Mining the Web of Data with Metaqueries

Francesca A. Lisi

Dipartimento di Informatica &
Centro Interdipartimentale di Logica e Applicazioni (CILA)
Università degli Studi di Bari "Aldo Moro", Italy
FrancescaAlessandra.Lisi@uniba.it

Abstract. The Web of Data uses the World Wide Web (WWW) infrastructure to represent and interrelate data sources. These sources are referred to as *knowledge graphs* (KGs) and correspond to huge collections of facts in the form of RDF triples. The analysis of data contained in a KG is propaedeutic to several crucial KG curation tasks, notably the automated completion of the graph, which pose several challenges due to the open and distributed environment of the WWW infrastructure. However, KG mining could take advantage of some useful meta-information about the data to be analyzed, for instance, the schema of the KG when available. In this paper, we resort to the notion of a metaquery, proposed in the 90s as a template for patterns one is interested to discover in a relational database. We propose to extend this notion to the novel context of the Web of Data, in particular to the case of KG mining. A distinguishing feature of metaquerying problems is the use of a second-order logic language. In this paper, we present a metaquery language based on second-order Description Logics but implementable with standard technologies underlying the Web of Data, and briefly describe mechanisms for answering such metaqueries in the context of interest.

Keywords: Metaquerying · Knowledge Graphs · Rule Mining.

1 Background and Motivation

The current vision of the World Wide Web (WWW) is that of a *Web of Data* (WoD), which highlights the central role of data in the WWW. More precisely, the WoD builds upon the WWW infrastructure to represent and interrelate data (aka *Linked Data*), with the aim of transforming the Web from a distributed file system into a distributed database system. The foundational standards of the WoD include the Uniform Resource Identifier (URI) and the Resource Description Framework (RDF)[1]. The former is used to identify resources whereas the latter is used to relate resources. In particular, RDF can be considered as a *data model* according to which data is represented in the form of triples $\langle subject\ predicate\ object \rangle$. Huge collections of these triples can be then organized into directed, labeled graphs known as *knowledge graphs* (KGs). A

[1] https://www.w3.org/RDF/

CEUR-WS.org/Vol-2206/paper8.pdf

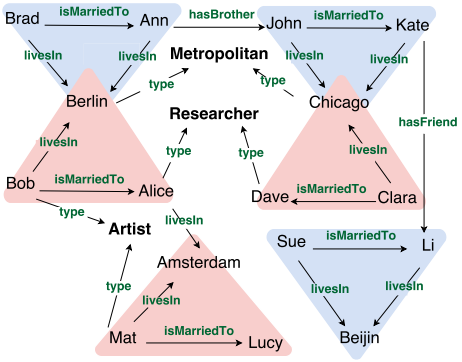

Fig. 1. Fragment of a knowledge graph (taken from [15]).

typical case of a large KG is *DBPedia*,[2] which, essentially, makes the content of *Wikipedia*[3] available in RDF and incorporates links to other datasets on the Web, *e.g.*, to *Geonames*[4]. An interesting point for the ILP community is that RDF triples can be straightforwardly represented by means of unary and binary first-order logic (FOL) predicates. More precisely, the unary predicates are the objects of the RDF *type* predicate, while the binary ones correspond to all other RDF predicates, *e.g.*, $\langle alice\ type\ researcher \rangle$ and $\langle bob\ isMarriedTo\ alice \rangle$ from the KG in Fig. 1 refer to $researcher(alice)$ and $isMarriedTo(bob, alice)$ respectively. KGs can be accessed by posing queries with the RDF query language SPARQL[5]. Several entailment regimes are available for query answering in SPARQL which are based on the existing link between RDF and the family of Description Logics (DLs) [1].

A distinguishing feature of KGs is their inherent *incompleteness*. Indeed, they are constructed by automatically extracting available information from existing Web information sources (see, *e.g.*, the above mentioned case of *DBPedia*). The curation of KGs is cumbersome due to their huge size. In particular, a major activity aims at the *completion* of the KG in hand, in order to address the issue of incompleteness. For instance, in *link prediction*, relational data mining algorithms can be exploited to automatically build rules able to make predictions on missing links. For example, the following rule

$$isMarriedTo(X, Y), livesIn(X, Z) \Rightarrow livesIn(Y, Z) \tag{1}$$

can be mined from the KG in Fig. 1 and applied to derive new facts such as $livesIn(alice, berlin)$, $livesIn(dave, chicago)$ and $livesIn(lucy, amsterdam)$ to be used for completing the graph. However, the extension of relational data

[2] `http://wiki.dbpedia.org/`
[3] `https://www.wikipedia.org/`
[4] `http://www.geonames.org/`
[5] `https://www.w3.org/TR/sparql11-overview/`

mining to the WoD context is not straightforward. Indeed, being intrinsically incomplete, KGs are naturally treated under the Open World Assumption (OWA) as opposed to databases for which the Closed World Assumption (CWA) holds. Nevertheless, KG mining algorithms could take advantage of some useful meta-information about the KG in hand, *e.g.*, domains, ranges and confidence values of relations inside the KG (*i.e.*, its schema). In this paper we resort to the notion of a *metaquery* which was proposed in the 90s as a template that describes the type of pattern one is interested to discover in a relational database [14,2,3]. A common feature to metaquerying problems is the use of a second-order logic language. For KG mining we devise a metaquery language based on second-order DLs. A first step towards this direction of research was taken in [11]. In the present paper we provide a more detailed description of the language (which was only sketched in [11]) and a preliminary analysis of the necessary steps for answering metaqueries in the proposed language.

The rest of the paper is structured as follows. Section 2 presents syntax and semantics of the metaquery language. Section 3 briefly describes mechanisms for answering metaqueries. Section 4 concludes the paper with final remarks and directions of future work.

2 A Metaquery Language for the Web of Data

In our proposal for the WoD context, a metaquery is a second-order DL conjunctive query. In the following we gently introduce the reader to this notion.

Syntax Let $\mathcal{L}$ be a DL language with syntax $(\mathsf{N}_\mathcal{C}, \mathsf{N}_\mathcal{R}, \mathsf{N}_\mathcal{O})$ where $\mathsf{N}_\mathcal{C}$, $\mathsf{N}_\mathcal{R}$, and $\mathsf{N}_\mathcal{O}$ are the alphabet of *concept*, *role*, and *individual* names, respectively. Note that concepts and roles in the DL terminology correspond to classes and properties in RDF.

First, we consider the extension of $\mathcal{L}$ so that we can enable the formulation of conjunctive queries (which go beyond the standard way of querying a DL knowledge base). For this purpose, let $\mathsf{V}_\mathcal{O}$ be a countably infinite set of *individual variables* disjoint from $\mathsf{N}_\mathcal{C}$, $\mathsf{N}_\mathcal{R}$, and $\mathsf{N}_\mathcal{O}$. A *term* t is an element from $\mathsf{V}_\mathcal{O} \cup \mathsf{N}_\mathcal{O}$. Let c be a concept, r a role, and t, t' terms.[6] An *atom* is an expression which can take three different forms: $c(t)$, $r(t, t')$, or $t \approx t'$. We refer to these three kinds of atoms as *concept atoms*, *role atoms*, and *equality atoms* respectively. A *conjunctive query* (CQ) of arity n is an expression of the form

$$q(X_1, \ldots, X_n) \leftarrow a_1, \ldots, a_m \tag{2}$$

where q, called the query predicate, does not belong to $\mathsf{N}_\mathcal{C} \cup \mathsf{N}_\mathcal{R} \cup \mathsf{N}_\mathcal{O} \cup \mathsf{V}_\mathcal{O}$, every X_i belongs to $\mathsf{V}_\mathcal{O}$, every a_j is a (possibly non-ground) atom, and all variables X_i occur in some a_j. The variables X_i are called the *free* variables (aka *distinguished* variables) of the query, whereas the other variables appearing in $a_1, \ldots, a_m$ are

[6] In DLs concept and role names are usually capitalized. However, for the sake of clarity, here we shall use capital letters only for variables.

called *existential* variables. A CQ is called Boolean if it has no free variable. An example of CQ is the following

$$q(Y, Z) \leftarrow isMarriedTo(X, Y), livesIn(X, Z), livesIn(Y, Z) \tag{3}$$

where Y and Z are the free variables, X is the existential one, and $isMarriedTo(X, Y)$, $livesIn(X, Z)$, $livesIn(Y, Z)$ are role atoms.

Since we are interested in second-order CQs, we need to introduce two further sets of variables (of second-order this time): $\mathsf{V}_{\mathcal{C}}$ of so-called *concept variables*, *i.e.* variables that can be quantified over $\mathsf{N}_{\mathcal{C}}$, and $\mathsf{V}_{\mathcal{R}}$ of so-called *role variables*, *i.e.* variables that can be quantified over $\mathsf{N}_{\mathcal{R}}$. Let then $\mathcal{MQ}(\mathcal{L})$ be the second-order DL language obtained by extending $\mathcal{L}$ with $\mathsf{V}_{\mathcal{C}}$ and $\mathsf{V}_{\mathcal{R}}$. For the purpose of this work, we can restrict $\mathcal{MQ}(\mathcal{L})$ to particular (second-order) CQs, *e.g.*, involving only role variables and individual variables. An example of one such metaquery is the following

$$MQ_1 : \ mq(Q, Y, Z) \leftarrow P(X, Y), Q(X, Z) \tag{4}$$

which looks for the properties (Q) holding for the individuals Y. Note that $P, Q \in \mathsf{V}_{\mathcal{R}}$ whereas $X, Y, Z \in \mathsf{V}_{\mathcal{O}}$ Metaqueries are the starting point for the definition of so-called *metaquery extensions*, *i.e.*, implications of the form

$$MQ_1 \rightarrow MQ_2 \tag{5}$$

which are actually a compact representation of two metaqueries, MQ_1 and MQ_2, where MQ_2 is longer than - we say *extends* - MQ_1. A shorter notation for (5) is the following which stresses how MQ_2 extends MQ_1

$$MQ_1 \Rightarrow (MQ_2 \setminus MQ_1) \tag{6}$$

The left-hand side and the right-hand side of (6) are called the *body* and the *head* of the metaquery extension, respectively. Note that in the case of query extensions, the head does not correspond to the conclusion (as with clauses). Following the standard terminology, one should rather bear in mind the unshortened notation, and call MQ_2 the *conclusion* of the metaquery extension. For instance, let us consider the following metaquery

$$MQ_2 : \ mq(Q, Y, Z) \leftarrow P(X, Y), Q(X, Z), Q(Y, Z) \tag{7}$$

which looks for the properties (Q) holding for the individuals Y and shared with the individuals X to which Y is related by some P. From (4) and (7) we can build a metaquery extension as shown below

$$P(X, Y), Q(X, Z) \Rightarrow Q(Y, Z) \tag{8}$$

Metaquery extensions serve as a template for rules we are interested in when applying rule mining algorithms to a given KG.

Semantics As for the semantics of $\mathcal{MQ}(\mathcal{L})$, we plan to follow the Henkin style [9] for the following reasons. As opposed to the Standard Semantics, in the *Henkin semantics* the expressive power of the language actually remains first-order. This is a desirable feature because it paves the way for the use of first-order solvers in spite of the second-order syntax. Also, this is a shared feature with RDF(S). Last, but not least, it makes possible an implementation with SPARQL.

A few remarks are necessary here about the use of the symbol $\Rightarrow$ in (1) and (8). Differently from $\leftarrow$, it does not represent the logical implication. However, as discussed in Sect. 3, it can be treated as such in contexts like the aforementioned link prediction problem, provided that an appropriate choice of rule evaluation measures is done.

3 Answering Metaqueries in the Web of Data

As said in the previous section, metaquery extensions are nothing but a compact representation of two metaqueries. Therefore in this section we limit our discussion to metaqueries.

The process of answering a metaquery can be divided into two stages. In the first stage, which we call the *instantiation stage*, we look for sets of concepts and roles that match the pattern determined by the metaquery. In the second stage, which we call the *filtration stage*, we filter out all the rules that match the pattern of the metaquery but do not satisfy some predefined evaluation criteria.

Instantiation Instantiating a metaquery is similar to solving a Constraint Satisfaction Problem (CSP) where one is interested in finding all solutions of the CSP problem. The instantiation step is practically possible when the schema of the KG in hand is available, which is not the case for every KG. Indeed the schema provides the signature of the relations occuring in the KG, thus making this step an informed search rather than a blind search. For instance, with reference to the KG depicted in Fig. 1, (1) is an instantiation of (8) obtained by substituting the role variables P and Q with the role names $isMarriedTo$ and $livesIn$, respectively.

Filtration At this stage the rules like (1) obtained by instantiating the given metaquery extension are evaluated according to some interestingness measures. For instance, we could aim at filtering out rules with low support and confidence values. In this case it is reasonable to compute confidence only for rules with sufficient support.

Following [7], the *(absolute) support* of a CQ Q in a KG $\mathcal{G}$ is the number of distinct tuples in the answer of Q on $\mathcal{G}$. The *relative support* of $h(X, Y)$ over $\mathcal{G}$ is defined as follows:

$$supp(h(X,Y),\mathcal{G}) = \frac{\#(X,Y) : h(X,Y) \in \mathcal{G}}{(\#X : \exists Y\ h(X,Y) \in \mathcal{G}) * (\#Y : \exists X\ h(X,Y) \in \mathcal{G})} \quad (9)$$

The confidence of a rule w.r.t. $\mathcal{G}$ can be defined starting from (9). However, in order to estimate the actual implication of the rule at hand, we could exploit the rule evaluation measure called *conviction* [4]. This choice is thus particularly attractive for the KG completion task.

4 Conclusions and Future Work

In this paper we have briefly presented a new approach to mining the Web of Data. The approach adapts the notion of metaquery introduced by [14] for relational data mining to the novel context of KG mining. The idea of considering extensions of metaqueries is inspired by [7] in their seminal work on ILP for association rule mining. However, differently from [14,7], our proposed metaquery language is based on second-order DLs but can be implemented with standard technologies of the Web of Data. The importance of metamodeling (of which metaquerying is a special case) in several applications has been recently recognized in the DL community. In particular, De Giacomo *et al.* [6] augment a DL with variables that may be interpreted - in a Henkin semantics - as individuals, concepts, and roles at the same time, obtaining a new logic $Hi(\mathcal{DL})$. Colucci *et al.* [5] introduce second-order features in DLs under Henkin semantics for modeling several forms of non-standard reasoning. Lisi [10,12] extends [5] to some variants of concept learning, thus being the first to propose higher-order DLs as a means for metamodeling in data mining.

In the KG community approaches for link prediction are divided into statistics-based (see [13] for an overview), and logic-based (e.g., [8,15]), which are the closest to our work. The latter basically extend and adapt previous work in ILP on relational association rule mining. However, they differ in the expressiveness of the mined rules. AMIE+ [8] can mine only Horn rules, whereas the methodology described in [15] can deal with the case of nonmonotonic rules.

In the future, several aspects of the proposed approach should be clarified before an implementation. First, we need to better define the semantics for the proposed metaquery language, also concerning the link with SPARQL. Second, we need to design algorithms for the instantiation stage and choose the most appropriate evaluation measures for the intended application.

References

1. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (eds.): The Description Logic Handbook: Theory, Implementation and Applications (2nd ed.). Cambridge University Press (2007)
2. Ben-Eliyahu-Zohary, R., Gudes, E.: Towards efficient metaquerying. In: Dean, T. (ed.) Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, IJCAI 99, Stockholm, Sweden, July 31 - August 6, 1999. 2 Volumes, 1450 pages. pp. 800–805. Morgan Kaufmann (1999)
3. Ben-Eliyahu-Zohary, R., Gudes, E., Ianni, G.: Metaqueries: Semantics, complexity, and efficient algorithms. Artificial Intelligence **149**(1), 61–87 (2003)

4. Brin, S., Motwani, R., Ullman, J.D., Tsur, S.: Dynamic itemset counting and implication rules for market basket data. In: Peckham, J. (ed.) SIGMOD 1997, Proceedings ACM SIGMOD International Conference on Management of Data, May 13-15, 1997, Tucson, Arizona, USA. pp. 255–264. ACM Press (1997). https://doi.org/10.1145/253260.253325
5. Colucci, S., Di Noia, T., Di Sciascio, E., Donini, F.M., Ragone, A.: A unified framework for non-standard reasoning services in description logics. In: Coelho, H., Studer, R., Wooldridge, M. (eds.) ECAI 2010 - 19th European Conference on Artificial Intelligence, Lisbon, Portugal, August 16-20, 2010, Proceedings. Frontiers in Artificial Intelligence and Applications, vol. 215, pp. 479–484. IOS Press (2010)
6. De Giacomo, G., Lenzerini, M., Rosati, R.: Higher-order description logics for domain metamodeling. In: Burgard, W., Roth, D. (eds.) Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011, San Francisco, California, USA, August 7-11, 2011 (2011)
7. Dehaspe, L., De Raedt, L.: Mining Association Rules in Multiple Relations. In: Lavrač, N., Džeroski, S. (eds.) Inductive Logic Programming. Lecture Notes in Artificial Intelligence, vol. 1297, pp. 125–132. Springer (1997)
8. Galárraga, L., Teflioudi, C., Hose, K., Suchanek, F.M.: Fast rule mining in ontological knowledge bases with AMIE+. VLDB Journal **24**(6), 707–730 (2015), https://doi.org/10.1007/s00778-015-0394-1
9. Henkin, L.: Completeness in the theory of types. Journal of Symbolic Logic **15**(2), 81–91 (1950)
10. Lisi, F.A.: A declarative modeling language for concept learning in description logics. In: Riguzzi, F., Zelezny, F. (eds.) Inductive Logic Programming, 22nd International Conference, ILP 2012, Dubrovnik, Croatia, September 17-19, 2012, Revised Selected Papers. Lecture Notes in Computer Science, vol. 7842. Springer Berlin Heidelberg (2013)
11. Lisi, F.A.: Towards a metaquery language for mining the web of data. In: Calì, A., Wood, P.T., Martin, N.J., Poulovassilis, A. (eds.) Data Analytics - 31st British International Conference on Databases, BICOD 2017, London, UK, July 10-12, 2017, Proceedings. Lecture Notes in Computer Science, vol. 10365, pp. 90–93. Springer (2017). https://doi.org/10.1007/978-3-319-60795-5_8
12. Lisi, F.A.: A model+solver approach to concept learning. In: Adorni, G., Cagnoni, S., Gori, M., Maratea, M. (eds.) AI*IA 2016: Advances in Artificial Intelligence - XVth International Conference of the Italian Association for Artificial Intelligence, Genova, Italy, November 29 - December 1, 2016, Proceedings. Lecture Notes in Computer Science, vol. 10037, pp. 266–279. Springer (2016), https://doi.org/10.1007/978-3-319-49130-1_20
13. Nickel, M., Murphy, K., Tresp, V., Gabrilovich, E.: A review of relational machine learning for knowledge graphs. Proceedings of the IEEE **104**(1), 11–33 (2016). https://doi.org/10.1109/JPROC.2015.2483592
14. Shen, W., Ong, K., Mitbander, B.G., Zaniolo, C.: Metaqueries for data mining. In: Advances in Knowledge Discovery and Data Mining, pp. 375–398. AAAI/MIT Press (1996)
15. Tran, H.D., Stepanova, D., Gad-Elrab, M.H., Lisi, F.A., Weikum, G.: Towards nonmonotonic relational learning from knowledge graphs. In: Cussens, J., Russo, A. (eds.) Inductive Logic Programming - 26th International Conference, ILP 2016, London, UK, September 4-6, 2016, Revised Selected Papers. Lecture Notes in Computer Science, vol. 10326, pp. 94–107. Springer (2017). https://doi.org/10.1007/978-3-319-63342-8_8