Explainable Deep RDFS Reasoner

Anonymous Author(s)
Anonymous Institution(s)

Abstract

Recent research efforts aiming to bridge the Neural-Symbolic gap for RDFS reasoning proved empirically that deep learning techniques can be used to learn RDFS inference rules. However, one of their main deficiencies compared to rule-based reasoners is the lack of derivations for the inferred triples (i.e. explainability in AI terms). In this paper, we build on these approaches to provide not only the inferred graph but also explain how these triples were inferred. We evaluated our justification model on two datasets: a synthetic dataset– LUBM1 benchmark– and a real-world dataset – ScholarlyData about conferences – where the lowest validation accuracy approached 96%.

Introduction

The famous saying in the Semantic Web (SW) community "A Little Semantics Goes a Long Way" (Hendler 1997) advocates for using light semantics such as RDFS (Hayes and Patel-Schneider 2014) in order to achieve the SW vision – where humans and machines interact to process, produce and reason about the Web of data. RDFS provides a way to describe the classes of RDF resources, their hierarchy, the relations among the different resources and the hierarchy of these relations. Given an RDF graph describing a set of resources (A-Box) and an RDFS ontology describing the classes and relations (T-Box), RDFS entailment rules generate new facts that extend our knowledge about the A-Box.

In many applications such as Question Answering (QA) (Unger et al. 2012; Zou et al. 2014; Unger et al. 2014), not only the inferred fact is required but also the derivation[1] of the triple. The derivation can be useful to justify the answer provided by the QA system which allows the Human-in-the-loop (HITL) to validate or reject the answer. An example of an RDFS derivation in LUBM (Guo, Pan, and Heflin 2005) is shown in listing 1.

Listing 1: RDFS derivation example in LUBM

```
_:UndergraduateStudent46 rdf:type
    LUBM:Student .
  ⟸
_:UndergraduateStudent46  rdf:type
    LUBM:UndergraduateStudent .
LUBM:UndergraduateStudent rdfs:subClassOf
    LUBM:Student .
```

Rule-based reasoners suffer from two main limitations:

1. Dealing with noisy data
2. Lack of support for approximate reasoning

These two limitations motivated the exploration of deep learning techniques for SW reasoning. In (Hohenecker and Lukasiewicz 2017), the authors propose Relational Tensor Networks (RTN) as an adaptation for Recursive Neural Tensor Networks (RNTN) (Socher et al. 2013). In (Ebrahimi et al. 2018), the authors extend the End-to-end memory networks (MemN2N) (Sukhbaatar et al. 2015) to store RDF triples and train the model to embed the knowledge graph and predict if a triple can be inferred from the input graph. (Makni and Hendler 2019) focus on the noise-tolerance aspect: by transforming the RDF graphs into a sequence of "graph words" they model the RDFS inference task as a machine translation task of graph words.

Unfortunately, deep learning based reasoners suffer from a problem that is common for deep learning approaches in general: the lack of explainability which translates into the inability to provide the derivation of the inferred triples. Towards combining the best of both worlds, this paper proposes a graph2seq model that learns the generation of justifications for the inferred triples.

Approach Overview

In order to input the RDF graph into the neural network– that learns the generation of the inference graph and/or the justification of the inferred triples– we first need to embed the RDF graph into a format that can be fed to neural networks. The literature contains many approaches for knowledge graph embedding such as: (Bordes et al. 2013; Wang et al. 2014; Nickel, Rosasco, and Poggio 2016; Ristoski and Paulheim 2016; Ji et al. 2015; Trouillon et al. 2017; Wang et al. 2017). However only a few are tailored for

[1]Derivation and justification are used interchangeably in this paper.

RDFS reasoning. One approach in particular is the "graph words" (Makni and Hendler 2019) technique that converts an RDF graph into a sequence of graph word IDs. Besides being tailored for RDFS reasoning, the choice of this embedding technique is motivated by the availability of the code and the expendability of the model as detailed further along.

The augmentation of the graph words model to support the generation of the justifications for the inferred triples is illustrated in Fig. 1. The upper segment depicts the graph words translation approach (Inference Generation) where each input graph is encoded into a 3D adjacency matrix (or Tensor). Each layer in the 3D adjacency matrix corresponds to the adjacency matrix between the RDF resources which are linked according to a single predicate. These layers are then assigned an ID that represents their layout. The sequence of these IDs represents the RDF graph hence the naming "graph words". The training set becomes a parallel corpus between the sequence of graph words of the input graph and the sequence of graph words of the inference. The authors then trained a seq2seq (Sutskever, Vinyals, and Le 2014) model that learns the inference generation.

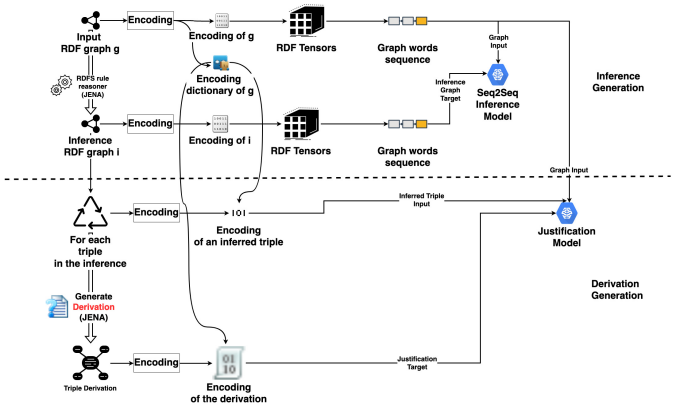

Figure 1: Explainable Deep RDFS reasoner (Approach Overview)

The lower segment of Fig. 1 (derivation generation) depicts the core contribution of this paper. Among the different explainability research efforts for deep learning networks, the closest to our approach is rationalization (Harrison, Ehsan, and Riedl 2018). Rationalization of neural networks consists of training the neural network not only on the input and target data, but also on the rationale behind the target data. In our case, the justification is provided by a rule-based reasoner (JENA (Carroll et al. 2004)). The training input consists of the input RDF graph represented as a sequence of graph words as well as the encoded inferred triple and the target is the encoded derivation. The justification model is a modified seq2seq model with two sequences as the input and one sequence as the output.

RDF Data Representation

In this section, we provide an overview of our data preparation process including ground-truthing and RDF tensors creation.

Ground-Truthing

This process is applied on the following datasets: LUBM (Guo, Pan, and Heflin 2005) and ScholarlyData (Gentile and Nuzzolese 2015). In order to generate the ground truth, we split each dataset into sub-graphs where each graph represents the description (using SPARQL DESCRIBE) of a subject resource. The ground-truthing procedure consists of the following steps:

1. Let L be the set of these sub-graphs.
2. For each graph l in L, we generate the inference i of l according to the corresponding ontology using JENA.
3. For each inferred triple t in the inference graph i, we generate the justification graph j using JENA.
4. The training input is (l, i) and the target is j

This results into pairs of input graphs and inferred triples (l, i).

In the case of LUBM1, the summarized derivations generated by JENA contain either one or at most two triples, which will simplify the design of the justification neural network model. On the contrary, the justifications generated from the real-world datasets can be longer as detailed later.

RDF Tensors Creation

The encoding and embedding stages aim to transform RDF graphs into a format suitable for neural network input. Given that the training data has three components to be encoded: the graph input, the inferred triple and the justification, we extended the encoding technique proposed in (Makni and Hendler 2019) (depicted in the upper segment of Fig. 1) to support the encoding of the derivations. The encoding technique described in (Makni and Hendler 2019) can be briefly described in the following steps:

1. Let P be the set of "active properties" which consists of the intersection between the properties in the ontology and the properties used in the A-Box. By using only the intersection and not the full set of properties in the ontology, the size of the embedding can be reduced dramatically without affecting the inference.
2. Let P^+ be the union of P and *rdf:type*.
3. A fixed arbitrary order is imposed on the set P^+ throughout the training.
4. Let OP be the ordered list resulting from this sorting. Each property in OP is assigned a integer ID p_i according to its index.
5. Let GR be the set of global resources (i.e. classes in the ontology). An arbitrary order is also imposed on GR and maintained throughout the training. Each class in GR is assigned an integer ID c_i according to its index.
6. Let LR be the set of local resources (i.e. the RDF resources that appear in the A-Box but not in the T-Box). Each resource in LR is assigned an ID l_i with an offset the size of GR. These IDs are reset when encoding the next RDF graph.

In this simplified encoding technique, the global GR and local LR resource sets are shared between the predicates. The authors also propose a more complex encoding technique where each group of predicates p has its own GR_p and LR_p resource dictionaries.

By design, this encoding technique ensures that every resource that can appear in the inference has an ID assigned to it when encoding the input graph. This allowed us to use the same encoding technique for the input graph and the inferred triple. In contrast, the properties that can appear in the justification such as *rdfs:subClassOf* and *rdfs:subPropertyOf* are not present in P^+ as they are not used in the A-Box. As a remedy, we had to add the properties that are used in the RDFS entailment rules (Hayes and Patel-Schneider 2014) to P^{++} and similarly impose a fixed arbitrary order to assign an ID to each of these properties. Knowing that the derivation can contain two triples, an order is also imposed to sort these two triples according to the IDs of their predicates.

Imposing these lists of orders is mandatory for the training, otherwise the same inferred triple can have two targets as its justification depending on the order of the derivation triples. Keeping in mind that neural networks are functions' approximators (Csáji 2001) but having the same input with two possible targets makes the problem outside function approximation realm.

At this stage, the input consists of two lists:

Graph words sequence A sequence of layers' IDs of size $P+$ where each layer ID represents the layout of the adjacency matrix according to the respective property.

Inferred triple encoding A sequence of 3IDs for the subject, property and object of the inferred triple. These IDs are assigned according to the same encoding technique for the input graph.

The target consists of:

Derivation encoding A padded sequence of size 6 corresponding to the IDs for each resource in the derivation.

Finally, the embedding of each layer in the graph words sequence is computed using High-Order Proximity preserved Embedding (HOPE) (Ou et al. 2016). For the inference and derivation sequences, we use one-hot vector encoding.

Model Architecture

The architecture of the justification neural network is provided in the supplementary material. It is a modified seq2seq model that takes two sequences as input. The graph input part is a tensor of size $(17, 280)$ where 17 represents the length of the sequence (i.e. the size of $P+$ in the case of LUBM). Each adjacency matrix in LUBM1 encoding is of size 70×70. When using the HOPE embedding with a dimension of 4, we reduce the size of the binary matrix to a real matrix of size 70×4 hence the 280 parameter.

The input triple tensor is of size $(3, 66)$ where 66 is the size of the one-hot vector. Using the local resources set LR played a major role in reducing the size of this one-hot vector making the training faster and even possible. Each sequence (the graph sequence and the triple sequence) go then through similar seq2seq layers with a Bidirectional Recurrent Neural Network (BRNN) (Schuster and Paliwal 1997) encoder which can be trained simultaneously in positive and negative time directions. The hidden representations are then merged in the *concatenate* layer before being decoded into a sequence of 6 items which should represent the derivation. Few *dropout* layers are introduced in the architecture with a dropout factor of 0.2 to prevent over-fitting and improve the generalization on the validation set.

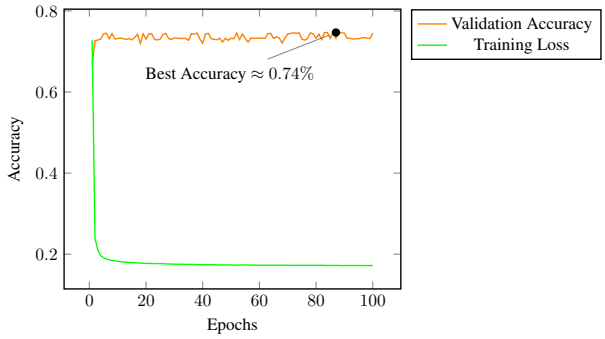

Figure 2: Baseline Model Performance on LUBM1 dataset

Evaluation

This section shows the evaluation results for the task of providing justifications for the inferred triples in LUBM1 both using a baseline model and a full model architecture. Both models were trained and evaluated using two NVIDIA Tesla P100 GPUs used in parallel. LUBM1 dataset has around one hundred thousand triples containing $17,189$ subject-resources within 15 classes. The total number of pairs of graphs and inferred triples is $78,766$. It is worth noting that the same triple can be inferred by multiple graphs and the total number of inferred triples in LUBM1 is smaller.

Baseline model

Our first experiment shows the performance of a baseline model which gets as input only the inferred triple and outputs the justification. Fig. 2 shows training and validation accuracy of the baseline model. In the first few iterations, the training loss dropped significantly but stagnated afterwards. Similarly, the validation accuracy increased in the first few iterations 66% to 73% but improved very little later on.

It might be surprising that such a naive model can even reach this accuracy especially that the justification of an inferred triple depends on the input graph from which the inferred triple was derived. For example, if the question is why an entity is of type *Person*, the answer can be:

1. because it is of type *Student*
2. or because it is of type *Professor*.

Given that the majority of entities of type persons in LUBM1 are of type student and not professor, a naive model that always outputs justification 1 will be correct on the majority of the cases–thus the surprising relatively high accuracy and the stagnation when the model fails to generalize for smaller classes.

Justification model

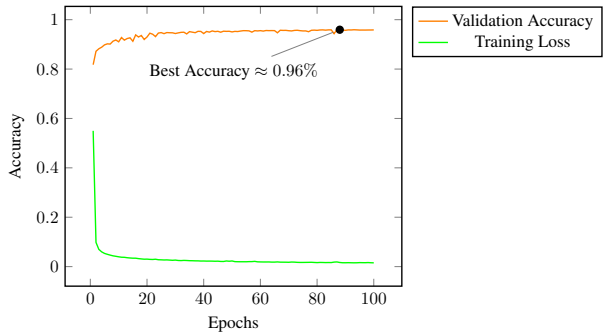

Figure 3: Performance of our model on LUBM1 dataset

Our final model has a much better performance as shown in Figures 3 and 4 showing a validation accuracy of 96% and 97.9% for LUBM1 and Scholarly datasets respectively.

Inference generation examples

Listing 2 shows an example of ground truth derivation and listing 3 shows an example where the justification model generates a wrong derivation by outputting the predicate *undergraduateDegreeFrom* rather than *doctoralDegreeFrom*.

Listing 2: Ground truth derivation

```
_:FullProfessor3 rdf:type  LUBM:Person .

  ⟸

_:FullProfessor3 lubm:doctoralDegreeFrom
    <http://www.University879.edu> .
```

Listing 3: Predicted wrong derivation

```
_:FullProfessor3 rdf:type  LUBM:Person .

  ⟸

_:FullProfessor3
    lubm:undergraduateDegreeFrom
    <http://www.University879.edu> .
```

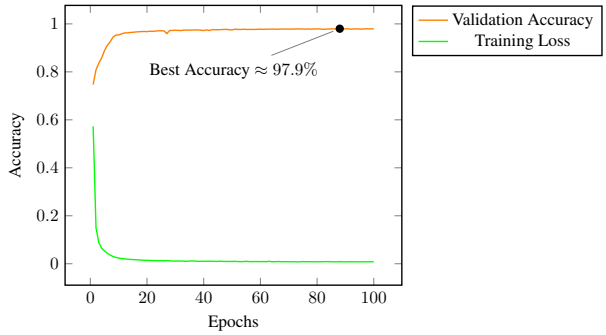

Figure 4: Performance of our model on ScholarlyData

Related Work

Opening the black box of deep neural networks Deep learning models are often described as black box models as they are not built in a way that allows for the backtracking of their decision-making process. However, recent research efforts are attempting to demystify the hidden processing of deep learning networks and opening its black box such as "Opening the Black Box of Deep Neural Networks via Information" (Shwartz-Ziv and Tishby 2017) and "Rationalization: A Neural Machine Translation Approach to Generating Natural Language Explanations" (Harrison, Ehsan, and Riedl 2018). In (Harrison, Ehsan, and Riedl 2018), the authors train a deep model to play the video game Frogger. They also feed the justifications generated by human players for each action. The model was able to not only make decisions that ultimately led to it winning the game, but it was also able to justify the sequence of actions it took.

Deep Learning for Ontological Reasoning (Makni and Hendler 2019) showed that neural network translation can be utilized effectively in symbolic reasoning in the presence of noise. The authors proposed an embedding technique tailored for RDFS reasoning where graphs are represented as layers where each layout is encoded as a 3D adjacency matrix and forms a graph word. The input graph and its entailment are then represented as a sequence of graph words and the task of RDFS inference is then formulated as as translation task achieved through neural machine translation. In this paper, we build on this approach (Makni and Hendler 2019) to provide the explainability advantage for RDFS reasoning. In (Ebrahimi et al. 2018), the authors extend the End-to-end memory networks (MemN2N) (Sukhbaatar et al. 2015) to store RDF triples and train the model to embed the knowledge graph and predict if a triple can be inferred from the input graph.

Conclusion

In this paper, we aimed at combining the benefits of both deep learning reasoners as well as the justification feature in rule-based reasoners. We proposed a novel approach for achieving explanability in RDFS reasoning which is inspired by the rationalization technique for explainable AI. The proposed grah2seq model takes as input an RDF graph as well as a sequence representing an inferred fact and generates a sequence representing the justification for the inferred triple. Our results show that this approach can provide a very high accuracy–up to 95.93% on LUBM1. Possible extension of this work is to test is on larger RDF datasets and compare the derivation generation timing. As noise-tolerance was one of the motivations for proposing deep learning based reasoners, it will also be worth-while to test the justification generator model in the presence of noisy input data.

References

Bordes, A.; Usunier, N.; Garcia-Duran, A.; Weston, J.; and Yakhnenko, O. 2013. Translating embeddings for modeling multi-relational data. In Burges, C. J. C.; Bottou, L.; Welling, M.; Ghahramani, Z.; and Weinberger, K. Q.,

eds., *Advances in Neural Information Processing Systems 26*. Curran Associates, Inc. 2787–2795.

Carroll, J. J.; Dickinson, I.; Dollin, C.; Reynolds, D.; Seaborne, A.; and Wilkinson, K. 2004. Jena: Implementing the Semantic Web recommendations. In *Proc. 13th Int. World Wide Web Conf. Alternate Track Papers Posters*, 74–83. New York, NY, USA: ACM.

Csáji, B. C. 2001. Approximation with artificial neural networks. *Faculty of Sciences, Etvs Lornd University, Hungary* 24:48.

Ebrahimi, M.; Sarker, M. K.; Bianchi, F.; Xie, N.; Doran, D.; and Hitzler, P. 2018. Reasoning over RDF knowledge bases using deep learning. *CoRR* abs/1811.04132.

Gentile, A. L., and Nuzzolese, A. G. 2015. clodg - conference linked open data generator. In Villata, S.; Pan, J. Z.; and Dragoni, M., eds., *Proceedings of the ISWC 2015 Posters & Demonstrations Track co-located with the 14th International Semantic Web Conference (ISWC-2015), Bethlehem, PA, USA, October 11, 2015.*, volume 1486 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Guo, Y.; Pan, Z.; and Heflin, J. 2005. LUBM: A benchmark for OWL knowledge base systems. *Web Semantics: Science, Services Agents World Wide Web* 3(2-3):158–182.

Harrison, B.; Ehsan, U.; and Riedl, M. O. 2018. Rationalization: A neural machine translation approach to generating natural language explanations. In *AI, Ethics, and Society, Papers from the 2018 AAAI WorkshopNew Orleans, Louisiana, USA, February 1, 2018.*, AAAI Workshops. AAAI Press.

Hayes, P. J., and Patel-Schneider, P. F. 2014. RDF 1.1 semantics. W3C recommendation 25 february 2014, W3C.

Hendler, J. 1997. A little semantics goes a long way.

Hohenecker, P., and Lukasiewicz, T. 2017. Deep learning for ontology reasoning.

Ji, G.; He, S.; Xu, L.; Liu, K.; and Zhao, J. 2015. Knowledge graph embedding via dynamic mapping matrix. In *Proc. 53rd Annu. Meeting Assoc. Computational Linguistics 7th Int. Joint Conf. Natural Language Process. (Volume 1: Long Papers)*, 687–696. Beijing, China: The Association for Computer Linguistics.

Makni, B., and Hendler, J. 2019. Deep learning for noise-tolerant RDFS reasoning. *Semantic Web* 10(5):823–862.

Nickel, M.; Rosasco, L.; and Poggio, T. 2016. Holographic Embeddings of Knowledge Graphs. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, 1955–1961. AAAI Press.

Ou, M.; Cui, P.; Pei, J.; Zhang, Z.; and Zhu, W. 2016. Asymmetric transitivity preserving graph embedding. In *Proc. 22Nd ACM SIGKDD Int. Conf. Knowledge Discovery Data Mining*, 1105–1114. New York, NY, USA: ACM.

Ristoski, P., and Paulheim, H. 2016. Rdf2vec: RDF graph embeddings for data mining. In Groth, P. T.; Simperl, E.; Gray, A. J. G.; Sabou, M.; Krötzsch, M.; Lécué, F.; Flöck, F.; and Gil, Y., eds., *Semantic Web - ISWC 2016 - 15th Int. Semantic Web Conference, Kobe, Japan, October 17-21, 2016, Proceedings, Part I*, volume 9981, 498–514.

Schuster, M., and Paliwal, K. K. 1997. Bidirectional recurrent neural networks. *IEEE Trans. Signal Process* 45(11):2673–2681.

Shwartz-Ziv, R., and Tishby, N. 2017. Opening the black box of deep neural networks via information.

Socher, R.; Chen, D.; Manning, C. D.; and Ng, A. Y. 2013. Reasoning with neural tensor networks for knowledge base completion. In Burges, C. J. C.; Bottou, L.; Ghahramani, Z.; and Weinberger, K. Q., 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.*, 926–934.

Sukhbaatar, S.; Szlam, A.; Weston, J.; and Fergus, R. 2015. End-to-end memory networks. In Cortes, C.; Lawrence, N. D.; Lee, D. D.; Sugiyama, M.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, 2440–2448.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to Sequence Learning with Neural Networks. In *Proc. 27th Int. Conf. Neural Inform. Process. Syst. - Volume 2*, 3104–3112. Cambridge, MA, USA: MIT Press.

Trouillon, T.; Dance, C. R.; Gaussier, É.; Welbl, J.; Riedel, S.; and Bouchard, G. 2017. Knowledge graph completion via complex tensor factorization. *Journal of Machine Learning Research (JMLR)* 18(130):1–38.

Unger, C.; Bühmann, L.; Lehmann, J.; Ngomo, A. N.; Gerber, D.; and Cimiano, P. 2012. Template-based question answering over RDF data. In *WWW*, 639–648. ACM.

Unger, C.; Forascu, C.; López, V.; Ngomo, A. N.; Cabrio, E.; Cimiano, P.; and Walter, S. 2014. Question answering over linked data (QALD-4). In *CLEF (Working Notes)*, volume 1180 of *CEUR Workshop Proceedings*, 1172–1180. CEUR-WS.org.

Wang, Z.; Zhang, J.; Feng, J.; and Chen, Z. 2014. Knowledge Graph Embedding by Translating on Hyperplanes. In Brodley, C. E., and Stone, P., eds., *Proc. Twenty-Eighth AAAI Conf. Artificial Intelligence*, 1112–1119. Cambridge, MA, USA: AAAI Press.

Wang, Q.; Mao, Z.; Wang, B.; and Guo, L. 2017. Knowledge graph embedding: A survey of approaches and applications. *IEEE Trans. Knowl. Data Eng* 29(12):2724–2743.

Zou, L.; Huang, R.; Wang, H.; Yu, J. X.; He, W.; and Zhao, D. 2014. Natural language question answering over RDF: a graph data driven approach. In *SIGMOD Conference*, 313–324. ACM.

Explainable Deep RDFS Reasoner (Supplementary material)

Anonymous Author(s)
Anonymous Institution(s)

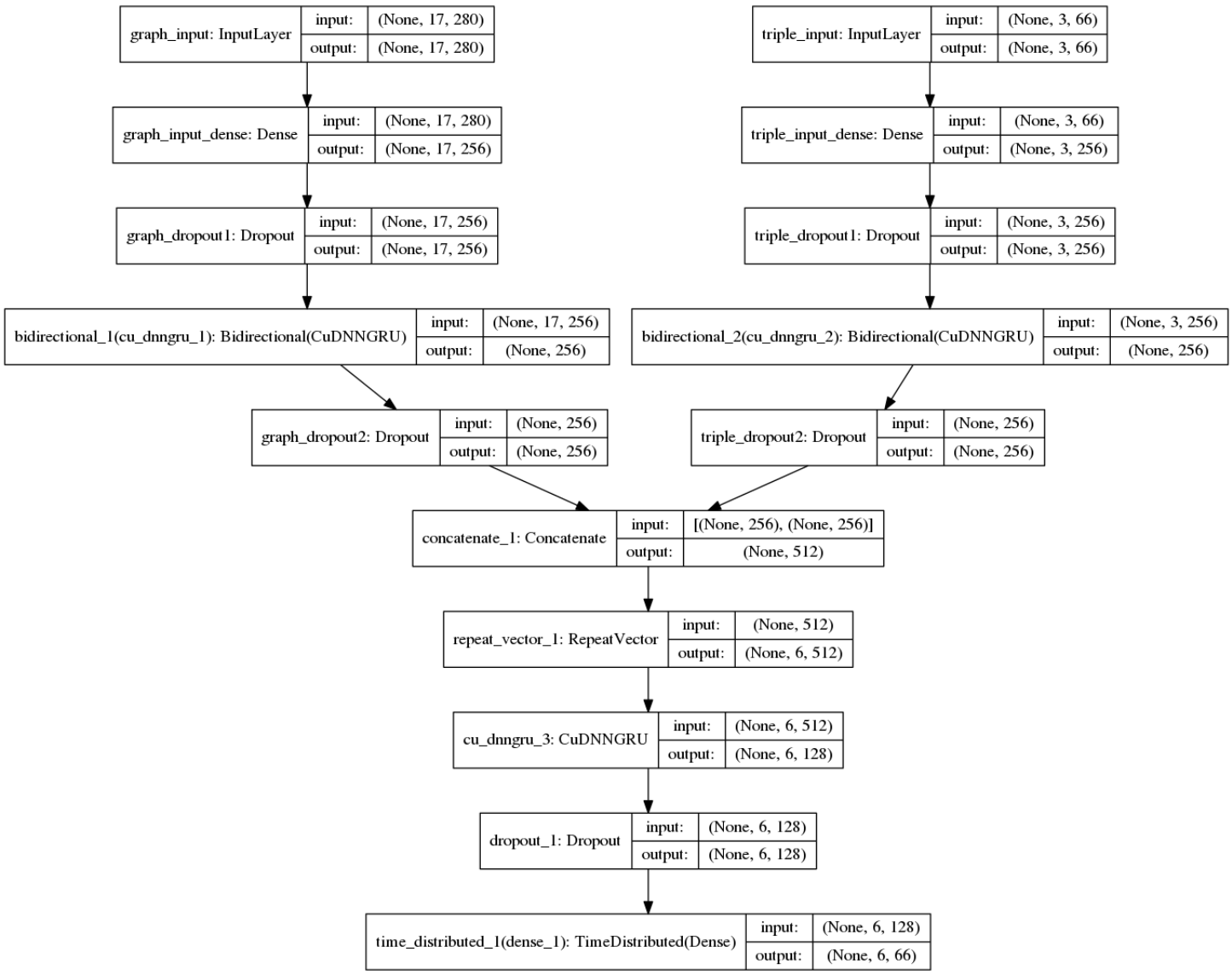

Figure 1: Model Architecture