Reflections on: Deep learning for noise-tolerant RDFS reasoning *

Bassem Makni[1] and James Hendler[2]

[1] IBM Research, 1101 Kitchawan Rd, Yorktown Heights, NY 10598, USA
bassem.makni@ibm.com
[2] TWC, Rensselaer Polytechnic Institute, 110 8th St, Troy, NY 12180, USA
hendler@cs.rpi.edu

Abstract. Since the 2001 envisioning of the Semantic Web (SW), the main research focus in SW reasoning has been on the soundness and completeness of reasoners. While these reasoners assume the veracity of input data, the reality is that the Web of data is inherently noisy. Although there has been recent work on noise-tolerant reasoning, it has focused on type inference rather than full RDFS reasoning. Even though RDFS closure generation can be seen as a Knowledge Graph (KG) completion problem, the problem setting is different— making KG embedding techniques that were designed for link prediction not suitable for RDFS reasoning. This paper documents a novel approach that extends noise-tolerance in the SW to full RDFS reasoning. Our embedding technique— that is tailored for RDFS reasoning— consists of layering RDF graphs 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 translation of these graph words sequences, achieved through neural machine translation. Our evaluation on LUBM1 synthetic dataset shows 97% validation accuracy and 87.76% on a subset of DBpedia while demonstrating a noise-tolerance unavailable with rule-based reasoners.

1 Introduction

The Web is inherently noisy and as such its extension is noisy as well. This noise is as a result of inevitable human error when creating the content, designing the tools that facilitate the data exchange, conceptualizing the ontologies that allow machines to understand the data content, mapping concepts from different ontologies, etc. This paper documents a novel approach that takes previous research efforts on noise-tolerance to the next level of full RDF Schema (RDFS) reasoning. The proposed approach utilizes the recent advances in deep learning- that showed robustness to noise in other machine learning applications such as computer vision and natural language understanding- for semantic reasoning. The first step towards bridging the Neural-Symbolic gap for RDFS reasoning is

* The code, models and datasets for this paper are available at: https://github.com/Bassem-Makni/NMT4RDFS

to represent Resource Description Framework (RDF) graphs in a format that can be fed to neural networks. The most intuitive representation to use is graph representation. However, RDF graphs differ from simple graphs as defined in the graph theory in a number of ways. The different graph models for RDF in the literature were neither designed for RDFS reasoning requirements nor are they suitable for neural network input. The proposed graph model for RDF consists of layering RDF graphs and encoding them in the form of 3D adjacency matrices. Each layer layout in the 3D adjacency matrices forms what we termed as a *graph word*. Every input graph and its corresponding inference are then represented as sequences of graph words. The RDFS inference becomes equivalent to the translation of graph words that is achieved through neural network translation. The main contributions in this paper are:

- **Noise Intolerance Conditions.** A taxonomy for noise types in SW data according to the impact of the noise on the inference is drawn along with the necessary conditions for a noise type to be propagable (i.e affect the inference).
- **Layered Graph Model for RDF.** We propose a layered graph model for RDF that is tailored for RDFS reasoning.
- **Graph Words.** Using the layered graph model, we propose a novel way of representing RDF graphs as a sequence of graph words.
- **Graph-to-Graph Learning.** By representing RDF graphs as a sequence of graph words, we were able to use neural network translation techniques for translation of graph words.
- **Full RDFS reasoning with noise tolerance.** Our evaluation shows not only comparable results with rule-based reasoners on intact data but also exceptional noise-tolerance compared to them: 99% for the deep reasoner *vs* 0% (by design) for Jena in the UGS_{100} dataset.

In Section 2, we use three aspects to position our research with respect to the related work. Section 3 draws a taxonomy for noise types in SW data and illustrates the process of ground truthing and noise induction for LUBM and a subset of DBpedia. We describe the layered graph model for RDF in Section 4. Then, the overall approach including the creation of the RDF tensors and the RDF graph words is presented in Section 5. The results of the experiments are described in the Section 6.

2 Background and Problem Statement

In this section we use three aspects to position our research with respect to related work:

- **Noise handling strategies: active vs adaptive**. Active noise handling consists of detecting noise and cleansing the data before performing any tasks that might be affected by the presence of noise, while adaptive noise handling approaches focus rather on building techniques that are noise-tolerant. The research described in this paper falls into the latter category.

- **Knowledge graph completion categories: schema-guided vs data-driven** RDFS closure can be seen as a Knowledge Graph Completion (KGC) problem— multi-relational link prediction problem in particular— where each RDFS rule generates different types of links. We refer to the RDFS closure computation as *schema-guided KGC* because the links are generated according to the ontology (TBox), unlike *data-driven KGC* where the links are predicted based on the analysis of the existing links in the KG.
- **Graph embedding output: Node/Edge embedding vs whole-graph embedding** Graph embedding approaches can be classified using several criteria. One particular criterion of interest is the "problem setting" [3], where the type of graph input as well as the embedding output are used to classify the embedding approach. The graph input can either be homogeneous or heterogeneous—where there are multiple types of nodes and/or multiple types of edges— which is the case for RDF graphs.
  The majority of graph embedding approaches yield node representation in a low dimensional space. This is why graph embedding and node embedding are often used interchangeably. However, there are other types of graph embedding outputs such as edge embedding and whole graph embedding— where the output is a vector representation of the whole graph not only node or edge vectors. The embedding vectors of similar graphs should be neighbors in the embedding space. The embedding of RDF graphs— in order to learn their inference— falls under this category.

2.1 Problem statement

Existing embedding techniques for KGs were not designed for RDFS reasoning and they raise two main challenges if they were to be used for this task.

1. The first challenge is the need to check the validity of every possible triple using the scoring function in order to generate the full materialization.
2. The second challenge is the embedding of the relations that are seen only in the inference such as the super-properties.

3 Ground Truthing and Noise Induction

For this research, the input is from one of two types of datasets: a synthetic dataset from LUBM and a real-world dataset from DBpedia [1]. The inferential target for these datasets is set using a rule-based SW reasoner (Jena [4]). Essentially, the goal for the deep reasoner is to learn the mapping between input RDF graphs and their entailed graphs in the presence of noise. Thus, noise was induced in the synthetic dataset to test the noise-tolerance of the deep reasoner.

3.1 Taxonomy of Semantic Web noise types

The literature contains a few taxonomies for the types of noise that can impact RDF graphs; however they are not drawn with respect to the impact of the noise on the inference. The taxonomy illustrated in Fig. 1 serves this purpose.

TBox Noise is the type of noise that resides within the ontology, such as in the class hierarchy or domain and range properties. This type of noise impacts inference over the whole dataset. Reasoning with tolerance to TBox noise is outside the scope of this research because using rule-based reasoners for ground truthing with noise in the TBox biases the whole ground truth. Non-propagable noise consists of any corrupted triple in the input graph that does not have any impact on the inference—for example when the original triple does not generate any inference nor does the corrupted triple. Propagable noise, on the other hand, is a corrupted triple in the input graph that changes the inference. The necessary conditions for RDFS rules to generate a noisy inference from a corrupted triple are identified in the extended journal paper.

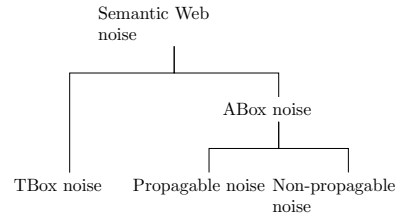

Fig. 1: Semantic Web noise taxonomy

3.2 Ground-Truthing in LUBM1

LUBM1 was generated according to the LUBM [6] ontology and contains 17,189 subject-resources within 15 classes. Let R be the set of these subject-resources. For each resource r in R, a graph g is built by running the SPARQL DESCRIBE query. Let G be the set of graphs g obtained after this step. For each graph g in G, the RDFS inferential closure is generated according to the LUBM ontology using Jena. Let I be the set of inference graphs. Finally, G and I are split into training (G_train,I_train), validation and testing sets using a stratified splitting technique where the resource class is used as the label for the stratification. The input of the supervised learning algorithm is the set of graphs G_train, the target is their corresponding inference graphs I_train and the goal is to learn the inference generation.

Noise Induction in LUBM1: In [11], a methodology for noise induction in LUBM was proposed in which three datasets were constructed by corrupting type assertions according to a given noise level. In RATA dataset, instances of type *TeachingAssistant* were corrupted to be of type *ResearchAssistant*, which is non-propagable because both concepts are sub-classes of the concept *Person*. In UGS, instances of type *GraduateStudent* were corrupted to be of type *University*, which is propagable by the *RDFS9* rule because these concepts are not siblings. In GCC, instances of type *Course* were corrupted to be of type *GraduateCourse*. This type of noise is also non-propagable.

As [11] focuses only on noisy type assertions, two additional datasets were created with noisy property assertions for the purpose of this research. In TEPA, the property *publicationAuthor* is corrupted to be *teachingAssistantOf*, which is propagable by *RDFS2* and *RDFS3* rules as the two properties have different

domains and ranges. In WOAD, the property *advisor* is corrupted to be *worksFor*. This noise is non-propagable as the property *worksFor* does not have any domain or range specification in the LUBM ontology.

3.3 Ground Truthing the Scientist Dataset from DBpedia

From DBpedia [2], a dataset of scientists' descriptions was built; 25,760 URIs for scientists' descriptions were retrieved. In order to diversify the types of classes in the scientists dataset, a few other classes that are related to the Scientist concept in DBpedia were also collected, namely: *EducationalInstitution*, *Place* and *Award*. The total number of triples obtained in the scientists dataset is $\simeq 5.5$ million. No artificial noise was induced in this dataset as it already has pre-existing noise. An example of noisy type assertion is the resource *dbr:United_States* being of type *dbo:Person*. There are 1,761 resources in DBpedia that are of types *dbo:Person* and *dbo:Place* simultaneously, which obviously indicates that one of them is a noisy triple.

4 Layered Graph Model for RDF

Even though the RDF conceptual model is designed as a graph, it differs from the graph theory definition of graphs in a number of ways. RDF graphs are heterogeneous multigraphs. Moreover, an edge in the ABox can be a node in the TBox (describing the properties hierarchy for example). Current research efforts to represent the RDF model as graphs— based on a: bipartite graph model [7], hypergraph model [9,8,10] or metagraph models [5]— target different goals ranging from storing and querying RDF graphs to reducing space and time complexity to solving the reification and provenance problem. Unfortunately, these goals do not coincide with RDFS reasoning. Moreover they use complex graph models which are not suitable for neural network input.

The proposed layered graph model for RDF consists of an ordered sequence of directed graphs, where each directed graph represents the relations between the resources of the RDF graph according to one property (from the set of properties in the ontology plus *rdf:Type*). It is important to note that the transformation of an RDF graph into its layered directed graph representation is not bijective as two non-isomorphic RDF graphs can have the same layered directed graph representation. However this transformation guarantees that if two RDF graphs have the same layered directed graph representation then their RDFS inference graphs according to the ontology O are isomorphic.

5 Approach overview

The overview of our approach is depicted in Fig. 2. It can be summarized in the following steps:

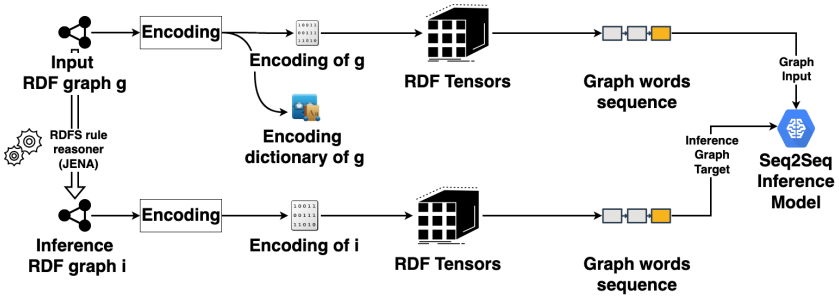

Fig. 2: Approach overview

1. **Layering the RDF graphs:** Using the layered graph model for RDF, we generate the layered graph version for each input and inference graph in the training set. The order of the properties corresponding to each layer can be chosen arbitrarily but it is crucial to maintain the same order across the dataset. In the simple version, each property in the ontology plus *rdf:Type* has its corresponding layer. While in the more efficient version, only the subset of "active" properties is considered—where an active property is a property from the ontology that is used in the dataset. This reduces the number of layers dramatically in the case of the scientists dataset where only a small subset of the DBpedia properties is used.
2. **RDF tensors creation** Using the layered version of the RDF graphs, an ID must be assigned to each resource in the RDF graph to allow it to be represented as a 3D adjacency matrix. In the simplified version, two dictionaries were created: one for the subject and object IDs— which is split into a global and a local dictionary— and one for the property IDs. The global resources dictionary contains the subject and object resources that are used throughout the G set (which are basically the RDFS classes in the ontology). While the local resources' dictionaries contain the IDs for the resources specific to each graph. In the more advanced version—required for complex ontologies— instead of using the same local dictionary for all the layers, each group of properties share their own dictionaries of local resources.
3. **Graph words generation** At this stage, every RDF graph is represented as a 3D adjacency matrix of size:
(active_properties_size, max_number_of_resources, max_number_of_resources).
In theory the maximum number of possible layer layouts in a dataset of size dataset_size is:

$$min(2^{max\_number\_of\_resources^2}, dataset\_size * active\_properties\_size)$$

However, in practice, the number of layouts is much smaller than this theoretical bound. For instance, in the LUBM1 dataset, we obtained 131 layouts versus 309,402 possible layouts. This observation is a good indication that the encoding algorithm has achieved one of its major goals of having similar encodings for "similar" graphs.

By assigning an ID for each layer's layout, the 3D adjacency matrix can be represented as a sequence of layouts' IDs as shown in Fig. 3. The layouts are termed *"graph words"*, as the sequence (or phrase) of graph words represents a 3D adjacency matrix and thus an RDF graph. Representing an RDF graph as a sequence of graph words has two main advantages:

(a) Reducing the size of the encoded dataset: only the ID of the layer's layout along with a catalog of layouts is saved.
(b) Exploitation of the research results in neural machine translation.

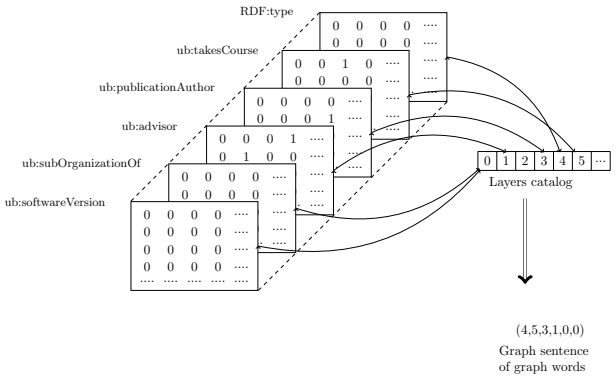

Fig. 3: From a 3D adjacency matrix to a sentence of graph words

The overall architecture of the model as well as its hyperparameters are detailed in the journal paper.

6 Evaluation

Fig. 4 shows the training process on the LUBM1 dataset. After approximately 12 minutes of training, 98.8% training accuracy was achieved. When testing the trained model on the intact LUBM1 test set, an overall per-graph accuracy of 97.7% was obtained.

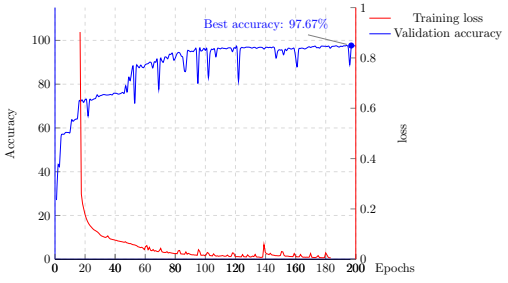

Fig. 4: Training results on intact LUBM1 data

The trained model was then tested on the noisy datasets created as described in Section 3. Two metrics were designed:

– **Macroscopic metric: Per-graph accuracy** : Inferences in this metric are scored correct when dr and i are isomorphic— in other words, when the deep reasoner inference from the corrupted graph is isomorphic to the Jena inference from the intact graph.
– **Microscopic metric: Per-triple precision/recall.** The previous metric overlooks the fact that some triples, generated by the deep reasoner and not by Jena, were in fact valid.

The macro and micro evaluation on the 5 noisy datasets (Fig. 5) shows exceptional noise-tolerance compared to rule-based reasoners: 99% for the deep reasoner *vs* 0% (by design) for Jena in the UGS_{100} dataset.

The model used for the scientists dataset is like the LUBM1 model, except for the hyper-parameters. Training to a validation accuracy of 87.76% takes over 16 hours. The 'scientists' dataset contains 94 graphs with the person-place noise out of which 38 inference graphs generated by the deep reasoner were *perfect*, containing exactly the inference from Jena minus the noisy triple. For the remaining person-place inferences, a few contain "false positive" triples not generated by Jena. For example, the deep reasoner inferred that *dbr:Big_Ben* is of type *dbo:HistoricPlace* even though this information is not explicitly (i.e. embedded in the DBpedia graph of the the resource *dbr:Big_Ben*) nor implicitly (i.e. can be inferred). The deep reasoner inferred this information by capturing the generalization that resources with similar links to *dbr:Big_Ben* are usually of type *dbo:HistoricPlace*.

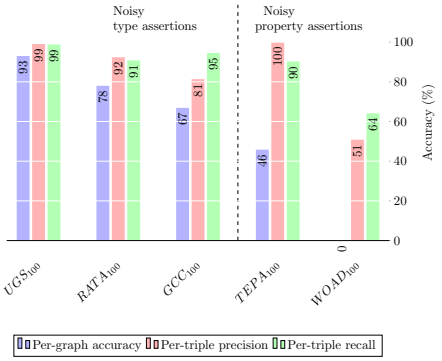

Fig. 5: Macro and micro evaluation on noisy LUBM1 datasets

7 Conclusions, Discussions and Future Work

The main contribution of this paper is the empirical evidence that deep learning (neural networks translation in particular) can in fact be used to learn semantic reasoning— RDFS rules specifically. The goal was not to reinvent the wheel and design a Yet another Semantic Reasoner (YaSR) using a new technology; it was rather to fill a gap that existing rule-based semantic reasoners could not satisfy, which is noise-tolerance. This research can be extended in a few directions. Currently the trained model on a specific domain with a given ontology cannot be used to generate inferences from a different domain. One promising research direction is to investigate transfer learning to bootstrap the adaption for new domains.

References

1. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: Dbpedia: A nucleus for a Web of open data. In: Aberer, K., Choi, K.S., Noy, N., Allemang, D., Lee, K.I., Nixon, L., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) Semantic Web. pp. 722–735. Springer Berlin Heidelberg, Berlin, Germany (2007)
2. Bizer, C., Heath, T., Berners-Lee, T.: Linked data - the story so far. Int. J. Semantic Web Inform. Syst. **5**(3), 1–22 (2009). https://doi.org/10.4018/jswis.2009081901, https://doi.org/10.4018/jswis.2009081901
3. Cai, H., Zheng, V.W., Chang, K.C.: A comprehensive survey of graph embedding: Problems, techniques and applications (2017), http://arxiv.org/abs/1709.07604
4. Carroll, J.J., Dickinson, I., Dollin, C., Reynolds, D., Seaborne, A., Wilkinson, K.: Jena: Implementing the Semantic Web recommendations. In: Proc. 13th Int. World Wide Web Conf. Alternate Track Papers Posters. pp. 74–83. ACM, New York, NY, USA (2004). https://doi.org/10.1145/1013367.1013381, http://doi.acm.org/10.1145/1013367.1013381
5. Chernenkiy, V., Gapanyuk, Y., Nardid, A., Skvortsova, M., Gushcha, A., Fedorenko, Y., Picking, R.: Using the metagraph approach for addressing RDF knowledge representation limitations. In: 2017 Internet Technologies Appl. (ITA). pp. 47–52 (9 2017). https://doi.org/10.1109/ITECHA.2017.8101909
6. Guo, Y., Pan, Z., Heflin, J.: LUBM: A benchmark for OWL knowledge base systems. Web Semantics: Science, Services Agents World Wide Web **3**(2-3), 158–182 (2005). https://doi.org/10.1016/j.websem.2005.06.005, https://doi.org/10.1016/j.websem.2005.06.005
7. Hayes, J., Gutiérrez, C.: Bipartite graphs as intermediate model for RDF. In: McIlraith, S.A., Plexousakis, D., van Harmelen, F. (eds.) Semantic Web - ISWC 2004: Third Int. Semantic Web Conference. vol. 3298, pp. 47–61. Springer, Berlin, Germany (2004). https://doi.org/10.1007/978-3-540-30475-3_5, https://doi.org/10.1007/978-3-540-30475-3_5
8. Liu, H., Dou, D., Jin, R., LePendu, P., Shah, N.: Mining biomedical ontologies and data using RDF hypergraphs. In: 12th Int. Conf. Mach. Learning Applications, ICMLA 2013. pp. 141–146. IEEE, Miami, FL, USA (12 2013). https://doi.org/10.1109/ICMLA.2013.31, https://doi.org/10.1109/ICMLA.2013.31
9. Morales, A.A.M.: A directed hypergraph model for RDF. In: Simperl, E.P.B., Diederich, J., Schreiber, G. (eds.) Proc. KWEPSY 2007 Knowledge Web PhD Symp. 2007. vol. 275, pp. 1–2. CEUR-WS.org, Innsbruck, Austria (6 2007), http://ceur-ws.org/Vol-275/paper24.pdf
10. Wu, G., Li, J., Hu, J., Wang, K.: System π: A native RDF repository based on the hypergraph representation for RDF data model. J. Computer Sci. Technology **24**(4), 652–664 (2009). https://doi.org/10.1007/s11390-009-9265-9, https://doi.org/10.1007/s11390-009-9265-9
11. Zhu, M., Gao, Z., Quan, Z.: Noisy type assertion detection in semantic datasets. In: Mika, P., Tudorache, T., Bernstein, A., Welty, C., Knoblock, C.A., Vrandecic, D., Groth, P.T., Noy, N.F., Janowicz, K., Goble, C.A. (eds.) The Semantic Web - ISWC 2014 - 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part I. Lecture Notes in Computer Science, vol. 8796, pp. 373–388. Springer (2014). https://doi.org/10.1007/978-3-319-11964-9_24, https://doi.org/10.1007/978-3-319-11964-9_24