A guided walk into link key candidate extraction with relational concept analysis
Manuel Atencia, Jérôme David, Jérôme Euzenat, Amedeo Napoli, Jérémy Vizzini

To cite this version:

HAL Id: hal-02984963
https://hal.archives-ouvertes.fr/hal-02984963
Submitted on 1 Nov 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers. L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
A guided walk into
link key candidate extraction
with relational concept analysis

Manuel Atencia, Jérôme David, Jérôme Euzenat, Amedeo Napoli, and Jérémy Vizzini

Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LIG, F-38000 Grenoble, France
Firstname.Lastname@inria.fr

Abstract. Data interlinking is an important task for linked data interoperability. One of the possible techniques for finding links is the use of link keys which generalise relational keys to pairs of RDF models. We show how link key candidates may be directly extracted from RDF data sets by encoding the extraction problem in relational concept analysis. This method deals with non functional properties and circular dependent link key expressions. As such, it generalises those presented for non dependent link keys and link keys over the relational model. The proposed method is able to return link key candidates involving several classes at once.

In this extended abstract, we describe the content of [2] by avoiding technical descriptions related to the domain. Interested readers are referred to the original paper for more details. To facilitate the compared reading, we explicitly refer to the original definitions and properties.

1 Interlinking RDF data sets

Interlinking RDF data sets is an important operation for linked (open) data on which research is currently very active.

1.1 RDF data sets and linked data

Linked data aims at publishing data expressed in RDF (Resource Description Framework) at the scale of the worldwide web [6,12]. In this paper we consider RDF data sets as sets of triples. We simply identify properties as the IRIs in predicate position in one of these triples and classes as the IRIs found in the codomain of the rdf:type predicates. This is the only schema information that is used. These data sets are not required to share vocabulary elements beside the RDF (for rdf:type) and OWL (for owl:sameAs) vocabularies.

Heterogeneous data sets interoperate through links which identify individuals across them. An important added value of linked data arises from the links
that identify the same entity in different data sets. They allow one to develop
innovative applications exploiting data cross-references and making inferences
between data sets.

1.2 Data interlinking

Data interlinking is the problem of finding IRIs described in RDF over different
data sets that refer to the same resource. It is thus a critical task for widening and
enhancing linked data. This is a knowledge discovery task as it infers knowledge
—the condition for identity of resources—from large volumes of data.

Different approaches and methods have been proposed to address the prob-
lem of automatic data interlinking [10,18]. Two main trends are used to tackle
this problem: numerical methods and logical methods. Numerical methods usually
compute a similarity between resources based on their property values to
establish links between those that are very similar [6,19,25]. Logical methods
use an axiomatic characterisation of what makes two resources the same so as
to find the owl:sameAs links between IRIs of different data sets [23,3,13].

2 Link key extraction

Our work belongs to the logic-based approach and is more precisely based on
link keys. Thus we first introduce them before considering approaches to extract
them from data.

2.1 Link keys

We introduced the notion of link keys as a way to find pairs of IRI identifying the
same resource [9,1]. Link keys generalise keys in relational algebra in three ways
adapted to RDF: (a) they apply across two data sets instead of a single one,
(b) they take into account multiple values for the same property, and (c) property
values may be other objects. The latter makes link keys eventually dependent
on each others.

A link key expression [2, Def. 4] specifies the pairs of properties to compare
for linking individuals belonging to different classes of the data sets. An example
of a link key expression is:

\[
\{ \langle \text{auteur}, \text{creator} \rangle \} \{ \langle \text{titre}, \text{title} \rangle \} \ linkkey \ (\text{Livre}, \text{Book})
\]

stating that whenever an instance of the class Livre has the same values for the
property auteur as an instance of class Book has for the property creator and they
share at least one value for their property titre and title, then they denote the
same entity.

A link key expression thus applies to a pair of classes ((Livre, Book)). In the
above expression, there are two sets of conditions (each reduces to a single-
ton): the eq-link key condition (\{ \langle \text{auteur}, \text{creator} \rangle \}) and the in-link key condition
(\{ \langle \text{titre}, \text{title} \rangle \}). The abbreviations eq- and in- stand for equality and intersection.
It is easy to understand what link set is generated by such a link key expression given two RDF data sets [2, Def. 6].

Such a link key may depend on another one as, for instance, properties auteur and creator have values in the Écrivain and Writer classes respectively. Identifying their values will then depend on another link key:

\[
\{\langle nom, lastname \rangle\}\{\langle prénom, firstname \rangle\} \text{ linkkey } \langle Écrivain, Writer \rangle
\]

This situation may be rendered even more intricate if Écrivain and Writer were instead identified from the values of their properties ouvrage and hasWritten referring to instances of Livre and Book. We would then face circularly dependent link keys.

2.2 Link key extraction

The goal of link key extraction is, given two data sets, to find which link keys hold.

Key-based data interlinking requires to have pairs of keys related by alignment (or schema mapping). Keys are either given or extracted from the data. Key extraction (and more generally functional dependencies discovery) has been addressed both in databases [24,14] and in the semantic web [3,20]. Systems for data interlinking from keys [20] can only extract a specific (strong) kind of in-link keys, which may not be the best candidates.

In [1], we have proposed to discover directly in-link keys between two classes from two data sets. The proposed algorithm does not require an initial alignment between properties of both data sets and also avoids the generation of keys that are specific to only one data set. It may be decomposed in two distinct steps: (i) identifying link key candidates, i.e. maximal link key expressions that would generate at least one link if used as a link key, followed by (ii) selecting the best link key candidate according to quality measures. To select the best link key candidate, we have introduced two sets of measures.

The purpose of this work is to reformulate Step (i) within the framework of formal concept analysis and to extend it towards dependent link key expressions.

3 Link key candidate extraction and formal concept analysis

The presented paper [2] aims at performing link key extraction through using Formal Concept Analysis (FCA). We thus first briefly introduce it. Then we summarize our previous work relating link key extraction algorithms and FCA. We finally present the full reformulation of link key extraction within FCA.

3.1 Formal Concept Analysis

Formal Concept Analysis (FCA) is a set of techniques, allowing data analysis and classification, based on solid mathematical foundations and efficient algorithms.
[11]. It starts with a binary table, called formal context, relating objects to attributes (or properties) and outputs a set of concepts which are pairs composed of a maximal set of objects sharing a corresponding maximal set of attributes. In addition, concepts are partially ordered within a concept lattice.

Using lattices is commonplace for extracting functional dependencies [16,8,17]. The problem was considered in FCA in a functional setting [11] and further refined in pattern structures, the extension of FCA to complex data [5,7]. However, FCA is defined for functional properties and must be adapted to the other cases, especially with multi-valued and relational attributes.

3.2 Link key candidates are formal concepts

We have previously shown how to encode the link key extraction problem in relational databases into formal concept analysis so that link key candidates correspond to formal concepts [4]. This was a simplification of the method used for extracting functional dependencies in [11] in which the formal context relates pairs of properties to pairs of objects. The relation holds between a pair of objects and a pair of properties if the objects have the same value for this property.

This work in the functional case of relational databases in which eq-conditions and in-conditions are equivalent (for non NULL cells).

This preliminary work showed that the approach was promising, but extensions were needed in several directions, among which:

1. Non-functional properties lead to distinguish eq- and in-conditions. This requires integrating them in the proposed formal contexts.
2. Relational properties lead to the possible existence of link keys depending of others link keys, e.g. individuals of the pair of classes ⟨Livre, Book⟩ will depend on the classes ⟨Écrivain, Writer⟩ as the value of the ⟨auteur, creator⟩ property pairs.
3. Moreover, some classes may depend on themselves, directly or indirectly. For instance, if identifying individuals of the pair of classes ⟨Écrivain, Writer⟩ may, in turn, depend on the pair of classes ⟨Livre, Book⟩ through the pair of properties ⟨ouvrage, hasWritten⟩.

3.3 Reformulation and extensions

A first contribution of [2] is the reformulation of the former encoding and the proof that it indeed finds all and only link key candidates.

We first define the notion of meet, join and subsumption between link key expressions [2, Def. 5] and reformulate link key candidates as closed by join [2, Def. 8]. This covers relevant link key expressions missed by the previous definitions. We prove that reformulated link key candidates are either previous link key candidates or join of link key candidates [2, Prop. 2].

We then define formal contexts for dependent link key candidates [2, Def. 9]. They extend the former contexts by qualifying the pairs of properties as acting either as an in-condition or an eq-condition. The former relates two objects sharing at least one value for these properties; the latter relates them if they
share all their values. We show that, as redefined, *link key candidates are formal concepts* [2, Prop. 3].

4 Dependent link key candidate extraction

So far, we were only concerned with extracting link key candidates between two identified classes independently of other classes. This means that, either this can only take into account literal values, i.e. data properties, or the equality between objects that appear as relation values require *owl:sameAs* statements.

A first enhancement is thus to express dependent link key expressions, e.g. link keys of the class Book that contains the *author* property will depend on link keys of the class Writer. For that purpose, we extend formal contexts to cover object properties such that a pair of object properties be qualified by the link key expression used for judging the equality of instances. For example, this can resort to expressing that the object should share a value in the property *auteur* and *creator* judging this equality on the equality between their property *noinsee* and *ssnumber*. This is integrated in the definition of *formal contexts for dependent link key candidates* [2, Def. 11] by adding for each pair of object properties the keys used for linking them.

This raises a subsequent problem: it is now possible to have the same pair of classes (Livre and Book) using different link keys for judging the equality of the same pair of classes (Personne and Person) depending on the pair of property that is compared, e.g. ⟨auteur, author⟩ and ⟨traducteur, translator⟩. This seems weird. It may be preferable to always compare the instances of the same pair of classes with the same link key expression.

This is the reason why we define *coherent families of link key candidates* [2, Def. 12] which assign at most one single link key candidate to any pair of classes.

5 Circular dependencies and relational concept analysis

Circular dependencies across link keys raise the problem of self-supported links, i.e. EEUU is the same as USA if Nueva York is the same as New-York which relies on EEUU being the same as USA. Since the goal of link keys is to find links between data, self-supported links should not be discarded upfront.

Circular dependencies are commonplace in ontologies, e.g. the class Person depending on itself through the property parent. The use of a relation and its inverse (author-hasWritten) is already a simple case of circular dependency.

Such characteristics require extensions of the genuine FCA framework. We have used relational concept analysis for that purpose.

5.1 Relational Concept Analysis

Relational Concept Analysis [21] has been designed for dealing with relational attributes and circular dependencies in the FCA framework. As for FCA, its
main goal is to find concept descriptions, i.e. descriptions of classes expressed in function of other class descriptions through properties, like in description logics. For that purpose, it deals with a relational context family which is made of a set of classical formal contexts, one per class, and a set of relational contexts, one per object property. The relational contexts is a binary table between objects of two contexts. It simply express which objects are in relation with which other objects through a particular object property. In addition, RCA uses scaling operators to introduce new discriminating properties within the formal contexts.

Its concept extraction process starts by \( (a) \) applying FCA to each formal context. This provides new concept lattices. Then \( (b) \) the scaling operators add new attributes to the formal contexts of classes depending on the extracted concepts for the co-domain of their object properties. The process returns to Step \( (a) \) until no change occurs. This has been proved to converge [22]. As a result, RCA returns one concept lattice for each formal context.

5.2 Extension to circular dependencies

Extracting link keys, i.e. conditions for pairs of objects to be the same, is not exactly like extracting class descriptions, i.e. conditions for an object to belong to a class. However, as we defined a suitable encoding for FCA, it is possible to extend it, even for RCA. In fact, the formal contexts are the same as in the previous section and the relational contexts for dependent link key candidates [2, Def. 13] are the same as in RCA except that they relate pairs of objects through pairs of object properties. A relational context family for dependent link key candidates [2, Def. 15] is made of such contexts, eventually restricted by an alignment.

The novelty lies on the introduction of two specific scaling operators, the link key condition scaling operators [2, Def. 14], which generate the corresponding in- and eq-conditions when necessary, i.e. when this discriminates between generated concepts. Coherent families of link key candidates can be extracted from the resulting lattices and evaluated.

6 Complexity and implementation

It may be strange that what the paper offers is a set of formal context definitions. This is the way a problem is expressed in formal concept analysis. If this is rightly done, the FCA/RCA process provides the solution to the problem, i.e. concept lattices.

We have implemented a full proof-of-concept in Python\(^1\) which supports all encodings described here and returns the concept lattice from data sets expressed in genuine RDF. It is also provided with many toy examples for torturing the model. This implementation is not particularly efficient.

\(^1\) https://moex.inria.fr/software/linkky/
We are currently reimplementing it in Java in our LinkEx tool\textsuperscript{2}. This is not yet fully completed.

The paper discusses the exponential data and ‘schema’ complexity of the proposed algorithm as well as the observation that, in practice, the data sets do not generate that many link keys. LinkEx is able to generate and evaluate 834 link key candidates from 99116 \((349 \times 284)\) possible links and 255 \((15 \times 17)\) pairs of properties within a few seconds on a laptop.

7 Conclusion

This paper provides a full account of using relational concept analysis to extract link key candidates from two RDF data sets containing various classes. All the contexts defined in this paper are extension of others, i.e. each considered problem is expressed in the same way in the context corresponding to it as in the more expressive contexts and all provide exactly the same results. The proposed approach works as well for unary link keys, i.e. keys, or \(n\)-ary link keys. No alignment is necessary, contrary to \cite{1}, though it is always possible to inject an alignment in the definitions to reduce the search space.

Once coherent families of link key candidates are extracted, they need to be evaluated. Available measures can apply in the same way because they are usually defined on the generated links.

Among further development of this approach, we plan to consider two research directions:

- Optimising the extraction process through interleaving evaluation and extraction and through using sampling techniques;

- Relaxing the data equality constraint by using similarity measures considered as tolerance relations \cite{15}.

8 Acknowledgements

This work has been partially supported by the ANR project Elker (ANR-17-CE23-0007-01, for the four first authors). Jérémy Vizzini has been partially supported by a grant from the PERSYVAL-Lab LabEx (ANR-11-LABX-0025-01) funded by the French “Investissement d’avenir” program.

References


\textsuperscript{2} https://gitlab.inria.fr/moex/linkex

