RDF* Knowledge Graph Completion by Translation

Linda Kwan[1,*], Pouya Ghiasnezhad Omran[1] and Armin Haller[1]

[1]Australian National University, Acton, ACT, 2601, Australia

Abstract

Knowledge graphs (KGs) are valuable for many applications, but they are incomplete due to their construction process or available information in a corresponding domain. Thus, Link Prediction (LP) techniques for inferring missing triples have been presented. Usually, such LP methods work on plain RDF triples, while more complex KGs like RDF*, where each fact can be qualified with another fact, are emerging. In this paper, we propose a translation-based method that can translate RDF* graphs to RDF graphs while the translation does not harm the performance of LP, whence we query the core facts (not qualifiers). We demonstrate that our translation-based method can help the link predictors that can handle RDF* directly like StarE to handle this specific kind of query more accurately. We also demonstrated that the extra complexity we create by translating could be manageable using more efficient link predictors like AnyBURL.

Keywords

RDF*, Knowledge Graph, Knowledge Graph Completion, Link Prediction

1. Introduction

Knowledge graphs (KGs) are important for modelling facts about real-world objects, but they are often incomplete. Thus, we require Link Prediction (LP) to infer facts which are not explicitly modelled in the KG. Many LP methods (e.g. [1]) can handle KGs represented in RDF that can model simple statements, while some like StarE [2] are capable of modelling complex RDF* KGs.

The Resource Description Framework (RDF) is a general-purpose framework used to model information on the Web [3]. An instance of RDF consists of a collection of statements called triples which consists of three elements: subject, predicate and object. Subjects and objects correspond to real-world entities, and predicates represent relations between those entities. However, using RDF to model complex relationships such as n-ary relations between entities is difficult. Thus, a more sophisticated model is required is express those types of statements. In Fig. 1 right side, we show an example of RDF KG.

RDF* (pronounced RDF-star) [4] extends from RDF. RDF* allows for more intuitive modelling of n-ary relations, making it more "expressive" than RDF. It allows statements about other statements to be represented as RDF* triples. In addition to the basic RDF structure, any RDF* triple can be a subject or an object of another RDF* triple. In particular, we can use qualifiers (predicate-entity pairs) to represent properties of an RDF* triple. In Fig. 1 left side, we show an

International Semantic Web Conference (ISWC) 2022: Posters, Demos, and Industry Tracks, October 23–27, 2022, Hangzhou, China

*Corresponding author.

linda.kwan@anu.edu.au (L. Kwan); p.g.omran@anu.edu.au (P. G. Omran); armin.haller@anu.edu.au (A. Haller)

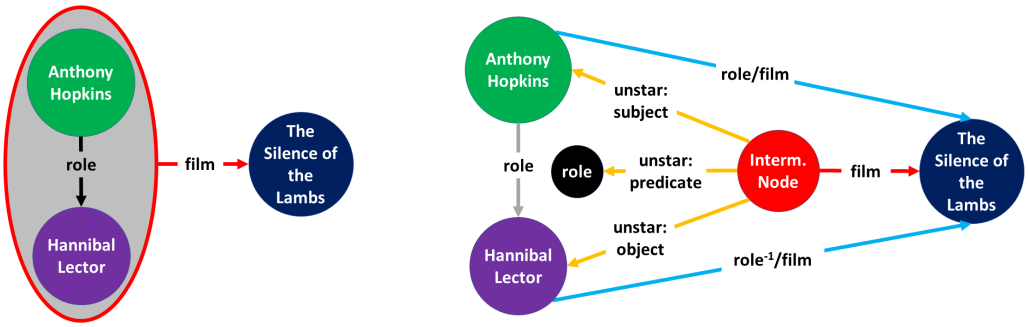

Figure 1: A visualisation of our proposed RDF* translation method (ExtRet). ExtRet extends standard reification (orange arrows) and unqualification (gray arrows).

Algorithm 1 Converting an RDF* KG to RDF KG using ExtRet

Input: RDF* KG $\mathcal{G}$, **Output**: RDF KG representation of $\mathcal{G}$

1: Let $\mathcal{H}$ be a blank RDF graph.
2: **for** RDF* triple (s, p, o) in $\mathcal{G}$ **do**
3: **if** (s, p, o) is an RDF triple **then**
4: Add (s, p, o) to $\mathcal{H}$
5: **if** s is an RDF triple **then**
6: Create a new intermediate node I for s if no existing node maps to s. Otherwise, use the existing node that maps to s.
7: Let $s = (c_s, c_p, c_o)$
8: Add the following triples to $\mathcal{H}$: $(I, \textit{unstar:S}, c_s)$, $(I, \textit{unstar:P}, c_p)$, $(I, \textit{unstar:O}, c_o)$, (c_s, c_p, c_o), $(c_s, c_p/p, o)$, $(c_o, c_p^{-1}/p, o)$, (I, p, o)
9: **if** o is an RDF triple **then**
10: Perform same as lines 6-7, but with o in lieu of s.
11: Add $(I, \textit{unstar:S}, c_s)$, $(I, \textit{unstar:P}, c_p)$, $(I, \textit{unstar:O}, c_o)$, (c_s, c_p, c_o), $(s, p/c_p, c_s)$, $(s, p/c_p^{-1}, c_o)$, (s, p, I) to $\mathcal{H}$.
12: **return** $\mathcal{H}$

example of RDF* KG. An RDF* triple without qualifiers is considered a core fact. While there are a vast number of methods for carrying out LPs for RDF KGs like AnyBURL [1], there are a few LP methods that can handle RDF* directly like StarE [2].

We propose a translation method called ExtRet (Extended Reification) which converts RDF* KG to RDF KG. We aim to improve the quality of LPs by applying our algorithm to RDF* KGs. We use existing state-of-the-art (SOTA) RDF link predictors to accomplish our research goals.

Our main contribution is proposing a novel translation method that can improve the performance of link predictors, specifically for queries of core facts. In our system, the input is RDF* and the queries are about core facts.

2. Our RDF* Translation Algorithm (ExtRet)

Standard reification [4] (orange arrows in Fig. 1) is a common method of converting RDF* KG to RDF KG. It works by replacing nested RDF* triples with intermediate nodes, then linking the intermediate nodes with subjects, predicates and objects of nested triples. However, it does not establish direct links between entities inside nested triples and entities outside them. Hence, link predictors might fail to recognise those sorts of links seen in RDF* triples.

Unqualification (grey arrow in Fig. 1) is not intended to produce an RDF KG that represents the whole RDF* KG but is used to set a minimum performance standard. It works by extracting the nested RDF triples from each RDF* triple in the KG, then constructing a separate KG from those. We introduce a novice translation algorithm named ExtRet (Extended Reification) which aims to minimise structural information loss while extending its ability to make LPs on unqualified facts. On top of standard reification and unqualification to maintain existing links, our algorithm involves adding direct relations between nested and outer entities within the same RDF* triple to the translated KG. The additional relations allow link predictors to recognise frequently occurring patterns in RDF* triples which results in higher quality LPs. Furthermore, ExtRet is designed to generalise to RDF* triples with multiple levels of nested RDF* triples.

The process of RDF* to RDF KG conversion using ExtRet is outlined in Algorithm 1. The algorithm takes an RDF* KG as an input and outputs the translated KG. First, a blank graph $\mathcal{H}$ is initialised to store the translated KG. The algorithm iterates through each RDF* triple in $\mathcal{G}$. When an RDF* triple, denoted by (s, p, o), is an RDF triple, then the triple itself is added to $\mathcal{H}$. Alternatively, if the subject s is an RDF triple, then a new intermediate node I is created to represent the core fact (c_s, c_p, c_o) = s unless such a node already exists. A set of RDF triples are added to $\mathcal{H}$ (see Line 8 of Alg. 1). Triples (I, *unstar:S*, c_s), (I, *unstar:P*, c_p) and (I, *unstar:O*, c_o) represent the links between each component of the core fact and the intermediate node, where metadata predicates with *unstar*-tags are used. (c_s, c_p/p, o) and (c_o, c_p^{-1}/p, o) represent direct links between the entities in the core fact and the object of the entire triple, where c_p/p and c_p^{-1}/p are distinct new predicates formed essentially by concatenating c_p and p. (s, p, I) represents the RDF* triple with the core fact replaced by the intermediate node. Also, the core fact itself is added to $\mathcal{H}$ to preserve the relationship between it entities. In the case where the object o is a core fact instead of s, the process of decomposing the RDF* triple into several RDF triples is similar, but with small changes to account for the position of the core fact (see Lines 9-11 of Alg. 1). The algorithm returns the translated KG after converting each RDF* triple in $\mathcal{G}$. ExtRet generalises to KGs with multi-levelled RDF* triples by repeatedly applying Algorithm 1 and treating each nested triple as a core fact.

3. Experiments

We conducted a set of experiments to evaluate our proposed system [1]. We demonstrate: (i) ExtRet can be used to generate a set of RDF facts that can be used to answer the queries about the core facts with higher accuracy than the original RDF* using SOTA RDF* LP StarE [2]. (ii) Although ExtRet generates more entities, predicates and facts, the complexity of LPs on the translated system is manageable by using more efficient link predictors like AnyBURL [1].

Link Prediction We performed our experiment using the JF17K dataset [5] as our benchmark. Due to our computational resource restriction, we prepared a sampled JF17K to run StarE on the translated version of sampled JF17K. Our train-valid-test split ratio is 64:16:20. The statistics of original and samples KG can be found in Table 1. The three algorithms used to convert those training and validation sets are standard reification, ExtRet and unqualification (which removes the qualifiers from each statement). Additionally, we applied the unqualification algorithm to

[1]Extensive results and code can be found at https://github.com/lindakwan/ExtRet

Table 1
Statistical information about the full and sampled knowledge graphs.

Dataset	Algorithm	#Entities	#Preds	#Facts (Train)	#Facts (Test)	w/Quals (%) (Train)
JF17K (sampled)	Original RDF*	2976	183	3631	887	1885 (51.9%)
	Unqualification	2524 (-15%)	121 (-34%)	3320 (-9%)	887	-
	Std reification	4497 (51%)	172 (-6%)	8053 (122%)	887	-
	ExtRet	4497 (51%)	310 (69%)	11929 (229%)	887	-
JF17K (full)	Original RDF*	25092	493	64955	15646	18286 (28.2%)
	Unqualification	22763 (-9%)	320 (-35%)	56722 (-13%)	15646	-
	Std reification	37738 (50%)	458 (-7%)	102491 (58%)	15646	-
	ExtRet	37738 (50%)	842 (71%)	141175 (117%)	15646	-

Table 2
Summary of experimental results.

LP model	Dataset	Algorithm	MRR	hits@1	hits@10	Train time
StarE	JF17K (sampled)	Original RDF*	0.32736	0.23545	0.50635	2h30m
		Unqualification	0.36824	0.27460	0.55503	**1h13m**
		Std reification	0.19000	0.14339	0.28095	8h33m
		ExtRet	**0.53018**	**0.45291**	**0.67778**	14h27m
AnyBURL	JF17K (sampled)	Unqualification	**0.2467**	**0.1794**	**0.4101**	1m40s
		Std reification	0.1367	0.1026	0.2021	1m40s
		ExtRet	**0.2467**	**0.1794**	**0.4101**	1m40s
AnyBURL	JF17K (full)	Unqualification	**0.1738**	**0.1159**	**0.3233**	1m40s
		Std reification	0.0605	0.0357	0.1300	1m40s
		ExtRet	**0.1738**	**0.1159**	**0.3233**	1m40s

each testing set since we are mainly interested in making predictions about the core facts. We used both StarE and AnyBURL to investigate their ability to make LPs on translated KGs. We evaluate the performance of LPs using the metrics MRR (Mean Reciprocal Rank) and hits@N adopted from [6].

KG Statistics In Table 1, percentages next to the number of entities, predicates and facts indicate the amount of change from the original KG. ExtRet involves constructing many new components, especially more new predicates and facts than standard reification, resulting in large percentage increases. Unqualification involves decreasing the number of components as qualifiers are deleted from facts, resulting in missing entities and predicates, and duplicate core facts. The percentages in the last column indicate the proportion of RDF* facts in the training set where core facts have qualifiers.

4. Results & Discussion

Our experiment in Table 2 demonstrates that ExtRet outperforms standard reification when tested on both StarE and AnyBURL in all three metrics and two datasets with significant margins. The extra triples added to the decomposed RDF* triple help to reinforce connections between entities in nested RDF* triples and outer entities. Those extra connections result in better LP results as the existence of those relations is taken into account. However, increasing the number of triples in the translated graph increases the training time of StarE despite the performance improvements. This is due to the need to update all the embedding matrices corresponding to the additional entities and relations added to the translated graph. 51.9% of statements in the training set of our sampled JF17K dataset consists of qualifiers and those require more than 14 hours of training. Thus, a training set with the same amount of statements, but with 100% of

them containing qualifiers, could potentially require more training time. Also, a dataset with the same amount of statements, but with more unique core facts, would result in a larger number of triples in the translated graph, and therefore, longer training time in StarE. Due to the long waiting time involved when using StarE to train a large dataset, only AnyBURL was run on the full JF17K dataset.

While training using AnyBURL is significantly faster than the StarE, its performance is much less optimal. Critics of rule-based approaches argue that rules generated become redundant when making predictions about sets of entities connected by different rules. Hence, our decision to use both AnyBURL and StarE to evaluate the performance of ExtRet is justified by the need to verify that our translation algorithm can result in good performance across LP systems with different limitations. As it is not ideal to rely on MRR and hits scores alone, we measure the performance of translation algorithms by comparing them against benchmarks.

We compare the results of both standard reification and ExtRet against the performance of the original RDF* KGs and the KGs generated by unqualification. We use unqualification as a performance benchmark for ExtRet to meet since AnyBURL does not support RDF* triples. It makes sense for a KG rich in metadata information to make better predictions than an unqualified graph. In most cases, our algorithm manages to meet those standards. Also, the performance of ExtRet exceeds the performance of the original RDF* graph where StarE is applied to the sampled JF17K dataset. Hence, ExtRet results in significant information gain.

5. Conclusion & Future Work

Our proposed translation algorithm ExtRet outperforms SOTA translation algorithms when using StarE to make LPs about core facts. AnyBURL is more scalable than StarE although their results are less optimal than StarE. For future work, we would consider using other benchmark datasets to compare the LP quality of ExtRet against standard reification. Additionally, we could run our experiment on a machine with GPUs that is capable of handling larger datasets.

References

[1] C. Meilicke, M. W. Chekol, D. Ruffinelli, H. Stuckenschmidt, Anytime bottom-up rule learning for knowledge graph completion, in: IJCAI, 2019.

[2] M. Galkin, P. Trivedi, G. Maheshwari, R. Usbeck, J. Lehmann, Message passing for hyper-relational knowledge graphs, in: EMNLP, 2020.

[3] R. Cyganiak, D. Wood, M. Lanthaler, G. Klyne, J. J. Carroll, B. McBride, Rdf 1.1 concepts and abstract syntax, 2014. URL: https://www.w3.org/TR/rdf11-concepts/.

[4] D. Arndt, J. Broekstra, B. DuCharme, O. Lassila, P. F. Patel-Schneider, E. Prud'hommeaux, T. Thibodeau, Jr., B. Thompson, Rdf-star and sparql-star, https://w3c.github.io/rdf-star/cg-spec/editors_draft.html, 2021.

[5] J. Wen, J. Li, Y. Mao, S. Chen, R. Zhang, On the representation and embedding of knowledge bases beyond binary relations, in: IJCAI, 2016.

[6] A. Bordes, N. Usunier, A. García-Durán, J. Weston, O. Yakhnenko, Translating embeddings for modeling multi-relational data, in: NIPS, 2013.