The Effect of Rule Injection in a Leakage Free Datasets

Mirza Mohtashim Alam[1], Mojtaba Nayyeri[1], Chengjin Xu[1], Md Rashad Al Hasan Rony[2], Hamed Shariat Yazdi[1], Afshin Sadeghi[1,2], and Jens Lehmann[1,2]

[1] SDA Research Group, University of Bonn, Bonn, Germany
[2] Fraunhofer IAIS, Dresden, Germany
s6mialam@uni-bonn.de
{Nayyeri, Xu, shariat, sadeghi}@cs.uni-bonn.de
{md.rashad.al.hasan.rony, jens.lehmann}@iais.fraunhofer.de

Abstract. Knowledge graph embedding (KGE) has become a prominent topic for many AI-based tasks such as recommendation systems, natural language processing, and link prediction. Inclusion of additional knowledge such as ontology, logical rules and text improves the learning process of KGE models. One of the main characteristics of knowledge graphs (KGs) is the existence of relational patterns (e.g., symmetric and inverse relations) which usually remain unseen by the embedding models. Inclusion of logical rules provides embedding models with additional information about the patterns already present in the KGs. The injection of logical rules has not yet been studied in depth for KGE models. In this paper, we propose an approach for rule-based learning on top of the two embedding models namely RotatE and TransE within this scope of the paper. We first study the effect of rule injection in the performance of the selected models. Second, we explore how the removal of leakage from popular KGs such as FB15k and WN18 affects the results. By leakage we are referring to the patterns exist in the training set from the test set (e.g. if the test set contains (h, r, t) then it also contains t, r, h in the training set which is considered as a symmetric leakage where t, r and h refers to *tail*, *relation* and *head* respectively). Empirical results suggest that incorporation of logical rules in the training process improves the performance of KGE models.

Keywords: Relational Patterns · Knowledge Graphs Embedding · Logic.

1 Introduction

Nowadays, knowledge graphs (KGs) are one of the leading technologies in knowledge representation and representation learning. The main characteristic of this technology is its representation of facts in a triple set $\mathcal{KG} = \{(h, r, t)|h, t \in \mathcal{E}, r \in \mathcal{R}\}$ in which h, t, r refer to the head and tail and relation respectively, where $\mathcal{E}$

is a set of entities (nodes), and $\mathcal{R}$ is the set of relations (links). Relational learning is the act of applying learning models on KGs. Among the already existing learning approaches, knowledge graph embedding (KGE) models have shown influential results in link prediction. An embedding model provides vector representations of entities and relations in a KG where a triple (h, r, t) is mapped to its corresponding d dimensional vector as $\mathbf{h}, \mathbf{r}, \mathbf{t} \in \mathbb{R}^d$.

In recent years, there have been a series of embedding models which were proposed with different learning skills. Some of the popular models are TransE [1], RESCAL [9], RotatE [11], and ComplEx [12]. However, not all the models are capable of capturing potential properties of KGs. One of the main characteristics of KGs, which originates from the relational representation of knowledge, is the existence of relational patterns. More precisely, the nodes and relations of a KG are very likely to form multiple types of patterns such as symmetric, inverse, transitive, and reflexive relationships - which can be necoded by many of the KGE models. For example a family of translation-based KGE models (namely TransE, TransR, TransH) is designed for encoding such patterns [6].

Empowerment of KGE models can be done internally by adding complementary knowledge to the underlying KGs. In the case of encoding relational patterns, the injection of logical rules is seen as an approach when the model cannot capture the patterns directly from the KG itself.

In order to further study the influence of logical rules in the learning process of KGE models, firstly, we have developed a pipeline where we are able to inject rules in popular embedding models. Research suggests that rule injection can aid learning of KGE models. After injection of logical rules, we have done experiments to get an intuition about how rule injection is affecting the performance of KGEs. Secondly, we removed known patterns (leakage) from the test set, which we call refined test set. In this step, we have additionally evaluated the performance of the underlying models. Results approve our hypothesis such that rule injection can be beneficial to KGE models such as TransE and RotatE. Removal of known patterns as leakage from the test set causes the results of the same KGE models to become worse. Unfortunately, rule injection is unable to help when the leakage has been removed from the test set. Within the scope of this paper we investigate on four types of rules, namely implication, symmetric, inverse and equivalence (Table 1).

2 Related Work

Since patterns are implicitly hidden all over the KGs, in order to empower KGEs with complementary knowledge of logical rules, one need to firstly extract them. Here we represent three of the related rule extraction and injection attempts in the context of KGEs. There are multiple rule extraction frameworks such as AMIE [3] and AMIE+ [2]. They are mainly designed for rule extraction. The extracted rules are comprehensive to some extent, however for encoding of KGEs, some steps of post-processing is needed. KALE [4] is another rule extraction and injection framework which focuses on two types of rules only: implication and composition. To model these rules explicitly, they first ground the rules. Grounding a rule means instantiating that rule with concrete en-

tities. For instance, grounding a rule $(h,\ isCapitalOf,\ t) \Rightarrow (h,\ locatedIn,\ t)$ would produce a set of groundings such as $(Helsinki,\ isCapitalOf,\ Finland) \Rightarrow (Helsinki,\ locatedIn,\ Finland)$. RUGE [5] uses an iterative approach for querying the KG and retrieving soft rules to label the unlabeled triples. RUGE aims at modeling logical rules directly. Soft rules are the one that are usually hold, even if they are sometimes mistaken. An example of a soft rule is $(h, bornIn, t) \Rightarrow (h, hasNationality, t)$. RUGE needs labeled triples which is additionally costly if the underlying KG does not have that. IterE [13] is a recent paper which focuses on learning embedding and rules in an iterative fashion. Our approach differs from their in many ways (e.g., we have our own grounding generation technique and our rule loss somehow acts as a regularizer to the model rule loss). There are other tools such as WARMR [7] and ALEPH [3] which are not suitable for KGEs due to the low scalability.

3 Rule Extraction

As mentioned before, KGs include relational patterns. However, they are often not explicitly modelled via schema axioms or rules. If we want to use the logical rules as complementary knowledge in KGEs, we need to first extract patterns from KGs and make them available for such models. Towards having a set of rules extracted from KGs, a set of certain steps have been followed in this work using AMIE [3] tool as a core tool. In this section, we discuss the required steps for rule extraction from KGs that will be used later for injection into KGEs.

3.1 Input: Knowledge Graph

The KGs that are considered as an input of our rule extractor step are seen as raw data in the form of RDF (Resource Description Framework) [4]. This representation is also in triple format (*s,p,o*) (Subject, Predicate and Object). Subject and object are considered as entities. Predicate is considered as relation between the two entities. We use two main standard datasets as input KGs namely FB15k [1] and WN18 [1] for our experimental benchmark. The selection of these datasets is due to the fact that both of these KGs contain leakage in the test set. For example, if any inverse pattern of a training triple exists in the test triple it becomes a lot easier for any model to infer. We call such inverse pattern as leakage in the test set. Generally a leakage containing test set yields better result than a leakage free test data. We removed such patterns to make those test triples leakage free. Hence, we wanted to investigate the fact that, whether the models are generalized to infer unknown data which does not contain any known pattern from the training set. In this paper, we deal with four types of rules namely, implication, inverse, symmetric and equivalence. Therefore, during this research, when considering the number of rules, we only refer to these four types of rules. The number of rules in FB15k and WN18 are 414 and 16 accordingly. After the generation of groundings we have found about 71% and 93% leakage in the test set of FB15k and WN18.

[3] http://www.cs.ox.ac.uk/activities/machlearn/Aleph/aleph_toc.html
[4] https://www.w3.org/RDF/

3.2 Output: Extracted Rules

The core of rule extraction step is based on AMIE [3], a rule mining tool to extract logical rules from underlying KGs without parameter tuning or expert input. With a systematic analysis we selected AMIE as in comparison to other rule extraction tools, it has a scalable algorithm which is employed for rule mining [10]. For other tools (e.g., WARMR [7]), we needed to provide specific information related to predicates but AMIE is more efficient in this regard. In some other tools (e.g., ALEPH), a user is required to input background knowledge and positive examples for the target predicates where in AMIE we just need to provide the KG as an input [3]. Hence, AMIE is particularly useful for the extraction of logical rules from large structured data or KGs. It also assigns confidence scores for each of the rules which specifically indicates the plausibility of the extracted rule. For a given KG as an input to AMIE, it generates output as statistical measures such as standard confidence, PCA (Principal Component Analysis) confidence, head coverage for each of the mined rules. In our study, we used its latest version named as AMIE+ [10]. In order to have a final set of rules, a set of steps needs to be performed (Fig 1).

Rule generation is a significant part for this pipeline. Not all of the extracted rules provide valuable information in the training process. Therefore, rules which have a lower confidence level in the dataset are discarded. Based on the comparisons of the results and evidences in the KGs, a threshold value of 0.8 is being set to abandon unwanted rules.

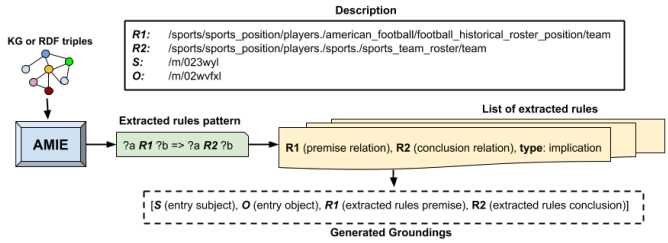

Fig. 1. The pipeline of grounding generation.

Grounding generation is required for the rule injection in the selected models. Grounding are the set of triples from the dataset that follows the patters of rules extracted by AMIE. After mining the rules from the underlying KG using AMIE+ and filtering out unwanted rules, we have generated groundings from those filtered rules, which are required for generating rule loss in the training phase. To do so, for each of the rules among the extracted results, the components are decomposed. After matching the corresponding patterns of the rules in the form of *premise* $\rightarrow$ *conclusion* for each of the class types (e.g., implication, inverse, symmetric and equivalence), the relations from premise and conclusion of the rule pattern is taken with matched class type together into a final set of rule (named as rule bag). Secondly, after the rule bag is created for each of the

items, it is taken and matched with the training triple of the respective KG. An example of how this task of grounding generation from original triple is depicted in Fig. 1. While extracting pattern of rules from KG using AMIE tool we enforce a confidence threshold value of 0.8 to filter out irrelevant pattern of rules.

4 Rule Injection

Rule injection is performed by adding groundings in the training phase. Grounding loss has been incorporated alongside of the KGE training by adding it up with the total loss. In this paper, two KGEs have been adopted in such a way that they learn grounding loss and triple score loss simultaneously as a total loss. Based on the total loss of the models (TransE and RotatE), back-propagation is performed and parameters are updated accordingly. In this section, each of the models that we use for rule injection are described in brief.

4.1 Selected KGE Models

TransE [1] calculates the plausibility of triples by measuring the distance between the addition of the head and relation $h + r$ and their respective tail t i.e. $d(h + r, t)$. The Margin Ranking Loss (MRL) is used to distinguish positive (h, r, t) and negative triples (h', r, t') by setting a margin between them.

RotatE [11] uses rotation of the head towards tail via relations in complex embedding space to compute score as $d_r = \|h \circ r - t\|$. Every embedding vector is complex and contains real and imaginary parts. RotatE uses the following loss $L = -log\sigma(\gamma - d_r(h,t)) - \sum_{i=1}^{n} \frac{1}{k} log\sigma(d_r(h'_i, t'_i) - \gamma)$. γ is the margin. The total loss has also been achieved by summing up the models loss and the rule loss as discussed previously. In both TransE and RotatE, negative triples are sampled from a probability distribution which works as a weight in the calculation of self adversarial negative sampling as proposed in RotatE [11].

4.2 Injection of Rules in Loss Function of KGE Model

In our context, we incorporate the groundings to the training phase which we obtained from the extracted rules. In this paper, we define this as rule injection. After obtaining the loss of the groundings in training phase, it is added with the mainstream loss. The principal goal of rule injection into KGE models is to

Rule	Definition	Formulation based on score function
Implication	$(h, r_1, t) \Rightarrow (h, r_2, t)$	$f^{r1}_{h,t} \leq f^{r2}_{h,t}$
Symmetric	$(h, r, t) \Longleftrightarrow (t, r, h)$	$f^{r}_{h,t} = f^{r}_{t,h} + \xi_{h,t}$
Inverse	$(h, r_1, t) \Rightarrow (t, r_2, h)$	$f^{r1}_{h,t} \leq f^{r2}_{t,h}$
Equivalence	$(h, r_1, t) \Longleftrightarrow (h, r_2, t)$	$f^{r1}_{h,t} = f^{r2}_{h,t} + \xi_{h,t}$

Table 1. Representation of rules

improve the models capability to capture the pattern of rules from knowledge graphs. Definition and the formulation of used rules are defined in Table 1 [8]. We denote the models score function $f_{h,t}$ (in table1) discussed in 4.1, as L. Hence,

it is further possible to obtain the scores, for each of the premise and conclusion represented by the definition of rules.

We call these scores as $output_1$ and $output_2$. $output_2$ is always bigger than $output_1$ (for TransE and RotatE) based on the formulation. Thus, it can act as a clear baseline for the grounding loss. Our grounding loss can be depicted in the equation $GL = \phi(L(output_2) - L(output_1) + \xi)$, where ϕ refers to the output function of the injected model and ξ is an additional constant value. Here, L can be stated as the abstraction of KGE model's (e.g., RotatE) score function.

5 Evaluation

Hyperparameter settings. A series of hyperparameters are used to achieve the result of table 2. For FB15K the learning rate α is 0.1, the rule loss multiplier ξ_{mp} is 0.05. The embedding dimension D is 200, batch size β is 2750, the rule loss multipliers {implication = 0, inverse = 0.1, symmetric = 0.01, equivalence = 0.05}, number of negative sample N is 10, γ is 24. For WN18 everything is the same except the rule loss multipliers and the margin γ. Since for WN18 we have only inverse and symmetric groundings, the multipliers are {inverse = 7, symmetric = 0.5}. The γ is kept at 10.

Results. In this section, we present the results of our evaluations. Selected comparison criteria are followed by the best practices of evaluation in embedding models. We consider Mean Rank (MR) and Hits@10 as the main criteria to evaluate the models [1]. We report the results in *Raw* and *Filtered* settings [1].

For both of these criteria, we consider looking into the details of the behaviour of KGE models with or without injection of rules. These are all evaluated on two sets of the data: raw and filtered as discussed in section 3. In table 2, we present the results of our evaluation on FB15k and WN18. The results show that rule injection has a positive influence in the result. In case of RotatE and TransE rule injection yields better result. It is clear that for FB15k dataset RotatE has a significant performance increment in both raw and filtered settings. I.e. the Hits@10 has increased from 0.85 to 0.88.

FB15K																
Models	**Standard**								**Leakage Free**							
	Without injection				**With injection**				**Without injection**				**With injection**			
	Raw		Filtered		Raw		Filtered		Raw		Filtered		Raw		Filtered	
	MR	Hits@10	MR	Hits@10	MR	Hits@10	MR	Hits@10	MR	Hits@10	MR	Hits@10	MR	Hits@10	MR	Hits@10
RotatE	190	0.5247	38	0.85	150	0.5705	35	0.8791	276	0.4355	113	0.6453	271	0.4354	108	0.6410
TransE	204	0.4605	68	0.6268	156	0.5199	51	0.6998	270	0.4304	123	0.5677	271	0.4251	125	0.5624
WN18																
Models	**Standard**								**Leakage Free**							
	Without injection				**With injection**				**Without injection**				**With injection**			
	Raw		Filtered		Raw		Filtered		Raw		Filtered		Raw		Filtered	
	MR	Hits@10	MR	Hits@10	MR	Hits@10	MR	Hits@10	MR	Hits@10	MR	Hits@10	MR	Hits@10	MR	Hits@10
RotatE	295	0.8171	281	0.9571	212	0.9262	209	0.9565	4077	0.3406	4058	0.3884	3034	0.3464	3015	0.3783
TransE	219	0.8102	207	0.9502	192	0.8978	190	0.9281	2991	0.2667	2972	0.3072	2429	0.2130	2412	0.2333

Table 2. Effect of rule injection on FB15k and WN18 test set

For TransE the Hits@10 has also been increased from 0.63 to 0.70 for FB15k dataset if we inject rules. Though it can be seen from the table that for WN18 dataset, the result does not improve upon injecting rules in filtered settings but upon injection of rules there is significant improvement in raw settings. I.e. the

Hits@10 has been improved from 0.81 to 0.90 for raw settings. However, if we refine the test set by removing the leakage, the result in every case is reduced by a significant number. Even the injection of the rules does not help in this regard. As seen in the table 2 the results are still poor even after injection.

6 Conclusion

In this work, we showcased the use of logical rules in extraction and injection of knowledge from knowledge graphs into embedding models. One of the main contributions of our work is that we removed leakage from testing set of FB15K, therefore, we have many inverse patterns in training dataset of the underlying KG, but not the leakage in testing. We mainly focused on the two main models namely: a) TransE (baseline model), and b) RotatE (current state-of-the-art model) designed for encoding the patterns. This study is a prototype of approving the role of rules in improving the learning process and performance of KGE models. As future work, we will extend the pipeline as a comprehensive framework to cover most of the other embedding models, and more datasets (FB15k-237, WN18RR) and real KGs. We also consider to automatize the rule extraction and injection which is currently done externally using AMIE with post-processing steps in grounding the rules. We also plan to train RUGE on these KGs that we have created and compare the results.

References

1. A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795, 2013.
2. L. Galárraga, C. Teflioudi, K. Hose, and F. M. Suchanek. Fast rule mining in ontological knowledge bases with AMIE+. *The VLDB Journal – The International Journal on Very Large Data Bases*, 24(6):707–730, 2015.
3. L. A. Galárraga, C. Teflioudi, K. Hose, and F. Suchanek. Amie: association rule mining under incomplete evidence in ontological knowledge bases. In *WWW*, pages 413–422, 2013.
4. S. Guo, Q. Wang, L. Wang, B. Wang, and L. Guo. Jointly embedding knowledge graphs and logical rules. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 192–202, 2016.
5. S. Guo, Q. Wang, L. Wang, B. Wang, and L. Guo. Knowledge graph embedding with iterative guidance from soft rules. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
6. S. M. Kazemi and D. Poole. Simple embedding for link prediction in knowledge graphs. In *Advances in neural information processing systems*, pages 4284–4295, 2018.
7. R. D. King, A. Srinivasan, and L. Dehaspe. Warmr: a data mining tool for chemical data. *Journal of Computer-Aided Molecular Design*, 15(2):173–181, 2001.
8. M. Nayyeri, C. Xu, J. Lehmann, and H. S. Yazdi. Logicenn: A neural based knowledge graphs embedding model with logical rules. *arXiv preprint arXiv:1908.07141*, 2019.

9. M. Nickel, V. Tresp, and H.-P. Kriegel. A three-way model for collective learning on multi-relational data. In *Icml*, volume 11, pages 809–816, 2011.
10. P. G. Omran, K. Wang, and Z. Wang. Scalable rule learning via learning representation. In *IJCAI*, pages 2149–2155, 2018.
11. Z. Sun, Z.-H. Deng, J.-Y. Nie, and J. Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197*, 2019.
12. T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard. Complex embeddings for simple link prediction. International Conference on Machine Learning (ICML), 2016.
13. W. Zhang, B. Paudel, L. Wang, J. Chen, H. Zhu, W. Zhang, A. Bernstein, and H. Chen. Iteratively learning embeddings and rules for knowledge graph reasoning. In *The World Wide Web Conference*, pages 2366–2377, 2019.