A Proposal for Classifying the Content of the Web of Data Based on FCA and Pattern Structures

Justine Reynaud[1(✉)], Mehwish Alam[2], Yannick Toussaint[1], and Amedeo Napoli[1]

[1] LORIA (CNRS – Inria Nancy-Grand Est – Université de Lorraine), 239, 54506 Vandoeuvre les Nancy, France
justine.reynaud@loria.fr
[2] Labortoire d'Informatique Paris Nord, Université Paris 13, Paris, France

Abstract. This paper focuses on a framework based on Formal Concept Analysis and the Pattern Structures for classifying sets of RDF triples. Firstly, this paper proposes a method to construct a pattern structure for the classification of RDF triples w.r.t. domain knowledge. More precisely, the poset of classes representing subjects and objects and the poset of predicates in RDF triples are taken into account. A similarity measure is also proposed based on these posets. Then, the paper discusses experimental details using a subset of DBpedia. It shows how the resulting pattern concept lattice is built and how it can be interpreted for discovering significant knowledge units from the obtained classes of RDF triples.

1 Introduction

The Web of Data (WOD) has become a very huge space of experimentation especially regarding knowledge discovery and knowledge engineering due to its rich and diverse nature. WOD is a database as it includes different kinds of data e.g. documents, images, videos etc. It can also be considered as a knowledge base because a major part of it relies on the Linked Data (LD) cloud. LD are further based on RDF triples of the form <subject, predicate, object> where each element in the triple denotes a resource (accessible through a URI). Moreover, the elements in a triple can be organized within partial orderings using the predefined vocabularies such as RDF Schema (RDFS), i.e. a subclass relation (`rdfs:subClassOf`) and a subproperty relation (`rdfs:subPropertyOf`, where a predicate in an RDF triple is also called a property). We rely on this double vision of WOD, as a database and as a knowledge base, for proposing a way of classifying the content of WOD thanks to Formal Concept Analysis (FCA) and its extension called as Pattern Structures. As a database, WOD can be navigated, searched, queried through SPARQL queries, and mined. As a knowledge base, WOD provides domain knowledge that can be used for guiding information retrieval, knowledge discovery and knowledge engineering. Regarding these

M. Kryszkiewicz et al. (Eds.): ISMIS 2017, LNAI 10352, pp. 684–694, 2017.
DOI: 10.1007/978-3-319-60438-1_67

tasks, questions are arising, e.g. "how to organize set of RDF triples such as triples returned as answers to a SPARQL query", "how to carry on a knowledge discovery process on WOD as a database and as a knowledge base at the same time". The first question has already been investigated by some authors of the present paper (see [3]) but improvements are still needed. The second question remains a challenge since knowledge discovery does not just amount to query processing, but can take advantage of partial orderings and of knowledge repositories lying in WOD (i.e. ontologies). Databases already define a certain schema but it is usually not as elaborate as a knowledge base, mainly due to the fact that a knowledge base shapes the human perception of the world in the form that a machine can understand thanks to an expressive knowledge representation language (e.g. OWL). Moreover, a knowledge repository is based on specific resources, e.g. ontologies, and can be seen as a set of facts and partial orderings (posets) organizing concepts and properties. Then, the posets supporting knowledge repository are of first importance for knowledge discovery within WOD.

Accordingly, we present in the following a knowledge discovery process based on Formal Concept Analysis and Pattern Structures that is applied to sets of RDF triples, taking into account the context, i.e. the knowledge resources related to the components of the RDF triples. Then, one main objective is to propose an operational mining process working on RDF triples w.r.t. domain knowledge. We extend preceding approaches by defining an order product able to organize pairs of properties and objects in the triples w.r.t. related posets of properties and objects. FCA and Pattern Structures are good candidates for mining the web of data and output concept lattices that can be explored, including concepts, implications and association rules. A concept lattice resulting from the discovery process can be considered as a new knowledge repository providing a well-founded organization to the original set of triples. Finally, the concept lattice offers a point of view on data from which an analyst can discover useful and significant knowledge units otherwise hidden in the data.

The paper is organized as follows. Section 2 motivates the proposed approach and presents WOD and Pattern Structures. Section 3 presents the existing work and the extension proposed in the current study. Section 4 discusses the experimental setup and finally Sect. 5 concludes the paper.

2 Preliminaries

2.1 The Web of Data

The Content of the Web of Data. The amount of data in the WOD has increased drastically over the past 10 years. Many important on-line resources are now represented as a part of Linked Data such as DBpedia, which represents Wikipedia Infoboxes in the form of RDF. All these data sources are represented in the form node-arc-labeled graph where each resource is connected to another through internal links and each data set is connected to another resource through external links forming Linked Open Data (LOD) cloud.

More formally, WOD can be seen as an oriented multigraph[1] $G = (V, E)$, where nodes correspond to resources and edges correspond to labeled links between those resources. Each resource can be represented as a URI, Blank Node or a literal. A literal represents a value (string, integer, date, ...) whereas a blank node designates an unidentified resource[2]. As a graph, WOD can also be considered as a set of triples (s, p, o), where s and o denote vertices, and p denotes an edge between them. Multiple RDF triples connect together to form a graph.

RDF and RDFS. Resource Description Framework[3] (RDF) allows the user to represent facts as statements, where each statement is a triple. This set of facts corresponds to an ABox in description logics. For example, (`Évariste_Galois, hasDeathPlace, Paris`) is an RDF triple which expresses a relation `hasDeath- Place` between the resources `Évariste_Galois` and `Paris`, meaning that Galois died in Paris. RDF also proposes special predicates such as `rdf:type`, which links an instance to its class, e.g. (`Évariste_Galois, rdf:type, Mathematician`).

RDF Schema[4] (RDFS) is the language including constructions for ordering RDF triples into a structure that corresponds to a TBox in description logics. The relation $\mathtt{C}_1$ `rdfs:subClassOf` $\mathtt{C}_2$ corresponds to the subsumption relation in description logics. A class C_1 is said to be *more specific than* a class C_2, declared as $\mathtt{C}_1$ `rdfs:subClassOf` $\mathtt{C}_2$, if the interpretation of $\mathtt{C}_1$, i.e. the set of instances of $\mathtt{C}_1$, is included in the interpretation of $\mathtt{C}_2$. Similarly, the relation $\mathtt{p}_1$ `rdfs:subPropertyOf` $\mathtt{p}_2$ means that if there is a relation p_1 between x and y, then there is a relation p_2 between x and y. Both relations `rdfs:subClassOf` and `rdfs:subPropertyOf` are transitive and have a logical semantics that can be operationalized as a set of inference rules [1].

The Posets of Classes and Predicates. The relations `rdfs:subClassOf` and `rdfs:subPropertyOf` are particularly interesting in the current scenario. The `rdfs:subClassOf` relation defines a partial order over classes, whereas the `rdfs:subPropertyOf` relation defines a partial order over properties.

Viewing WOD as a graph $G = (V, E)$, these two relations define two partial orders, the first over the set of vertices $(V, \leqslant_V)$ and the second over the set of edges $(E, \leqslant_E)$. In the following, we assume that both posets $(V, \leqslant_V)$ and $(E, \leqslant_E)$ are trees, an multiple inheritance will not be discussed here. We work with DBpedia, which does not make use of multiple inheritance, hence this approach is still relevant.

[1] Having more than one edges between two nodes.
[2] In this work we do not consider blank nodes.
[3] https://www.w3.org/TR/2014/REC-rdf11-mt-20140225/.
[4] https://www.w3.org/TR/2014/REC-rdf-schema-20140225/.

Querying Web of Data. SPARQL[5] is the standard language for querying WOD. The queries can be constructed with the help of graph patterns represented as a set of triples, formally termed as Basic Graph Patterns (BGP). The answer of such a query is the set of all subgraphs matching the BGP. Then, the variables are replaced by the resources of the graph. An example is presented in Fig. 1.

```
Q = SELECT DISTINCT ?s ?p ?o WHERE {
      ?s ?p ?o .
      ?s dct:subject dbc:Smartphone.
      ?p rdf:type owl:ObjectProperty.
    }
```

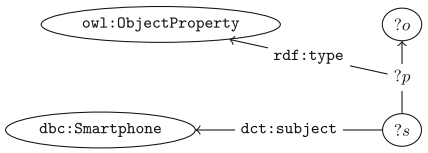


Fig. 1. Query for extracting the data and the associated basic graph pattern. Every triple extracted is connected to some subject *?s* which is an element (`dc:subject`) of the category which deals with smartphones (`dbc:Smartphone`). The prefix `dbc` is for all the DBpedia categories, whereas the prefix `dc` represents Dublin Core, a generic vocabulary for describing resources.

2.2 Pattern Structures

Pattern structures (PS) [8] are a generalization of Formal Concept Analysis[6] (FCA) [9] for dealing with complex data. While FCA is based on a binary relation between objects (G) and attributes (M), PS consider that objects in G have a description. Descriptions are partially ordered in a meet-semilattice, thanks to a subsumption relation $\sqsubseteq$ which is associated to a similarity relation denoted as $\sqcap$. More precisely, if c and d are two descriptions, then $c \sqcap d = c \Leftrightarrow c \sqsubseteq d$. Formally, a pattern structure is defined as follows:

Definition 1 (Pattern structure). *Let G be a set of objects, $(D, \sqcap)$ a semi-lattice of descriptions and $\delta : G \to D$ a mapping associating a description to an object. Then $(G, (D, \sqcap), \delta)$ is called a pattern structure. The Galois connections are the following:*

$$A^{\square} = \bigsqcap_{g \in A} \delta(g) \qquad \text{for } A \subseteq G$$

$$d^{\square} = \{g \in G \mid d \sqsubseteq \delta(g)\} \qquad \text{for } d \in D$$

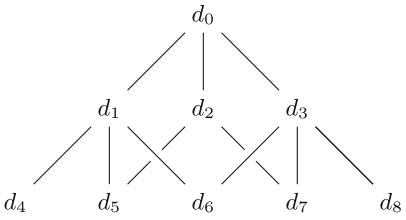
As in FCA, the composition of these mappings are closure operators: given a set of objects $A \subseteq G$ we have that $A \subseteq A^{\square\square}$ and A is closed when $A = A^{\square\square}$ (the same for $d \subseteq (D, \sqcap)$, $d \subseteq d^{\square\square}$ and d is closed when $d = d^{\square\square}$).

[5] https://www.w3.org/TR/rdf-sparql-query/.

[6] We assume that the reader is familiar with the basics of FCA thus we directly detail the basics of pattern structures.

A pattern concept (A, d) verifies that $A^{\square} = d$ and $d^{\square} = A$ where A and d are closed. Given a set of objects $A \in G, (A^{\square\square}, A^{\square})$ is a pattern concept. Similarly, if $d \in (D, \sqcap)$ is a description, $(d^{\square}, d^{\square\square})$ is a pattern concept. A partial order on pattern concepts is defined in a way similar to FCA: $(A_1, d_1) \leqslant (A_2, d_2) \Leftrightarrow A_1 \subseteq A_2 \Leftrightarrow d_2 \sqsubseteq d_1$. This partial order gives rise to a pattern concept lattice.

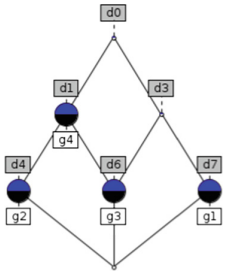
Example 1. Given the objects and their descriptions in Fig. 2, we have $\delta(g_2) = d_4$ and $\delta(g_3) = d_6$. We have $\delta(g_2) \sqcap \delta(g_3) = d_1$. Thus, $(\{g_2, g_3, g_4\}, d_1)$ is a pattern concept.



(a) The meet-semilattice $(D, \sqcap)$

obj.	descr.
g_1	d_7
g_2	d_4
g_3	d_6
g_4	d_1

(b) Objects and their descriptions



(c) Resulting lattice built with ConExp [12]

Fig. 2. Example of formal context and the resulting lattice for pattern structures.

3 Building a Pattern Structure for RDF Data

3.1 Preliminaries

FCA and patterns structures have already been used for classifying RDF data using graph structure [7,10,11] and using RDF triples [2,3]. In [2], the authors aim to provide a navigation space over RDF resources. The extent of a concept is a set of resources, and the intent is a set of pairs (predicates, objects). The similarity between two descriptions is computed pairwise. The relation `rdfs:subClassOf` is taken into account as domain knowledge.

The work in [2] is the starting point of the present work. We present hereafter an example to give the intuition on how RDF triples are taken into account and how we generalize the work in [2].

Example 2. The first part of this example gives an intuition of the pattern structure used in [2]. Given the example Fig. 3, we have:

$$\delta(Paris) = \{\underbrace{(cityOf, \{Europe\})}_{P1}, \underbrace{(capitalOf, \{France\})}_{P2}\}$$

$$\delta(Nancy) = \{\underbrace{(hasLocation, \{Europe\})}_{N1}, \underbrace{(cityOf, \{France\})}_{N2}\}$$

$$\delta(Paris) \sqcap \delta(Nancy) = \{P1 \sqcap N1, P1 \sqcap N2, P2 \sqcap N1, P2 \sqcap N2\}$$

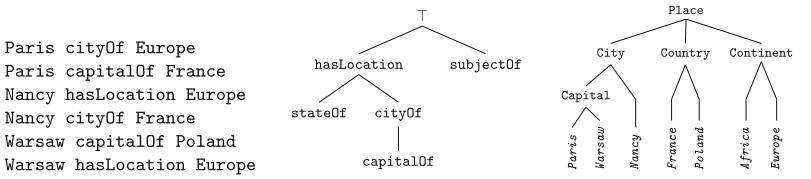
According to [2], the similarity is the following:

$$\delta(Paris) \sqcap \delta(Nancy) = \{(cityOf, \{Place\})\} \qquad \text{from } P1 \sqcap N2$$

The comparison between two pairs (predicate, object) is possible only if the predicates are the same. In the following, we extend this pattern structure to take into account the `rdfs:subPropertyOf` relation, leading to this similarity:

$$\delta(Paris) \sqcap \delta(Nancy) = \{(hasLocation, \{Europe\}), (cityOf, \{France\})\}$$
$$\text{from } P1 \sqcap N1 \text{ and } P2 \sqcap N2, \text{ which are the most specific}$$

This leads to a more accurate similarity between the two descriptions. In the next section, we show how to take into account both `rdfs:subClassOf` and `rdfs:subPropertyOf`.



(a) Set of facts **(b)** Tree of predicates **(c)** Tree of classes with their instances

Fig. 3. Toy knowledge base. Subfigure (a) illustrates a set of facts. Subfigure (b) illustrates a poset of properties w.r.t. `rdfs:subPropertyOf` relation. Subfigure (c) shows a poset of classes with their instances.

3.2 A Pattern Structure for RDF Triples

In this section, we present a pattern structure to classify RDF triples, considering the posets of classes and of predicates as domain knowledge. The data set $\mathcal{B}$ is extracted from DBpedia with a SPARQL query `Q`: all the triples satisfying the constraints expressed in the query `Q` are kept.

$$\mathcal{B} = \{(s, p, o) \mid \texttt{Q} \vDash (s, p, o)\}$$

In order to avoid confusion between the objects in FCA and the objects in RDF, objects in FCA are called entities. Then, the set G of entities corresponds to the set of subjects in the RDF triples:

$$G = \{s \mid (s, p, o) \in \mathcal{B}\}$$

For descriptions, we have a set M of pairs (p, o) corresponding to the pairs in data set $\mathcal{B}$.

$$M = \{(p, o) \mid (s, p, o) \in \mathcal{B}\}$$

This set is structured w.r.t two partial orders, contrasting with [2] where only one order is considered. Indeed, the order on predicates and the order on classes are taken into account. The resulting poset $(V \times E, \leqslant_\pi)$ is the Cartesian product of the posets $(V, \leqslant_V)$ and $(E, \leqslant_E)$:

$$(p_i, o_i) \leqslant_\pi (p_j, o_j) \Leftrightarrow p_i \leqslant_E p_j \text{ and } o_i \leqslant_V o_j$$

We define the extended set M^* as the set of all pairs (p, o) which are in M together with all pairs (p_i, o_j) such as $(p, o) \leqslant_\pi (p_i, o_j)$:

$$M^* = M \cup \bigcup_{(p,o)\in M} \{(p_i, o_j) \mid (p, o) \leqslant_\pi (p_i, o_j)\}$$

The set M^* plays the same role as the extended set of attributes introduced in [5,6] including all attributes and their subsumers.

Example 3. Considering Fig. 3, if $(capitalOf, Country)$ is contained in M, then $(capitalOf, Place)$, $(hasLocation, Country)$ and $(hasLocation, Place)$ are included in M^*.

The descriptions of entities are mappings from G to M^*, such that if a pair (p, o) is in the description of a subject s, then (s, p, o) belongs to the data set $\mathcal{B}$. From this set, we keep only the most specific elements, i.e. if $\delta(s) = \{(p, C_1), (p, C_0)\}$ and C_1 `rdfs:subClassOf` C_0, then (p, C_0) follows from (p, C_1) and $\delta(s) = \{(p, C_1)\}$. Thus, the description of a subject is the antichain of the minimal pairs in its description:

$$\delta(s) = \min\{(p, o) \mid (s, p, o) \in \mathcal{B}\}$$

where min selects the pairs which are minimal w.r.t the order defined on pairs (p, o). The intuition is the following. A description in M^* is a filter, i.e. a pair (p,o) and all subsumers of (p, o) in M^*. The filter then can be "represented" by its minimal elements. The order on descriptions is written as $\delta(s_1) \sqsubseteq \delta(s_2)$ and is interpreted as "$\delta(s_1)$ is more specific than $\delta(s_2)$":

$$\delta(s_1) \sqsubseteq \delta(s_2) \Leftrightarrow \forall(p_1, o_1) \in \delta(s_1), \exists(p_2, o_2) \in \delta(s_2) \text{ s.t. } (p_1, o_1) \leqslant_\pi (p_2, o_2)$$

Since a description such as $\delta(s_1)$ or $\delta(s_2)$ is an antichain of (p, o) pairs, when a pair $(p_1, o_1) \in \delta(s_1)$ is lower than a pair $(p_2, o_2) \in \delta(s_2)$, there does not exist any pair $(p, o) \in \delta(s_2)$ which is lower than (p_1, o_1). We can now define the similarity operator as:

$$\delta(s_1) \sqcap \delta(s_2) = \min_{\substack{(p_{1_i}, o_{1_j})\in\delta(s_1) \\ (p_{2_i}, o_{2_j})\in\delta(s_2)}} \{(lcs_E(p_{i_1}, p_{i_2}), lcs_V(o_{j_1}, o_{j_2}))\}$$

where *lcs* is the *least common subsumer* of two elements in the tree of classes or of properties.

Definition 2 (Least common subsumer). *Given a tree* $(H, \leqslant)$*, the least common subsumer of two nodes* x *and* y *of that tree is the node* z *s.t.* $x \leqslant z, y \leqslant z, \nexists z_1 \leqslant z$ *s.t.* $x \leqslant z_1$ *and* $y \leqslant z_1$*.*

The pair $(lcs_E(p_{i_1}, p_{i_2}), lcs_V(o_{j_1}, o_{j_2}))$ belongs to M^* since lcs_E relies on the `rdfs:subPropertyOf` relation and lcs_V relies on the `rdfs:subClassOf` relation. Finally, we have that:

Proposition 1. $\delta(s_1) \sqsubseteq \delta(s_2) \Leftrightarrow \delta(s_1) \sqcap \delta(s_2) = \delta(s_2)$.

This equation ensures that the resulting construction has all the good properties of a lattice, which is mandatory in the knowledge discovery process. It is reversed from the usual equation – $\delta(s_1) \sqsubseteq \delta(s_2) \Leftrightarrow \delta(s_1) \sqcap \delta(s_2) = delta(s_1)$, since the two connections are order-isomorphisms.

4 Experiments

This section illustrates our approach with the help of an experiment. Pattern concept lattice was built on a set of RDF triples extracted from DBpedia. The main points discussed are the construction of the data set and the construction of the pattern concept lattice.

Table 1. Statistics on DBpedia (April, 2016) and the smartphones corpus (January, 2017). The number of predicates and classes correspond to the number of nodes in each of the domain knowledge trees.

	Triples	Entities	Predicates	Classes	Concepts
DBpedia	9.5 billion	5.2 million	1103	754	–
Smartphones	566	3423	25	17	775
Toy ex.	5	54	4	13	14

Building the Data Set. DBpedia contains more than 9 billion triples. In the current work we focus on extracting domain specific subset of RDF triples about smartphones. The data set was extracted using the query given in Fig. 1, i.e. triples (s, p, o) such that s represents smartphones and p is an `objectProperty`, i.e. a property whose range is a resource (and not a literal). The extracted data set contains 3423 triples and is detailed Table 1. In order to present the resulting pattern concept lattice, we use a toy example with only 5 entities, described Table 1. The resulting pattern concept lattice is presented in Fig. 4. The experiment has also been run on the full corpus of Smartphones.

Resulting Lattice. Using the previous toy example, the pattern structure built a pattern concept lattice with 14 formal concepts. The extent of a formal concept is a set of instances occurring in a subject position of a triple. The intent of a pattern concept is a set of couples (predicate, object). From this resulting lattice, we can make several observations. First, object concepts were located at a very low level in the lattice, just above the bottom concept. Theoretically, object concepts could end up higher in the lattice, depending of course on the data set, but it is a rare occurrence. This means that the lattice cannot be used to rank answers to a SPARQL query as suggested in [5], as there is no way to decide if one instance fits the query better than another. Instead, the lattice provides context to the answers, highlighting similarities and differences between the entities.

Second, descriptions of instances through triples vary a lot from one to another and does not follow a regular schema. Thus, the *Blackberry* is located to the side of the lattice, sharing very few similarities with other smartphones. It is described as being a *multitouch screen* phone but not a *touch screen* phone while the *IPhone* is both a *multitouch* and a *touch screen* phone. This is probably due to some missing information in the data set. Moreover, some pairs (predicate,object) should be assigned to more instances. For example, `(type,Merchandise)`, should be shared by all instances.

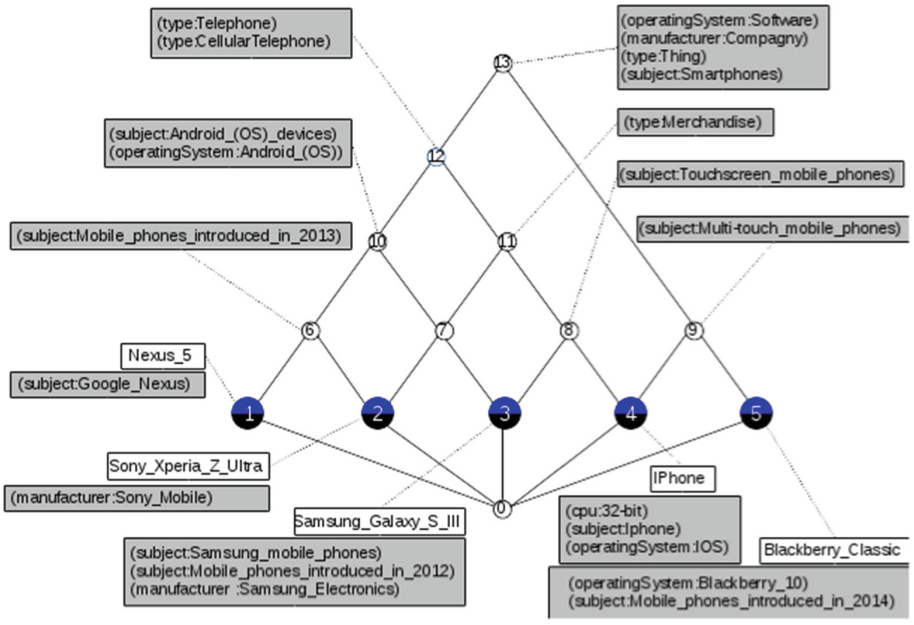


Fig. 4. Lattice built from the toy example.

We noticed that, the date of introduction is encoded in a string in such a way that we cannot formally reason about dates. However, following the informal

meaning of the strings, there is a disjunction between phones introduced in 2013 and those introduced in 2014.

Another interesting observation is that, the operating systems leads to a partition of the instances and all the concepts subsumed by `(operatingSystem,x)` form disjoint sub-lattices with different `x`. This is more obvious when looking at the lattice built with the complete data set as there is currently no phone with more than one operating system.

Finally, pattern structures, like FCA, define implication rules. Thus, we found that phones introduced in 2013 are all under Android system in our toy example. We may also learn some equivalence inside a description. For example, (`subject:Android_(OS)_devices`) $\equiv$ (`operatingSystem:Android_(OS)`). As a matter of fact, the two properties correspond to two formulations of the same property, one coming from a Wikipedia encoding, the other one from DBpedia.

From these observations, we can conclude that the lattice is of great help to add context to the data extracted from WOD using some external knowledge, giving a synthetic and structured view of the data extracted by a SPARQL query. The approach highlights some descriptions that play a major role in structuring the lattice and, conversely, highlights descriptions that are meaningless, or those that are not associated with instances where they should be. To avoid the above problems that weaken the interpretation of the lattice, two questions arise: *how can we improve data set collection from WOD to identify non-relevant or noisy properties?* and *how can we identify within the lattice, missing associations between properties and instances and then improve the data set?* One possible answer is to rely on association rules with high confidence for finding possible missing definitions. This works has been discussed in [4] and should be extended.

5 Conclusion and Future Work

In this work, we defined a pattern structure in the continuity of [2,3]. This approach is relevant for WOD, especially in the case of DBpedia, since entities (i.e. subjects of the triples) correspond to Wikipedia pages. We showed that pattern structures are relevant for taking into account domain knowledge, even with more than one partial order. Finally, we presented the resulting pattern concept lattice and discussed the observed results. An interesting extension to our work would be to consider the triples with literals having numeric values (for ages and dates). Ongoing work is using association rule mining for generating pseudo definitions using the formalism of description logics.

Acknowledgments. This work has been conducted with the support of "Région Lorraine" and "Délégation Générale de l'Armement".

References

1. Abiteboul, S., Manolescu, I., Rigaux, P., Rousset, M.C., Senellart, P.: Web Data Management. Cambridge University Press, Cambridge (2011)
2. Alam, M., Buzmakov, A., Napoli, A., Sailanbayev, A.: Revisiting pattern structures for structured attribute sets. In: CLA Proceedings, pp. 241–252 (2015)
3. Alam, M., Napoli, A.: Interactive exploration over RDF data using formal concept analysis. In: DSAA Proceedings (2015)
4. Alam, M., Buzmakov, A., Codocedo, V., Napoli, A.: Mining definitions from RDF annotations using formal concept analysis. In: IJCAI Proceedings, pp. 823–829 (2015)
5. Carpineto, C., Romano, G.: A lattice conceptual clustering system and its application to browsing retrieval. Mach. Learn. **24**(2), 95–122 (1996)
6. Carpineto, C., Romano, G.: Concept Data Analysis: Theory and Applications. Wiley, Chichester (2004)
7. Ferré, S.: A proposal for extending formal concept analysis to knowledge graphs. In: Baixeries, J., Sacarea, C., Ojeda-Aciego, M. (eds.) ICFCA 2015. LNCS (LNAI), vol. 9113, pp. 271–286. Springer, Cham (2015). doi:10.1007/978-3-319-19545-2_17
8. Ganter, B., Kuznetsov, S.O.: Pattern structures and their projections. In: Delugach, H.S., Stumme, G. (eds.) ICCS-ConceptStruct 2001. LNCS (LNAI), vol. 2120, pp. 129–142. Springer, Heidelberg (2001). doi:10.1007/3-540-44583-8_10
9. Ganter, B., Wille, R.: Formal Concept Analysis - Mathematical Foundations. Springer, Heidelberg (1999)
10. Kötters, J.: Concept lattices of RDF graphs. In: Proceedings of FCA&A@ICFCA, pp. 81–91 (2015)
11. Kuznetsov, S.O.: Computing graph-based lattices from smallest projections. In: ICCS Proceedings, pp. 35–47 (2007)
12. Yevtushenko, S.A.: System of data analysis "concept explorer". In: Proceedings of 7th national Conference on Artificial Intelligence KII, pp. 127–134 (2000)