Discrete Applied Mathematics 249 (2018) 2–17

Contents lists available at ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

# Exploratory knowledge discovery over Web of Data

Mehwish Alam [a,*], Aleksey Buzmakov [b], Amedeo Napoli [c]

[a] Institute of Cognitive Sciences and Technologies, Via San Martino della Battaglia 44, 00185 Rome, Italy
[b] National Research University, Higher School of Economics, 37 Gagarina Bulvar, Perm, Russia
[c] LORIA (CNRS –Inria Nancy Grand Est - Université de Lorraine), BP 239, 54506 Vandoeuvre les Nancy, France

ARTICLE INFO

*Article history:*
Received 13 February 2016
Received in revised form 13 March 2018
Accepted 15 March 2018
Available online 8 April 2018

*Keywords:*
Formal concept analysis
Pattern structures
Exploratory data analysis and knowledge discovery
Web of Data
Resource description framework (RDF)

ABSTRACT

With an increased interest in machine processable data and with the progress of semantic technologies, many datasets are now published in the form of RDF triples for constituting the so-called Web of Data. Data can be queried using SPARQL but there are still needs for integrating, classifying and exploring the data for data analysis and knowledge discovery purposes. This research work proposes a new approach based on Formal Concept Analysis and Pattern Structures for building a pattern concept lattice from a set of RDF triples. This lattice can be used for data exploration and visualized thanks to an adapted tool. The specific pattern structure introduced for RDF data allows to make a bridge with other studies on the use of structured attribute sets when building concept lattices. Our approach is experimentally validated on the classification of RDF data showing the efficiency of the underlying algorithms.



## 1. Introduction

World Wide Web (WWW) started as a web of documents where HTML and textual documents (resources) are connected through hyperlinks and can be identified. This web of documents is much more easily processable by humans than by machines. A way of making the web of documents machine processable is to represent the content of the web in the form of triples where one resource is connected with another resource. Resources and links between resources hold a "name" (URI). Moreover, there are two formalisms for representing triples and organization of triples, namely RDF – for representing triples and RDF Schema – for organizing resources and links. The resulting (huge) dataset in the form of entity-relationship triples is known as the "Linked Open Data" (LOD) cloud or "Web of Data" (WOD) [5].

WOD follows a *decentralized* publication model meaning that several distributed graphs of resources are published by different contributors. Most of the time, these graphs have nothing in common except some shared resources. Moreover, external data schemas in the form of ontologies or concept hierarchies are also published independently and are linked to WOD to facilitate the data analysis. Some resources only contain a schema without instances such as the SWRC ontology [33]. Some other resources may only contain triples without any schema information such as DBLP.[1]

Then, a main challenge is to provide a framework for guided navigation and exploration along with knowledge discovery over these graphs of resources. In other words, these decentralized graphs should be "centralized enough" for enabling domain specific applications. For example, when building domain specific applications, it is important to give an analyst, i.e. a domain expert or a user, an insight into what these distributed resources contain. Based on analyst-requirements and

* Corresponding author.
*E-mail addresses:* mehwish.alam@istc.cnr.it (M. Alam), AVBuzmakov@hse.ru (A. Buzmakov), amedeo.napoli@loria.fr (A. Napoli).

[1] http://dblp.l3s.de/d2r/.

https://doi.org/10.1016/j.dam.2018.03.041
0166-218X/

task-specific information, data analysis can then be carried out through exploration, following the tracks of "exploratory data analysis" [34].

To allow data analysis and not only information retrieval, an important task is to classify triples w.r.t. their associated schema. This classification can be performed over relevant datasets based on analyst and task specifications. In addition, it is valuable to combine the classification operation with visualization tools for providing human–computer interaction. Interaction and exploration are intertwined, allowing the analyst to focus on elements of interest and to select those classes of triples in which she/he is interested by providing feedback to the system.

This paper introduces a framework, namely "RDF-Pattern Structures", based on interactive data exploration [27] and Pattern Structures [21] which are an extension of Formal Concept Analysis (FCA) [22]. The proposed framework takes into account samples from datasets published as a part of Web of Data and distributed over independent resources by directly involving the analyst and user/task specifications. For classifying these selected sets of triples, we define an RDF-Pattern Structure which is based on a specific similarity measure for comparing triples in taking into account a reference schema. This way, similarity between triples amounts to an intersection of antichains. Accordingly, we also present a way of efficiently working with intersection of antichains, especially in large sets of data. An RDF-Pattern Structure generates a pattern concept lattice, i.e. a partially ordered organization of classes of triples based on a reference schema called an *RDF-Index*. This RDF-Index provides a "centralized view" over distributed resources and serves as a navigation and exploration space for the analyst. For allowing interactive operations w.r.t. the RDF-Index, we introduce a visualization tool, namely *RV-Xplorer* (Rdf-View Explorer), which enables visualization and interactions.

The main contributions of the paper can be summarized as follows:

- An original definition of the so-called RDF-Pattern Structure based on the pattern structure formalism, where RDF data are described in terms of objects and descriptions.
- An original way of defining and computing similarity among RDF-pattern descriptions based on the intersection of antichains and the RMQ procedure, revisiting and extending the seminal work of Ganter and Kuznetsov in [21].
- An interactive exploration of RDF data supported by the RV-Xplorer visualization tool.

This paper, which extends and completes several previous publications [1–3], is structured as follows. Section 2 introduces the background and the context of the present research work. Section 3 details the construction of the navigation space for RDF data. Section 4 explains the process of interactive data exploration over the navigation space and introduces the interactive visualization tool RV-Xplorer. Section 5 describes some experimental results. Finally, Section 6 discusses related work while Section 7 concludes the paper.

## 2. Preliminaries

### 2.1. Web of Data

Web of Data follows the entity-relationship model and contains two types of information i.e., schema information and factual information. *Schema* information is referred to as the already defined classes and their properties and relations between the classes built from top to bottom based on human conceptualization of a domain. One such example is `Schema.org`,[2] which is a joint effort introduced by major search engines i.e., Google, Yahoo and Bing. It defines a set of generic classes for several domains along with the properties of each class. If an HTML document is tagged with these classes then it is detected by the search engines and is shown in the form of "Google Knowledge Graphs"[3] to provide direct answers to the user queries. Resource Description Framework (RDF)[4] and SKOS[5] provide specific vocabularies for defining the schema. *Facts* keep information about specific domain such as "*car hasColor blue*". One such effort is Linked Data [5] which has become a standard for publishing data on-line in the form of entities and relationships which can further be linked to other data sources published in the same format. It uses RDF which is used for representing and storing statements, where each statement is represented as a triple $\langle subject, predicate, object \rangle$. A set of linked statements constitutes an "RDF graph" or an "RDF triple store".

For instance, Table 2 shows an example of RDF triples for papers with their keywords and authors from DBLP i.e., $t1$, $t2$, $t3$, $t4$, $t5$, and $t6$. The prefixes and full forms of all the abbreviations used in this paper are shown in Table 1. In triple $t1$ i.e., $\langle s_1, p_1, o_{11} \rangle$, $s_1$ is the subject, $p_1$ is the predicate and $o_{11}$ is the object. Here, $s$ represents the titles of the paper, $p$ represents the predicates $p_1, p_2, p_3, p_4$ and $o$ represents the authors or keywords. The subject denotes the *resource* and the predicate denotes properties of the resource and defines relationship between the subject and the object. Each resource is defined by a URI ("Uniform Resource Identifier"). In the rest of the paper we use "dereferenced" resources i.e., $s_1$ instead of a complete URI.

The background knowledge about topics in the papers is related to the keywords of the papers. It is represented in the ACM Computing Classification System (ACCS[6]) and is shown in triples $t7$, $t8$ and $t9$. For the sake of simplicity we use only the two resources DBLP and ACCS in the examples.

2 http://schema.org/.

3 A knowledge base used by Google to enhance the search engine with semantic search.

4 http://www.w3.org/RDF/.

5 http://www.w3.org/TR/swbp-skos-core-spec.

6 https://www.acm.org/about/class/2012.

**Table 1**
Prefixes and Abbreviations of the terms used in the paper.

| Abbreviation | Term |
|---|---|
| $p_1$ (dc:subject) | http://purl.org/dc/elements/1.1/subject |
| $p_2$ (dc:creator) | http://purl.org/dc/elements/1.1/creator |
| $p_3$ (dc:title) | http://purl.org/dc/elements/1.1/title |
| $p_4$ | skos:broader |
| $C_1$ | Web Crawling |
| $C_2$ | Web Indexing |
| $C_3$ | Page and Site Ranking |
| $C_4$ | RDF |
| $C_5$ | OWL |
| $C_6$ | Similarity Measure |
| $C_7$ | Question Answering |
| $C_8$ | Recommender Systems |
| $C_9$ | Clustering and Classification |
| $C_{10}$ | Web Search Engines |
| $C_{11}$ | Semantic Web |
| $C_{12}$ | World Wide Web |
| $C_{13}$ | Retrieval Models and Ranking |
| $C_{14}$ | Retrieval Tasks and Goals |

**Table 2**
RDF triples about papers with authors and keywords from DBLP.

| *tid* | Subject | Predicate | Object | Dataset |
|---|---|---|---|---|
| $t1$ | $s_1$ | $p_1$ | $o_{11}$ | DBLP |
| $t2$ | $s_1$ | $p_2$ | $o_{12}$ | DBLP |
| $t3$ | $s_2$ | $p_1$ | $o_{16}$ | DBLP |
| $t4$ | $s_2$ | $p_2$ | $o_{22}$ | DBLP |
| $t5$ | $s_1$ | rdf:type | Publication | DBLP |
| $t6$ | $o_{12}$ | rdf:type | Author | DBLP |
| $t7$ | $o_{11}$ | $p_4$ | $C_1$ | ACCS |
| $t8$ | $o_{16}$ | $p_4$ | $C_6$ | ACCS |
| $t9$ | $C_1$ | $p_4$ | $C_{10}$ | ACCS |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |

### 2.2. SPARQL

A standard query language for RDF graphs is SPARQL[7] which mainly focuses on graph matching. A SPARQL query is composed of two parts, the head and the body. The body of the query contains Basic Graph Patterns present in the WHERE clause of the query. It is composed of complex graph patterns defined by means of RDF triples with variables, conjunctions, disjunctions and constraints over the values of the variables. These graph patterns are matched against the RDF graph and the matched graph is retrieved and manipulated according to the conditions given in the query. The head of the query is an expression which indicates how the answers of the query should be constructed. A subset of these triples is selected based on analyst specifications. For example, a SPARQL query for papers from the field of classification is given in Listing 1.

```
PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#>
PREFIX dc:<http://purl.org/dc/terms/>

SELECT distinct ?title ?keywords ?author
where {
?paper dc:creator ?a .
?a rdfs:label ?author .
?paper dc:subject ?keywords .
?paper dc:title ?title .
FILTER(
  regex(STR(?keywords), "supervised classification", "i")
|| regex(STR(?keywords), "unsupervised classification", "i"))
```

Listing 1: SPARQL Query for extracting triples.

[7] http://www.w3.org/TR/rdf-sparql-query/.

**Table 3**
A formal context $\mathcal{K}$.

| | $p_2$ | | | | |
|---|---|---|---|---|---|
| Titles | $o_{21}$ | $o_{22}$ | $o_{23}$ | $o_{24}$ | $o_{25}$ |
| $s_1$ | × | | | | |
| $s_2$ | | × | × | | |
| $s_3$ | | × | | × | × |
| $s_4$ | | | × | | |
| $s_5$ | | × | × | | × |

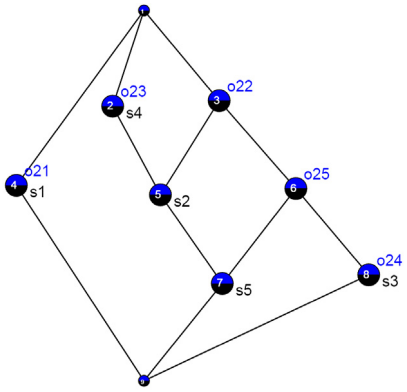


**Fig. 1.** The concept lattice for Table 3.

### 2.3. Formal concept analysis

Formal Concept Analysis (FCA) [22] is a mathematical framework used for a number of purposes, among which are classification, data analysis, information retrieval and knowledge discovery [8]. A formal context $\mathcal{K} = (G, M, I)$, consists of $G$, a set of "entities", $M$, a set of attributes, and $I$, a binary relation between entities in $G$ and attributes in $M$. It should be noticed that we rename "objects" in FCA as "entities" to avoid any confusion with the "objects" in RDF triples. Table 3 presents a formal context related to papers and their authors. The titles of the papers are considered as entities while their authors are considered as attributes. The fact that a paper has an author is represented as a cross in the binary context. According to the first row in Table 3, paper $s_1$ has author $o_{21}$.

From this context formal concepts are computed by applying derivation operators. Given $A \subseteq G$ and $B \subseteq M$, two derivation operators, both denoted by $(\cdot)'$, formalize the sharing of attributes by objects, and dually, the sharing of objects by attributes:

$$A' = \{m \in M \mid gIm \text{ for all } g \in A\} \tag{1}$$

$$B' = \{g \in G \mid gIm \text{ for all } m \in B\}. \tag{2}$$

The two derivation operators form a *Galois connection* between the powersets $\wp(G)$ and $\wp(M)$. A formal concept of the context $\mathcal{K}$ is a pair $(A, B)$ where $A \subseteq G, B \subseteq M, A' = B$ and $B' = A$. Moreover, $A$ is called the "extent" and $B$ the "intent" of the $(A, B)$ concept. Considering the context in Table 3, the pair $(\{s_2, s_5\}, \{o_{22}, o_{23}\})$ is a formal concept because $\{s_2\}' = \{o_{22}, o_{23}\}$ and $\{o_{22}, o_{23}\}' = \{s_2, s_5\}$, meaning that the set of authors common to $s_2$ and $s_5$ are $\{o_{22}, o_{23}\}$. It is represented as a maximal rectangle, highlighted in gray in Table 3.

Let $\mathcal{B}(G, M, I)$ be the set of all formal concepts for $\mathcal{K} = (G, M, I)$. Given two concepts $(A_1, B_1)$ and $(A_2, B_2)$, then $(A_1, B_1)$ is a subconcept of $(A_2, B_2)$ – dually $(A_2, B_2)$ is a superconcept of $(A_1, B_1)$– denoted by $(A_1, B_1) \leqslant (A_2, B_2)$, iff $A_1 \subseteq A_2$ – dually $B_2 \subseteq B_1$. For example, in Fig. 1, we have that $(\{s_5\}, \{o_{22}, o_{23}, o_{25}\}) \leqslant (\{s_2, s_5\}, \{o_{22}, o_{23}\})$ Fig. 1 shows a complete lattice for Table 3. In this figure we use "reduced labeling", which means that every subconcept of a concept say $X$ also contains the attributes present in the intent of $X$. Dually, every superconcept of a concept $X$ contains the objects present in the extent of $X$.

### 2.4. Pattern structures

FCA [22] can process only binary contexts and more complex data such as graphs cannot be directly processed. Pattern structures [21] provide an extension of FCA which allows direct processing of more complex data such as numbers, intervals, trees and graphs. Intuitively, pattern structures generalize the classical FCA setting in the following way. Let us consider two

**Table 4**
Context with about the temperatures of the cities in different weather.

| | Summer | Winter | Spring | Autumn |
|---|---|---|---|---|
| Paris | 30 | −5 | 18 | 12 |
| Prague | 25 | −10 | 7 | 9 |
| Rome | 35 | 2 | 12 | 15 |

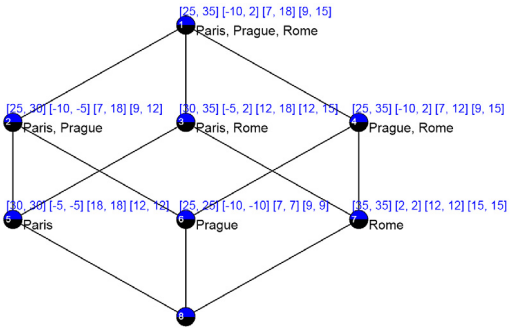


**Fig. 2.** Pattern Concept lattice for Table 4.

entities with their attributes in the formal context in Table 3, say $(\{s_2\}, \{o_{22}, o_{23}\})$ and $(\{s_5\}, \{o_{22}, o_{23}, o_{25}\})$. Attribute sharing and then concept building in FCA is based on "intersection of sets of attributes", e.g. $(\{s_2, s_5\}, \{o_{22}, o_{23}\})$ forms a concept. Moreover, intersection is related to inclusion as follows: if $X$ and $Y$ are two sets, we have $X \cap Y = X \iff X \subseteq Y$.

Generalizing these ideas, let us suppose that we have two entities $g_i$ and $g_j$ with their descriptions $d_i$ and $d_j$. The common description of $d_i$ and $d_j$ will be captured by a so-called "similarity operator", denoted by $d_i \sqcap d_j$, which can be understood as a generalization of intersection. In addition, descriptions can be organized thanks to a partial ordering denoted by $\sqsubseteq$ which verifies, for any two descriptions $d_1$ and $d_2$, $d_1 \sqcap d_2 = d_1 \iff d_1 \sqsubseteq d_2$. For example, going back to the binary case, if we assume that $d_2 = \{o_{22}, o_{23}\}$ and $d_5 = \{o_{22}, o_{23}, o_{25}\}$, then $d_2 \sqsubseteq d_5 = \{o_{22}, o_{23}\} \cap \{o_{22}, o_{23}, o_{25}\} = \{o_{22}, o_{23}\} = d_2$, i.e. $d_2 \sqsubseteq d_5$. The description $d_2 = \{o_{22}, o_{23}\}$ is smaller than the description $d_5 = \{o_{22}, o_{23}, o_{25}\}$ w.r.t. the partial ordering $\sqsubseteq$.

More formally, a pattern structure is a triple $(G, (D, \sqcap), \delta)$, where $G$ is the set of entities, $(D, \sqcap)$ is a "meet-semilattice" of descriptions $D$ and $\delta : G \to D$ maps an entity to its description. A meet-semilattice is a partially ordered set having a meet or a greatest lower bound, in which all pairs have a meet.[8] The fact that $(D, \sqcap)$ is a meet-semilattice guarantees that the meet of any two descriptions always exist. In a pattern structure $(G, (D, \sqcap), \delta)$, the derivation operators are defined as follows:

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

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

An element in $D$ is referred to as a *pattern*, and the subsumption order over these patterns verifies, for any two descriptions $c$ and $d$: $c \sqsubseteq d \Leftrightarrow c \sqcap d = c$. The two operators $(.)^{\square}$ form a Galois connection as introduced in Section 2.3, and a pattern concept is defined as follows. A "pattern concept" of a pattern structure $(G, (D, \sqcap), \delta)$ is a pair $(A, d)$ where $A \subseteq G$ and $d \in D$ such that $A^{\square} = d$ and $A = d^{\square}$, where $A$ is called the concept "extent" and $d$ is called the concept "intent".

We illustrate pattern structures with numerical and interval data. Let us consider a data table about temperatures in some European cities at different periods of year (see Table 4). The first record indicates that the average temperature in Paris during Summer is 30. Actually, a description is defined as a vector of intervals rather than a vector of numbers (an interval shows the possible variations of temperatures). Then, the mapping $\delta : G \longrightarrow D$ is given for the *Paris* entity by $\delta(Paris) = \langle [30, 30], [-5, -5], [18, 18], [12, 12] \rangle$. The similarity operation for $(D, \sqcap)$ is defined for any two intervals as the "convex hull" of the intervals.

Given that $\delta(Paris) = \langle [30, 30], [-5, -5], [18, 18], [12, 12] \rangle$ and $\delta(Prague) = \langle [25, 25], [-10, -10], [7, 7], [9, 9] \rangle$, the similarity between both descriptions is $\delta(Paris) \sqcap \delta(Prague) = \langle [25, 30], [-10, -5], [7, 18], [9, 12] \rangle$. The resulting pattern concept is $(\{Paris, Prague\}, \langle [25, 30], [-10, -5], [7, 18], [9, 12] \rangle)$.

The partial ordering between pattern concepts is defined in (quite) the same way as in classical FCA, i.e. $(A_1, d_1) \leqslant (A_2, d_2)$ as soon as $A_1 \subseteq A_2$ or dually $d_2 \sqsubseteq d_1$. Then we can build a *pattern concept lattice* (see Fig. 2). A smooth and complete introduction to the interval pattern structure for numerical data is given in [24,25].

[8] The meet operation is idempotent ($x \sqcap x = x$), commutative ($x \sqcap y = y \sqcap x$) and associative. A partially ordered set in which all pairs have a "join", i.e. a lowest greater bound, is a join-semilattice. A partially ordered set that is both a join-semilattice and a meet-semilattice is a lattice.

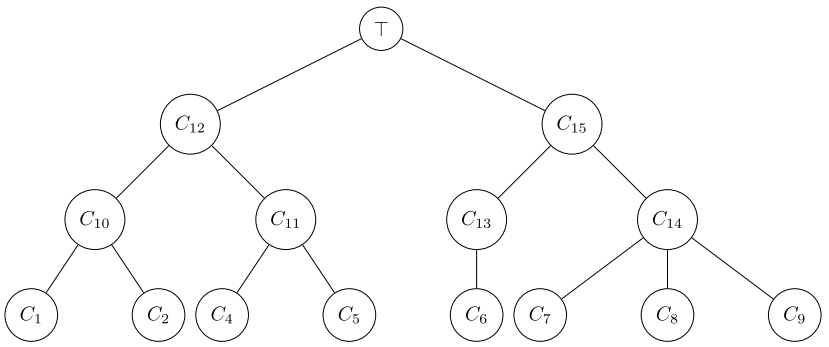


**Fig. 3.** A tiny part from ACM Computing Classification System.

**Table 5**
RDF Triples as entities $S$ and semantic descriptions $D$.

| Entities $S$ | $d_{i1}$ | $d_{i2}$ |
|---|---|---|
| $s_1$ | $(p_1 : \{C_1, C_2, C_7\})$ | $(p_2 : \{o_{21}\})$ |
| $s_2$ | $(p_1 : \{C_6, C_8, C_9\})$ | $(p_2 : \{o_{22}, o_{23}\})$ |
| $s_3$ | $(p_1 : \{C_4, C_5\})$ | $(p_2 : \{o_{22}, o_{24}, o_{25}\})$ |
| $s_4$ | $(p_1 : \{C_4, C_7, C_8\})$ | $(p_2 : \{o_{23}\})$ |
| $s_5$ | $(p_1 : \{C_8, C_9\})$ | $(p_2 : \{o_{22}, o_{23}, o_{25}\})$ |

## 3. Building an RDF-pattern structure

Below, we explain how to define a suitable pattern structure $(G, (D, \sqcap), \delta)$ for dealing with sets of RDF triples. RDF data are based on triples of the form $(s, p, o)$ where subject $s$ and object $o$ related by predicate $p$ can be organized within a class hierarchy. This can be the case for example in RDF Schema which includes many constructs among which "subclass" and "subproperty". Here, we only consider predicates such as `rdfs:subClassOf` and `skos:broader` which organize classes of subjects or objects into a tree structure. This tree structure is called the *reference schema* and is denoted as $(S, \leqslant_s)$, where $C_1 \leqslant_s C_2$ means that class $C_2$ is more general than class $C_1$ in $(S, \leqslant_s)$. Hence, the $(S, \leqslant_s)$ tree structure is used for comparing subjects and objects in the RDF triples.

Then a similarity operator can be defined for comparing RDF triples with the same subjects and the same predicates but different objects. This allows us to build an organization of RDF triples w.r.t a reference schema, into a pattern concept lattice, also called an RDF-Index. The RDF-Index can be used for navigation and interactive exploration purposes.

### 3.1. From RDF triples to an RDF-pattern structure

Hereafter we consider Listing 1 and we show how to represent RDF triples extracted by this SPARQL query as entities and their descriptions in a pattern structure $(G, (D, \sqcap), \delta)$. A subject $s$ in an RDF triple $(s, p, o)$ is mapped to an entity $g$ in the set of entities $G$, and the predicate–object pair $(p, o)$ is mapped to a description $d \in D$. More precisely, the set of RDF triples $(s, p, o)$ in which $s$ is a subject is rewritten as $(s, \{p_i : \{o_1, o_2, \ldots, o_{|i|}\}\})$ with $i \in \{1, \ldots, n\}$ and $|i|$ denoting the cardinality of the set of objects related to $s$ through the predicate $p_i$.

For example, in Table 2, the object related to $s_1$ through $p_1$ is $o_{11}$ and belongs to the reference schema ACCS, while the object related to $s_1$ through $p_2$ is $o_{12}$ and denotes names of authors (names do not belong to any reference schema and cannot be compared). The schema associated with ACCS is shown in Fig. 3 and is used for comparing objects related to topics of papers. The circles represent classes of entities and the lines represent the ordering relation $\leqslant_s$. As the ordering $\leqslant_s$ is defined at the class level, each object is identified with its corresponding classes, e.g. $o_{11}$ is identified with $C_1$ meaning that $o_{11}$ is an instance of class $C_1$. Then, the description $\{(p_1 : \{o_{11}\})\}$ becomes $\{(p_1 : \{C_1\})\}$. This identification is only performed for descriptions for which there is an available reference schema.

Continuing with the example and considering the triples $t1 = (s_1, p_1, o_{11})$ and $t2 = (s_1, p_2, o_{12})$ in Table 2, the description of $s_1$ is $\delta(s_1) = \{d_{11}, d_{12}\}$ where $d_{11} = p_1 : \{C_1, C_2, C_7\}$ and $d_{12} = p_2 : \{o_{12}\}$ (the classes of $o_{11}$ are $\{C_1, C_2, C_7\}$ and $o_{12}$ is an author name). It should be noticed that a description such as $d_{11}$ should form an antichain w.r.t. the reference schema i.e. elements in $d_{11}$ – as in each description – are not comparable w.r.t. class ordering. When ranges of predicates contain ordered elements, they are always transformed to become antichains, retaining only minimal elements between comparable elements. Table 5 shows a final representation of the RDF triples.

### 3.2. Similarity as the LCS operation

The similarity operation between two different classes is based on their "Least Common Subsumer" or LCS in the class hierarchy. Actually, this operation is related to "structured sets of attributes", i.e. attributes in a context are partially ordered, and was already studied in [7,8] for plain FCA and in [21] for pattern structures.

In [7,8], the authors consider a formal context $(G, M, I)$ and an extended set of attributes $M^*$ of $M$ where attributes are organized within a subsumption hierarchy according to a partial ordering denoted by $\leqslant_{M^*}$. The subsumption hierarchy can be either a tree or an acyclic graph with a unique maximal element. Then the construction of the concept lattice from such a context can be done in two main ways. A first one is to use a scaling and to complete the description of an object with all attributes implied by the original attributes. The problem is the space necessary to store the scaled context, especially in case of large datasets. A second way is to use an "extended intersection operation" between sets of attributes which is defined as follows. The intersection of two sets of attributes $Y_1$ and $Y_2$ is obtained by finding for each pair $(m_1, m_2)$, $m_1 \in Y_1$, $m_2 \in Y_2$, the most specific attributes in $M^*$ that are more general than $m_1$ and $m_2$, and then retaining only the most specific elements of the set of attributes generated in this way, i.e. the LCS of $m_1$ and $m_2$.

In [21], the authors introduce a pattern structure $(G, (D, \sqcap), \delta)$ for structured sets of attributes. It is assumed that the attribute set $(M, \leqslant_M)$ is finite and partially ordered, and that all attribute combinations that can occur must be order ideals (downsets) of this order. Any order ideal $O$ is described by the set of its maximal elements, i.e. $O = \{x | \exists y \in M,\ x \leqslant y\}$, which is an antichain. The set $D$ of descriptions includes these antichains and the similarity operation $\sqcap$ is based on the intersection of two antichains (details are given in [21] and in [1]).

In the present work, we adapt the pattern structure introduced in [21] but we keep the ordering of attribute descriptions as in [7,8], i.e. the most general attribute descriptions are higher than the most specific attribute descriptions. Thus, the similarity operation between two descriptions is defined as the LCS operation and it returns the most specific description which is more general than two descriptions. The LCS gives an idea of the "closeness" between two descriptions. Practically, the LCS operation is implemented using the "Range Minimum Query" algorithm which is discussed in Appendix.

### 3.3. The practical definition of the similarity operation

In this section, we discuss the structure of the meet-semi-lattice of descriptions along with the similarity and subsumption order on descriptions. We consider two descriptions of the form $p_i : A$ and $p_i : B$. $A$ and $B$ are the "range" of the predicate $p_i$ and, as noticed above, are antichains of the reference schema $(S, \leqslant_s)$. Then, it should be noticed that the similarity $p_i : A \sqcap p_j : B$ is not computed whenever $i \neq j$, and that $p_i : A \sqcap p_i : B = p_i : (A \sqcap B)$, where $(A \sqcap B)$ is the intersection of antichains $A$ and $B$.

Two main cases are considered here, the antichains are singletons or not. In the first case, let us consider two descriptions $c = p_i : A$ and $d = p_i : B$, where $A$ and $B$ correspond to classes in the reference schema $(S, \leqslant_s)$. Then, we have the following definition of similarity and the associated ordering relation (subsumption order) where the LCS operation is computed in $(S, \leqslant_s)$:

$$p_i : A \sqcap p_i : B = p_i : (A \sqcap B) = p_i : LCS(A, B)$$

$$p_i : A \sqcap p_i : B = p_i : A \Leftrightarrow p_i : A \sqsubseteq p_i : B.$$

For example, based on the reference schema shown in Fig. 3, it comes:

$$p_1 : C_4 \sqcap p_1 : C_5 = p_1 : (C_4 \sqcap C_5) = p_1 : LCS(C_4, C_5) = p_1 : C_{11}.$$

$$p_1 : C_{11} \sqcap p_1 : C_4 = p_1 : C_{11} \Leftrightarrow p_1 : C_{11} \sqsubseteq p_1 : C_4$$

$$p_1 : C_{11} \sqcap p_1 : C_5 = p_1 : C_{11} \Leftrightarrow p_1 : C_{11} \sqsubseteq p_1 : C_5$$

In the second case, we consider descriptions $c = p_i : A$ and $d = p_i : B$, where $A$ and $B$ correspond to set of classes, actually antichains, in the reference schema $(S, \leqslant_s)$. Intuitively, we have to compute the LCS of all mutual pairs of classes and only retain the minimal classes of the resulting set. Working on all the pairs would not be efficient and we rely on an elegant and efficient way of computing the LCS of two antichains by means of the RMQ algorithm (see Appendix and [1]).

For continuing the intuition, let us consider two antichains based on the running reference schema (Fig. 3). If $A = \{C_1\}$ and $B = \{C_4, C_7, C_8\}$ then we should compute $LCS(C_1, C_4) = C_{12}$, $LCS(C_1, C_7) = \top$ and $LCS(C_1, C_8) = \top$. The two last operations return $\top$, i.e. the most general class, and in this case we consider that the LCS does not exist (in any case, it can be noticed that $\top$ would be discarded as we only retain the minimal elements in the final LCS). In the same way, if now we consider $A = \{C_1, C_2, C_7\}$ and $B = \{C_4, C_7, C_8\}$ and compute the mutual LCS of each pair, we obtain the set $\{C_{12}, C_7, C_{14}\}$ and retain the final set $\{C_{12}, C_7\}$ as $\{C_{14}\}$ is not minimal ($C_7 \leqslant_s C_{14}$ in $(S, \leqslant_s)$).

Finally, let us remark that the LCS of two antichains verifies the following property:

$$\forall \ell \in LCS(A, B), \exists a \in A, \exists b \in B, a \leqslant_s \ell, b \leqslant_s \ell.$$

It means that all element in $LCS(A, B)$ has a corresponding lower element in each set $A$ and $B$, as it is the case for an intersection, i.e. an element in the intersection is included in both intersected sets.

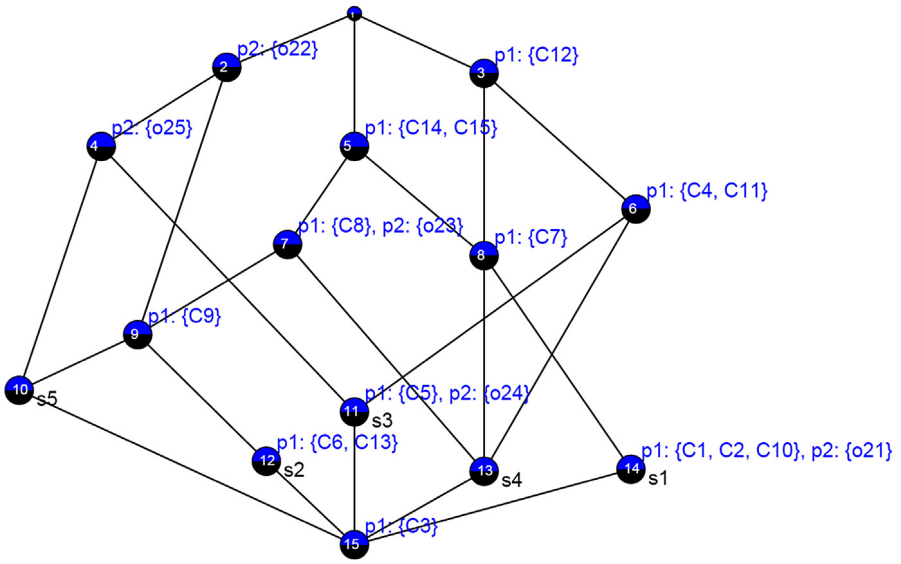


**Fig. 4.** The pattern concept lattice or RDF-Index for descriptions in Table 5.

### *3.4. Building the pattern concept lattice in an RDF-pattern structure*

In this section, we show how a pattern concept lattice can be constructed. Following Section 2.4, given a subset of objects $A \subseteq G$, $A^{\square}$ returns the set of descriptions representing the similarity between all subjects in $A$. This similarity as detailed above relies on intersection of antichains constituting the range of the predicates in the RDF triples. Moreover, when the objects in the ranges of the predicates have no reference schema, then the ranges are considered as antichains themselves. Then the similarity of such antichains amounts to a simple intersection of sets. For example, let us consider the computation of $\{s_1, s_3\}^{\square}$:

$$\begin{aligned}
\{s_1, s_3\}^{\square} &= \prod_{s \in \{s_1, s_3\}} \delta(s) \\
&= \delta(s_1) \sqcap \delta(s_3) \\
&= \langle (p_1 : \{C_1, C_2, C_7\})(p_2 : \{o_{12}\}) \\
&\quad \sqcap (p_1 : \{C_4, C_5\})(p_2 : \{o_{22}, o_{24}, o_{25}\}) \rangle \\
&= \langle (p_1 : \{C_1, C_2, C_7\}) \sqcap (p_1 : \{C_4, C_5\}), \\
&\quad (p_2 : \{o_{21}\}) \sqcap (p_2 : \{o_{22}, o_{24}, o_{25}\}) \rangle \\
&= \langle (p_1 : \{C_{12}\})(p_2 : \{\}) \rangle
\end{aligned}$$

$$\begin{aligned}
\langle (p_1 : \{C_{12}\})(p_2 : \{\}) \rangle^{\square} &= \{s \in G \mid \langle (p_1 : \{C_{12}\})(p_2 : \{\}) \rangle \sqsubseteq \delta(s)\} \\
&= \{s_1, s_3, s_4\}.
\end{aligned}$$

The pair $(A, d) = (\{s_1, s_3, s_4\}, \langle (p_1 : \{C_{12}\})(p_2 : \{\}) \rangle)$ is a pattern concept (i.e. $A^{\square} = d$ and $d^{\square} = A$), denoted as $K\#3$ in the final pattern concept lattice shown in Fig. 4. The subsumption order $\sqsubseteq$ between two pattern concepts $(A_1, d_1)$ and $(A_2, d_2)$ is given as follows: $(A_1, d_1) \sqsubseteq (A_2, d_2) \iff A_1 \subseteq A_2$ or dually $d_2 \sqsubseteq d_1$. This pattern concept lattice is called an "RDF-Index" and can be navigated and explored.

## 4. Navigation and interactive exploration over the RDF-index

In order to support exploration in Linked Data, it is necessary to provide the analyst some tools for classifying and exploring the data, interpreting the results and providing feedback. We illustrate these tasks with the help of a scenario. Moreover, we will also give details on the visualization tool RV-Xplorer especially designed for data exploration.

### *4.1. Motivating scenario*

Consider the scenario where an analyst wants to search for the papers published in conferences or journals related to a given field of research. Some of the problems faced by the analyst for retrieving and visualizing such papers are as follows:

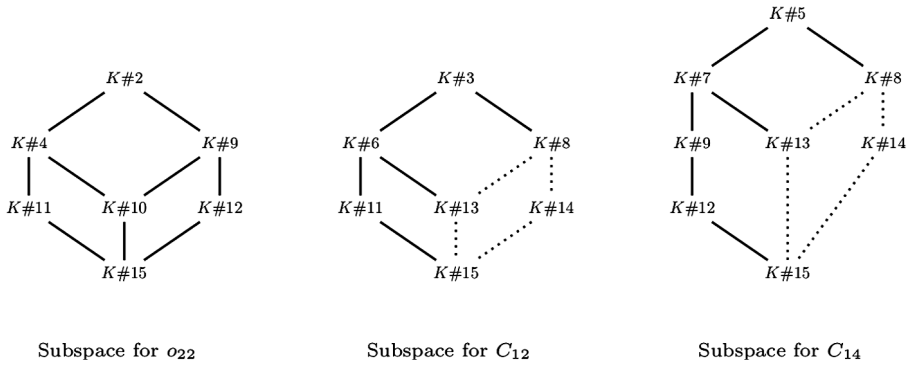


**Fig. 5.** Sublattices from Fig. 4.

- The analyst looks-up the DBLP page of some authors working in the reference field. For a complete view, the analyst has to go through all the publications of each author and then browse through the DBLP pages of the co-authors.
- If the analyst is searching for the papers which are targeting more than one field, such as "Information Retrieval" and "World Wide Web", then it should be desirable to retrieve such papers directly.
- It can be interesting for the analyst to detect the communities of authors who often work together to retrieve more relevant papers or to envision possible collaborations with authors in these communities.
- Finally, detecting the "diversity" of an author can give an idea of the competencies of this author.

Accordingly we try to guide this kind of exploration based on an RDF-Index which is built from an initial set of Linked Data and then is explored according to some preferences.

### *4.2. Interactive data exploration over the RDF-index*

Several navigation operations can be applied over the RDF-Index for obtaining precise information. In the RDF-Index, every concept $C$ contains a group of subjects (extent of $C$) connected to classes of the objects through predicates (intent of $C$). The most general concepts in the higher levels of the pattern concept lattice have extents of larger size (i.e. higher number of subjects or entities) and a smaller number of classes – in the range of predicates – in the intents, i.e. descriptions are very general. Then, two basic navigation operations are *upward* and *downward navigation*. Moreover, the operation of hiding a part of the concept lattice is provided to focus only on relevant classes, while a sublattice in the RDF-Index can be interpreted as a community of authors. Below, we provide details on each aspect.

*Downward/Upward navigation.* Downward navigation allows the analyst to move from more general to more specific concepts. For example, if an analyst wants to retrieve the scientific papers on some topic such as "World Wide Web", she/he locates the concept containing only papers about this topic i.e. $K\#3$ in Fig. 5. For narrowing down to the papers related to "World Wide Web" and "Question Answering", the lattice can be navigated downwards to obtain $K\#8$ which contains more specialized papers. By contrast, the analyst may want to go back to a more general concept, e.g. from $K\#8$ to $K\#3$, using an *upward navigation*.

*Hiding non relevant concepts/sublattices.* The analyst can explore the RDF-Index from any of the dimensions, e.g. authors and topics. Then, the analyst can mark a concept as irrelevant and then all the subconcepts in the RDF-Index will be marked as irrelevant as well and will be hidden.

For example, during the navigation of the RDF-Index, the analyst visits $K\#3$ which contains papers on "World Wide Web". If the analyst is not interested in papers on this topic, then $K\#3$ is marked as irrelevant and then the subconcepts $K\#6$, $K\#8$, $K\#11$, $K\#13$, and $K\#14$, are marked as irrelevant as well.

Moreover, continuing the exploration w.r.t the author dimension, let us suppose that the analyst marks $K\#2$ as irrelevant (e.g. $K\#2$ is related to author $o_{22}$), then the concepts in the sublattice whose $K\#2$ is the top are marked as irrelevant as well, i.e. $K\#4$, $K\#9$, $K\#10$, $K\#11$, and $K\#12$ (see Fig. 5).

*Sublattices as community of authors.* Some sublattices can be interpreted as subspaces related to a topic or an author. Fig. 5 shows three examples of such subspaces. The first sublattice is related to the author $o_{22}$ and represents the community of authors working with author $o_{22}$. The concept $K\#2$ contains all the papers published by the author $o_{22}$. Then this sublattice can be navigated downwards to visit more specific concepts such as $K\#4$ and $K\#9$. Moreover, $K\#4$ and $K\#9$ provide information about co-authors of $o_{22}$, e.g. $o_{23}$ and $o_{25}$ and represent a community of authors that work with $o_{22}$. Based on the cardinality of the extent of $K\#4$ and $K\#9$, the importance of the community can be measured, i.e. the number of common papers is high or not. Missing relations between authors can also be detected, e.g. $o_{22}$ shares papers with $o_{23}$ and $o_{25}$, but not with both authors. Then collaborations can be suggested.

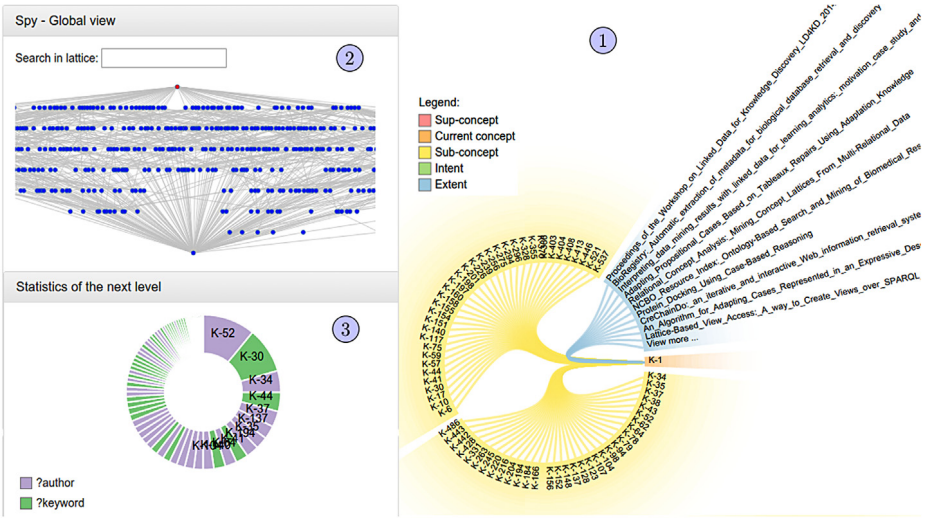


**Fig. 6.** The basic interface of RV-Xplorer displaying the top concept $K\#1$.

Finally, Fig. 5 shows two subspaces, one w.r.t. the topic "World Wide Web", and the second w.r.t. the topic "Information Retrieval". The dotted parts in both subspaces represent the subspace common to the two topics, i.e. these concepts include the papers depending on both topics.

### 4.3. Visualization

An experiment was performed on the papers published by the Data Mining Team in the LORIA Lab.[9] For this purpose, all the papers of the team published from 2010 to 2014 in international journals and conferences were selected. An RDF-Index was built using the paper titles, their keywords and authors and the reference schema is ACCS. The results were visualized using the tool RV-Xplorer (Rdf View eXplorer[10] ) [3].

Fig. 6 shows the interface of RV-Xplorer which consists of three parts:

- ① is called the *local view* and shows a detailed description of the selected concept for allowing interaction and navigation.
- ② is called the *spy* and shows the global view of the pattern concept lattice.
- ③ is called the *summarization index* and can be used to guide the analyst when navigating level by level in the pattern concept lattice, showing the statistics of the next level to visit.

Fig. 7 shows the selected concept displaying its contents, i.e., the extent, intent, parent concepts and children concepts. The pink and yellow parts in the selected concept ($K\#52$) show the parent ($K\#1$) and child concepts ($K\#342$, $K\#53$, ...) respectively. The zone displaying the children concepts is broken into parts based on the intent type, e.g., intents containing only authors, only topics and a mix of both authors and topics. This is further distinguished in the *summarization index* with the help of different colors. The green and blue parts show the intent and the extent of the concept respectively i.e., the group of papers sharing some authors and topics. For example, this selected concept includes all the papers published by the author "*Amedeo Napoli*". Suppose that the analyst wants to check with which author *Amedeo Napoli* published most of his papers during the period of 2010–2014. With the help of the summarization index, it can be seen that this author is in concept $K\#321$.

As the number of subconcepts can be very large in number, RV-Xplorer shows the intent of each subconcept on mouse-over, in the present case $K\#321$. Such information may guide the analyst and suggest some concepts to visit. This way the analyst can navigate upwards and downwards in the RDF-Index to access specific as well as general information. Finally, if the analyst wants to narrow down papers written by *Amedeo Napoli* and *Sergei O. Kuznetsov* together, she/he will click on $K\#321$ in the yellow part. Which then opens the selected concept (see Fig. 8).

The spy (② in Fig. 6) shows the complete lattice to track the position of the selected concept, highlighted in red. If the analyst wants to check details about a particular paper, then on mouse over the concept is highlighted in red in the spy (see Fig. 6).

9 Laboratoire Lorrain de Recherche en Informatique et ses Applications, Nancy, France.

10 A dedicated web page to visualize and interact with the index is available at http://rv-xplorer.loria.fr/#/graph/orpailleur_paper/1/.

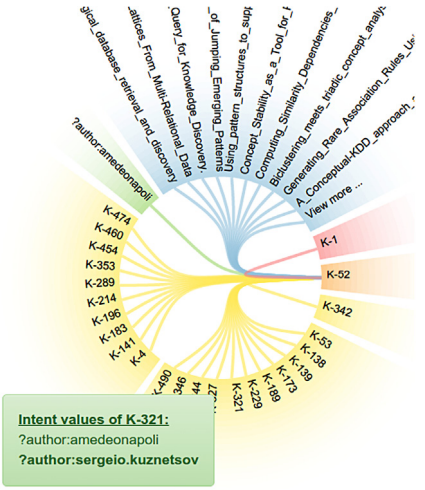


**Fig. 7.** Concept displaying all the papers of one author.

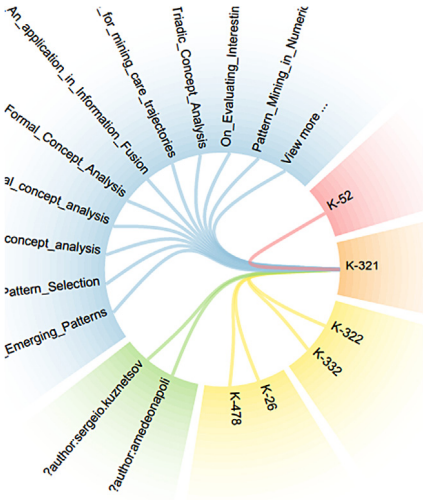


**Fig. 8.** Concept displaying author collaborations.

Finally, it helps in decreasing the navigation space by enabling to focus only on the interesting parts in the RDF-Index and hide the rest of the lattice (see Section 4.2). Using the right-click on a concept allows to mark it as irrelevant and to hide it. Once marked irrelevant the hidden part cannot be accessed unless marked relevant. Further navigation operations implemented in RV-Xplorer are discussed in [3].

## 5. Experimentation

Several experiments have been conducted using publicly available data on a MacBook with a 1.3 GHz Intel Core i5, 4 GB of RAM running OS X Yosemite 10.3. We have used the FCAPS[11] software developed in C++ for dealing with different kinds of pattern structures. FCAPS can build a concept lattice starting from a standard formal context and a pattern concept lattice from RDF data.

The first dataset used for experimentation was DBLP which records bibliographic information about journals, conferences and authors. The triple store used is the RDF data dump for DBLP, which is made available at RDF-HDT[12] [16]. RDF-HDT ("Header, Dictionary, Triples") is a compact data structure for RDF data which provides efficient storage by compressing

11 https://github.com/AlekseyBuzmakov/FCAPS.

12 http://www.rdfhdt.org/datasets/.

**Table 6**
Results of the experiments with different kinds of data.

(a) Real data experiments.

| Dataset | $\lvert G\rvert$ | $\lvert\mathcal{T}\rvert$ | $\mathtt{Leaves}(\mathcal{T})$ | $\lvert\mathcal{L}\rvert$ | $t_{\mathcal{T}}$ | $t_{\mathbb{K}}$ |
|---|---|---|---|---|---|---|
| DBLP | 5293 | 33207 | 33198 | 10134 | 45 s | 21 s |
| Biomedical Data | 63 | 1490 | 933 | 1725582 | 145 s | 162 s |

(b) Numerical data experiments.

| Dataset | $\lvert G\rvert$ | $\lvert\mathcal{T}\rvert$ | $\lvert\mathtt{Leaves}(\mathcal{T})\rvert$ | $\lvert\mathcal{L}\rvert$ | $t_{\mathcal{T}}$ | $t_{\mathbb{K}}$ |
|---|---|---|---|---|---|---|
| BK | 35 | 626 | 10 | 840897 | 37 s | 42 s* |
| LO | 16 | 224 | 26 | 1875 | 0.043 s | 0.088 s |
| NT | 131 | 140 | 6 | 128624 | 3.6 s | 6.8 s |
| PO | 22 | 1236 | 58 | 416837 | 49 s | 57 s* |
| PT | 22 | 4084 | 60 | 452316 | 50 s | 38 s* |
| PW | 94 | 436 | 21 | 1148656 | 60 s | 49 s* |
| PY | 36 | 340 | 53 | 771569 | 46 s | 40 s* |
| QU | 44 | 8212 | 8 | 783013 | 28 s | 30 s* |
| TZ | 31 | 626 | 88 | 650041 | 58 s | 43 s* |
| VY | 52 | 202 | 15 | 202666 | 5.9 s | 11.6 s |

$\lvert G\rvert$ is the number of entities. $\lvert\mathcal{T}\rvert$ is the size of the attribute tree and the number of attributes in the scaled context $\lvert M\rvert$. $\mathtt{Leaves}(\mathcal{T})$ is the number of leaves in the attribute tree. $\lvert\mathcal{L}\rvert$ is the size of the concept lattice for the corresponding data. $t_{\mathcal{T}}$ is the computational time for data represented as a set of antichains in the attribute tree. $t_{\mathbb{K}}$ is the computational time represented by a scaled context, i.e., by a set of filters in the attribute tree.
* Shows that the we are not able to build the whole lattice.

big datasets. The experimentation was based on a subset of papers whose topic was about "machine learning". The titles of the papers were considered as entities and the keywords were taken as descriptions, with ACCS as a reference schema for keywords.

The second dataset belongs to the domain of life sciences, and contains information about drugs, their side effects (SIDER[13]) , and their categories (DrugBank[14] ). The reference schemas related to this second dataset are MedDRA[15] for side effects and MeSH[16] for drug categories.

We compute a concept lattice in two different ways, i.e. by computing the intersection of antichains with RMQ and by scaling (see Appendix). Indeed, the number of leaves in a tree can be much smaller than the number of vertices in this tree. For example, the number of vertices in Fig. 3 is 15, while the number of leaves is only 8. Thus, the direct intersection of antichains can be more efficient than the intersection of antichains by means of a scaling procedure.

The parameters of the datasets and the computational results are shown in Table 6a. For DBLP, the context consists of 5293 entities and 33207 attributes, where we have 33198 leaves in the taxonomy of the attributes, meaning that most of attributes are mutually incomparable. It took 45 s to produce a pattern concept lattice having 10134 concepts directly from the descriptions given by antichains of the reference schema. To produce the same lattice starting from a scaled context the program only takes 21 s.

By contrast, the approach based on pattern structures is better for the biomedical data. Indeed, it takes 145 s, while the computation starting from the scaled contexts takes 162 s. In this case, the dataset contains 1490 attributes with 933 leaves. Thus, the approach based on pattern structures works faster if the number of leaves is significantly smaller than the number of vertices. It is worth noticing that the size of antichains is much smaller than the size of the filters used for scaling, explaining the efficiency in this case. However, when the number of leaves is comparable to the number of vertices, the approach based on pattern structures is slower, because the antichain intersection requires more efforts with pattern structures than with set intersections.

Since the efficiency of the pattern structure approach is higher for the trees with a low number of leaves, we can use this method to increase efficiency of standard FCA for special kinds of contexts. In a context $(G, M, I)$, an attribute $m_1$ can be considered as an ancestor of another attribute $m_2$ if any entity containing the attribute $m_2$ also contains the attribute $m_1$. Accordingly we can construct an attribute tree $\mathcal{T}$ based on this principle and rely on it for computing intersection of antichains. In this case the set of attributes $M$ and the set of vertices of $\mathcal{T}$ are the same and $|M| = |\mathcal{T}|$. The second part of the experiment was based on this observation.

We used numerical data from Bilkent University in the second part of the experiments.[17] The datasets were converted to formal contexts by standard interordinal scaling [22]. The scaled attributes are closely connected, i.e., there is a lot of pairs of attributes $(m_1, m_2)$ such that the set of entities described by $m_1$ is a subset of entities described by $m_2$, i.e., $(m_1)' \subseteq (m_2)'$, allowing to state that $m_1 \leqslant m_2$. Using this property, we built attribute trees from the scaled contexts. These trees have

13 http://sideeffects.embl.de/.
14 http://www.drugbank.ca/.
15 http://meddra.org/.
16 http://www.ncbi.nlm.nih.gov/mesh/.
17 http://funapp.cs.bilkent.edu.tr/DataSets/.

many more vertices than leaves, thus, the approach based on pattern structures should be efficient. The results of the experiments comparing both approaches are shown in Table 6b. It should be noticed that in some cases, when building the lattice with standard FCA, the lattice was so large that the memory was swapping and the computation was stopped. The fact of swapping is shown by a star "*" next to computational time in column $t_{\mathbb{K}}$. This did not happen with pattern structures because computing the similarity of antichains requires less memory to store than the corresponding filters.

Finally this experiment shows that the approach based on pattern structures takes not only less time to compute a pattern concept lattice, but also requires less memory, since there is no memory swapping.

## 6. Related work

There are several studies about exploratory data analysis and information retrieval based on FCA. The associated tools facilitate the interactive exploration of the data at hand. One of the earliest tools is CREDO [9], which displays the concept lattice as a tree-folder and bounds the search space through user constraints. Several other tools were proposed afterward which were based on similar functionalities as CREDO, such as CreChainDo [29] which enables the user to reduce the search space by providing user feedback to the system. Another evolution over CREDO is FooCA [26] which allows the user to interact with formal contexts and with concept lattices. SearchSleuth [12,14] employs the paradigm of "conceptual neighborhood" for displaying clustered web results. In addition to providing the basic functionalities of web clustering engines, SearchSleuth allows the reduction of attribute sets based on support threshold (in a way similar to iceberg lattices [32]). All these tools are built on the basis of web clustering engines [6], which cluster the answers returned by search engines based on the snippets obtained during the search. These tools provide information retrieval and basic exploration capabilities but they lack the support for data analysis as provided by RV-Xplorer.

A series of tools for visualization with concept lattices were built under the supervision of Peter Eklund. In particular, CEM [10] provides visualization of the personal emails using FCA where the objects are emails and attributes are keywords extracted from emails. Insertion and deletion operation over keywords is allowed to the user. In [36], authors present an iPad application *"A Place for Art"* which allows the user to explore an art collection with the help of links generated using FCA. Image Sleuth [15] is a tool for browsing and searching annotated collections of images. The set of objects are the images and their annotated features are the set of attributes. The thumbnails of the images are the extent of the concept. This allows the user to restrict the set of attributes, move to upper and lower neighbors, search for similar objects and similar concepts. The concept lattice is displayed with the help of a tree display just for ensuring user readability. The Hasse diagram is only displayed for the neighborhood of the selected concept. An extension of Image Sleuth is DVD Sleuth [13] which was applied to the information space built from the dynamic DVD collection in amazon.com. All these tools provide a user friendly interface using FCA on the back-end and are built for specific retrieval purposes. By contrast, RV-Xplorer can be used for data exploration, interaction with the analyst, plus data analysis and interactive exploration of a pattern concept lattice generated from complex data such as RDF Data. However, RV-Xplorer is experimented users having knowledge about FCA and pattern structures.

Another FCA-based tool, OntoComP [31], has been developed for knowledge base completion with the help of attribute exploration. The system asks questions to the ontology engineer whose answers are used to complete a knowledge base under study. This is not the objective of RV-Xplorer which allows the exploration of a pattern concept lattice generated from RDF data. Moreover, in the pattern recognition side, NAVIGALA [35] is a system for navigating concept lattices applied to noisy symbol recognition. A common ground to these systems is that they are based on plain FCA which is much less adapted than pattern structures to deal with complex data such as RDF triples.

Sébastien Ferré has also conducted a lot of research work based on FCA on information retrieval, and on the classification of RDF and graph data. Logical Concept Analysis (LCA) was introduced in [20] for dealing with complex data and in particular logical formulas. In [17], the lattice is considered as an exploration space over RDF data. A query language with similar expressivity as SPARQL which is consistent meaning that it is complete and have no dead-ends i.e., every concept is reachable by navigation is also proposed. Following the same line, Sparklis [19] is another system for dealing with RDF data with the help of concept lattice. Sparklis helps a user in exploring a SPARQL endpoint without any prior knowledge to the query language. The user is guided at each step to build questions and answers by interaction. At each step suggestions are given to the user to perform refinement hence allowing exploratory search and feedback. Finally, to complete the overview, [18] introduces another approach for dealing with complex RDF graphs termed as Graph-FCA. By contrast, RV-Xplorer provides means to perform exploratory data analysis and guides the user navigation in a very simple and rough way compared the capabilities of the above systems. The objectives are not exactly the same either, as in the current work, the emphasis is more on knowledge discovery and on the definition of a pattern structure adapted to RDF data.

Another variation of pattern structures able to work on web data is discussed in [11] and called "ontological pattern structures" (OPS). The authors use OPS for analyzing web data and building or completing annotations w.r.t. $\mathcal{EL}$ ontologies. The similarity measure in this pattern structure is based on the convex hull of pairs of classes lying in an ontology (classes are partially ordered). There are some commonalities between the current work and [11], but the definition of RDF-Pattern Structures is quite different and the similarity relies on the intersection of antichains. The purpose is also different and more oriented towards navigation and exploration of RDF data rather than annotation.

Finally, we would like to mention "triadic analysis" (see [23,28]) that is defined for classifying objects involved in ternary relations and thus is able to take into account the three dimensions of RDF triples. Currently, we only consider

$$
\begin{array}{lccccccccccccccccccccccccccccc}
\mathcal{D} & [ & 0 & 1 & 2 & \mathbf{3} & 2 & 3 & 2 & \mathbf{1} & 2 & 3 & 2 & \mathbf{3} & 2 & 1 & 0 & 1 & 2 & 3 & 2 & 1 & 2 & 3 & 2 & 3 & 2 & 3 & 2 & 1 & 0 ] \\
 & & \top & C_{12} & C_{10} & C_1 & C_{10} & C_2 & C_{10} & C_{12} & C_{11} & C_4 & C_{11} & C_5 & C_{11} & C_{12} & \top & C_{15} & C_{13} & C_6 & C_{13} & C_{15} & C_{14} & C_7 & C_{14} & C_8 & C_{14} & C_9 & C_{14} & C_{15} & \top \\
 & & 1 & 2 & 3 & \mathbf{4} & 5 & 6 & 7 & \mathbf{8} & 9 & 10 & 11 & \mathbf{12} & 13 & 14 & 15 & 16 & 17 & 18 & 19 & 20 & 21 & 22 & 23 & 24 & 25 & 26 & 27 & 28 & 29
\end{array}
$$

**Fig. A.9.** The three-dimensional array $\mathcal{D}$ including the depths, the list of the corresponding vertices, and the ranks of the vertices for the tree in Fig. 3.

two dimensions as we split a triple $(s, p, o)$ into the subject $s$ and the predicate–object pair $(p, o)$. Thanks to the definition of RDF-Pattern Structures, we gain in efficiency and computational power what we probably loose in precision w.r.t. RDF triples. However, this is precisely the objective of another current research work to define a suitable pattern structure able to deal with the three dimensions of RDF triples and to benefit from the computational power of pattern structures [30].

## 7. Conclusion

This paper proposes a new approach based on RDF-Pattern Structures for building a pattern concept lattice from a set of RDF triples. This pattern concept lattice provides an index over RDF data by organizing RDF triples with respect to a reference schema and allows the navigation in the lattice and the exploration of RDF data. We show how to define a similarity operation, based on an intersection of antichains, which is applied to RDF triples and which supports an RDF-Pattern Structure.

Experiments have been performed where RDF-Pattern Structure is compared to an approach based on scaling. The comparison shows that the RDF-Pattern Structure is more efficient when the reference schema is deep and has a small number of leaves.

The proposed framework is general and can be applied to any RDF dataset. One of the future directions is to use the complete RDF Schema, i.e. to take into account the subclass relation between classes and the subproperty relation between predicates. This would be a way of effectively dealing with every component of the triples and to take advantage of the semantics related to predicates.

## Appendix A. Using range minimum query for computing LCS

Range Minimum Query (RMQ) [4] is an efficient procedure for finding a minimal element in an array of comparable objects. Considering a set of partially ordered vertices – for simplifying, we will consider that this partial order is a tree – the RMQ procedure operates on a three-dimensional data structure denoted by $\mathcal{D}$ including the depth of every vertex $\mathtt{v_i}$ from the top vertex in the tree, the label of $\mathtt{v_i}$ and the rank of $\mathtt{v_i}$ in the array. The set of partially ordered vertices, i.e. the tree, is traversed using depth-first search and this produces the first dimension of $\mathcal{D}$ recording the list of depths of the vertices. Every time the procedure considers a vertex, say $\mathtt{v_i}$, i.e. the first visit time or a return to the vertex, the depth of $\mathtt{v_i}$ is added at the end of the first dimension of $\mathcal{D}$. The second dimension of $\mathcal{D}$ corresponds to the list of the labels of the vertices. The third dimension of $\mathcal{D}$ includes an index starting from 1 until the whole set of partially ordered vertices has been explored. An example of such an array $\mathcal{D}$ is given in Fig. A.9 showing the three dimensions, the depth array, the list of corresponding vertices, and the ranks of the vertices, for the tree given in Fig. 3.

For example, let us compute the intersection of two antichains of the tree in Fig. 3, say $A = \{C_1, C_5, C_8\}$ and $B = \{C_1, C_7, C_9\}$. Based on the three-dimensional array $\mathcal{D}$ in Fig. A.9, $A$ and $B$ are transformed into the list of indices corresponding to the first occurrence of the considered vertex in the second dimension of $\mathcal{D}$, i.e. $A = \{4, 12, 24\}$ and $B = \{4, 22, 26\}$. Then, a "union" of the two sets is based on a special "ordered merging" alternating an element in each set, i.e. $Z = \{4_A, 4_B, 12_A, 22_B, 24_A, 26_B\}$. Then, RMQ – which given two vertices returns the vertex of minimal depth is computed only for consecutive elements of $Z$, thus $\mathtt{RMQ}(4, 4) = 4$, $\mathtt{RMQ}(4, 12) = 8$ (the minimal depth from position 4 to 12 exists on position 8), $\mathtt{RMQ}(12, 22) = 15$, $\mathtt{RMQ}(22, 24) = 23$, and $\mathtt{RMQ}(24, 26) = 25$. In the resulting set $\{4, 8, 15, 23, 25\}$, we should remove non minimal classes. The classes on position 8, i.e. $C_{12}$, and 15, i.e. $\top$, are not minimal as they are the super-classes of $C_1$ at position 4. The class at positions 23 and 25 is the same, namely $C_{14}$, and we only retain the first occurrence. Finally, RMQ returns the set $\{4, 23\}$ which corresponds to the $\{C_1, C_{14}\}$, i.e. the antichain which is the intersection of $A = \{C_1, C_5, C_8\}$ and $B = \{C_1, C_7, C_9\}$.

The same answer is obtained if RMQ is computed pairwise for each element in the sets $A$ and $B$, but the approach is less efficient. Actually, the number of calls to RMQ in the "consecutive approach" is $O(|A| + |B|)$. By contrast, the number of calls to RMQ in the "pairwise approach" is $O(|A|.|B|)$, where $|A|.|B| \geqslant |A| + |B|$ (see details in [1].

For completing the above example, we give the main lines of the computation for the example detailed in Section 3.3, i.e. $A = \{C_1, C_2, C_7\}$ and $B = \{C_4, C_7, C_8\}$. The two lists of indices are $A = \{4, 6, 22\}$ and $B = \{10, 22, 24\}$ and the ordered union set is $Z = \{4_A, 10_B, 6_A, 22_B, 22_A, 24_B\}$. Computing RMQ is done as follows: $\mathtt{RMQ}(4, 10) = 8$, $\mathtt{RMQ}(10, 6) = 8$, $\mathtt{RMQ}(6, 22) = 15$, $\mathtt{RMQ}(22, 22) = 22$, and $\mathtt{RMQ}(22, 24) = 23$. Finally, the resulting set is $\{8, 22\}$ which corresponds to the $\{C_{12}, C_7\}$.

## Appendix B. Intersection of antichains by scaling

Another approach for computing intersection of antichains is to scale the antichains to the corresponding "filters". A *filter corresponding to an antichain* in a poset is the set of all elements of the poset that are greater than at least one element from the antichain. For example, let us consider the tree in Fig. 3. A filter corresponding to the antichain $A = \{C_1, C_5, C_8\}$ is the set of all subsumers of all elements from the antichain, i.e. $Fil(A) = \{C_1, C_{10}, C_{12}, \top, C_5, C_{11}, C_8, C_{14}, C_{15}\}$. The filter corresponding to the antichain $B = \{C_1, C_7, C_9\}$ is the set $Fil(B) = \{C_1, C_{10}, C_{12}, \top, C_7, C_{14}, C_{15}, C_9\}$. The intersection $Fil(A) \cap Fil(B)$ and the resulting set of minimal elements are $Fil(A) \cap Fil(B) = \{C_1, C_{10}, C_{12}, \top, C_{14}, C_{15}\} = \{C_1, C_{14}\}$.

Considering again the example with $A = \{C_1, C_2, C_7\}$ and $B = \{C_4, C_7, C_8\}$. The filter related to antichain $A$ is $Fil(A) = \{C_1, C_{10}, C_{12}, \top, C_2, C_7, C_{14}, C_{15}\}$ and the one related with $B$ is $Fil(B) = \{C_4, C_{11}, C_{12}, \top, C_7, C_{14}, C_{15}, C_8\}$. Then the intersection of the two filters and the resulting set of minimal elements are $Fil(A) \cap Fil(B) = \{C_{12}, \top, C_7, C_{14}, C_{15}\} = \{C_{12}, C_7\}$.

It should be noticed that the approach based on scaling has a higher complexity. Indeed, the size of a filter is $O(|\mathcal{T}|)$ and, thus, the computational complexity of intersecting two antichains by means of a scaling is $O(|\mathcal{T}|)$. Other details can be found in [1], while the scaling approach is introduced and detailed in [7,8].

## References


[1] M. Alam, A. Buzmakov, A. Napoli, A. Sailanbayev, Revisiting pattern structures for structured attribute sets, in: S.B. Yahia, J. Konecny (Eds.), The Twelth International Conference on Concept Lattices and their Applications, CLA, in: CEUR Workshop Proceedings, vol. 1466, 2015, pp. 241–252.
[2] M. Alam, A. Napoli, Interactive exploration over RDF data using formal concept analysis, in: International Conference on Data Science and Advanced Analytics, DSAA, IEEE, 2015, pp. 1–10.
[3] M. Alam, M. Osmuk, A. Napoli, RV-Xplorer: A way to navigate lattice-based views over RDF graphs, in: S.B. Yahia, J. Konecny (Eds.), The Twelth International Conference on Concept Lattices and their Applications, CLA, in: CEUR Workshop Proceedings, vol. 1466, 2015, pp. 23–34.
[4] M.A. Bender, M. Farach-Colton, G. Pemmasani, S. Skiena, P. Sumazin, Lowest common ancestors in trees and directed acyclic graphs, J. Algorithms 57 (2) (2005) 75–94.
[5] C. Bizer, T. Heath, T. Berners-Lee, Linked data – The story so far, Int. J. Semantic Web Inf. Syst. 5 (3) (2009) 1–22.
[6] C. Carpineto, S. Osinski, G. Romano, D. Weiss, A survey of web clustering engines, ACM Comput. Surv. 41 (3) (2009) 17:1–17:38.
[7] C. Carpineto, G. Romano, A lattice conceptual clustering system and its application to browsing retrieval, Mach. Learn. 24 (2) (1996) 95–122.
[8] C. Carpineto, G. Romano, Concept Data Analysis: Theory and Applications, John Wiley & Sons, Chichester (UK), 2004.
[9] C. Carpineto, G. Romano, Exploiting the potential of concept lattices for information retrieval with CREDO, J. UCS 10 (8) (2004) 985–1013.
[10] R.J. Cole, P.W. Eklund, G. Stumme, CEM-Visualisation and discovery in email, in: D.A. Zighed, H.J. Komorowski, J.M. Zytkow (Eds.), Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery, PKDD, in: Lecture Notes in Computer Science, vol. 1910, Springer, 2000, pp. 367–374.
[11] A. Coulet, F. Domenach, M. Kaytoue, A. Napoli, Using pattern structures for analyzing ontology-based annotations of biomedical data, in: P. Cellier, F. Distel, B. Ganter (Eds.), Proceedings of the 11th International Conference on Formal Concept Analysis, ICFCA, in: Lecture Notes in Computer Science, vol. 7880, Springer, 2013, pp. 76–91.
[12] F. Dau, J. Ducrou, P.W. Eklund, Concept similarity and related categories in searchsleuth, in: P.W. Eklund, O. Haemmerlé (Eds.), Proceedings of the 16th International Conference on Conceptual Structures, ICCS, in: Lecture Notes in Computer Science, vol. 5113, Springer, 2008, pp. 255–268.
[13] J. Ducrou, DVDSleuth: A case study in applied formal concept analysis for navigating web catalogs, in: U. Priss, S. Polovina, R. Hill (Eds.), Proceedings of the 15th International Conference on Conceptual Structures, ICCS, in: Lecture Notes in Computer Science, vol. 4604, Springer, 2007, pp. 496–500.
[14] J. Ducrou, P.W. Eklund, SearchSleuth: The conceptual neighbourhood of an web query, in: P.W. Eklund, J. Diatta, M. Liquière (Eds.), Proceedings of the Fifth International Conference on Concept Lattices and Their Applications, CLA, in: CEUR Workshop Proceedings, vol. 331, 2007.
[15] J. Ducrou, B. Vormbrock, P.W. Eklund, FCA-based browsing and searching of a collection of images, in: H. Schärfe, P. Hitzler, P. Øhrstrøm (Eds.), Proceedings of the 14th International Conference on Conceptual Structures, ICCS, in: Lecture Notes in Computer Science, vol. 4068, Springer, 2006, pp. 203–214.
[16] J.D. Fernández, M.A. Martínez-Prieto, C. Gutiérrez, A. Polleres, M. Arias, Binary RDF representation for publication and exchange (HDT), J. Web Semantics 19 (2013) 22–41.
[17] S. Ferré, Conceptual navigation in RDF graphs with SPARQL-like queries, in: L. Kwuida, B. Sertkaya (Eds.), Proceedings of 8th International Conference on Formal Concept Analysis, ICFCA, in: Lecture Notes in Computer Science, vol. 5986, Springer, 2010, pp. 193–208.
[18] S. Ferré, A proposal for extending formal concept analysis to knowledge graphs, in: J. Baixeries, C. Sacarea, M. Ojeda-Aciego (Eds.), Proceedings of the 13th International Conference on Formal Concept Analysis, ICFCA, in: Lecture Notes in Computer Science, vol. 9113, Springer, 2015, pp. 271–286.
[19] S. Ferré, Sparklis: An expressive query builder for SPARQL endpoints with guidance in natural language, Semantic Web 8 (3) (2017) 405–418.
[20] S. Ferré, O. Ridoux, A logical generalization of formal concept analysis, in: B. Ganter, G.W. Mineau (Eds.), Proceedings of the 8th International Conference on Conceptual Structures, ICCS, in: Lecture Notes in Computer Science, vol. 1867, Springer, 2000, pp. 371–384.
[21] B. Ganter, S.O. Kuznetsov, Pattern structures and their projections, in: H.S. Delugach, G. Stumme (Eds.), Proceedings of the 9th International Conference on Conceptual Structures, ICCS, in: Lecture Notes in Computer Science, vol. 2120, Springer, 2001, pp. 129–142.
[22] B. Ganter, R. Wille, Formal Concept Analysis, Springer, Berlin, 1999.
[23] R. Jäschke, A. Hotho, C. Schmitz, B. Ganter, G. Stumme, Discovering shared conceptualizations in folksonomies, J. Web Semantics 6 (1) (2008) 38–53.
[24] M. Kaytoue, S.O. Kuznetsov, A. Napoli, Revisiting numerical pattern mining with formal concept analysis, in: T Walsh (Ed.), Proceedings of the 22nd International Joint Conference on Artificial Intelligence, IJCAI, IJCAI/AAAI, 2011, pp. 1342–1347.
[25] M. Kaytoue, S.O. Kuznetsov, A. Napoli, S. Duplessis, Mining gene expression data with pattern structures in formal concept analysis, Inf. Sci. 181 (10) (2011) 1989–2001.
[26] B. Koester, Conceptual knowledge retrieval with FooCA: Improving web search engine results with contexts and concept hierarchies, in: P. Perner (Ed.), Proceedings of the 6th Industrial Conference on Data Mining, in: Lecture Notes in Computer Science, vol. 4065, Springer, 2006, pp. 176–190.
[27] M. van Leeuwen, Interactive data exploration using pattern mining, in: A. Holzinger, I. Jurisica (Eds.), Interactive Knowledge Discovery and Data Mining in Biomedical Informatics - State-of-the-Art and Future Challenges, in: Lecture Notes in Computer Science, vol. 8401, Springer, 2014, pp. 169–182.
[28] F. Lehmann, R. Wille, A triadic approach to formal concept analysis, in: G. Ellis, R. Levinson, W. Rich, J.F. Sowa (Eds.), Proceedings of the 3rd International Conference on Conceptual Structures, ICCS, in: Lecture Notes in Computer Science, vol. 954, Springer, 1995, pp. 32–43.
[29] E. Nauer, Y. Toussaint, Crechaindo: an iterative and interactive web information retrieval system based on lattices, Int. J. General Syst. 38 (4) (2009) 363–378.

[30] J. Reynaud, M. Alam, Y. Toussaint, A. Napoli, A proposal for classifying the content of the web of data based on FCA and pattern structures, in: M. Kryszkiewicz, A. Appice, D. Slezak, H. Rybinski, A. Skowron, Z.W. Ras (Eds.), Proceedings of the 23rd International Symposium on Foundations of Intelligent Systems, ISMIS, in: Lecture Notes in Computer Science, vol. 10352, Springer, 2017, pp. 684–694.
[31] B. Sertkaya, OntoComP: A Protégé plugin for completing OWL ontologies, in: Proceedings of the 6th European Semantic Web Conference, ESWC, in: Lecture Notes in Computer Science, vol. 5554, Springer, 2009, pp. 898–902.
[32] G. Stumme, R. Taouil, Y. Bastide, N. Pasquier, L. Lakhal, Computing iceberg concept lattices with Titanic, Data Knowl. Eng. 42 (2) (2002) 189–222.
[33] Y. Sure, S. Bloehdorn, P. Haase, J. Hartmann, D. Oberle, The SWRC ontology – Semantic web for research communities, in: C. Bento, A. Cardoso, G. Dias (Eds.), Proceedings of the 12th Portuguese Conference on Artificial Intelligence, EPIA, in: Lecture Notes in Computer Science, vol. 3808, Springer, 2005, pp. 218–231.
[34] J.W. Tukey, Exploratory Data Analysis, Addison-Wesley, 1977.
[35] M. Visani, K. Bertet, J. Ogier, Navigala: an original symbol classifier based on navigation through a galois lattice, Int. J. Pattern Recogn. Artif. Intel. 25 (4) (2011) 449–473.
[36] T. Wray, P.W. Eklund, K. Kautz, Pathways through information landscapes: Alternative design criteria for digital art collections, in: R. Baskerville, M. Chau (Eds.), Proceedings of the International Conference on Information Systems, ICIS, Association for Information Systems, 2013.