Sandwich: An Algorithm for Discovering Relevant Link Keys in an LKPS Concept Lattice

Nacira Abbas, Alexandre Bazin, Jérôme David, Amedeo Napoli

Sandwich: an Algorithm for Discovering Relevant Link Keys in an LKPS Concept Lattice

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. The discovery of link keys between two RDF datasets allows the identification of individuals which share common key characteristics. Actually link keys correspond to closed sets of a specific Galois connection and can be discovered thanks to an FCA-based algorithm. In this paper, given a pattern concept lattice where each concept intent is a link key candidate, we aim at identifying the most relevant candidates w.r.t adapted quality measures. To achieve this task, we introduce the "Sandwich" algorithm which is based on a combination of two dual bottom-up and top-down strategies for traversing the pattern concept lattice. The output of the Sandwich algorithm is a poset of the most relevant link key candidates. We provide details about the quality measures applicable to the selection of link keys, the Sandwich algorithm, and as well a discussion on the benefit of our approach.

1 Introduction

Linked data are structured data expressed in the RDF (Resource Description Framework) model where resources are identified by Internationalized Resources Identifiers (IRIs) [7]. Data interlinking is a critical task for ensuring the wide use of linked data. It consists in finding pairs of IRIs representing the same entity among different RDF datasets and returning a set of identity links between these IRIs. Many approaches have been proposed for data interlinking [11,9,8,10]. In this paper, we focus on the discovery of *link keys* [2]. Link keys extend the notion of a key as used in databases and allow the inference of identity links between RDF datasets. A link key is based on two sets of pairs of properties and a pair of classes. The pairs of properties express sufficient conditions for two subjects, instances of the classes, to be the identical. The link key

$$k = (\{\langle \texttt{designation}, \texttt{titre}\rangle\}, \{\langle \texttt{designation}, \texttt{titre}\rangle, \langle \texttt{author}, \texttt{auteur}\rangle\}, \langle \texttt{Book}, \texttt{Livre}\rangle)$$

states that whenever an instance a_1 from the class `Book` and an instance b_1 from the class `Livre` have the same values for the property `designation` and for the property `titre`, and that a_1 and b_1 share at least one value for the properties `author` and `auteur`, then a_1 and b_1 denote the same entity. We say that a link key k generates the identity link $\langle a_1, b_1\rangle$.

⋆ This work is supported by the French ANR Elker Project ANR-17-CE23-0007-01.

Link keys are in general not provided and they have to be discovered in the datasets under study. The discovery of link keys consists in extracting link key candidates from a pair of datasets and then to evaluate their relevance for the interlinking task. The relevance of a given link key is measured w.r.t. two criteria, (i) correctness, and (ii) completeness.

Link key candidates correspond to closed sets of a specific Galois connection. For this reason the question of using Formal Concept Analysis (FCA) [6] to discover link keys was naturally raised [3]. In [4] authors proposed a formal context for the discovery of link key candidates for a given pair of classes. However, when there is no alignment between classes, the choice of the right pair of classes is not necessarily straightforward. To overcome this limitation, a generalization of link key discovery based on Pattern Structures [5] was proposed in [1]. The authors introduced a specific pattern structure for link key candidate discovery over two datasets D_1 and D_2 called LK-pattern structure without requiring an a priori alignment. An LKPS-lattice is the lattice of pattern concepts generated from an LK-pattern structure. Each concept intent is a link key candidate and each extent is the link set generated by the link key in the intent.

The size of an LKPS-lattice may be prohibitively large and not all link key candidates are relevant for the interlinking task. Our purpose in this paper is to identify the relevant link keys in the LKPS-lattice and to discard the irrelevant ones. The evaluation criteria of a link key candidate, i.e., completeness and correctness, are based on adapted evaluation measures that can be used for selecting the relevant link keys, i.e., the value taken by a candidate for such a measure is above a given threshold. The evaluation measures should also be monotone, i.e., increasing or decreasing, w.r.t. the order of the LKPS-lattice. Moreover, completeness and correctness verify an "inverse" relationship, as correctness tends to decrease when completeness increases. In this paper we rely on this observation and we show that the upper part of an LKPS-lattice contains the most complete but the least correct link keys, while the lower part of the LKPS-lattice contains the most correct but the least complete link keys.

Starting from this observation, we introduce an original pruning strategy combining a "bottom-up" and a "top-down" pruning strategies, while the most relevant link key candidates are lying "in the middle" and achieve the best compromise between completeness and correctness. Accordingly, we propose the `Sandwich` algorithm that traverse and prune the LKPS-lattice w.r.t. two measures estimating correctness and completeness. The input of this algorithm is an LKPS-lattice, a correctness measure and a threshold, and as well a completeness measure and a threshold. The output of the `Sandwich` algorithm is a poset of relevant link key candidates. To the best of our knowledge, this is the first time that such an algorithm is proposed for selecting the best link key candidates w.r.t. adapted measures. In addition, this is also an elegant way of taking advantage of the fact that link key candidates correspond to the closed sets of a given Galois connection which is made explicit in the following.

The organization of the paper is as follows. First we make precise definitions and notations, and we briefly present the problem of link key discovery in a pattern structure framework. Then we present the correctness and the completeness of link keys, and the `Sandwich` algorithm for pruning an LKPS-lattice. Finally we discuss on the benefit of our strategy.

2 The Discovery of Link Keys with Pattern Structures

2.1 A Definition of Link Keys

We aim to discover identity links among two RDF datasets D_1 and D_2. An identity link is a statement of the form $\langle s_1, \texttt{owl:sameAs}, s_2 \rangle$ expressing that the subject $s_1 \in S(D_1)$ and the subject $s_2 \in S(D_2)$ represent the same real-world entity. For example, given D_1 and D_2 in Figure 1, the data interlinking task should discover the identity link $\langle a_1, \texttt{owl:sameAs}, b_1 \rangle$ because the subjects a_1 and b_1 both represent the same vaccine `"Pfizer-BioNTech"`. For short, we write $\langle a_1, b_1 \rangle$ and we call this pair a *link*. A link key is used to generate such links.

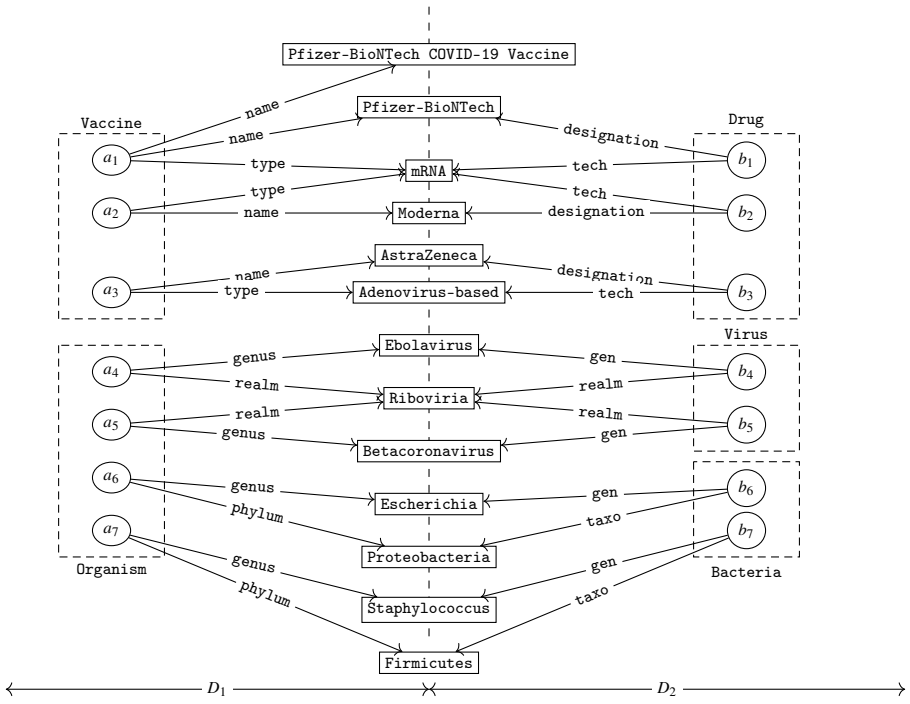

Fig. 1. Example of two RDF datasets. On the left-hand side, the dataset D_1 populated with instances of the classes: `Vaccine` and `Organism`. On the right-hand side, the dataset D_2 populated with instances of the classes: `Drug`, `Virus` and `Bacteria`.

Let us consider two RDF datasets D_1 and D_2, two non empty subsets of pairs of properties, namely Eq and In, such that $Eq \subseteq P(D_1) \times P(D_2)$, $In \subseteq P(D_1) \times P(D_2)$, $Eq \subseteq In$, two class expressions –conjunction or disjunction– C_1 over D_1 and C_2 over D_2. Then $k = (Eq, In, \langle C_1, C_2 \rangle)$ is a "link key" over D_1 and D_2. An example of a link key is $k = (\{\langle \texttt{type,tech} \rangle\}, \{\langle \texttt{type,tech} \rangle, \langle \texttt{name,designation} \rangle\}, \langle \texttt{Vaccine,Drug} \rangle)$.

A link key may or may not generate links among two datasets as made precise here after. Let $k = (Eq, In, \langle C_1, C_2 \rangle)$ be a link key over D_1 and D_2. The link key k

generates a link $\langle s_1, s_2 \rangle \in S(C_1) \times S(C_2)$ iff $\langle s_1, s_2 \rangle$ verifies the link key k, i.e. (i) $p_1(s_1)$ and $p_2(s_2)$ should be non empty, (ii) $p_1(s_1) = p_2(s_2)$ for all $\langle p_1, p_2 \rangle \in Eq$, and (iii) $p_1(s_1) \cap p_2(s_2) \neq \emptyset$ for all $\langle p_1, p_2 \rangle \in In$.

The set of pairs of subjects $\langle s_1, s_2 \rangle \in S(C_1) \times S(C_2)$ verifying k is called the link set of k and denoted by $L(k)$. As the properties in RDF are not functional, we compare the values of subjects in two ways, (i) Eq are pairs of properties for which two subjects share all their values, and (ii) In are those pairs of properties for which two subjects share at least one value. Then $\langle a_1, b_1 \rangle$ verifies $k = (\{\langle \texttt{type,tech} \rangle\}, \{\langle \texttt{type,tech} \rangle, \langle \texttt{name,designation} \rangle\}, \langle \texttt{Vaccine,Drug} \rangle)$, because $\langle a_1, b_1 \rangle \in S(\texttt{Vaccine}) \times S(\texttt{Drug})$, and $\texttt{type}(a_1) = \texttt{tech}(b_1)$, and $\texttt{name}(a_1) \cap \texttt{designation}(b_1) \neq \emptyset$.

Algorithms for link key discovery [2,4,1] discover firstly the so-called "link key candidates" and then evaluate each candidate using quality measures. The relevant link key candidates are selected to generate identity links between datasets.

A link key candidate is defined in [4] as the intent of a formal concept computed within a particular formal context for link key candidate discovery, given pair of classes $\langle C_1, C_2 \rangle \in Cl(D_1) \times Cl(D_2)$. Actually, link keys are equivalent w.r.t. their link set, i.e. if two link keys k_1 and k_2 generate the same link set then they are equivalent. Link key candidates are maximal elements of their equivalence classes and thus correspond to closed sets. Moreover, the set of links must not be empty for a link key candidate. This explains the use of Formal Concept Analysis [6] in the discovery of link key candidate.

However, the pair of classes $\langle C_1, C_2 \rangle \in Cl(D_1) \times Cl(D_2)$ is not always known in advance and thus a generalization of the existing algorithms based on Pattern Structures was proposed in [1], as explained in the following.

2.2 A Pattern Structure for Link Key Discovery

In a pattern structure designed for the discovery of link key candidates over two datasets [1], the set of objects is the set of pairs of subjects from the two datasets and the descriptions of objects are potential link keys over these datasets. A link key candidate corresponds to an intent of a pattern concept in the lattice generated from this pattern structure. Moreover the link set of a link key candidate corresponds to the extent of the formal concept. In the following, we do not provide any definition but we recall some important results from [1]. Moreover, for simplicity, we consider only the In set of pairs of properties in a link key, i.e. $k = (In, \langle C_1, C_2 \rangle)$ (as $Eq \subseteq In$).

The Lk-pattern structure for the datasets in Figure 1 is given in Table 1. The associated concept lattice, called an LKPS-Lattice, is displayed in Figure 2. An example of link key candidate is given by $k_2 = (\{\langle \texttt{name,designation} \rangle, \langle \texttt{type,tech} \rangle\}, \langle \texttt{Vaccine,Drug} \rangle)$, and the related link set is $L(k_2) = \{\langle a_1, b_1 \rangle, \langle a_2, b_2 \rangle, \langle a_3, b_3 \rangle\}$.

3 The Pruning of an LKPS-lattice

3.1 Correction and Completeness of a Link Key Candidate

Given an LKPS-lattice, we aim at identifying the most relevant link key based on a set of adapted interest measures. Link key relevance depends on two main criteria, namely

$S(D_1) \times S(D_2)$	In	$\langle C_1, C_2 \rangle$
$\langle a_1, b_1 \rangle$	$\{\langle \texttt{name}, \texttt{designation} \rangle, \langle \texttt{type}, \texttt{tech} \rangle\}$	$\langle \texttt{Vaccine}, \texttt{Drug} \rangle$
$\langle a_1, b_2 \rangle$	$\{\langle \texttt{type}, \texttt{tech} \rangle\}$	$\langle \texttt{Vaccine}, \texttt{Drug} \rangle$
$\langle a_2, b_2 \rangle$	$\{\langle \texttt{name}, \texttt{designation} \rangle, \langle \texttt{type}, \texttt{tech} \rangle\}$	$\langle \texttt{Vaccine}, \texttt{Drug} \rangle$
$\langle a_2, b_1 \rangle$	$\{\langle \texttt{type}, \texttt{tech} \rangle\}$	$\langle \texttt{Vaccine}, \texttt{Drug} \rangle$
$\langle a_3, b_3 \rangle$	$\{\langle \texttt{name}, \texttt{designation} \rangle, \langle \texttt{type}, \texttt{tech} \rangle\}$	$\langle \texttt{Vaccine}, \texttt{Drug} \rangle$
$\langle a_4, b_4 \rangle$	$\{\langle \texttt{genus}, \texttt{gen} \rangle, \langle \texttt{realm}, \texttt{realm} \rangle\}$	$\langle \texttt{Organism}, \texttt{Virus} \rangle$
$\langle a_4, b_5 \rangle$	$\{\langle \texttt{realm}, \texttt{realm} \rangle\}$	$\langle \texttt{Organism}, \texttt{Virus} \rangle$
$\langle a_5, b_5 \rangle$	$\{\langle \texttt{genus}, \texttt{gen} \rangle, \langle \texttt{realm}, \texttt{realm} \rangle\}$	$\langle \texttt{Organism}, \texttt{Virus} \rangle$
$\langle a_5, b_4 \rangle$	$\{\langle \texttt{realm}, \texttt{realm} \rangle\}$	$\langle \texttt{Organism}, \texttt{Virus} \rangle$
$\langle a_6, b_6 \rangle$	$\{\langle \texttt{genus}, \texttt{gen} \rangle, \langle \texttt{phylum}, \texttt{taxo} \rangle\}$	$\langle \texttt{Organism}, \texttt{Bacteria} \rangle$
$\langle a_7, b_7 \rangle$	$\{\langle \texttt{genus}, \texttt{gen} \rangle, \langle \texttt{phylum}, \texttt{taxo} \rangle\}$	$\langle \texttt{Organism}, \texttt{Bacteria} \rangle$

Table 1. The Lk-pattern structure over the datasets D_1 and D_2 represented in Figure 1.

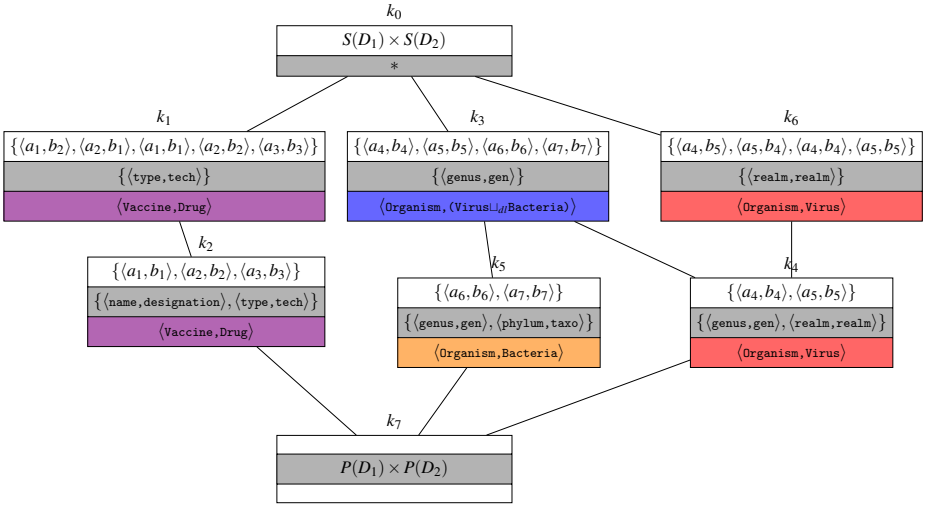

Fig. 2. The LKPS-lattice generated from the Lk-pattern structure in Table 1. The different colors distinguish the different pairs of classes.

"correctness" and "completeness". The correctness of a link key is its ability to generate correct links, while its completeness is its ability to generate all the correct links.

Firstly, we start from the hypothesis that the higher the number of pairs of properties for which two subjects s_1, s_2 share a value is, the greater is the probability that the link $\langle s_1, s_2 \rangle$ is correct. Then the *size* of the link key candidate $k = (In, \langle C_1, C_2 \rangle)$ is $|k| = |In|$. Moreover, the size of link keys is monotone w.r.t the intents of pattern concept intents in an LKPS-lattice.

In [2], the measure of coverage was proposed to evaluate the completeness of a link key candidate. The coverage of a link key $k = (In, \langle C_1, C_2 \rangle)$ is defined as follows: $co(k) = |\pi_1(L(k)) \cup \pi_2(L(k))| / |S(C_1) \cup S(C_2)|$, where $\pi_1(L(k)) = \{s_1 | \langle s_1, s_2 \rangle \in L(k)\}$ and $\pi_2(L(k)) = \{s_2 | \langle s_1, s_2 \rangle \in L(k)\}$. The coverage is locally monotone, i.e. when the

link key candidates are associated with the same pairs of classes, the coverage is monotone w.r.t extents of these candidates in the LKPS-lattice.

3.2 Sandwich: an Algorithm for Selecting the Most Relevant Link Key Candidates

Given an LKPS-lattice, we propose the `Sandwich` algorithm for identifying the relevant link key candidates (intents) and discarding the irrelevant candidates. The input of `Sandwich` is a LKPS-lattice, a correctness measure σ_{cor} and a minimum threshold μ_{cor}, and a completeness measure σ_{comp} and a minimum threshold μ_{comp}. The output of `Sandwich` is the poset of all relevant link key candidates. It should be noticed that a link key candidate may be relevant for a given pair of classes and not relevant for another pair, i.e. a given link key candidate may generate all the correct links over a pair of classes and no correct link over another pair. Accordingly, it is more appropriate to identify relevant link keys associated with each pair of classes. Thus, in a first step, `Sandwich` splits the lattice into sub-lattices where all intents are link key candidates associated with the same pairs of classes. In a second step, `Sandwich` prunes the sub-lattices based on correctness and completeness measures.

Regarding correctness, `Sandwich` retains the link key candidates k for which the score of correctness measure $\sigma_{cor}(k) \geq \mu_{cor}$. The correctness measure should be monotone w.r.t. the intents in the LKPS-lattice, and the larger intents are at in the "bottom part" of the lattice (w.r.t. the standard concept lattice order). Therefore, the most correct link keys are lying in the lower part of the considered given LKPS-lattice and a "bottom-up pruning strategy" is carried out. The intents of the retained concepts correspond to link key candidates k verifying $\sigma_{cor}(k) \geq \mu_{cor}$.

The strategy for retaining the complete kink keys is roughly the same, i.e., `Sandwich` retains link key candidates k for which the score of completeness measure $\sigma_{comp}(k) \geq \mu_{comp}$. However, by contrast, the most complete link keys are having the better covering w.r.t. the extents of concepts, which are lying in the "upper part" of a given LKPS-sub-lattice. This time, a "top-down pruning strategy" is carried out, and the extents of the retained concepts correspond to link key candidates k verifying $\sigma_{comp}(k) \geq \mu_{comp}$.

Finally, the `Sandwich` algorithm retains the concepts which are selected at the same time by both pruning strategies.

For illustrating the pruning strategy[3], let us consider the example of LKPS(D_3, D_4) displayed in Figure 3. The correctness measure which is monotone w.r.t. intents is the size of the link key and the threshold is set to $\mu_{cor} = 3$ (minimum size). The completeness measure which is monotone w.r.t extents is the coverage of a link key and the threshold is set to $\mu_{comp} = 0.9$ (minimum coverage). For the pair of classes $\langle$`Person,Personne`$\rangle$, the bottom-up pruning strategy returns all the pattern concepts whose intent size is greater than 3, i.e., $\{k_5, k_6, k_7\}$. The top-down pruning strategy returns all the pattern concepts whose coverage is above 0.9, i.e., $\{k_4, k_5\}$. Finally, the best link key w.r.t. to both strategies is k_5, and it can be used to find identity links over the RDF datasets D_3 and D_4.

[3] The datasets and the implementation generating the lattice can be checked at `https://gitlab.inria.fr/nabbas/sandwich\_algorithm`

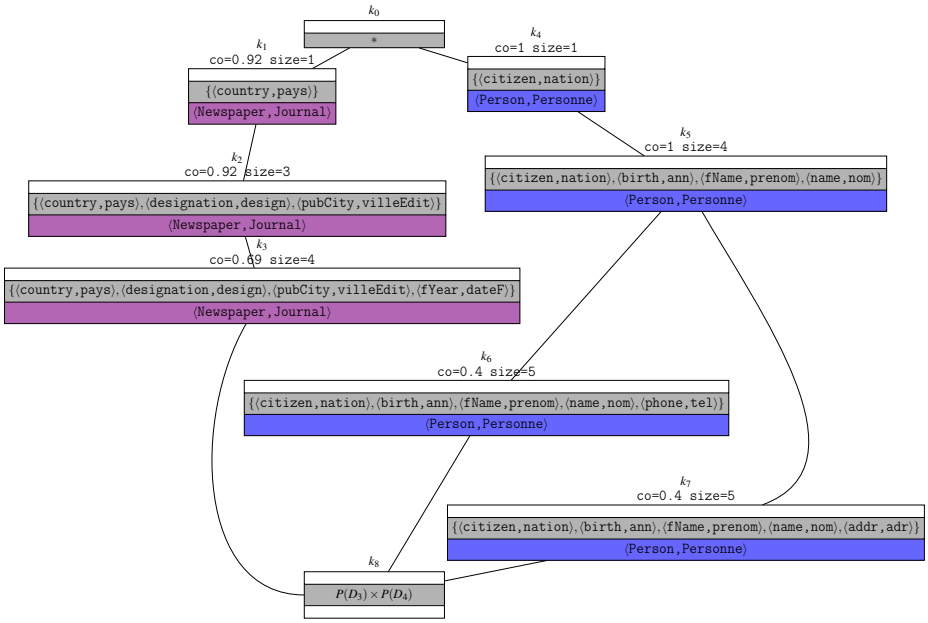

Fig. 3. The LKPS-lattice of link key candidate over the datasets D_3 and D_4.

4 Discussion and conclusion

In this paper, we have studied the problem of the discovery of link keys in an FCA framework. We have proposed an algorithm based on pattern structures which returns a pattern concept lattice where the intent of a pattern concept corresponds to a link key candidate and the associated extent corresponds to the set of links related with the link key candidate. Indeed, FCA and pattern structures are well adapted to study the discovery of link keys above two datasets as link keys can be considered as "closed sets". Actually they are introduced as the maximum element in an equivalence class in [4]. Making a parallel with equivalence classes of itemsets this emphasizes the fact that a link key corresponds to a closed set.

Then, given the pattern concept lattice including all link key candidates, one crucial problem is to select the best link keys, in the same way as one could be interested in some "good concepts" extracted from the lattice to be checked by a domain expert. For that we introduce a set of quality measures that can be used for checking two main properties, namely correctness and completeness of the link keys. As a (pattern) concept lattice is based on duality and two anti-isomorphic orders, the measures of correctness and completeness are also behaving in a dual way. Intuitively, the correctness of a link key measures the capability to generate correct links, and in this way, the largest link key will be among the best link keys (w.r.t. a reference pair of classes). Dually, the completeness of a link key measures the capability to generating the largest set of links, and such a link key will also be among the best link keys (w.r.t. a reference pair of classes).

Furthermore, following the duality principle, designing an algorithm able to discover the best link keys, i.e. reaching the best compromise between correctness and completeness, amounts to exploring the pattern concept lattice in both and dual ways, namely top-down and bottom-up. This is precisely the work of the `Sandwich` algorithm which combines two pruning strategies, a top-down traversal and a bottom-up traversal, for reaching the most complete set of links and at the same time the most correct link key candidates. This is a straightforward illustration of the duality principle in a (pattern) concept lattice.

Now, the `Sandwich` is generic and is able to work with different quality measures as soon as they are monotone or locally monotone. For the next step, we should improve the present research work in a number of directions. First, we should enlarge the collection of measures and include more current measures used in the FCA and data mining communities, such as "stability" or "lift" for example. They could provide interesting directions of investigations for characterizing link keys. Second, we should improve the global traversal strategy and combine the characterization of link key candidates at the construction of the pattern concept lattice if possible. In this way, each pattern concept could be tagged with its characteristics w.r.t. a pair of classes and some quality measures. Finally, we have run a complete set of experiments for validating the current proposal on a real-world basis, and as well check the foundations and improve the algorithmic part of the `Sandwich` algorithm.

References

1. Abbas, N., David, J., Napoli, A.: Discovery of Link Keys in RDF Data Based on Pattern Structures: Preliminary Steps. In: Proceedings of CLA. pp. 235–246. CEUR Workshop Proceedings 2668 (2020)
2. Atencia, M., David, J., Euzenat, J.: Data interlinking through robust linkkey extraction. In: Proceedings of ECAI. pp. 15–20. IOS Press (2014)
3. Atencia, M., David, J., Euzenat, J.: What can FCA do for database linkkey extraction? In: Proceedings of FCA4AI Workshop. pp. 85–92. CEUR Workshop Proceedings 1257 (2014)
4. 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)
5. Ganter, B., Kuznetsov, S.O.: Pattern Structures and Their Projections. In: Proceedings of ICCS. pp. 129–142. LNCS 2120, Springer (2001)
6. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer - Verlag (1999)
7. Heath, T., Bizer, C.: Linked Data: Evolving the Web into a Global Data Space. Synthesis Lectures on the Semantic Web, Morgan & Claypool Publishers (2011)
8. Nentwig, M., Hartung, M., Ngomo, A.N., Rahm, E.: A survey of current Link Discovery frameworks. Semantic Web **8**(3), 419–436 (2017)
9. Ngomo, A.N., Auer, S.: LIMES - A Time-Efficient Approach for Large-Scale Link Discovery on the Web of Data. In: Proceedings of IJCAI. pp. 2312–2317. IJCAI/AAAI (2011)
10. Symeonidou, D., Armant, V., Pernelle, N.: BECKEY: understanding, comparing and discovering keys of different semantics in knowledge bases. Knowledge-Based Systems **195** (2020)
11. Volz, J., Bizer, C., Gaedke, M., Kobilarov, G.: Silk - A Link Discovery Framework for the Web of Data. In: Proceedings of the WWW2009 Workshop on Linked Data on the Web (LDOW). CEUR Workshop Proceedings 538 (2009)