Non-Redundant Link Keys in RDF Data: Preliminary Steps

Nacira Abbas[1], Alexandre Bazin[1], Jérôme David[2], and Amedeo Napoli[1]

[1] Université de Lorraine, CNRS, Inria, Loria, F-54000 Nancy, France
Nacira.Abbas@inria.fr, Alexandre.Bazin@loria.fr, Amedeo.Napoli@loria.fr
[2] Université Grenoble Alpes, Inria, CNRS, Grenoble INP, LIG, F-38000 Grenoble, France Jerome.David@inria.fr

Abstract. A link key between two RDF datasets D_1 and D_2 is a set of pairs of properties allowing to identify pairs of individuals, say x_1 in D_1 and x_2 in D_2, which can be materialized as a x_1 `owl:sameAs` x_2 identity link. There exist several ways to mine such link keys but no one takes into account the fact that `owl:sameAs` is an equivalence relation, which leads to the discovery of non-redundant link keys. Accordingly, in this paper, we present the link key discovery based on Pattern Structures (PS). PS output a pattern concept lattice where every concept has an extent representing a set of pairs of individuals and an intent representing the related link key candidate. Then, we discuss the equivalence relation induced by a link key and we introduce the notion of non-redundant link key candidate.

Keywords: Linked Data · RDF · Link Key · Formal Concept Analysis · Pattern Structures.

1 Introduction

In this paper, we are interested in data interlinking which goal is to discover identity links across two RDF datasets over the web of data [5,8]. The same real world entity can be represented in two RDF datasets by different subjects in RDF triples `(subject,property,value)` (instead of "object" usually used in RDF data we will use "value"). It is important to be able to detect such identities, for example using rules expressing sufficient conditions for two subjects to be identical. A link key takes the form of two sets of pairs of properties associated with a pair of classes. The pairs of properties express sufficient conditions for two subjects, from the associated pair of classes, to be the same. An example of a link key is $(\{(\texttt{designation}, \texttt{title})\}, \{(\texttt{designation}, \texttt{title}), (\texttt{creator}, \texttt{author})\}, (\texttt{Book}, \texttt{Novel}))$ which states that whenever an instance a of the class `Book` has the same (non empty) values for the property `designation` as an instance b of the class `Novel` for the property `title` (universal quantification), and that a and b share at least one value for the properties `creator` and `author` (existential quantification), then a and b denote the same entity, i.e., an `owl:sameAs` relation can be established between a and b.

A link key can be understood as a "closed set" in the sense that it is maximal w.r.t. the set of pairs of individuals to which it applies. This was firstly discussed in [2] and then extended in [3]. Hence the question of relying on Formal Concept Analysis (FCA [7]) to discover link keys is straightforward as FCA is based on a closure operator. Then, given two RDF datasets, FCA is applied in [3] to a binary table where rows correspond to pairs of individuals and columns to pairs of properties. The intent of a concept is a link key candidate which should be validated thanks to suitable quality measures. The extent of the concept is the set of identity links between individuals. Furthermore, a generalization of the former approach proposed in [1] is based on pattern structures [6] and takes into account different pairs of classes at the same time in the discovery of link keys.

Link key candidates over two RDF datasets have to generate different and maximal link sets. However it appears that two different link key candidates may generate the same link set. This means that there exists some redundancy between the two link key candidates, that they should be considered as equivalent and merged. This can be achieved by looking at `owl:sameAs` which is an equivalence relation stating that two individual should be identified. The `owl:sameAs` relation generates partitions among pairs of individuals that can be used to detect redundant link key candidates and thus reduce their number, i.e., two candidates relying on the same partition are declared as redundant and thus merged.

In this paper, we present the discovery of link key candidates within the framework of pattern structure. Then, we introduce the notion of non-redundant link key candidate based on the equivalence relation induced by a link key candidate. Finally, we discuss how these candidates can be merged to reduce the search space of link keys.

2 Basics and Notations

2.1 RDF data

In this work, we deal with RDF datasets which are defined as follows:

Definition 1 (RDF dataset).
Let U be a set of IRIs (Internationalized Resource Identifier), B a set of blank nodes and L a set of literals. An RDF dataset is a set of triples $(s, p, v) \in (U \cup B) \times U \times (U \cup B \cup L)$.

Given a dataset D, we denote by:

- $I(D) = \{s \mid \exists p, v \ (s, p, v) \in D\}$ the set of individual identifiers,
- $P(D) = \{p \mid \exists s, v \ (s, p, v) \in D\}$ the set of property identifiers,
- $C(D) = \{c \mid \exists s \ (s, \texttt{rdf:type}, c) \in D\}$ the set of class identifiers. A triple $(s, \texttt{rdf:type}, c)$ means that the subject s is an instance of the class c.
- $I(c) = \{s \mid (s, \texttt{rdf:type}, c) \in D\}$ the set of instances of $c \in C(D)$,
- $p(s) = \{v \mid (s, p, v) \in D\}$ is the set of values (or "RDF objects") related to s through p.

An identity link is an RDF triple $(a, \texttt{owl:sameAs}, b)$ stating that the IRIs a and b refer to the same real-world entity. Fig. 1 represents two RDF datasets D_1 and D_2, where $P(D_1) = \{p_1, p_2, p_3, p_4\}$ and $P(D_2) = \{q_1, q_2, q_3, q_4\}$. Then $C(D_1) = \{c_1\}$ and $C(D_2) = \{c_2\}$ with $I(c_1) = \{a_1, a_2, a_3, a_4, a_5\}$ and $I(c_2) = \{b_1, b_2, b_3, b_4, b_5\}$. For example, the set of values of b_3 for the property q_2 is $q_2(b_3) = \{v_8, v_9\}$.

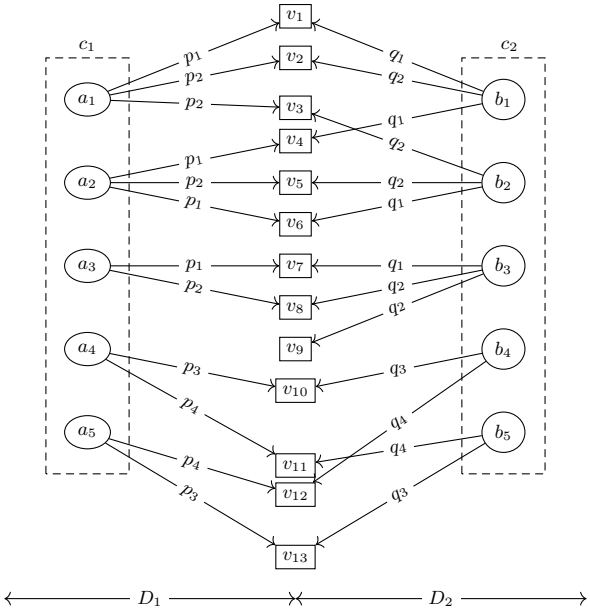

Fig. 1. Example of two RDF datasets. On the left-hand side, the dataset D_1 is populated with instances of the class c_1, and on the right-hand side the dataset D_2 is populated with instances of the class c_2.

2.2 Link Keys

Link keys are logical constructors allowing to deduce identity links (`owl:sameAs`). In this paper we will call *link key expression* the syntactic formulation of a link key, when we do not know whether the expression is a valid link key. For example, the link key expression $(\{\}, \{(\texttt{nbrOfPages}, \texttt{nbrOfBeds})\}, (\texttt{Book}, \texttt{Hospital}))$ will not satisfy the link key semantics since an instance of class `Book` and an instance of class `Hospital` cannot represent the same entity since `Book` and `Hospital` are disjoint classes.

Definition 2 (Link key expression, link key candidate). *Let D_1 and D_2 be two RDF datasets, $k = (Eq, In, (c_1, c_2))$ is a* link key expression *(over D_1 and D_2) iff $In \subseteq P(D_1) \times P(D_2)$, $Eq \subseteq In$, $c_1 \in C(D_1)$ and $c_2 \in C(D_2)$.*

The set of links $L(k)$ (directly) generated by k is the set of pairs of instances $(a, b) \in I(c_1) \times I(c_2)$ satisfying:

(i) for all $(p, q) \in Eq$, $p(a) = q(b)$ and $p(a) \neq \emptyset$,
(ii) for all $(p, q) \in In \setminus Eq$, $p(a) \cap q(b) \neq \emptyset$.

A link key expression $k_1 = (Eq_1, In_1, (c_1, c_2))$ is a link key candidate *if:*

(iii) $L(k_1) \neq \emptyset$,
(iv) k_1 is maximal i.e. there does not exist another link key expression $k_2 = (Eq_2, In_2, (c_1, c_2))$ such that $Eq_1 \subset Eq_2$, $In_1 \subset In_2$, and $L(k_1) = L(k_2)$.

The number of link key expressions may be exponential w.r.t. the number of properties. Then link key discovery algorithms only consider link key candidates which are link key expressions generating at least one link and that are maximal w.r.t. the set of links they generate.

3 Link Key Discovery

Here after we assume that all link key expressions are defined on the same pair of datasets D_1 and D_2 w.r.t. one pair of classes, yielding link key expressions of the form $k = (Eq, In, (c_1, c_2))$. In the following, we show how link keys may be discovered within the formalism of pattern structures (see details in [1]) and then we discuss the notion of non-redundant link keys.

Example 1. Let us consider the pattern structure $(G, (E, \sqcap), \delta)$ displayed in Table 1. Here we skip the details for building this table and the related PS lattice which can be found in [1].

The rows termed "PS objects" correspond to the set of objects G of the pattern structure and include pairs of related instances. The set of descriptions $(E, \sqcap)$ includes all possible pairs of properties preceded either by $\forall$ or $\exists$. The mapping δ relates a pair of instances $(a, b) \in I(c_1) \times I(c_2)$ to a description as follows: (i) $\delta(a, b)$ includes $\forall(p, q)$ whenever $p(a) = q(b)$ and $p(a) \neq \emptyset$, (ii) $\delta(a, b)$ includes $\exists(p, q)$ whenever $p(a) \cap q(b) \neq \emptyset$. Then the descriptions correspond to link key expressions (Eq, In) w.r.t. the pairs of classes (c_1, c_2). It should be noticed that it is possible to simultaneously work with several pairs of classes as explained in [1].

We have that $\delta(a_1, b_1) = \{\exists(p_1, q_1), \exists(p_2, q_2)\}$ because $p_1(a_1) \cap q_1(b_1) \neq \emptyset$ and $p_2(a_1) \cap q_2(b_1) \neq \emptyset$ while $\delta(a_2, b_1) = \{\exists(p_1, q_1)\}$ because $p_1(a_2) \cap q_1(b_1) \neq \emptyset$. Then $\delta(a_1, b_1) \sqcap \delta(a_2, b_1) = \{\exists(p_1, q_1)\}$ and thus $\delta(a_2, b_1) \sqsubseteq \delta(a_1, b_1)$. This can be read in the pattern concept lattice where the pattern concept pc_5 is subsumed by the pattern concept pc_4, i.e., the extent of pc_5 $\{(a_1, b_1), (a_2, b_2), (a_3, b_3)\}$ is included in the extent of pc_4 $\{(a_1, b_1), (a_2, b_1), (a_2, b_2), (a_3, b_3)\}$, while the intent $\{\exists(p_1, q_1)\}$ of pc_4 is included in the intent of pc_5, $\{\exists(p_1, q_1), \exists(p_2, q_2)\}$.

The set of all pattern concepts is organized within the pattern concept lattice LKPS-lattice displayed in Fig. 2. Moreover, all potential link key candidates are lying in the intents of the pattern concepts in the lattice. The corresponding set of link key candidates is denoted by LKC.

PS objects (g)	descriptions ($\delta(g)$)
(a_1, b_1)	$\{\exists(p_1, q_1), \exists(p_2, q_2)\}$
(a_1, b_2)	$\{\exists(p_2, q_2)\}$
(a_2, b_1)	$\{\exists(p_1, q_1)\}$
(a_2, b_2)	$\{\exists(p_1, q_1), \exists(p_2, q_2)\}$
(a_3, b_3)	$\{\forall(p_1, q_1), \exists(p_1, q_1), \exists(p_2, q_2)\}$
(a_4, b_4)	$\{\forall(p_3, q_3), \exists(p_3, q_3)\}$
(a_4, b_5)	$\{\forall(p_4, q_4), \exists(p_4, q_4)\}$
(a_5, b_4)	$\{\forall(p_4, q_4), \exists(p_4, q_4)\}$
(a_5, b_5)	$\{\forall(p_3, q_3), \exists(p_3, q_3)\}$

Table 1. The pattern structure related to link key discovery over c_1 and c_2 introduced in Fig. 1.

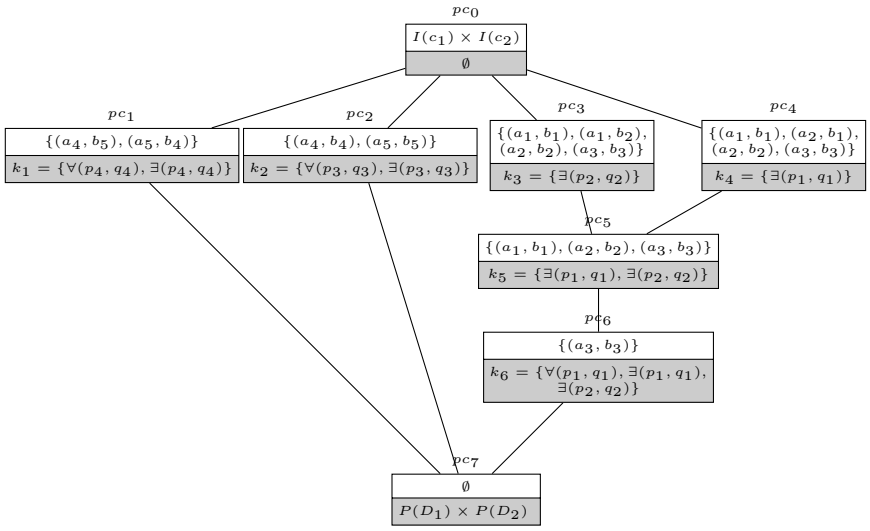

Fig. 2. The pattern concept intents in the pattern concept lattice LKPS-lattice include the complete set of link key candidates.

Let us consider the so-called LKPS-lattice and $pc = (L(k), k)$ a pattern concept, where the extent $L(k)$ corresponds to the set of links generated by k, and the intent k corresponds to a link key candidate. Let I denotes the set of instances $I = I(c_1) \cup I(c_2)$ and the binary relation $\simeq_k \subseteq I \times I$ such as $(a, b) \in L(k) \rightarrow a \simeq_k b$. The interpretation of $a \simeq_k b$ is: "k states that there exists a `owl:sameAs` relation between a and b". Actually $\simeq_k$ is an equivalence relation based on the fact that `owl:sameAs` itself is an equivalence relation. We say that k induces the equivalence relation $\simeq_k$ over I. Moreover $\simeq_k$ forms a partition over I where each element of this partition is an equivalence class. In fact the $\simeq_k$ equivalence relation will help us to build more concise set of link key candidates since it allows to identify *non-redundant link key candidates* termed nr-LKC. A link key candidate k_1 is a nr-LKC in LKC if there is no other candidate k_2 in LKC such that $\simeq_{k_1}$ and $\simeq_{k_2}$ form the same partition. Otherwise, k_1 is redundant.

In Fig. 2, it can be observed that $\simeq_{k_3}$ and $\simeq_{k_4}$ form the same partition, namely $\{(a_1, b_1, a_2, b_2), (a_3, b_3)\}$ (it should be noticed that singletons are omitted for the sake of readability). Then the link key candidates k_3 and k_4 are redundant. By contrast, k_1 is a nr-LKC because there is no other candidate k in LKC such that $\simeq_{k_1}$ and $\simeq_k$ form the same partition.

Let us briefly explain how $\simeq_{k_3}$ and $\simeq_{k_4}$ are inducing the same partition, namely $\{(a_1, b_1, a_2, b_2), (a_3, b_3)\}$. The extent of k_3 in LKPS-lattice is given by $\{(a_1, b_1), (a_1, b_2), (a_2, b_2), (a_3, b_3)\}$. By transitivity and symmetry of `owl:sameAs`, we have that (a_1, b_2) and (b_2, a_2) yields (a_1, a_2), then (a_2, a_1) and (a_1, b_1) yields (a_2, b_1), and finally (b_1, a_2) and (a_2, b_2) yields (b_1, b_2) and the complete graph between (a_1, a_2, b_1, b_2). The same thing applies when we consider k_4 instead of k_3. This intuitively shows how $\simeq_{k_3}$ and $\simeq_{k_4}$ are inducing the same partition.

One main straightforward application of identifying nr-LKC is the ability to reduce the search space of link keys since the set of nr-LKC is included in LKC. Indeed, this can be seen as a refinement where redundant link key candidates inducing the same partition are merged. For example, since $\simeq_{k_3}$ and $\simeq_{k_4}$ form the same partition, then, k_3 and k_4 can be merged into a nr-LKC $k_{34} = \{k_3, k_4\}$. Among the perspectives is to consolidate the theory and practice of link key discovery based on partition pattern structures initially introduced for mining functional dependencies in [4].

References

1. Abbas, N., David, J., Napoli, A.: Discovery of link keys in RDF data based on pattern structures: Preliminary steps. In: Proceedings of ICFCA. CEUR Workshop Proceedings, vol. 2668, pp. 235–246. CEUR-WS.org (2020)
2. Atencia, M., David, J., Euzenat, J.: Data interlinking through robust linkkey extraction. In: Proceedings of ECAI. pp. 15–20 (2014)
3. Atencia, M., David, J., Euzenat, J., Napoli, A., Vizzini, J.: Link key candidate extraction with relational concept analysis. Discrete applied mathematics **273**, 2–20 (2020)
4. Baixeries, J., Kaytoue, M., Napoli, A.: Characterizing functional dependencies in formal concept analysis with pattern structures. Annals of Mathematics and Artificial Intelligence **72**, 129–149 (2014)
5. Ferrara, A., Nikolov, A., Scharffe, F.: Data Linking for the Semantic Web. International Journal of Semantic Web and Information Systems **7**(3), 46–76 (2011)
6. Ganter, B., Kuznetsov, S.O.: Pattern Structures and Their Projections. In: Proceedings of the International Conference on Conceptual Structures (ICCS). pp. 129–142. LNCS 2120, Springer (2001)
7. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer (1999)
8. Nentwig, M., Hartung, M., Ngonga Ngomo, A.C., Rahm, E.: A survey of current link discovery frameworks. Semantic Web **8**(3), 419–436 (2017). https://doi.org/10.3233/SW-150210