation over different test sets. We did not retrain, e.g., on Linked Data for the Linked Data test set.

Quantitative Results We now present and discuss our evaluation results. Our evaluation metrics are average of precision and recall and f-measure over all the KGs in the test set, obtained for both valid and invalid sets of triples. We also report the recall for the class of negatives (specificity) to interpret the result more carefully by counting the number of true negatives. Additionally, as mentioned earlier, we have done zero-padding for each batch of queries with size less than 100. This implies the need for introducing another class label for such zero paddings both in the training and test phase. 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.

[4]https://jena.apache.org/

To the best of our knowledge there is no architecture capable of conducting deductive reasoning on completely unseen RDF KGs. In addition, NTP and LTNs appear to have severe scalability issues, which means we cannot compare them to our system at scale. Neighbourhood approximated Neural Theorem Provers [30] heavily rely on entity embeddings, making it unsuitable for our goal, as discussed. That is why we have considered the non-normalized embedding version of our memory network as a baseline. Similarly, Graph-to-Graph learning architecture [1] is ontology-specific model. In fact, after training such model on one domain, you need to adapt the model hyper-parameters for another one and start the training from scratch on a different width model. Beside that, the Graph-to-Graph model is not scalable to large ontologies like DBpedia; instead it restricts the vocabulary to small restricted domain datasets. These inherent limitations for cross-ontology adaptation and the generative nature of the model (as opposed to classification in our setup) makes the comparison impossible.

Our technique shows a significant advantage over the baseline as shown in Table 1. A further even more important benefit of using our normalization model is its training time. In fact, this considerable time complexity difference is the result of the remarkable size difference of embedding matrices in the original and normalized cases. For instance, the size of embedding matrices to be learned by our algorithm for the normalized OWL-Centric dataset is $3,033 \times 20$ as opposed to $811,261 \times 20$ for the non-normalized one (and $1,974,062 \times 20$ for Linked Data which is prohibitively big). That has caused a remarkably high decrease in training time and space complexity and consequently has helped the scalability of our memory networks. In case of the OWL-Centric dataset, for instance, the space required for saving the normalized model is 80 times less than the intact model ($\approx 4GB$ after compression). Nevertheless, the normalized model is almost 40 times faster to train than the non-normalized one for this dataset. Our normalized model trained for just a day on OWL-Centric data but achieves better accuracy, whereas it trained on the same non-normalized dataset more than a week on a 12-core machine. Hence, the importance of using normalization cannot be emphasized enough.

To further get an idea of how our model performs on different data sources, we have applied our approach on multiple datasets with various characteristics. The result across all variations are given in Table 1. From this Table we can see that, apart from our strikingly good performance compared to the baseline, there are number of other interesting points: Our model gets even better results on the Linked Data task while it has trained on the OWL-Centric dataset. We hypothesize that this may be due to a generally simpler structure of Linked Data, but validating this will need further research.

The large portion of our few false negative instances come from the inability of our model to infer that all classes are subclass of themselves. Another interesting observation is the poor performance of our algorithm when it has trained on the OWL-Centric dataset and tested on a tricky version of the Linked Data. In that case our model has classified most of the triples to the "yes" class and this has led to low specificity (recall for "no" class) of 16%. This seems inevitable because in this case the negative instances bear close resemblance to the positives ones, making differentiation more challenging. However, training the model on the tricky OWL-Centric dataset has improved that by a substantial margin (more than three times). In case of our particularly challenging synthetic data, performance is not as good, and this may be due to the very different nature of this dataset which would require much longer reasoning chains than the non-synthetic data. Our training so far has only been done on real-world datasets; it may

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

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

Table 1
Experimental results of proposed model

be interesting to more closely investigate our approach when trained on synthetic data, but that was not the purpose of our study.

It appears natural to analyze the reasoning depth acquired by our network. We conjecture that reasoning depth acquired by the network will correspond both to (1) the number of layers in the deep network, and (2) the ratio of deep versus shallow reasoning required to perform 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 account. To gain a first understanding of what our model has learned in this respect, we have mimicked this symbolic reasoner behavior in creating our test set. We first started from our input KG K_0 in hop 0. We then produced, subsequently, KGs of K_1,..., K_n until no new triples are added (i.e. K_{n+1} is empty) by applying the RDF inference rules from the specification: The hop 0 dataset contains the original KG's triples in the inferred axioms, hop 1 contains the RDF(S) axiomatic triples. The real inference steps start with K_n where $n >= 2$. Table 2 summarizes our results in this setup. Unsurprisingly, we observe that result over our 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 2. 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 [31, 9, 32] also corroborate 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.

General Embeddings Visualization In order to gain some insight on the nature of our normalized embeddings, we have plotted a Principal Component Analysis (PCA) two-dimensional vector visualization of embeddings computed for the RDF(S) terms and all normalized words in the KGs, in Figure 2. The embeddings were fetched from the matrix B (embedding query lookup table) in the hop 1 of our model trained over the OWL-Centric dataset. Words are positioned in the plot based on the similarity of their embedding vectors. As anticipated, all the normalized

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

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

Table 2
F-measure and Data Distribution over each reasoning hop

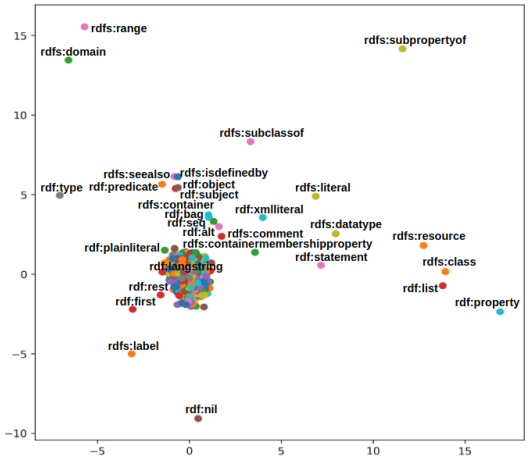

Figure 2: PCA projection of embeddings for the vocabulary

words tend to form one cluster as opposed to multiple ones. The PCA projection illustrates the ability of our model to automatically organize RDF(S) concepts and learn implicitly the relationships between them. For instance, *rdfs:domain* and *rdfs:range* have been located very close together and far from normalized entities. *rdf:subject*, *rdf:predicate* and *rdf:object* vectors are very similar, and the same for *rdf:seeAlso* and *rdf:isDefinedBy*. Likewise, *rdfs:container*, *rdf:bag*, *rdf:seq*, and *rdf:alt* are in the vicinity of each other. *rdf:langstring* is the only RDF(S) entity which is inside the normalized entities cluster. We believe that it is because *rdf:langString*'s domain and range is string and consequently it has mainly co-occurred with normalized instances in the KGs. Another possible reason for this is its low frequency in our data.

5. Conclusions and Future Work

We have demonstrated that a deep learning architecture based on memory networks and pre-embedding normalization is capable of learning how to perform deductive reason over previously unseen RDF KGs with high accuracy. We believe that we have thus provided the first

deep learning approach that is capable of high accuracy RDF deductive reasoning over previously unseen KGs. Normalization appears to be a critical component for high performance of our system. We plan to investigate its scalability and to adapt it to other, more complex, logics.

6. Acknowledgements

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

References

[1] B. Makni, J. Hendler, Deep Learning for Noise-tolerant RDFS Reasoning, Ph.D. thesis, Rensselaer Polytechnic Institute, 2018.

[2] S. Riedel, L. Yao, A. McCallum, B. M. Marlin, Relation extraction with matrix factorization and universal schemas, in: Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2013, pp. 74–84.

[3] R. Socher, D. Chen, C. D. Manning, A. Ng, Reasoning with neural tensor networks for knowledge base completion, in: Advances in neural information processing systems, 2013, pp. 926–934.

[4] B. Yang, W.-t. Yih, X. He, J. Gao, L. Deng, Embedding entities and relations for learning and inference in knowledge bases, arXiv preprint arXiv:1412.6575 (2014).

[5] K. Toutanova, D. Chen, P. Pantel, H. Poon, P. Choudhury, M. Gamon, Representing text for joint embedding of text and knowledge bases, in: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, 2015, pp. 1499–1509.

[6] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, G. Bouchard, Complex embeddings for simple link prediction, in: International Conference on Machine Learning, 2016, pp. 2071–2080.

[7] B. Peng, Z. Lu, H. Li, K.-F. Wong, Towards neural network-based reasoning, arXiv preprint arXiv:1508.05508 (2015).

[8] R. Das, A. Neelakantan, D. Belanger, A. McCallum, Chains of reasoning over entities, relations, and text using recurrent neural networks, arXiv preprint arXiv:1607.01426 (2016).

[9] T. Rocktäschel, S. Riedel, End-to-end differentiable proving, in: Advances in Neural Information Processing Systems, 2017, pp. 3788–3800.

[10] W. Ling, T. Luís, L. Marujo, R. F. Astudillo, S. Amir, C. Dyer, A. W. Black, I. Trancoso, Finding function in form: Compositional character models for open vocabulary word representation, arXiv preprint arXiv:1508.02096 (2015).

[11] T. Kudo, Subword regularization: Improving neural network translation models with multiple subword candidates, arXiv preprint arXiv:1804.10959 (2018).

[12] R. Sennrich, B. Haddow, A. Birch, Neural machine translation of rare words with subword units, arXiv preprint arXiv:1508.07909 (2015).

[13] D. Bahdanau, T. Bosc, S. Jastrzębski, E. Grefenstette, P. Vincent, Y. Bengio, Learning to compute word embeddings on the fly, arXiv preprint arXiv:1706.00286 (2017).
[14] M. Eric, C. D. Manning, A copy-augmented sequence-to-sequence architecture gives good performance on task-oriented dialogue, arXiv preprint arXiv:1701.04024 (2017).
[15] D. Raghu, N. Gupta, et al., Hierarchical pointer memory network for task oriented dialogue, arXiv preprint arXiv:1805.01216 (2018).
[16] A. Talman, S. Chatzikyriakidis, Testing the generalization power of neural network models across nli benchmarks, arXiv preprint arXiv:1810.09774 (2018).
[17] W. Xiong, T. Hoang, W. Y. Wang, Deeppath: A reinforcement learning method for knowledge graph reasoning, arXiv preprint arXiv:1707.06690 (2017).
[18] A. Bordes, J. Weston, R. Collobert, Y. Bengio, Learning structured embeddings of knowledge bases, in: AAAI, AAAI Press, 2011.
[19] S. Sukhbaatar, J. Weston, R. Fergus, et al., End-to-end memory networks, in: Advances in neural information processing systems, 2015, pp. 2440–2448.
[20] R. Evans, D. Saxton, D. Amos, P. Kohli, E. Grefenstette, Can neural networks understand logical entailment?, arXiv preprint arXiv:1802.08535 (2018).
[21] L. Serafini, A. S. d. Garcez, Learning and reasoning with logic tensor networks, in: Conference of the Italian Association for Artificial Intelligence, Springer, 2016, pp. 334–348.
[22] F. Bianchi, P. Hitzler, On the capabilities of logic tensor networks for deductive reasoning., in: AAAI Spring Symposium: Combining Machine Learning with Knowledge Engineering, 2019.
[23] P. Hohenecker, T. Lukasiewicz, Ontology reasoning with deep neural networks, arXiv preprint arXiv:1808.07980 (2018).
[24] P. Hitzler, M. Krötzsch, B. Parsia, P. F. Patel-Schneider, S. Rudolph (Eds.), OWL 2 Web Ontology Language: Primer (Second Edition), W3C Recommendation 11 December 2012, 2012. Available from http://www.w3.org/TR/owl2-primer/.
[25] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, O. Yakhnenko, Translating embeddings for modeling multi-relational data, in: Advances in neural information processing systems, 2013, pp. 2787–2795.
[26] Z. Wang, J. Zhang, J. Feng, Z. Chen, Knowledge graph embedding by translating on hyperplanes., in: AAAI, volume 14, 2014, pp. 1112–1119.
[27] P. Hitzler, M. Krotzsch, S. Rudolph, Foundations of semantic web technologies, Chapman and Hall/CRC, 2009.
[28] P. J. Hayes, P. F. Patel-Schneider (Eds.), RDF 1.1 Semantics, 2014. Available from http://www.w3.org/TR/rdf11-mt/.
[29] S. Sam, P. Hitzler, K. Janowicz, On the quality of vocabularies for linked dataset papers published in the semantic web journal, Semantic Web 9 (2018) 207–220.
[30] P. Minervini, M. Bosnjak, T. Rocktäschel, E. Grefenstette, S. Riedel, Scalable neural theorem proving on knowledge bases and natural language (2018).
[31] R. Das, S. Dhuliawala, M. Zaheer, L. Vilnis, I. Durugkar, A. Krishnamurthy, A. Smola, A. McCallum, Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning, arXiv preprint arXiv:1711.05851 (2017).
[32] F. Yang, Z. Yang, W. W. Cohen, Differentiable learning of logical rules for knowledge base reasoning, in: Advances in Neural Information Processing Systems, 2017, pp. 2319–2328.

On the Generalization Capability of Memory Networks for Reasoning

Monireh Ebrahimi [1] **Md Kamruzzaman Sarker** [1] **Federico Bianchi** [1 2] **Ning Xie** [1] **Aaron Eberhart** [1]
Derek Doran [1] **Pascal Hitzler** [1]

Abstract

A significant and recent development in neural-symbolic learning are deep neural networks that can reason over symbolic knowledge bases (KBs) and perform scalable reasoning tasks. Initial neural-symbolic systems that can deduce the entailment of a KB have been presented, but they are theoretically and practically limited: current systems learn fact relations and entailment patterns specific to a particular KB and hence do not truly learn to reason, and must be retrained for each KB they are tasked with entailing. To address this generalization limitation, we propose a differentiable end-to-end deep memory network that learns over abstract, generic symbols to discover entailment patterns common to any reasoning task. A key component of the system is a simple but highly effective normalization process for continuous representation learning of KB entities within memory networks. Our results show how the model, trained over a set of KBs, can effectively entail facts from test KBs, even when the domain of test KBs is completely different from the training KBs.

1. Background

With the recent revival of interest in artificial neural networks, they have been applied vastly for the completion of KBs. These methods (Chang et al., 2014; Nickel et al., 2012; Riedel et al., 2013; Socher et al., 2013; Toutanova et al., 2015; Trouillon et al., 2016; Yang et al., 2014) heavily rely on the subsymbolic representation of entities and relations learned through maximization of a scoring objective function over valid factual triples. Thus, the current success of such models hinges primarily on the power of those subsymbolic continuous real-valued representations in encoding the similarity/relatedness of entities and relations. Recent attempts have focused on neural multi-hop reasoners (Das et al., 2016; Neelakantan et al., 2015; Peng et al., 2015; Shen et al., 2017; Weissenborn, 2016) to equip the model to deal with more complex reasoning. More recently, a Neural Theorem Prover (Rocktäschel & Riedel, 2017) has been proposed in an attempt to take advantage of both symbolic and sub-symbolic reasoning.

Despite their success, the main restriction common to neural reasoners is that they are unable to generalize to new domains. This inherent limitation follows from both the representation functions used and the learning process. The major issue comes from the mere reliance of these models on the representation of entities learned during the training or in the pre-training phase stored in a lookup table. Consequently, these models have difficulty to deal with out-of-vocabulary(OOV) entities. Although the small-scale OOV problem has been addressed in part in the natural language processing (NLP) domain by taking advantage of character-level embedding (Ling et al., 2015), learning embeddings on the fly by leveraging text descriptions or spelling (Bahdanau et al., 2017), copy mechanism (Eric & Manning, 2017) or pointer networks (Raghu et al., 2018), still these solutions are insufficient for transferring purposes. (Talman & Chatzikyriakidis, 2018) shows the success of natural language inference (NLI) methods is heavily benchmark specific. An even greater source of concern is that reasoning in most of the above sub-symbolic approaches hinges more on the notion of similarity and geometric-based proximity of real-valued vectors (induction) as opposed to performing transitive reasoning (deduction) over them. In short, to the best of our knowledge, to date, there is no sub-symbolic reasoning work, which is able to transfer the learning capability from one KB to unseen one. In fact, since previous works have focused to conduct reasoning on the unseen part of the same KB, they have tried to gain generalization ability through induction and robustness to missing edges(Guu et al., 2015) as opposed to deduction. Likewise, recent years have seen some progress in zero-shot relation learning in sub-symbolic reasoning domain(Neelakantan et al., 2015; Rocktäschel et al., 2015; Xiong et al., 2017). Zero-shot learning refers to the ability of the model to infer new relations where that relation has not been seen before in

[1]Department of Computer Science & Engineering, Wright State University [2]University of Milan - Bicocca. Correspondence to: Monireh Ebrahimi <ebrahimi.2@wright.edu>.

ICML 2019 Workshop on Understanding and Improving Generalization in Deep Learning, Long Beach, California, 2019.

training set(Bordes et al., 2011). This generalization capability is still quite limited and fundamentally different from our work in terms of both methodology and purpose.

Inspired by these observations, we take a different approach by investigating the emulation of deductive symbolic reasoning using memory networks. Memory networks (Weston et al., 2014) are a class of learning models capable of conducting multiple computational steps over an explicit memory component before returning an answer. They have been recently applied successfully to a range of NLP tasks such as question answering (Hill et al., 2015; Sukhbaatar et al., 2015), language modeling (Sukhbaatar et al., 2015), and dialogue tasks (Bordes et al., 2016; Dodge et al., 2015). End-to-end memory networks (MemN2N) (Sukhbaatar et al., 2015) are a less-supervised, more general version of these networks, applicable to the settings where labeled supporting memories are not available. We have selected such networks since we believe that they are a primary candidate to perform well for 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 NLI is also promising: while NLI 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 get insights into the "reasoning" underlying the network output, as we will be able to see which pieces of the input KB are taken into account at each step.

The main contribution of this paper, however, is a recipe involving a simple but effective KB triple normalization before learning their representation within a MemN2N. To perform logical inference in more abstract level, and thereby facilitating the transfer of reasoning expertise from one KB to another, the normalization maps entities and predicates in a knowledge to a generic vocabulary. Facts in additional KBs are normalized using the same vocabulary, so that the network does not learn to overfit its learning to entity and predicate names in a specific KB. This emulates symbolic reasoning by neural embeddings as the actual names (as strings) of entities from the underlying logic such as variables, constants, functions, and predicates are insubstantial for logical entailment in the sense that a consistent renaming across a theory does not change the set of entailed formulas (under the same renaming). Thanks to the term-agnostic feature of our representation, we are able to create a reasoning system capable of performing reasoning over an unseen set of vocabularies in the test phase.

2. Problem Formulation

We wish to train a neural model that will learn to reason over one set of theories, and can then transfer that learning to new theories over the same logic. One of the key obstacles we face with our task is to understand how to represent training and test data. To use standard neural approaches, symbols will have to be represented over the real coordinate space R as vectors (points), matrices or tensors. Many embeddings for KBs have been proposed (Bordes et al., 2013; Lin et al., 2015; Trouillon et al., 2016; Wang et al., 2014), but we are not aware of an existing embedding that captures what seems important for the deductive reasoning scenario. Indeed, the prominent use case explored for KB embeddings is not deductive in nature; rather, it concerns the problem of the discovery or suggestion of additional links or edges in the graph, together with appropriate edge labels. In this link discovery setting, the actual labels for nodes or edges in the graph, and as such their commonsense meanings, are likely important, and most existing embeddings reflect this. However, for deductive reasoning the names of entities are insubstantial and should not be captured by an embedding. Another inherent problem in the use of such representations across KBs is the OOV problem. While a word lookup table can be initialized with vectors in an unsupervised task or during training of the reasoner, it still cannot generate vector representations for unseen terms. It is further impractical to store the vectors of all words when vocabulary size is huge (Ling et al., 2015). Similarly, memory networks usually rely on word-level embedding lookup tables, i.e., learned with the underlying rationale that words that occur in similar supervised scenarios should be represented by similar vectors. That is why they are known to have difficulties dealing with OOV, as a word lookup table cannot provide a representation for the unseen, and thus has difficulty to do NLI over new words (Bahdanau et al., 2017), and for us this would pose a challenge in the transfer to new KBs.

We thus need representations that are agnostic to the terms used as primitives in the KB. To build such a representation, we use syntactic normalization: a renaming of primitives from the logical symbols to a set of predefined entity names that are used across different normalized theories. By randomly assigning the mapping for the renaming, the network's learning will be based on the structural information within the theories, and not on the actual names of the primitives. Note that this normalization not only plays the role of "forgetting" irrelevant label names, but also makes it possible to transfer learning from one KB to the other. Indeed, the network can be trained with many KBs, and then subsequently tested on completely new ones.

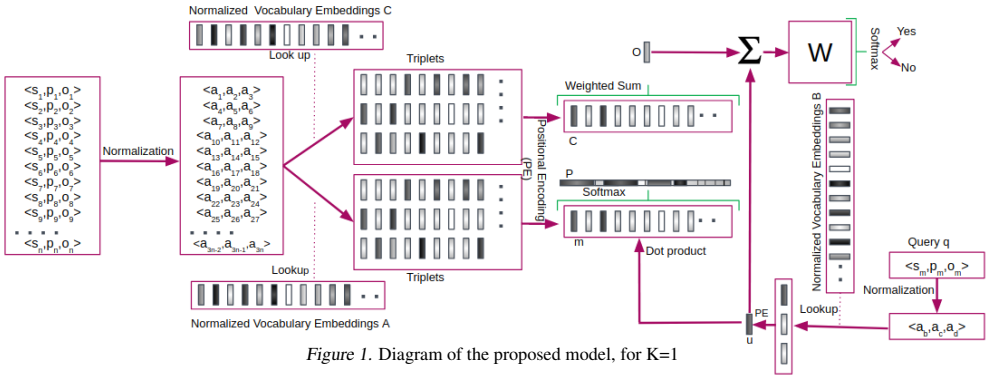

Figure 1. Diagram of the proposed model, for K=1

3. Model Architecture

We consider a model architecture that adapts the MemN2N with fundamental alterations necessary for abstract reasoning. A high-level view of our model is shown in Figure 1. It takes a discrete set G of normalized RDFS statements (called *triples*) $t_1, ..., t_n$ that are stored in memory, a query q, and outputs a "yes" or "no" answer to determine if q is entailed by G. Each of the normalized t_i and q contains symbols coming from a general dictionary with V normalized words shared among all of the normalized RDFS theories in both training and test sets. The model writes all triples to the memory and then calculates a continuous embedding for G and q. Through multiple hop attention over those continuous representations, the model then classifies the query. The model is trained by back-propagation of error from output to the input through multiple memory accesses. More Specifically, the model is augmented with an external memory that stores the embeddings of the normalized triples in our KB. This memory is defined as an $n \times d$ tensor where n denotes the number of triples in the KB and d is the dimensionality of the embeddings. The KB is stored in the memory vectors from two continuous representations of m_i and c_i obtained from two input and output embedding matrices of A and C with size $d \times V$ where V is the size of vocabulary. Similarly, the query q is embedded via a matrix B to obtain an internal state u. In each reasoning step, those memory slots useful for finding the correct answers should have their contents retrieved. To enable this, we use an attention mechanism for q over memory input representations by taking an internal product followed by a softmax:

$$p_i = \text{Softmax}(u^T(m_i)) \quad (1)$$

Equation (1)calculates a probability vector p over the memory inputs, the output vector o is computed as the weighted sum of the transformed memory contents c_i with respect to their corresponding probabilities p_i by $o = \sum_i p_i c_i$. This describes the computation within a single hop. The internal state of the query vector updates for the next hop as $u^{k+1} = u^k + o^k$. The process repeats K times where K is the number of computational hops. The output of the K^{th} hop is used to predict the label $\hat{a}$ by passing o^K and u^K through a weight matrix of size $V \times d$ and a softmax:

$$\hat{a} = \text{Softmax}(W(u^{K+1})) = \text{Softmax}(W(u^k + o^k)).$$

Figure 1 shows the model for $K = 1$. The parameters to be learned by backpropagation are A, B, C, and W matrices.

Memory Content An RDFS KB is a collection of statements stored as triples $(e1, r, e2)$ where $e1$ and $e2$ are called *subject* and *object*, respectively, while r is a relation binding $e1$ and $e2$ together. Every entity in an RDFS KB is represented by a unique Universal Resource Identifier (URI). We normalize these triples by systematically renaming all URIs which are not in the RDF or RDFS namespaces as discussed previously. Each such URI is mapped to a set of arbitrary strings in a predefined set $\mathcal{A} = \{a_1, ..., a_n\}$, where n is taken as a training hyper-parameter giving an upper bound for the largest number of entities in a KB the system will be able to handle. Note that URIs in the RDF/RDFS namespaces are not renamed, as they are important for the deductive reasoning according to the RDFS model-theoretic semantics. Consequently, each normalized RDFS KB will be a collection of facts stored as triples $\{(a_i, a_j, a_k)\}$.

It is important to note that each symbol is mapped into an element of $\mathcal{A}$ regardless of its position in the triple. Yet the position of an element within a triple is an important feature to consider. Thus we employ a positional encoding (PE) (Sukhbaatar et al., 2015) to encode the position of each element within the triple. Each memory slot thus represents the positional-weighted summation of each triplet. The PE ensures that the order of the elements now affects the encoding of each memory slot m_i.

Training Dataset	Test Dataset	Accuracy	
		Our model	Baseline
OWL-Centric	Linked Data	**96**	43
OWL-Centric(90%)	OWL-Centric (10%)	**90**	82
OWL-Centric	OWL-Centric Test Set *	**69**	61
OWL-Centric	Synthetic Data	**52**	48

* Completely Different Domain.

Table 1. Experimental results of the proposed model

4. Evaluation

Dataset We have collected RDFS datasets from the Linked Data Cloud[1] and the Data Hub[2].Our training set ("OWL-centric") is comprised of a set of RDFS KBs each of size 1,000 triples, sampled from populating around 20 OWL ontologies with different data. In order to test our model's ability to generalize to completely different datasets, we have collected another dataset called the OWL-Centric Test Set. To assure our evaluation represents real-world RDFS data completely independent of the training data, we have used almost all RDFS KBs listed in (Sam et al., 2018); we call this the Linked Data test set. Furthermore, to test the limitations of our model on artificially difficult data, we have created a small synthetic dataset which requires long reasoning chains if done with a symbolic reasoner. For each KB we have created the finite set of inferred triples using the Apache Jena[3] API. These inferred triples comprise our positive class instances. We generated non-inferred triples by random permutation of triple entities and removing those triples which were entailed.

Results Trainings were done over 10 epochs using the Adam optimizer with a learning rate of $\eta = 0.005$, a learning rate decay of $\eta/2$, and a batch size of 100 over triples. All embeddings are vectors of size 20. We have used $K = 10$. Adjacent weight sharing was used where the output embedding of one layer is the input embedding of the next one. All the weights are initialized by a Gaussian distribution with $\mu = 0$ and $\sigma = 0.1$. Here we report the average accuracy over all the KBs in the test set, obtained for both valid and invalid sets of triples. We have considered the non-normalized embedding version of our memory network as a baseline. Our technique shows a significant advantage over the baseline as shown in Table 1. A further even more important benefit of using our normalization model is its training time. In fact, this considerable time complexity difference is the result of the remarkable size difference of embedding matrices in the original and normalized cases. For instance, the size of embedding matrices to be learned by our algorithm for the normalized OWL-Centric dataset is $3,033 \times 20$ as opposed to $811,261 \times 20$ for the non-normalized one (and $1,974,062 \times 20$ for Linked Data which is prohibitively big). That has caused a remarkably high decrease in training time and space complexity. In case of the OWL-Centric dataset, for instance, the space required for saving the normalized model is 80 times less than the intact model. Likewise, the normalized model is almost 40 times faster to train than the non-normalized one for this dataset. Hence, the importance of using normalization cannot be emphasized enough.

[1]https://lod-cloud.net/

[2]https://datahub.io/

[3]https://jena.apache.org/

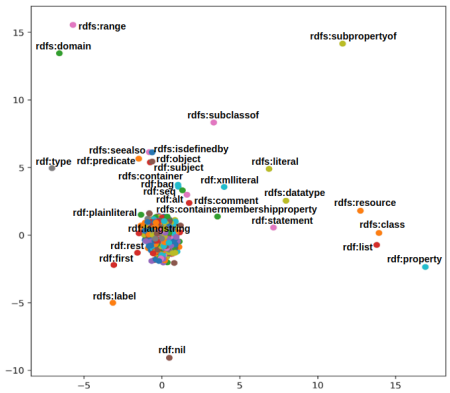

Figure 2. PCA projection of embeddings for the whole vocabulary

General Embeddings Visualization We have plotted a Principal Component Analysis (PCA) two-dimensional vector visualization of embeddings computed for the RDF(S) terms and all normalized words in the KBs, in Figure 2. The embeddings were fetched from the matrix B (embedding query lookup table) in the hop 1 of our model trained over the OWL-Centric dataset. Words are positioned in the plot based on the similarity of their embedding vectors. As anticipated, all the normalized words tend to form one cluster as opposed to multiple ones. The PCA projection illustrates the ability of our model to automatically organize RDF(S) concepts and learn implicitly the relationships between them. For instance, rdfs:domain and rdfs:range have been located very close together and far from normalized entities. rdf:subject, rdf:predicate and rdf:object vectors are very similar, and the same for rdf:seesAlso and rdf:isDefinedBy. Likewise, rdfs:container, rdf:bag, rdf:seq, and rdf:alt are in the vicinity of each other.

5. Conclusions and Future Work

We have demonstrated that a deep learning architecture based on memory networks and pre-embedding normalization is capable of learning how to perform deductive reasoning over previously unseen RDFS KBs with high accuracy. We believe that we have thus provided the first deep learning approach that is capable of high accuracy RDFS deductive reasoning over previously unseen KBs. Normalization appears to be a critical component for high performance of our system. This obviates the need for supervised retraining over the task of interest or unsupervised pretraining over the external source of data for learning the representations when encountered with a new KB. It also provides insights into representation learning for rare or OOV words, transfer learning, zero-shot learning, and domain adaptation in the reasoning domain. We plan to properly investigate scalability of our approach and to adapt it to other, more complex, logics.

References

Bahdanau, D., Bosc, T., Jastrzebski, S., Grefenstette, E., Vincent, P., and Bengio, Y. Learning to compute word embeddings on the fly. *arXiv preprint arXiv:1706.00286*, 2017.

Bordes, A., Weston, J., Collobert, R., Bengio, Y., et al. Learning structured embeddings of knowledge bases. In *AAAI*, volume 6, pp. 6, 2011.

Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., and Yakhnenko, O. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pp. 2787–2795, 2013.

Bordes, A., Boureau, Y.-L., and Weston, J. Learning end-to-end goal-oriented dialog. *arXiv preprint arXiv:1605.07683*, 2016.

Chang, K.-W., Yih, S. W.-t., Yang, B., and Meek, C. Typed tensor decomposition of knowledge bases for relation extraction. 2014.

Das, R., Neelakantan, A., Belanger, D., and McCallum, A. Chains of reasoning over entities, relations, and text using recurrent neural networks. *arXiv preprint arXiv:1607.01426*, 2016.

Dodge, J., Gane, A., Zhang, X., Bordes, A., Chopra, S., Miller, A., Szlam, A., and Weston, J. Evaluating prerequisite qualities for learning end-to-end dialog systems. *arXiv preprint arXiv:1511.06931*, 2015.

Eric, M. and Manning, C. D. A copy-augmented sequence-to-sequence architecture gives good performance on task-oriented dialogue. *arXiv preprint arXiv:1701.04024*, 2017.

Guu, K., Miller, J., and Liang, P. Traversing knowledge graphs in vector space. *arXiv preprint arXiv:1506.01094*, 2015.

Hill, F., Bordes, A., Chopra, S., and Weston, J. The goldilocks principle: Reading children's books with explicit memory representations. *arXiv preprint arXiv:1511.02301*, 2015.

Lin, Y., Liu, Z., Sun, M., Liu, Y., and Zhu, X. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, volume 15, pp. 2181–2187, 2015.

Ling, W., Luís, T., Marujo, L., Astudillo, R. F., Amir, S., Dyer, C., Black, A. W., and Trancoso, I. Finding function in form: Compositional character models for open vocabulary word representation. *arXiv preprint arXiv:1508.02096*, 2015.

Neelakantan, A., Roth, B., and McCallum, A. Compositional vector space models for knowledge base completion. *arXiv preprint arXiv:1504.06662*, 2015.

Nickel, M., Tresp, V., and Kriegel, H.-P. Factorizing yago: scalable machine learning for linked data. In *Proceedings of the 21st international conference on World Wide Web*, pp. 271–280. ACM, 2012.

Peng, B., Lu, Z., Li, H., and Wong, K.-F. Towards neural network-based reasoning. *arXiv preprint arXiv:1508.05508*, 2015.

Raghu, D., Gupta, N., et al. Hierarchical pointer memory network for task oriented dialogue. *arXiv preprint arXiv:1805.01216*, 2018.

Riedel, S., Yao, L., McCallum, A., and Marlin, B. M. Relation extraction with matrix factorization and universal schemas. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 74–84, 2013.

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

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

Sam, S., Hitzler, P., and Janowicz, K. On the quality of vocabularies for linked dataset papers published in the semantic web journal. *Semantic Web*, 9(2):207–220, 2018.

Shen, Y., Huang, P.-S., Gao, J., and Chen, W. Reasonet: Learning to stop reading in machine comprehension. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1047–1055. ACM, 2017.

Socher, R., Chen, D., Manning, C. D., and Ng, A. Reasoning with neural tensor networks for knowledge base completion. In *Advances in neural information processing systems*, pp. 926–934, 2013.

Sukhbaatar, S., Weston, J., Fergus, R., et al. End-to-end memory networks. In *Advances in neural information processing systems*, pp. 2440–2448, 2015.

Talman, A. and Chatzikyriakidis, S. Testing the generalization power of neural network models across nli benchmarks. *arXiv preprint arXiv:1810.09774*, 2018.

Toutanova, K., Chen, D., Pantel, P., Poon, H., Choudhury, P., and Gamon, M. Representing text for joint embedding of text and knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1499–1509, 2015.

Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., and Bouchard, G. Complex embeddings for simple link prediction. In *International Conference on Machine Learning*, pp. 2071–2080, 2016.

Wang, Z., Zhang, J., Feng, J., and Chen, Z. Knowledge graph embedding by translating on hyperplanes. In *AAAI*, volume 14, pp. 1112–1119, 2014.

Weissenborn, D. Separating answers from queries for neural reading comprehension. *arXiv preprint arXiv:1607.03316*, 2016.

Weston, J., Chopra, S., and Bordes, A. Memory networks. corr abs/1410.3916, 2014.

Xiong, W., Hoang, T., and Wang, W. Y. Deeppath: A reinforcement learning method for knowledge graph reasoning. *arXiv preprint arXiv:1707.06690*, 2017.

Yang, B., Yih, W.-t., He, X., Gao, J., and Deng, L. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*, 2014.