An Introduction to AnyBURL

Christian Meilicke(✉), Melisachew Wudage Chekol, Daniel Ruffinelli, and Heiner Stuckenschmidt

University of Mannheim, Mannheim, Germany
christian@informatik.uni-mannheim.de

1 Introduction

Current research on knowledge graph completion is often concerned with latent approaches that are based on the idea to embed a knowledge graph into a low dimensional vector space. At the same time symbolic approaches have attracted less attention [13]. However, such approaches have a big advantage: they yield an explanation in terms of the rules that trigger a prediction. In this paper we propose a bottom-up technique for efficiently learning logical rules from large knowledge graphs inspired by classic bottom-up rule learning approaches as Golem [8] and Aleph [10]. Our approach is called AnyBURL (**A**nytime **B**ottom-**U**p **R**ule **Le**arning). We report on experiments where we evaluated AnyBURL on datasets that have been labelled as hard cases for simple (rule-based) approaches. Our approach performs as good as and sometimes better than most models that have been proposed recently. Moreover, the required resources in terms of memory and runtime are significantly smaller compared to latent approaches. This paper is an extended abstract of an IJCAI 2019 paper [6].

2 Language Bias and Algorithm

A small subset of a knowledge graph is shown in Fig. 1. Suppose that we are interested in finding rules that explain why Ed speaks Dutch, which corresponds to the fact $speaks(ed, d)$. To construct useful rules, we look at all paths of length n that start at ed or d. Note that we allow a path to be constructed by following in- and outgoing edges. We have marked three paths starting at ed in Fig. 1. Two of these paths are acyclic paths ending somewhere in the knowledge graph, while the third path is, together with $speaks(ed, d)$, cyclic. On the left side of the figure these paths are shown in terms of their corresponding bottom rules.

In [6] we argued that any useful rule, that can be generalized from such a bottom rule, must belong to a certain type of rule. Thus, we can directly instantiate these types instead of building up a complete generalization lattice. We list in the following all rules that result from generalizing the second and the third bottom rule. We use upper-case letters to refer to variables.

C. Benzmüller and H. Stuckenschmidt (Eds.): KI 2019, LNAI 11793, pp. 244–248, 2019.
https://doi.org/10.1007/978-3-030-30179-8_20

$speaks(ed, d) \leftarrow born(ed, a)$
$speaks(ed, d) \leftarrow mar(ed, lisa), born(lisa, a)$
$speaks(ed, d) \leftarrow lives(ed, nl), lang(nl, d)$

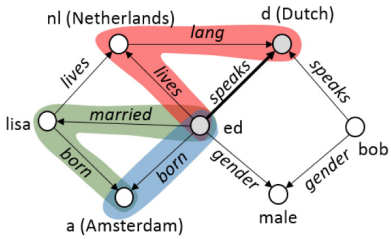

Fig. 1. Three paths sampled from a knowledge graph

$$speaks(X, d) \leftarrow married(X, A_2), born(A_2, a) \quad (1)$$
$$speaks(X, d) \leftarrow married(X, A_2), born(A_2, A_3) \quad (2)$$
$$speaks(X, Y) \leftarrow lives(X, A_2), lang(A_2, Y) \quad (3)$$
$$speaks(X, d) \leftarrow lives(X, A_2), lang(A_2, d) \quad (4)$$
$$speaks(ed, Y) \leftarrow lives(ed, A_2), lang(A_2, Y) \quad (5)$$

The main algorithm of AnyBURL samples random paths of length n and generalizes them to rules as the ones shown above. Then AnyBURL computes confidence and support of each rule. If the rule fulfils a quality criteria defined by the user, e.g., being above minimal support or confidence threshold, it is stored. AnyBURL checks how much new rules are learned within a certain interval. If the fraction of new rules learned within this interval is below a threshold it increases n and continues to search for longer rules.

Given a completion task as $r(a, ?)$, we have to compute a ranking of the top-k candidates that can substitute the question mark. It would be straight forward to create such a ranking based on the rules that have been learned, if each entity would be generated by at most one rule. We could just order the proposed entities by the confidence values of the rules that suggested them. However, an entity is usually suggested by several rules. If we would assume that these rules are independent, we could use a Noisy-Or aggregation. However, the assumption that the rules are independent is often not valid. In [6] we report also about experiments that support this claim. Instead of that we apply a rather simple but efficient approach. We order the candidates via the maximum of the confidences of all rules that have generated the candidates. If the maximum score of several candidates is the same, we order these candidates via the second best rule that generates them, and so on, until we find a rule that makes a difference. The results of this approach can be computed without grounding all relevant rules.

3 Results

In our experiments we have used the standard knowledge base completion evaluation datasets FB15k and WN18 [1] plus their harder variants FB15-237 [11] and

Table 1. Comparing our approach against current state of the art

Approach	WN18			WN18RR			FB15			FB15-237		
	h@1	h@10	MRR	h@1	h@10	MRR	h@1	h@10	MRR	h@1	h@10	MRR
SimplE [4]	93.9	94.7	94.2				66.0	83.8	72.7			
ConvE [2]	93.5	95.5	94.2	39	48	46	67.0	87.3	74.5	23.9	49.1	31.6
ComplEx-N3 [5]		96	95		57	48		91	86		56	37
R-GCN+ [9]	69.7	96.4	81.9				60.1	84.2	69.6	15.1	41.7	24.9
CrossE [14]	74.1	95.0	83.0				63.4	87.5	72.8	21.1	47.4	29.9
AMIE+ [3]	87.2	94.8		35.8	38.8		64.7	85.8		17.4	40.9	
AnyBURL, 10 s	94.2	94.9	≥94	43.2	52.7	≥46	79.6	83.8	≥81	13.4	25.9	≥17
AnyBURL, 100 s	94.6	95.9	≥95	44.5	54.9	≥48	80.8	87.6	≥83	19.6	41.0	≥26
AnyBURL, 1000 s	93.9	95.6	≥95	44.6	55.5	≥48	80.4	89.0	≥83	23.0	47.9	≥30
AnyBURL, 10000 s	93.5	95.4	≥94	44.1	55.2	≥47	79.6	88.7	≥82	23.3	48.6	≥31

WN18RR [2]. The first block in Table 1 lists the results of current state of the art completion techniques. We selected five models, which achieved very good results published within the last year at top conferences. AnyBURL achieves after a 1000 seconds learning phase results that are at the same level and sometimes slightly better than most of these models. AnyBURL has, for example, better hits@1, hits@10, and MMR results[1] than ConvE [2] on all datasets except FB15-237, where the results of AnyBURL are only slightly worse. AnyBURL outperforms SimplE [4] already when using the rules that have been learned after 10 s.

Exceptionally, the ComplEx-N3 model proposed in [5], originally introduced in [12], achieves better results than any other model including AnyBURL. While we ran AnyBURL on a standard laptop, the ComplEx-N3 runtimes are based on the use of a Quadro GP100 GPU. Moreover, AnyBURL runtimes are significantly lower.

In line six we present results for the rule learner AMIE+. These results have been reported in [7] in an environment similar to ours. AMIE+ does not have an anytime behaviour, however, the parameters have been chosen dataset specific to exploit a time frame of 10 h as good as possible. Thus, the results are roughly comparable to the 10000 s results of AnyBURL. AnyBURL generates significantly better results than AMIE+ on all datasets.

4 Conclusion

AnyBURL generates in short time results that are as good and better than many recent state of the art techniques, while using only limited resources (both in

[1] As in [1] we compute filtered hits@k. We omit the adjective filtered. Since AnyBURL cannot compute a complete ranking efficiently, we assume that any candidate ranked at a position >k is not a correct prediction. We use this assumption to compute a lower bound for the MRR.

terms of memory and computational power). Compared to latent representation models AnyBURL has several advantages in common with other rule based approaches: (1) The candidate ranking can be explained in terms of the rules that generated this ranking. (2) The generated model (= rule set) can be reused for a dataset using the same predicates and an overlapping set of (important) constants. (3) A rule based approach does not require to learn dataset specific hyper parameters. In the future we plan to extend the language bias of AnyBURL. AnyBURL is implemented as a Java program without any dependencies. It is available at http://web.informatik.uni-mannheim.de/AnyBURL/.

References

1. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: Advances in Neural Information Processing Systems, pp. 2787–2795 (2013)
2. Dettmers, T., Minervini, P., Stenetorp, P., Riedel, S.: Convolutional 2D knowledge graph embeddings. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
3. Galárraga, L., Teflioudi, C., Hose, K., Suchanek, F.M.: Fast rule mining in ontological knowledge bases with AMIE+. VLDB J.- Int. J. Very Large Data Bases **24**(6), 707–730 (2015)
4. Kazemi, S.M., Poole, D.: Simple embedding for link prediction in knowledge graphs. In: Advances in Neural Information Processing Systems, pp. 4289–4300 (2018)
5. Lacroix, T., Usunier, N., Obozinski, G.: Canonical tensor decomposition for knowledge base completion. In: ICML, pp. 2869–2878 (2018)
6. Meilicke, C., Chekol, M.W., Ruffinelli, D., Stuckenschmidt, H.: Anytime bottom-up rule learning for knowledge graph completion. In: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI 2019) (2019)
7. Meilicke, C., Fink, M., Wang, Y., Ruffinelli, D., Gemulla, R., Stuckenschmidt, H.: Fine-grained evaluation of rule- and embedding-based systems for knowledge graph completion. In: Vrandečić, D., et al. (eds.) ISWC 2018. LNCS, vol. 11136, pp. 3–20. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-00671-6_1
8. Muggleton, S.H., Feng, C.: Efficient induction of logic programs. In: Proceedings of the First Conference on Algorithmic Learning Theory, pp. 368–381 (1990)
9. Schlichtkrull, M., Kipf, T.N., Bloem, P., van den Berg, R., Titov, I., Welling, M.: Modeling relational data with graph convolutional networks. In: Gangemi, A., et al. (eds.) ESWC 2018. LNCS, vol. 10843, pp. 593–607. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-93417-4_38
10. Srinivasan, A.: The aleph manual. Technical report, Computing Laboratory, Oxford University (2000)
11. Toutanova, K., Chen, D.: Observed versus latent features for knowledge base and text inference. In: Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality, pp. 57–66 (2015)
12. Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., Bouchard, G.: Complex embeddings for simple link prediction. In: International Conference on Machine Learning, pp. 2071–2080 (2016)

13. Wang, Q., Mao, Z., Wang, B., Guo, L.: Knowledge graph embedding: a survey of approaches and applications. IEEE Trans. Knowl. Data Eng. **29**(12), 2724–2743 (2017)
14. Zhang, W., Paudel, B., Zhang, W., Bernstein, A., Chen, H.: Interaction embeddings for prediction and explanation in knowledge graphs. In: Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, pp. 96–104. ACM (2019)