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.

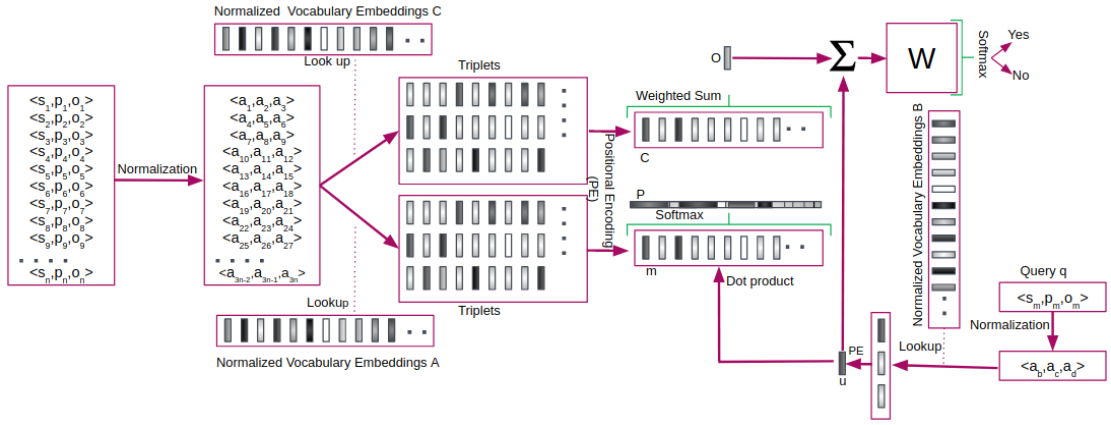

Figure 1: Diagram of the proposed model, for K=1

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

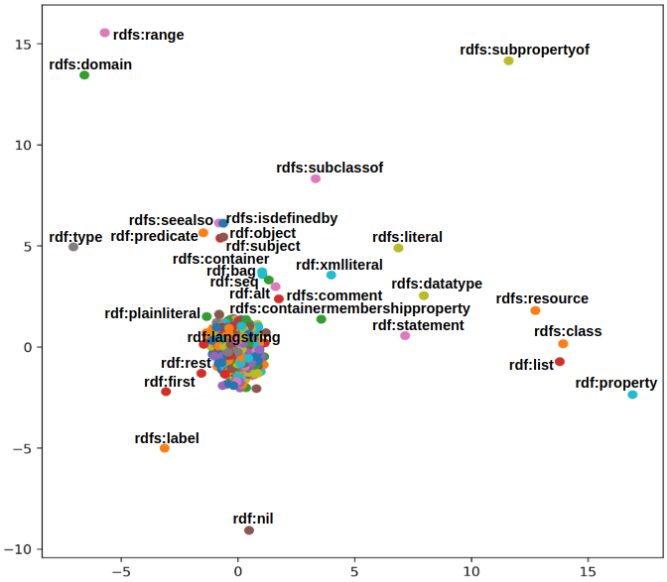

Figure 2: PCA projection of embeddings for the vocabulary

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.