Knowledge Graph Completion with Probabilistic Logic Programming*

Elisabetta Gentili[1,*]

[1]Department of Engineering, University of Ferrara, Via Saragat, 1, 44124, Ferrara, Italy

Abstract

Knowledge Graphs have gained popularity in the last decade, given their ability to represent huge structured knowledge bases. However, they are often incomplete and thus Knowledge Graph Completion (KGC) is currently a hot topic. In this paper we present our idea of performing KGC by learning liftable probabilistic logic programs via regularization, using LIFTCOVER+, with the aim of obtaining more accurate results while learning a smaller set of rules.

Keywords

Knowledge Graphs Completion, Probabilistic Inductive Logic Programming, Regularization

1. Background

Even though the term Knowledge Graph (KG) has been used from 1973 [1], there is still no universally accepted formal definition for it [2]. Nevertheless, we can say that KGs are graph-based representations of knowledge in terms of relationships between entities. More practically, following the Resource Description Framework (RDF) data model [3], a KG is a set of triples $\langle s, p, o\rangle$ where s is the subject, p is the predicate, and o is the object. An example of such a triple is $\langle ed, speaks, dutch\rangle$. Figure 1 [4] shows an example of a KG.

Data can be naturally and effectively represented with graphs in many real-world domains, such as computer networks, social networks, healthcare (diseases, molecules), transportation, and so on. For this reason, KGs are employed for different tasks like query answering, recommender systems, chatbots and voice assistants.

KGs have become popular over the last decade thanks to the introduction of Google's Knowledge Graph in 2012 [5], used to automatically generate knowledge panels. Knowledge panels are boxes containing information coming from various sources on the web, and are meant to give the user an overview of the researched topic. Aside from Google's, other popular examples of KGs, both proprietary and open, are Amazon Product graph, Facebook Graph API, IBM Watson, Microsoft Satori, Wikimedia's Wikidata [6], YAGO [7], and FreeBase [8].

AIxIA'23: 22nd International Conference of the Italian Association for Artificial Intelligence - Doctoral Consortium, November 06–09, 2023, Rome, Italy

*Doctoral project supervised by Evelina Lamma (Department of Engineering, University of Ferrara) and Fabrizio Riguzzi (Department of Mathematics and Computer Science, University of Ferrara).

*Corresponding author.

elisabetta.gentili1@unife.it (E. Gentili)

009-0006-6901-0540 (E. Gentili)

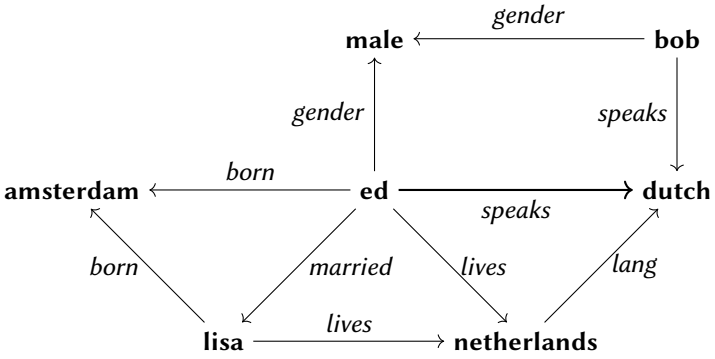

Figure 1: An example of Knowledge Graph, from [4].

Since KGs cannot contain all the possible knowledge in the domain, they are usually incomplete and sparse, thus it is often necessary to infer missing information (entities or relationships). This task is referred to as Knowledge Graph Completion (KGC). According to what is missing, KGC can be divided into specific tasks [9], such as link prediction, entity prediction, or relation prediction.

KGC is a very active field of research and many algorithms have been proposed to solve the problem. They can be divided into traditional and representation learning-based methods [9]. Rule-based reasoning methods and probabilistic graphical models, such as Markov Logic Networks, are examples of techniques that fall under the first category. On the other hand, KGC methods based on embeddings or neural network models are an example of technique belonging to the second category.

2. Methodology

Our goal is to perform KGC with a Probabilistic Logic Programming (PLP) algorithm [10], to learn logical rules representing paths in large KGs, which will allow the ranking of candidates in terms of probabilities. PLP combines logic-based languages and uncertainty [11]. In the last years PLP under the distribution semantics [12] in particular has gained high popularity thanks to its expressiveness, especially in domains where uncertainty plays a relevant role [13, 14, 15]. Logic Programs with Annotated Disjunctions (LPADs) [16] are a PLP language under the distribution semantics. In LPADs, heads of clauses are disjunctions in which each atom is annotated with a probability. Liftable Probabilistic Logic Programs [17] have been proposed to perform lifted inference [18] in an efficient way by taking into consideration populations of individuals instead of considering each individual separately. LIFTCOVER+ [19] performs structure and parameter learning of liftable probabilistic logic programs, and it is an improved version of LIFTCOVER [17] that adds regularization and gradient descent for parameter learning, to improve the quality of the solutions and prevent overfitting.

The triples of a KG can be represented by First-Order Logic (FOL) atoms. For example, the triple $\langle ed, speaks, dutch \rangle$ can be represented by the atom $speaks(ed, dutch)$. Therefore, we

consider a KG $\mathbb{G}$ as a set of ground atoms or facts:

$$\mathbb{G} = \{relation(subject, object) \mid relation \in \mathbb{R},\ subject,\ object \in \mathbb{C}\}$$

where $\mathbb{C}$ is a set of constants (entities) and $\mathbb{R}$ is a set of binary predicates (relations).

Following the approach of AnyBURL [4], we want to learn chain rules of increasing length of the form:

$$h(X_0, X_1) \leftarrow b_1(X_1, X_2), \ldots, b_n(X_n, X_{n+1}).$$

Here $h(\ldots)$ is the *head* of the rule, while the $b_i(\ldots)$ form the *body*. Upper-case letters are variables.

AnyBURL is an anytime algorithm designed to learn rules from knowledge graphs by following the bottom-up paradigm. With n referring to the number of body atoms and starting from $n = 2$, AnyBURL iteratively samples random paths of length n, and learns rules of length $n - 1$, until a certain saturation is reached. For each rule, the confidence is computed as the number of head and body groundings that are true divided by the number of body groundings that are true. Considering the KG in Figure 1 from [4], in order to explain the fact $speaks(ed, dutch)$, AnyBURL finds all the paths starting from *ed* or *netherlands*, and their corresponding bottom rule. An example of bottom rule to be generalized is the following:

$$speaks(ed, dutch) \leftarrow lives(ed, netherlands),\ lang(dutch, netherlands).$$

Then, starting from these bottom rules, AnyBURL extracts generalized rules.

Given a ground path rule of the form $h(c_0, c_1) \leftarrow b_1(c_1, c_2), \ldots, b_n(c_n, c_{n+1})$, extracted rules can be of one of three types:

1. rules that generalize acyclic ground path rules, i.e., rules where $c_0 \neq c_{n+1}$:

$$h(c_0, X) \leftarrow b_1(X, A_2), \ldots, b_n(A_n, A_{n+1});$$

2. rules that generalize cyclic ground path rules, i.e., rules where $c_0 = c_{n+1}$:

$$h(Y, X) \leftarrow b_1(X, A_2), \ldots, b_n(A_n, Y);$$

3. rules that generalize both acyclic and cyclic ground path rules:

$$h(c_0, X) \leftarrow b_1(X, A_2), \ldots, b_n(A_n, c_{n+1});$$

where X, Y are variables that appear in the head, while A_i can appear only in the body.

A rule is stored only if a certain quality criteria is met (e.g., the confidence above a specified threshold). Then, n is increased by 1 and the loop is repeated. Given a completion task $r(a, ?)$, the learnt rules are then used to find an entity c such that $r(a, c) \notin \mathbb{G}$ is true, with $r \in \mathbb{R}$ and $a, c \in \mathbb{C}$. The candidate values for c are ordered according to the maximum confidence of all the rules that generated them. In case of a tie for some candidates, the second rule that generated them is considered, and so on. The following generalized rules can be obtained from the above-mentioned bottom rule:

$$speaks(ed, Y) \leftarrow lives(ed, A_2),\ lang(A_2, Y)$$

$$speaks(ed,\ dutch) \leftarrow lives(ed, A_2),\ lang(A_2, dutch)$$
$$speaks(ed, Y) \leftarrow lives(ed, A_2),\ lang(A_2, Y)$$

where upper-case letters are variables.

In our approach, we learn rules with the AnyBURL algorithm but we attach to each rule a probability instead of a confidence and we tune the probabilities of the set of rules with parameter learning. We use LIFTCOVER+ that uses regularization: we try to bring the parameters close to 0 as much as possible and we remove the rules with a probability below a threshold, because they have a small influence on the final result. By doing so, the ranking should be more accurate and the number of rules learnt smaller. Furthermore, being a rule-based approach, the resulting candidate ranking is explained by the rules, and thus easily understandable.

3. Related work

Aside from AnyBURL [20, 4], several other approaches have been proposed for KGC.

AMIE [21] and its improved version AMIE++ [22] are top-down rule learning systems tailored to support the Open World Assumption, that is, a scenario in which absent data cannot be used as counterexamples. AMIE++ was developed to work with larger knowledge bases. The authors of [23] proposed a neural model for Existential Positive First-Order logical queries represented via box embeddings. In [24], the authors developed an approach for mining relational nonmonotonic rules from KGs under the Open World Assumption, that combines rule learning and nonmonotonic reasoning. DRUM [25] is an approach used for mining first-order logical rules from KGs by performing inductive link prediction and thus able to manage previously unseen entities. The authors of [26] proposed a rule learning approach to learn typed rules using type information to guide the rule search.

4. Conclusions

In this paper, we presented our approach for performing KGC with PLP. The approach is based on the AnyBURL algorithm, however, it differs from it in the ranking method. In fact, we use probabilities instead of confidence values. Furthermore, the algorithm we employ, LIFTCOVER+, uses regularization to prune rules with negligible probabilities. In this way we should be able to obtain a more accurate ranking and a smaller set of learnt rules.

Acknowledgments

This article was produced while attending the PhD programme in Engineering Science at the University of Ferrara, Cycle XXXVIII, with the support of a scholarship financed by the Ministerial Decree no. 351 of 9th April 2022, based on the NRRP - funded by the European Union - NextGenerationEU - Mission 4 “Education and Research”, Component 1 “Enhancement of the offer of educational services: from nurseries to universities” - Investment 4.1 “Extension of the number of research doctorates and innovative doctorates for public administration and cultural heritage”. This work has been partially supported by the Spoke 1 “FutureHPC & Big-Data” of the

Italian Research Center on High-Performance Computing, Big Data and Quantum Computing (ICSC) funded by MUR Missione 4 - Next Generation EU (NGEU), by TAILOR, a project funded by EU Horizon 2020 research and innovation programme under GA No. 952215, and by the National Group of Computing Science (GNCS-INDAM).

References

[1] E. W. Schneider, Course modularization applied: The interface system and its implications for sequence control and data analysis., 1973.

[2] L. Ehrlinger, W. Wöß, Towards a definition of knowledge graphs., SEMANTiCS (Posters, Demos, SuCCESS) 48 (2016) 2.

[3] O. Lassila, R. R. Swick, Resource description framework (rdf) model and syntax specification, 1999.

[4] C. Meilicke, M. W. Chekol, D. Ruffinelli, H. Stuckenschmidt, Anytime bottom-up rule learning for knowledge graph completion., in: IJCAI, 2019, pp. 3137–3143.

[5] A. Singhal, et al., Introducing the knowledge graph: things, not strings, Official Google blog 5 (2012) 3.

[6] T. Bayer, The Wikidata revolution is here: enabling structured data on Wikipedia, 2013.

[7] F. M. Suchanek, G. Kasneci, G. Weikum, Yago: A large ontology from wikipedia and wordnet, Journal of Web Semantics 6 (2008) 203–217.

[8] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, J. Taylor, Freebase: a collaboratively created graph database for structuring human knowledge, in: Proceedings of the 2008 ACM SIGMOD international conference on Management of data, 2008, pp. 1247–1250.

[9] Z. Chen, Y. Wang, B. Zhao, J. Cheng, X. Zhao, Z. Duan, Knowledge graph completion: A review, IEEE Access 8 (2020) 192435–192456.

[10] D. Azzolini, E. Gentili, F. Riguzzi, Link Prediction in Knowledge Graphs with Probabilistic Logic Programming: Work in Progress, in: J. Arias, S. Batsakis, W. Faber, G. Gupta, F. Pacenza, E. Papadakis, L. Robaldo, K. Ruckschloss, E. Salazar, Z. G. Saribatur, I. Tachmazidis, F. Weitkamper, A. Wyner (Eds.), Proceedings of the International Conference on Logic Programming 2023 Workshops co-located with the 39th International Conference on Logic Programming (ICLP 2023), volume 3437 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2023, pp. 1–4.

[11] F. Riguzzi, Foundations of Probabilistic Logic Programming Languages, Semantics, Inference and Learning, Second Edition, River Publishers, Gistrup, Denmark, 2023.

[12] T. Sato, A statistical learning method for logic programs with distribution semantics, in: L. Sterling (Ed.), Logic Programming, Proceedings of the Twelfth International Conference on Logic Programming, Tokyo, Japan, June 13-16, 1995, MIT Press, 1995, pp. 715–729. doi:10.7551/mitpress/4298.003.0069.

[13] L. De Raedt, A. Kimmig, Probabilistic (logic) programming concepts, Machine Learning 100 (2015) 5–47. doi:10.1007/s10994-015-5494-z.

[14] F. Riguzzi, E. Lamma, M. Alberti, E. Bellodi, R. Zese, G. Cota, Probabilistic logic programming for natural language processing, in: F. Chesani, P. Mello, M. Milano (Eds.), Workshop

on Deep Understanding and Reasoning, URANIA 2016, volume 1802 of *CEUR Workshop Proceedings*, Sun SITE Central Europe, 2017, pp. 30–37.

[15] A. Nguembang Fadja, F. Riguzzi, Probabilistic logic programming in action, in: A. Holzinger, R. Goebel, M. Ferri, V. Palade (Eds.), Towards Integrative Machine Learning and Knowledge Extraction, volume 10344 of *Lecture Notes in Computer Science*, Springer, 2017, pp. 89–116. doi:10.1007/978-3-319-69775-8_5.

[16] J. Vennekens, S. Verbaeten, M. Bruynooghe, Logic Programs With Annotated Disjunctions, in: 20th International Conference on Logic Programming (ICLP 2004), volume 3132 of *Lecture Notes in Computer Science*, Springer, 2004, pp. 431–445.

[17] A. Nguembang Fadja, F. Riguzzi, Lifted discriminative learning of probabilistic logic programs, Machine Learning 108 (2019) 1111–1135.

[18] D. Poole, First-order probabilistic inference, in: G. Gottlob, T. Walsh (Eds.), IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003, Morgan Kaufmann Publishers, 2003, pp. 985–991.

[19] E. Gentili, A. Bizzarri, D. Azzolini, R. Zese, F. Riguzzi, Regularization in probabilistic inductive logic programming, in: International Conference on Inductive Logic Programming, 2023. (in press).

[20] C. Meilicke, M. W. Chekol, P. Betz, M. Fink, H. Stuckeschmidt, Anytime bottom-up rule learning for large-scale knowledge graph completion, The VLDB Journal (2023) 1–31.

[21] L. A. Galárraga, C. Teflioudi, K. Hose, F. Suchanek, Amie: association rule mining under incomplete evidence in ontological knowledge bases, in: Proceedings of the 22nd international conference on World Wide Web, 2013, pp. 413–422.

[22] L. Galárraga, C. Teflioudi, K. Hose, F. M. Suchanek, Fast rule mining in ontological knowledge bases with AMIE++, The VLDB Journal 24 (2015) 707–730.

[23] H. Ren, W. Hu, J. Leskovec, Query2box: Reasoning over knowledge graphs in vector space using box embeddings, arXiv preprint arXiv:2002.05969 (2020).

[24] H. D. Tran, D. Stepanova, M. H. Gad-Elrab, F. A. Lisi, G. Weikum, Towards nonmonotonic relational learning from knowledge graphs, in: Inductive Logic Programming: 26th International Conference, ILP 2016, London, UK, September 4-6, 2016, Revised Selected Papers 26, Springer, 2017, pp. 94–107.

[25] A. Sadeghian, M. Armandpour, P. Ding, D. Z. Wang, Drum: End-to-end differentiable rule mining on knowledge graphs, Advances in Neural Information Processing Systems 32 (2019).

[26] H. Wu, Z. Wang, K. Wang, Y.-D. Shen, Learning typed rules over knowledge graphs, in: Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning, volume 19, 2022, pp. 494–503.