Modular Neuro-Symbolic Knowledge Graph Completion

Abelardo Carlos Martinez Lorenzo*
TU Berlin
Germany

Alexander Perfilyev
University of Copenhagen
Denmark

Volker Markl
TU Berlin, BIFOLD, DFKI
Germany

Martha Clokie
University of Leicester
United Kingdom

Thomas Sicheritz-Ponten
University of Copenhagen
Denmark

Zoi Kaoudi
IT University of Copenhagen
Denmark

ABSTRACT

Knowledge graph completion (a.k.a. link prediction), i.e., the task of inferring missing edges in knowledge graphs, is a widely used task in many applications, such as product recommendation and question answering. State-of-the-art approaches include knowledge graph embeddings and rule mining which are data-driven and, thus, solely based on the information contained in the input knowledge graph. This leads to unsatisfactory prediction results and ignores domain expertise making such solutions inefficient for domains such as healthcare and bioinformatics. To enhance the accuracy of knowledge graph completion we propose Poderoso, a modular neuro-symbolic framework that loosely integrates the data-driven power of knowledge graph embeddings with rule-based reasoning. Poderoso not only enhances the prediction accuracy with domain knowledge via rules stemming from experts but also allows users to plug their own knowledge graph embedding models and reasoning engines. In our preliminary results we show that Poderoso enhances the MRR accuracy of vanilla knowledge graph embeddings and outperforms hybrid solutions that combine knowledge graph embeddings with rule mining. We also discuss how Poderoso can be used in bionformatics, in particular how it can advance research in bacteriophage therapy.

VLDB Workshop Reference Format:
Abelardo Carlos Martinez Lorenzo*, Alexander Perfilyev, Volker Markl, Martha Clokie, Thomas Sicheritz-Ponten, and Zoi Kaoudi. Modular Neuro-Symbolic Knowledge Graph Completion. VLDB 2025 Workshop: New Ideas for Large-Scale Neurosymbolic Learning Systems.

1 INTRODUCTION

Knowledge graphs (KGs) are extensively used in many application domains, ranging from search engines to bioinformatics. Nodes in the graph denote entities and edges represent the relations between entities. KGs are typically stored in data systems that allow for efficient retrieval of parts of the graphs based on a declarative query [3, 11]. As KGs are usually created from incomplete data sources, it is often the case that there is missing information in the results of such queries. The task of *KG completion* (a.k.a. *link prediction*) aims to infer missing edges in a KG.

Two known methods used for the problem of KG completion is knowledge graph embeddings (KGE) and rule mining. KGEs are representations of entities and relations into a low-dimensional space which can be used to predict whether a missing link in the graph shall be true or not. Rule-mining methods mine logical rules from the KG and then apply them to infer new information. Both methods rely on the existing patterns present in the KG and are thus unable to generalize in cases where there is inadequate information. For instance, KGEs work well in particular for predicting links of popular entities and relations, i.e., dense parts of the graph [9]. Similarly, rule-mining methods work well when there is enough evidence supporting the patterns they mine. However, real-world KGs consist of both dense and sparse areas and thus, current approaches fail to provide satisfactory prediction results in the general case.

We argue that an effective direction to improve KG prediction performance is the inclusion of domain expertise encoded by ontologies and rules. For example, imagine Bob, a bioinformatician working on bacteriophage therapy who has been able to encode available datasets about phages into a KG. Although Bob can use KGEs to predict missing links and thus determine which lab experiments to conduct, there is information known in the field which is not found in the available datasets, e.g., that a certain type of phages can burst the bacterial cell. Such information cannot be extracted by KGEs because there is no much evidence in the available datasets. This hinders the effectiveness of current link prediction methods.

Related work. There have been few proposals on hybrid solutions, combining KGEs with rule mining and reasoning [2, 4, 13]. KALE [4] embeds first-order logic rules in the same mathematical framework of TransE, a specific KGE model, by devising a new loss function. The authors of [2] use mined rules to impose constraints into the KGE models. The drawback of these approaches is that the embedding and reasoning processes are tailored to a specific KGE model and specific rules which makes the system difficult to adapt to more efficient KGE models and incorporate domain knowledge. pLogicNet [13] is based on Markov Logic Networks (MLN) trained with the variational EM algorithm and leverages KGEs to infer missing triples during the inference step (E-step) which are then used in the learning step (M-step) until convergence. However, such approach inherits the inefficiency of MLNs to scale to large KGs. None of the aforementioned approaches can leverage domain specific knowledge in the form of ontologies and rules. In fact, [5]

Proceedings of the VLDB Endowment. ISSN 2150-8097.

*Work done while conducting master thesis at TU Berlin.

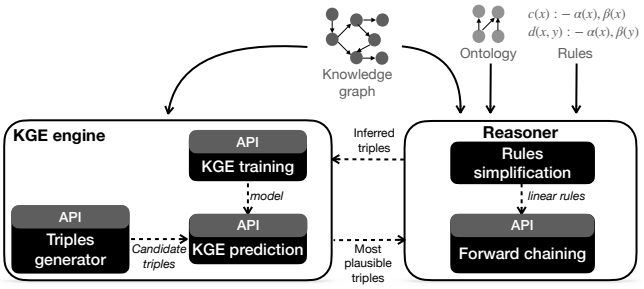


Figure 1: Overview of Poderoso.

states that most existing KGEs are not capable of encoding ontological information. In [6], the authors use ontological information but only for improving the negative samples that KGEs require.

Proposed solution. We are working towards building Poderoso, a modular neurosymbolic framework which allows for KGEs and rule-based reasoning to work in tandem. Poderoso consists of two main components: a KGE engine and a reasoner. The KGE engine produces new triples based on the embeddings and the reasoner produces new triples based on the KG, the ontologies and the rules. We aim for rules that are defined by domain experts but they can also be specified by different entailment regimes. The novelty of our framework over previous hybrid approaches is twofold. First, it leverages the predictive power of the data-driven KGE approaches (neural approach) while at the same time it is able to capture domain expert knowledge via deductive reasoning (symbolic approach). This allows to infer information even for sparse regions of a KG. Second, it combines KGEs with reasoning in a loosely-coupled and modular way and, thus, does not depend on any specific KGE model or reasoning algorithm: Users can easily plug their own KGE model or reasoning engine and the system will seamlessly work out of the box. Thanks to its loosely-coupled and modular integration of KGEs and reasoning, it can achieve better predictive performance.

2 PROPOSED FRAMEWORK

Our goal is to combine KGEs with rule-based reasoning in a loosely-coupled and modular way so that we can infer missing information, i.e., extract triples that do not exist in a KG. Our framework, Poderoso, consists of two main components: the *KGE engine* and the *Reasoner*. Both components are able to output inferred triples (missing links) using different ways. The KGE engine uses embeddings to output missing triples, while the reasoner uses a forward chaining reasoning algorithm to output inferred triples. Figure 1 shows the internals of the two components and how they interact with each other. Each component passes its output as input to the other component in an iterative manner (constant loop) until no new information can be inferred.

2.1 KGE engine

The KGE engine is responsible for the embedding-based learning. At each iteration, it receives as input the initial KG together with any triples inferred by the reasoner and trains a KGE model. The model is then able to predict with certain probability whether a given test triple, not contained in the training input set, is true or not. The challenge we had to overcome is how to generate the test triples. Ideally, we would like to pass through the KGE prediction the complement of the input KG, i.e., all triples that do not exist. Although theoretically it is possible to create the complement of the input KG by taking all pairs of nodes that are not directly connected and all possible edge labels, it is infeasible in practice as the number of test triples would be $O(N^2 \times R)$ for a KG with N entities and R relations. For this reason, the KGE engine includes a triples generator submodule which is responsible to output a subset of triples contained in the complement of the KG which have a good chance of being true. Once the triples generator module outputs candidates triples, it passes them to the KGE prediction module which outputs whether a triple is true with a certain probability. The KGE engine outputs the triples that are true with a probability above a certain threshold.

The triples generator is built in a generic manner to be able to support various methods for generating candidate triples. The simplest method is using uniform random sampling to draw from the set of entities and relations. This strategy, however, leads to triples that are with high probability not true as connecting randomly entities with random relations creates meaningless triples. An intuitive strategy is to explore sparse regions of the graph as entities that are densely connected are less probable to have missing true relations. For this we exploit the cluster coefficient of the entities, i.e., the fraction of triangles passing through a node w.r.t. its degree. We then use the cluster coefficient as a weight to sample entities for each type of relation in KG. In [1], we have evaluated different sampling methods for extracting plausible facts from a KG and concluded that more work is required in this direction.

The goal of the KGE engine is to be independent of any KGE model and triples generation strategy. In other words, users can plug their own implementations. To do so, Poderoso exposes the following primitives:

```
model = fit(X)
Y = generateTriples(X)
Y' = predict(Y, model)
```

Function `fit` receives a set `X` with the triples of the input KG and returns the fitted KGE model. Function `generateTriples` takes as input a set `X` of triples and outputs another set of candidate triples `Y` which are not contained in `X`. Function `predict` receives a set of triples `Y` and a model and outputs the input triples with an extra column that specifies the probability of the triple being true.

2.2 Reasoner

The reasoning engine is responsible to infer new triples given an initial set of triples (initial KG plus triples output by the KGE engine), a set of rules, and optionally an ontology. Rules can be user-defined or can be stemming from an entailment regime, such as RDFS or OWL2. Our framework can use any forward chaining reasoner that exhaustively applies the given rules to the input and inferred triples until a fixpoint is reached, i.e., no new triples can be inferred. Poderoso achieves this by providing the following simple primitive:

```
Y = infer(X, rules, [ontology])
```

The type of rules supported strongly depends on the underlying reasoner used by the framework. The only requirement is that they

Table 1: Datasets statistics.

Dataset	Entities	Relations	Triples (total)
DBPedia20k	20,143	12	120,000
LUBM-2	53,860	32	268,136

should be monotonic, i.e., adding new rules or triples never results in removing triples or contradicting already output results.

A problem that may arise with the input rules is that they are redundant, i.e., they produce redundant triples. Although this does not change the correctness of the results, it adds a significant overhead on the runtime of the reasoning process. Even the rules included in the RDFS entailment regime are redundant and can lead to a large number of redundant triples [7]. For this reason, Poderoso includes a subcomponent that simplifies the input rules.

3 VALIDATION AND USE CASE

We validate Poderoso with a preliminary set of experiments and describe a real use case we are currently working on.

3.1 Experiments

We use both synthetic and real world KGs. The questions we answer are: (i) how link prediction can be improved by a hybrid modular approach and (ii) what is the training time overhead that reasoning incurs to the KGE computation.

Datasets and rules. We use one real-world KG that is accompanied with an ontology, namely DBPedia, and the popular synthetic KG generator, LUBM. As the original DBPedia KG is too large for KGE models to handle, we use a subset of it: we extracted a well-connected subgraph that contains the 12 most popular relations and information around them. Table 1 summarizes the characteristics of our datasets. We utilize the simple RDFS entailment rules [10].

Baselines. Poderoso's current implementation supports TransE, DistMult, ComplEx, HolE, and ConvE KGE models and Pellet [14] and HermiT[1] OWL2 DL reasoners. We compare Poderoso against each KGE model in addition to KALE [4] and plogicNet [13] which use a different approach to incorporate reasoning in KGEs. We use the same hyperparameter settings for the vanilla KGEs and our approach. For KALE and plogicNet, we use the hyperparameters recommended in their repository.

Hardware and Software. We ran all our experiments in a machine with 32x2.3 GHz AMD Opteron(tm) processor 6376, 62GB RAM memory, a GPU NVIDIA-SMI 440.33.01, the driver version 440.33.01 and the CUDA version 10.0. We used Python 3.7 and Tensorflow 1.15. For the baselines plogicNet and KALE, we used Python 3.7 with PyTorch 1.5 and Java 1.8.0, respectively.

Metrics. We evaluate the accuracy by computing the filtered mean reciprocal rank (MRR), and the Hits@1, Hits@3, and Hits@10 for the link prediction task. In addition, we measure training time.

3.1.1 Model performance. We first compare the predictive performance of Poderoso compared to vanilla KGE models. Table 2 shows the results for both datasets. We observe that for both LUBM and DBPedia20k Poderoso almost always improves the quality of the model (shown by the underlined times). In particular, Poderoso improves the MRR, Hits@1, Hits@3, and Hits@10 of HolE [12] by

[1]http://www.hermit-reasoner.com/

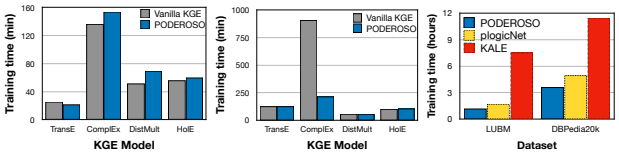
Table 2: Accuracy among different KGE models and approaches. Poderoso performs better than vanilla KGEs for both LUBM and DBPedia20k. It also significantly outperforms the hybrid approches of plogicNet [13] and KALE [4].

LUBM

Method	MRR	Hits@1	Hits@3	Hits@10
TransE	0.28	0.28	0.26	0.31
Poderoso (TransE)	0.24	0.22	0.24	0.27
ComplEx	0.24	0.22	0.24	0.27
Poderoso (ComplEx)	0.32	0.27	0.32	0.41
DistMult	0.28	0.26	0.28	0.32
Poderoso (DistMult)	**0.35**	**0.32**	**0.36**	**0.40**
HolE	0.09	0.08	0.09	0.10
Poderoso (HolE)	0.27	0.25	0.27	0.29
plogicNet	0.10	0.09	0.11	0.14
KALE	0.01	0	0	0.05

DBPedia20k

Method	MRR	Hits@1	Hits@3	Hits@10
TransE	0.11	0.09	0.12	0.15
Poderoso (TransE)	0.14	0.10	0.15	0.20
ComplEx	0.25	0.18	0.27	0.38
Poderoso (ComplEx)	**0.29**	**0.22**	**0.33**	**0.43**
DistMult	0.25	0.20	0.29	0.34
Poderoso (DistMult)	0.28	0.20	0.33	0.41
HolE	0.13	0.11	0.14	0.17
Poderoso (HolE)	0.13	0.11	0.14	0.18
plogicNet	0.20	0.16	0.24	0.31
KALE	0.18	0	0	0.25



(a) LUBM (b) DBPedia20k (c) SOTA comparison

Figure 2: (a) and (b): Training times for vanilla KGE and Poderoso: Poderoso keeps the reasoning overhead low for most of the cases. (c): Poderoso is much faster than plogicNet and KALE for both datasets.

a factor of 3. The best model for LUBM scored an MRR of 0.35 with DistMult. For DBpedia20k, Poderoso with ComplEx scored the best MRR with value of 0.29. As the best KGE model differs for each dataset, it is fundamental to give the opportunity to users to use any KGE model. This is achieved through the modular design of Poderoso. In addition, we observe that Poderoso significantly outperforms the other hybrid baselines, namely plogicNet [13] and KALE [4]. It achieves up to 3.5× better MRR performance than plogicNet and 1.5× better MRR than KALE. This is not only because plogicNet and KALE deeply embed reasoning with KGEs and thus may lose some information but also because our approach can effortlessly use any KGE model and improve over it.

3.1.2 Training time. We now evaluate what is the overhead of adding the reasoning process when training KGEs. Figure 2 shows the training time of Poderoso for (i) different KGE models and (ii) the baselines compared to Poderoso with the KGE model that yields the best prediction accuracy. In Figure 2(a), we observe that the overhead posed by the reasoning engine is very small for the LUBM

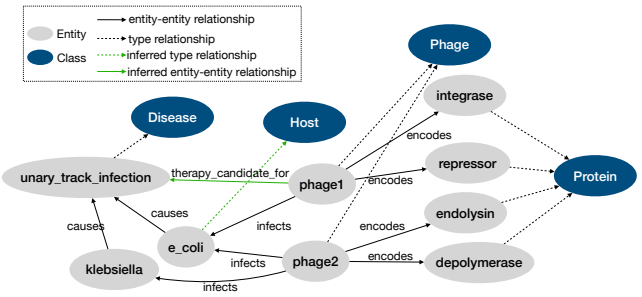


Figure 3: Subset of Phage Knowledge Graph.

dataset (up to 34% for DistMult), while for TransE the inferred triples resulting from the inference lead to faster convergence of the algorithm. Similar results we observe for the DBpedia20k dataset in Figure 2(b). Finally, Figure 2(c) shows the difference in training time between PODEROSO and the hybrid baselines KALE and plogicNet. For both datasets, PODEROSO is faster to converge.

3.2 PODEROSO for phage therapy

A common application of KG completion is in the area of bioinformatics. In particular, we are working on using PODEROSO in the field of bacteriophage (phage for short) therapy. Phages comprise one of the most abundant organisms in earth and offer a promising solution to antimicrobial resistance. Their complexity in genome diversity and unknown functions of the genes they encode have made them an under-explored organism so far. However, advances in computational biology and sequencing methods makes it more possible to understand them and harness their therapeutic potential. Still, their diversity, complexity, and unknown functionality make it hard for scientists to determine the experiments to conduct in the lab (i.e., find the hypothesis). KG completion comes as a natural solution to this problem: Given a KG that encodes all known information about phages and bacteria hosts, KG completion can provide the hypotheses to be validated with lab experiments.

We are currently constructing a phage knowledge graph and a rule base which we then plan to utilize for KG completion using PODEROSO. A subset of the KG is shown in Figure 3, while a small set of rules are shown below:

```
R1: host(Y) :- infects(X,Y).
R2: broad_host_range(X) :- infects(X, Y1), infects(X, Y2),
                genus(Y1), genus(Y2), notEqual(Y1, Y2).
R3: lysogeny(X, Y) :- phage(X), infects(X, Y),
                 encodes(X, integrase), encodes(X, repressor).
R4: anti_biofilm(X) :- encodes(X, depolymerase).
R5: therapy_candidate_for(X, Y) :- lysogeny(X, Y),
                broad_host_range(X), anti_biofilm(X).
```

These rules formulate the conditions under which a phage can be used as therapy for certain bacterium hosts. We expect that the links being inferred from PODEROSO will help us design targeted laboratory experiments to validate these predictions. This may further lead to novel and insightful findings for phage therapy.

4 OPEN CHALLENGES & CONCLUSION

To fully realize the power of PODEROSO we need to tackle a set of data systems challenges: (i) A main scalability bottleneck is the training time of KGEs and the reasoning time which depends on the rules complexity and input KG. A distributed version for both training [8] and reasoning [7] would lead to a larger scale neuro-symbolic approach. (ii) As triples output from the KGE engine contain some uncertainty it is natural to include this uncertainty into the reasoner by utilizing probabilistic reasoners. However, in some preliminary results we conducted we found probabilistic reasoners to hinder scalability even further. (iii) A naive way of communication between the two main components (e.g., sequential execution) could also contribute to huge training times. (iv) As ML models can only provide predictions to specific questions (i.e., test/inference data), it is not straightforward on how to extract "plausible" data from ML models out of the box (i.e., without any test data). We have done an initial evaluation of different sampling algorithms [1] but there is a lot of room for improvement. (v) Finding the right combinations between different ML models, fact extraction strategies, and RR methods for each task would require an optimization layer, potentially similar to query optimizers.

We proposed PODEROSO, a modular neurosymbolic framework that loosely couples KGEs with symbolic logical reasoning. In contrast to state-of-the-art approaches that rely solely on the information available in the input knowledge graph, we can incorporate domain expertise in the KG completion task which leads to increased accuracy. The modular design also allows users to plug any KGE or reasoning algorithm allowing for further optimizations. Our results showed that PODEROSO can improve accuracy while keeping training time low. The application of PODEROSO in the field of phage therapy will demonstrate its practical utility in discovering meaningful relationships among phages and bacteria hosts, potentially guiding experimental design and accelerating discovery.

REFERENCES

[1] Rama Widyadhana Bhagaskoro, Volker Markl, and Zoi Kaoudi. 2024. Evaluation of Sampling Methods for Discovering Facts from Knowledge Graph Embeddings. In *EDBT*. 664–675.

[2] Boyang Ding, Quan Wang, Bin Wang, and Li Guo. 2018. Improving Knowledge Graph Embedding Using Simple Constraints. In *ACL*. 110–121.

[3] François Goasdoué, Zoi Kaoudi, Ioana Manolescu, Jorge-Arnulfo Quiané-Ruiz, and Stamatis Zampetakis. 2015. CliqueSquare: Flat plans for massively parallel RDF queries. In *ICDE*. 771–782.

[4] Shu Guo, Quan Wang, Lihong Wang, Bin Wang, and Li Guo. 2016. Jointly Embedding Knowledge Graphs and Logical Rules. In *EMNLP*. 192–202.

[5] Victor Gutierrez-Basulto and Steven Schockae. 2018. From Knowledge Graph Embedding to Ontology Embedding? An Analysis of the Compatibility between Vector Space Representations and Rules. In *KR*. 379–388.

[6] Nitisha Jain, Trung-Kien Tran, Mohamed H. Gad-Elrab, and Daria Stepanova. 2021. Improving Knowledge Graph Embeddings with Ontological Reasoning. In *ISWC*. 379–388.

[7] Zoi Kaoudi and Manolis Koubarakis. 2013. Distributed RDFS Reasoning over Structured Overlay Networks. *Journal of Data Semantics* (2013).

[8] Christian Meilicke, Manuel Fink, Yanjie Wang, Daniel Ruffinelli, Rainer Gemulla, and Heiner Stuckenschmidt. 2018. Fine-Grained Evaluation of Rule- and Embedding-Based Systems for Knowledge Graph Completion. In *ISWC*. 3–20.

[9] Aisha Mohamed, Shameem Puthiya Parambath, Zoi Kaoudi, and Ashraf Aboulnaga. 2020. Popularity Agnostic Evaluation of Knowledge Graph Embeddings. In *UAI*. 1059–1068.

[10] Sergio Muñoz, Jorge Pérez, and Claudio Gutierrez. 2009. Simple and efficient minimal RDFS. *Journal of Web Semantics* (2009).

[11] Thomas Neumann and Gerhard Weikum. 2010. The RDF-3X engine for scalable management of RDF data. *VLDB Journal* 19, 1 (2010), 91–113.

[12] Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. 2016. Holographic Embeddings of Knowledge Graphs. In *AAAI*. 1955–1961.

[13] Meng Qu and Jian Tang. 2019. Probabilistic Logic Neural Networks for Reasoning. In *NeurIPS*. 7710–7720.

[14] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. 2007. Pellet: A practical OWL-DL reasoner. *Journal of Web Semantics* (2007).