Lattice-Based View Access: A way to Create Views over SPARQL Query for Knowledge Discovery.

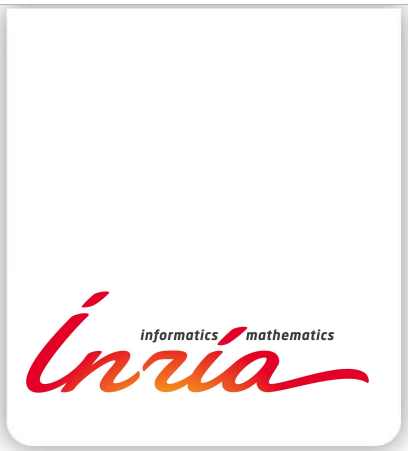
Mehwish Alam, Amedeo Napoli

Lattice-Based View Access: A way to Create Views over SPARQL Query for Knowledge Discovery

Mehwish Alam , Amedeo Napoli

RESEARCH REPORT
N° 8591
August 2014
Project-Teams Orpailleur

ISSN 0249-6399 ISRN INRIA/RR--8591--FR+ENG



Lattice-Based View Access: A way to Create Views over SPARQL Query for Knowledge Discovery

Mehwish Alam *, Amedeo Napoli *

Project-Teams Orpailleur

Research Report n° 8591 — August 2014 — 28 pages

Abstract: The data published in the form of RDF resources is increasing day by day. This mode of data sharing facilitates the exchange of information across the domains. Although it provides easier ways in the use of data, it also gives rise to new challenges. In order to access these data general as well as specific queries can be posed with the help of SPARQL. These queries over semantic web data usually produce list of tuples as answers which may be huge in number or may require further manipulation so that it can be understood and interpreted. Accordingly, this paper introduces a new clause `View By` in the SPARQL query for creating semantic views over the raw SPARQL query answers. This approach namely, Lattice Based View Access (LBVA), is a framework based on Formal Concept Analysis. It provides a classification of the answers of SPARQL queries based on a concept lattice, that can be navigated for retrieving or mining specific patterns in query results w.r.t. user constraints. In this way, the concept lattice can be considered as a materialized view of the data resulting from a SPARQL query.

Key-words: Formal Concept Analysis, SPARQL Query vues, Lattice-Based Views, SPARQL, Classification

* LORIA (CNRS – Inria Nancy Grand Est – Université de Lorraine) Vandoeuvre-lès-Nancy, France

RESEARCH CENTRE
NANCY – GRAND EST

615 rue du Jardin Botanique
CS20101
54603 Villers-lès-Nancy Cedex

Lattice-Based View Access: A way to Create Views over SPARQL Query for Knowledge Discovery

Résumé : Les données publiées sous la forme de ressources RDF augmentent de jour en jour. Ce mode de partage donne la facilite d'échange d'informations entre les domaines. Bien qu'il offre des moyens plus faciles de l'utilisation des données, il donne également lieu à de nouveaux défis. Pour accéder à ces données générales, ainsi que les requêtes spécifiques peuvent être posées à l'aide de SPARQL. Ces requêtes produisent habituellement liste de tuples que sont les réponses. Ces tuples peuvent être énorme en nombre ou peuvent nécessiter une manipulation supplémentaire pour quÕil puisse être compris et interprété. En conséquence, cet article présente une nouvelle clause `VIEW BY` dans la requête SPARQL pour la création de vues sémantiques sur les premiéres réponses d'interrogation SPARQL. Cette approche, Lattice-Based View Access (LBVA), est un cadre basé sur l'analyse formelle de concept. Il fournit une classification des réponses de requêtes SPARQL basé sur un concept de réseau. LBVA peut être navigué pour récupérer ou l'exploitation minière des modèles spécifiques dans les résultats de la requête en ce qui concerne les contraintes de l'utilisateur. De cette façon, le concept de réseau peut être considéré comme une vue matérialisée des données issues d'une requête SPARQL.

Mots-clés : Analyse Formelle de Concept, SPARQL Query Views, Lattice-Based Views, SPARQL, Classification

Contents

1 Introduction

At present, Web has become a potentially large repository of knowledge, which is becoming main stream for querying and extracting useful information. In particular, Linked Open Data (LOD) [1] provides a method for publishing structured data in the form of RDF resources. These RDF resources are interlinked with each other to form a cloud. SPARQL queries are used in order to make these resources usable, i.e., queried. In some cases, queries in natural language against standard search engines can be simple but in some others they are complex and may require integration of data sources. Then the standard search engines will not be able to easily answer these queries, e.g., *Currencies of all G8 countries*. Such a complex query can be formalized as a SPARQL query over data sources present in LOD cloud through SPARQL endpoints for retrieving answers. Moreover, users may sometimes execute queries which generate huge amount of results giving rise to the problem of information overload [2]. A typical example is given by the answers retrieved by search engines, which mix between several meanings of one keyword. In case of huge results, user will have to go through a lot of results to find the interesting ones, which can be overwhelming without any specific navigation tool. Same is the case with the answers obtained by SPARQL queries, which are huge in number and it may be harder for the user to extract the most interesting patterns. This problem of information overload raises new challenges for data access, information retrieval and knowledge discovery w.r.t web querying.

This paper proposes a new approach based on Formal Concept Analysis (FCA [3]). It describes a lattice-based classification of the results obtained by SPARQL queries by introducing a new clause `VIEW BY` in SPARQL query. This framework, called Lattice-Based View Access (LBVA), allows the classification of SPARQL query results into a concept lattice, referred to as a *view*, for data analysis, navigation, knowledge discovery and information retrieval purposes. In the current study we introduce a new clause `VIEW BY` which enhances the functionality of already existing `GROUP BY` clause in SPARQL query by adding sophisticated classification and Knowledge Discovery aspects. Here after, we describe how a lattice-based view can be designed from a SPARQL query. Afterwards, a view is accessed for analysis and interpretation purposes which are totally supported by the concept lattice. In case of large data only a part of the lattice [4] can be considered for the analysis.

In order to make interpretation operations, LBVA also provides faceted browsing over the obtained *view*. Along with browsing and navigation this *view* enables the reduction of the navigation space for the navigational facets in two ways. First, it introduces the stability index for each of the concepts obtained, second, it provides level wise reduction of the navigational facets. In this way, this paper investigates also the capabilities of FCA to deal with Semantic Web (SW) data.

The paper is structured as follows: Section 2 discusses the related work. Section 3 gives a brief overview of Linked Open Data while section 4 introduces a motivating example. Section 5 defines LBVA and gives the overall architecture of the framework. Section 6 discusses some experiments conducted using LBVA. Section 7 shows the visualization. Section 8 explains how the associated software can be used. Finally, Section 9 concludes the paper.

2 Related work

The intuition of classifying results obtained by querying LOD is inspired by web clustering engines [5] such as Carrot2[1], Clusty[2]. The general idea behind web clustering engines is to group the

[1] http://project.carrot2.org/index.html
[2] http://www.clusty.com

results obtained by query posed by the user based on the different meanings of the terms related to a query. The general idea underlying the web clustering engines is to provide a complementary view over the search results obtained by standard search engines such as Google and Yahoo. These web clustering engines extract features from the snippets in HTML/textual documents by preprocessing the snippets using standard Natural Language Processing techniques. These features are further fed to the clustering algorithm where the clusters of the documents are generated. Finally, these clusters are visualised to provide navigational purposes.

However, such systems deal with unstructured textual data on web. By contrast, there are some studies conducted to deal with structured RDF data. One such system is ASPARAGUS [6] which deductively groups the results by taking into account the subsumption hierarchy deduced by knowledge bases. In [2], the authors introduce a clause `Categorize By` to target the problem of managing large amounts of results obtained by conjunctive queries with the help of subsumption hierarchy present in the knowledge base. By contrast, the `View By` clause generates lattice-based views which provide a mathematically well-founded classification based on formal concepts and an associated concept lattice. It also paves way for navigation or information retrieval by traversing the concept lattice through stability based faceted browsing. Moreover, data analysis is performed by allowing the extraction of association rules from the lattice. Such data analysis operations allow discovery of new knowledge. Additionally, unlike `Categorize By`, `View By` can deal with data that has no schema (which is often the case with linked data). Moreover, `View By` has been evaluated over very large set of answers (roughly 100,000 results) obtained over real datasets. In case of larger number of answers, `Categorize By` does not provide any pruning mechanism while this paper describes how the views can be pruned using icerberg lattices.

There exists some research work on the application of FCA to SW data. For example, [7] discusses such an application by focusing on the performance of several algorithms designed to overcome the efficiency issues in the construction of concept lattices based on the data in SW. Moreover, unlike our proposal, this work does not provide any formal or definite transformations from semantic web data to formal context.

3 Preliminaries

3.1 Linked Open Data

Linked Open Data (LOD) [1] is the way of publishing structured data for data sharing purposes. LOD represents RDF data in the form of node-and-arc-labeled directed graphs. This representation helps in the connection between several resources through their schema. RDF allows the specification of the named entities with the help of URIs which are further re-used across the Web. These entities are further grouped into named classes which are related to each other through named relations. The attributes related to each entity is defined with the help of literal values.

Definition 1 (RDF Triple)**.** *Given a set of URIs U, blank nodes B and literals L, an RDF triple is represented as $t = (s, p, o) \in (\mathbf{U} \cup \mathbf{B}) \times \mathbf{U} \times (\mathbf{U} \cup \mathbf{B} \cup \mathbf{L})$, where s is a subject, p is a predicate and o is an object.*

Definition 2 (RDF Graph)**.** *A finite set of RDF triples is called as RDF Graph $\mathcal{G}$ such that $\mathcal{G} = (V, E)$, where V is a set of vertices and E is a set of labeled edges and $\mathcal{G} \in \mathbf{G}$, such that $\mathbf{G} = (\mathbf{U} \cup \mathbf{B}) \times \mathbf{U} \times (\mathbf{U} \cup \mathbf{B} \cup \mathbf{L})$.*

Each pair of vertices connected through a labeled edge keeps the information of a statement. Each statement is represented as $\langle$*subject*, *predicate*, *object*$\rangle$ referred to as an RDF Triple. In $\mathcal{G} = (V, E)$, V includes *subject* and *object* while E includes the *predicate*.

SPARQL[3] is the standard query language for RDF. A SPARQL query is basically composed of three parts, namely *pattern matching*, *solution modifiers* and *output*. The *pattern matching* part takes into account pattern matching features used by the graphs such as union of patterns, filtering of values etc. The *solution modifiers* include the operations which are applied after the output from the pattern matching part is obtained such as limit, distinct (other clauses like Group By, Having are also included in this part). The third part is the *output* which is of many types such as boolean queries returning yes/no answers (ASK keyword), selection of values of the variables matching the patterns, construction of new RDF data (through CONSTRUCT clause) and descriptions of resources (DESCRIBE keyword).

A SPARQL query can also be represented in the form of head ← body, where body includes the RDF graph patterns including conjunction disjunction and other constraints also containing variables. While the head of the query organizes the construction of answer of the query. A SPARQL query `Q` is matched against a graph $\mathcal{G}$ to obtain a set of values bound to the variables in the body. These values are then processed based on the information given in the head of `Q` to produce answers.

In the current work we will focus more on the type of queries whose output performs value selection over the variables matching the patterns. Such queries contain $SELECT$ clause with the projection over set of variables while evaluating a SPARQL query.

Now let us assume that there exists a set of variables V disjoint from U in the above definition of RDF, then $(U \cup V) \times (U \cup V) \times (U \cup V)$ is a graph pattern called a triple pattern. Let us consider a variable $?X \in V$ and $?X = c$ then $c \in U$. Given U and V a mapping μ is a partial function $\mu : V \rightarrow U$. If t is the triple pattern then $\mu(t)$ would be the triple obtained by replacing variables in t with respect to μ.

$[\![.]\!]_G$ takes an expression of patterns and returns a set of mappings. Given a mapping $\mu : V \rightarrow U$ and a set of variables $W \subseteq V$, μ is represented as $\mu_{|W}$, which is described as a mapping such that $dom(\mu_{|W}) = dom(\mu) \cap W$ and $\mu_{|W}(?X) = \mu(?X)$ for every $?X \in dom(\mu) \cap W$. Finally, the SELECT SPARQL query is defined as follows:

Definition 3. *A SPARQL SELECT query is a tuple (W, P), where P is a graph pattern and W is a set of variables such that $W \subseteq var(P)$. The answer of (W, P) over an RDF graph G, denoted by $[\![(W, P)]\!]_G$, is the set of mappings:*

$$[\![(W, P)]\!]_G = \{\mu_{|W} | \mu \in [\![P]\!]_G\}$$

In the above definition $var(P)$ is the set of variables in pattern P where as W represents the variables in the SELECT clause of the SPARQL query. In the rest of the paper we denote W as V to avoid overlap between the attribute values W in many-valued context and variables W in SELECT clause of SPARQL query. The related example is discussed in section 4. Further details on the formalization and foundations of RDF databases are discussed in [8].

3.2 Formal Concept Analysis (FCA)

In this section we introduce the basics of Formal Concept Analysis (FCA) [3] which are necessary for understanding the rest of the paper. FCA is a mathematical framework used for a number of purposes, among which classification and data analysis, information retrieval and knowledge discovery [9]. Let G be a set of objects and M a set of attributes, and $I \subseteq G \times M$ a relation where gIm is true iff object $g \in G$ has attribute $m \in M$. The triple $\mathcal{K} = (G, M, I)$ is called a "formal context". Given $A \subseteq G$ and $B \subseteq M$, two derivation operators, both denoted by $'$, formalize the sharing of attributes for objects, and, in a dual way, the sharing of objects for attributes:

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

$$A' = \{m \in M \mid gIm \ for\ all\ g \in A\} \quad (1)$$
$$B' = \{g \in G \mid gIm \ for\ all\ m \in B\} \quad (2)$$

The two derivation operators $'$ form a *Galois connection* between the powersets $\wp(G)$ and $\wp(M)$. Maximal sets of objects related to maximal set of attributes correspond to closed sets of the composition of both operators $'$ (denoted by $''$). Then a pair (A, B) is a formal concept iff $A' = B$ and $B' = A$. The set A is the "extent" and the set B is the "intent" of the formal concept (A, B). The set $\mathcal{C}_\mathcal{K}$ of all concepts from $\mathcal{K}$ is partially ordered by extent inclusion (or dually intent inclusion), denoted by $\leqslant_\mathcal{K}$ as follows:

$$(A_1, B_1) \leqslant (A_2, B_2) \Leftrightarrow A_1 \subseteq A_2 (\Leftrightarrow B_2 \subseteq B_1) \quad (3)$$

Consequently, $\mathcal{L}_\mathcal{K} = \langle \mathcal{C}_\mathcal{K}, \leqslant_\mathcal{K} \rangle$ forms the *concept lattice* of $\mathcal{K}$. There exist several algorithms [10, 11] to build a concept lattice which also focus on efficiency of building the lattices for large number of objects.

Many-valued Context: In some cases, a many-valued context is obtained instead of a formal context. A many-valued context is defined as follows:

Definition 4. *A many-valued context is denoted by (G, M, W, I) and consists of G the set of objects, M the set of (many-valued) attributes, W the set of attribute values and a ternary relation I between G, M and W i.e., $I \subseteq G \times M \times W$.*

Here $(g, m, w) \in I$ is read as *"the attribute m has the value w for the object g"*. If W has n values than G, M, W, I is called the n-valued context.

Conceptual Scaling: In order to obtain a one-valued binary context from the many-valued context, scaling procedure is adopted. A scale S_m of an attribute m of a many-valued context is a one-valued context (G_m, M_m, I_m) with $m(G) = S_m$ for $m \in M$ and then the new set of attributes is $M_s = \bigcup_{m \in M} S_m$. During plain scaling the object set G remains unchanged, every many-valued attribute m is replaced by the scale attributes of scale S_m.

Definition 5. *If (G, M, W, I) is a many-valued context and S_m, $m \in M$ are scale contexts, then the derived context with respect to plain scaling is the context (G, N, J) with*

$$N := \bigcup_{m \in M} M_m$$

and

$$gJ(m, n) :\Longleftrightarrow m(g) = w \ and \ wI_m n$$

Iceberg Lattices: In order to restrict the number of concepts in some cases iceberg concept lattices can be used [4]. Iceberg concept lattices contain only the top most part of the lattice. Formally, let $B \subseteq M$ and let minimum support, denoted by *minsupp*, be an integer representing a support threshold value. For a given concept (A,B), the support of B is the cardinality of A denoted by $|A|$. Relative support is given by $|A|/|G|$ and belongs to the interval $[0, 1]$. An intent B in concept (A,B) is said to be frequent as soon as $supp(B) = |A|/|G| \geqslant minsupp$. Likewise, a concept is called a frequent concept if its intent is frequent. The set of all frequent concepts of $\mathcal{K}$, for a given threshold, is called an iceberg concept lattice of $\mathcal{K}$.

Stability: The stability [12] of a formal concept indicates how much an intent in the concept depends on the objects in the extent of the concept and vice versa. The intensional stability of a concept measures the likelihood that if a random set of object is removed from the extent of a concept the intent of the concept would change. Similarly, extensional stability of a concept measures the likelihood of change in its extent if a random set of attributes is removed from the intent of the concept.

More formally, the intensional and extensional stability indexes for a concept (A, B) are defined as follows:

$$\sigma_i(A, B) = \frac{|\{C \subseteq A | C' = B\}|}{2^{|A|}}$$

$$\sigma_e(A, B) = \frac{|\{D \subseteq B | D' = A\}|}{2^{|B|}}$$

The intuition underlying these definitions, can be explained as follows: each concept (A, B) has $|A|$ objects in its extent. Total number of subsets of such objects is $2|A|$. Suppose that $C \subseteq A$, is used to construct the lattice whose context had otherwise remained unchanged. Such a lattice would then contain a concept (C, C'), and the intent of this concept would be related to the intent of the original concept by $B \subseteq C'$. Now the intentional stability of the original concept, $\sigma_i(A, B)$, is given by the proportion of object subsets in the given context, such as C, that have the specific property that $C' = B$. The intent of concept (A, B) is therefore stable whenever any one (or more) of these subsets of objects such as C is used to construct a lattice, assuming that the rest of the context remains unchanged. Conversely, if a new lattice is built whose context does not have any single such subset of objects, then the resulting lattice will no longer have a concept whose intent is B. Formally, when $\sigma_i(A, B) = 0$, each and every object in these subsets has at least one attribute that is not in the intent of (A, B).

$\mathcal{DG}$-Basis for Implications: FCA also allows knowledge discovery using association rules. An implication over the attribute set M in a formal context is of the form $B_1 \rightarrow B_2$, where $B_1, B_2 \subseteq M$. The implication holds iff every object in the context with an attribute in B_1 also has all the attributes in B_2. For example, when $(A_1, B_1) \leqslant (A_2, B_2)$ in the lattice, we have that $B_1 \rightarrow B_2$. Duquenne-Guigues ($\mathcal{DG}$) basis for implications [13] is the minimal set of implications equivalent to the set of all valid implications for a formal context $\mathcal{K} = (G, M, I)$. Actually, the $\mathcal{DG}$-basis can be considered as a possible representation of all information lying in the concept lattice.

4 Need for Classifying SPARQL Query Results

In this section we introduce a motivating example focusing on why LOD should be queried and why the SPARQL query results need classification. Consider a query `Q` *all the bands which play different stringed instruments along with their origin.* The RDF graph (i.e., set of triples) related to this query are shown in Table 1. These triples belong to DBpedia, a central hub of LOD which extracts data from Wikipedia info boxes and makes it available in the structured format. The current version of DBpedia 3.9 contains 2.46 billion RDF triples. Out of these triples 470 million are extracted from Wikipedia in English language, 1.98 billion are are in other languages and about 45 million triples link DBpedia to the external data sets.

Let us name this query `Q`, then `Q` can not be answered by standard search engines as it generates a separate list of bands and stringed instruments requiring multiple resources to be integrated. However, `Q` can be answered by SPARQL queries over LOD. For example, let us

	Triples About Stringed Musical Instruments
t1	db:RHCP[4] rdf:type dbo::Band
t2	db:Disturbed rdf:type dbo:Band
t3	db:RHCP dbp:origin db:United States
t4	db:Disturbed dbp:origin db:United States
t5	db:RHCP dbo:bandMember db:Josh_Klinghoffer
t6	db:Disturbed dbo:bandMember db:John_Moyer
t7	db:Disturbed dbo:bandMember db:Dan_Donegan
t8	db:John_Moyer dbo:instrument db:Bass_Guitar
t9	db:Dan_Donegan dbo:instrument db:Electronic_Keyboard
t10	db:Josh_Klinghoffer dbo:intsrument db:Banjo
t11	db:Josh_Klinghoffer dbo:intsrument db:Accordion
t12	db:Banjo dcterms:subject Category:Stringed_Instrument
t13	db:Bass_Guitar dcterms:subject Category:Stringed_Instrument
t14	db:Accordion dcterms:subject Category:Keyboard_Instrument
t15	db:Electronic_Keyboard dcterms:subject Category:Keyboard_Instrument

Table 1: RDF Triples for Musical Instruments from DBpedia. The prefixes dbo:, db:, dbp: stand for dbpedia-owl:, dbpedia: and dbpprop: respectively.

?band	?instrument	?origin
dbpedia:RHCP	dbpedia:Banjo	dbpedia:US
dbpedia:Disturbed	dbpedia:Bass_Guitar	dbpedia:US
dbpedia:The_Solution	dbpedia:Banjo	dbpedia:Sweden

Table 2: Small section of results obtained as an answer to query `Q`.

consider the SPARQL query `Q` over DBpedia[5] shown in Listing 3. This query retrieves all the bands with the instruments played by their band members along with the origin of the band.

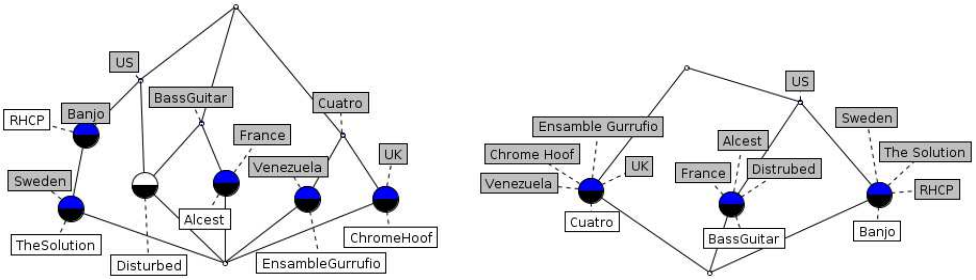
Listing 1: SPARQL Query Q

```
1  SELECT ?band ?instrument ?origin WHERE {
2        ?band rdf:type dbpedia-owl:Band.
3        ?band dbpprop:origin ?origin.
4        ?band dbpedia-owl:bandMember ?member.
5        ?member dbpedia-owl:instrument ?instrument .
6        ?instrument dcterms:subject dbpedia:Category:String_instruments .}
7 GROUP BY ?instrument ?origin
```

Here the head of the query contains `SELECT` clause, $V = \{?\texttt{band}, ?\texttt{instrument}, ?\texttt{origin}\}$ The pattern in line 2 of `Q` matches the triples {t1,t2}, line 3 select the triples {t3,t4} while line 4 and 5 select {t5,t6,t7} and {t8,t9,t10,t11} respectively. However, line 6 reduces the number of triples selected by line 3 and 4 with the help of the constraint that the instruments to be selected should be strictly stringed instruments. Finally, the selected triples by line 4, 5 and 6 are {t5,t6,t8,t7}. Moreover, line 4 and line 5 integrate the answer variable `?band` with the answer variable `?instrument` with the help of the free variable `?member`. The above SPARQL query returns a set of tuples representing a list of bands along with the instruments they play and their origin as an answer. An excerpt of the answers is shown in Table 2.

In case of too many origins `GROUP BY` clause will lead to many small groups which would be hard for the user to observe with respect to origin or instrument, failing in the task of grouping. A classification technique can be used for navigation or interpretation. This classification technique based on FCA serves as a view given a SPARQL query `Q`. Given `Q` and output tuple `t` a *view* is partially ordered set of classes with groups of objects sharing some properties. For example, Figure 1(a) shows a concept lattice for a small part of query answers. Here we can see classes such as the concept which contains all the bands which play `Cuatro`. If the search is more specified

[5] http://dbpedia.org/sparql



(a) Classes of Bands w.r.t. Musical Instruments and Countries, e.g., the concept on the top right corner with the attribute `Cuatro` contains all the bands which play Cuatro.

(b) Classes of Musical Instruments w.r.t Bands and their Origin

Figure 1: Concept Lattices w.r.t Musical Instrument's and Band's Perspective.

then the origin of each of the bands can also be retrieved. It is possible to retrieve bands which play `Cuatro` and are from `UK`, here `Chrome Hoof` is the band which plays `Cuatro` in the current small example. On the other hand, Figure 1(b) shows a concept lattice where musical instruments are classified with respect to bands and their origin, giving a totally different perspective over the same set of answers.

5 Lattice-Based View Access

In this paper, we propose an approach called Lattice-Based View Access for classification of SPARQL query results in the form of a concept lattice referred to as view. This view provides users with analysis, navigation, classification and question/answering capabilities over these results. In the scenario of LOD, the RDF data and query processing procedure can not be controlled, so, in our algorithm we do not process RDF triples (RDF graph) and the SPARQL query. Here we define views over RDF data by processing the set of tuples returned by the SPARQL query.

5.1 SPARQL Queries with Classification Capabilities

The idea of introducing a `VIEW BY` clause is to provide classification of the results and add a knowledge discovery aspect to the results w.r.t the variables appearing in `VIEW BY` clause.

Let `Q` be a SPARQL SELECT query of the form `Q = SELECT ?X ?Y ?Z WHERE {pattern P} VIEW BY ?X` then the set of variables $V = \{?X, ?Y, ?Z\}$ (As W represents set of attribute values in the definition of a many-valued formal context, we represent the variables in select clause as V to avoid confusion). According to definition 3 the answer of the tuple (V, P) is represented as $[\![(\{?X, ?Y, ?Z\}, P)]\!] = \mu_i$ where $i \in \{1, \ldots, k\}$ and k is the number of mappings obtained for the query `Q`. For the sake of simplicity, $\mu_{|W}$ is given as μ. Here, $dom(\mu_i) = \{?X, ?Y, ?Z\}$ which means that $\mu(?X) = X_i$, $\mu(?Y) = Y_i$ and $\mu(?Z) = Z_i$. Finally, a complete set of mappings can be given as $\{\{?X \to X_i, ?Y \to Y_i, ?Z \to Z_i\}\}$.

Now, variables appearing in the `VIEW BY` clause are referred to as object variable[6] and is denoted as Ov such that $Ov \in V$. In the current scenario $Ov = \{?X\}$. The remaining variables are referred to as attribute variables and are denoted as Av where $Av \in V$ such that $Ov \cup Av = V$ and $Ov \cap Av = \varnothing$.

[6]The object here refers to the object in FCA.

	?band	?instrument	?origin
μ_1	RHCP	Banjo	US
μ_2	Disturbed	Bass_Guitar	US
⋮	⋮	⋮	⋮

Table 3: Generated Mappings for SPARQL Query `Q`

Example 1. *Following the example in section 4, an alternate query with the* `VIEW BY` *clause can be given as:*

```
SELECT ?band ?instrument ?origin WHERE {
   P1 ?band rdf:type dbpedia-owl:Band.
   P2 ?band dbpprop:origin ?origin.
   P3 ?band dbpedia-owl:bandMember ?member .
   P4 ?member dbpedia-owl:instrument ?instrument .
   P5 ?instrument dcterms:subject dbpedia[7]:Category:String_instruments .}
VIEW BY ?band
```

Let us continue the sample query discussed in section 4. Here, $V = \{?\texttt{band}, ?\texttt{instrument}, ?\texttt{origin}\}$ *and* $P = (P_1\ AND\ P_2\ AND\ P_3\ AND\ P_4\ AND\ P_5)$ *then the evaluation of the SELECT query* $[\![(\{?band, ?instrument, ?origin\}, P)]\!]$ *will generate the mappings shown in Table 3. Accordingly,* $dom(\mu_i) = \{?band, ?instrument, ?origin\}$. *Here,* $\mu_1(?band) = RHCP$, $\mu_1(?instrument) = Banjo$ *and* $\mu_1(?origin) = US$. *In the current example, we have,* $Ov = \{?band\}$ *because it appears in the* `VIEW BY` *clause and* $Av = \{?instrument, ?origin\}$. *Figure 1a shows the generated view when* $Ov = \{?band\}$ *and in Figure 1b, we have;* $Ov = \{?instrument\}$ *and* $Av = \{?band, ?origin\}$.

5.2 Designing a Formal Context (G, M, W, I)

The results obtained by the query are in the form of set of tuples, which are then organized as a many-valued context. The design of a formal context relies on the selection of the object variable and some of the attribute variables.

Obtaining Object and Attribute Sets (G, M, W)**:** As described previously, we have $Ov = \{?X\}$ then $\mu(?X) = \{X_i\}_{i\in\{1,\ldots,k\}}$, where X_i denote the values obtained for the object variable and the corresponding mapping is given as $\{\{?X \rightarrow X_i\}\}$. Finally, $G = \mu(?X) = \{X_i\}_{i\in\{1,\ldots,k\}}$. Let $Av = \{?Y, ?Z\}$ then $M = Av$ and the attribute values $W = \{\mu(?Y), \mu(?Z)\} = \{\{Y_i\}, \{Z_i\}\}_{i\in\{1,\ldots,k\}}$. The corresponding mapping for attribute variables are $\{\{?Y \rightarrow Y_i, ?Z \rightarrow Z_i\}\}$

Obtaining Ternary Relation (I)**:** Consider an object value $g_i \in G$ and an attribute value $w_i \in W$ then we have $(g_i, \text{“}?Y'', w_i) \in I$ iff $?Y(g_i) = w_i$, i.e., the value of g_i for attribute $?Y$ is w_i, $i \in \{1, \ldots, k\}$ as we have k values for $?Y$.

Example 2. *In the example* $Ov = \{?band\}$, $Av = \{?instrument, ?origin\}$. *The answers obtained by this query are organized into a many-valued context as follows: the distinct values of the object variable ?band are kept as a set of objects, so* $G = \{RHCP, Disturbed, \ldots\}$, *attribute variables*

[7] http://dbpedia.org/resource/

Band	Instrument	Origin
RHCP	Banjo	US
Disturbed	Bass Guitar	US
Alcest	Bass Guitar	France
The Solution	Banjo	Sweden, US
Chrome Hoof	Cuatro	UK
Ensamble Gurrufio	Cuatro	Venezuela

Table 4: Many-Valued Context representing the answer tuple (X_i, Y_i, Z_i).

	Instrument			*Origin*				
Band	Banjo	Bass Guitar	Cuatro	US	Sweden	UK	France	Venezuela
RHCP	×			×				
Disturbed		×		×				
Alcestr		×					×	
The Solution	×			×	×			
Chrome Hoof			×			×		
Ensamble Gurrufio			×					×

Table 5: Formal Context $\mathcal{K}_{DBpedia}$.

provide $M = \{instrument, origin\}$, $W_1 = \{Banjo, BassGuitar, \dots\}$ *and* $W_2 = \{US, UK, France \dots\}$ *in a many-valued context. The obtained many-valued context is shown in Table 4.*

Obtaining Binary Context (G, M, I)**:** Afterwards, a conceptual scaling used for binarizing the many-valued context, in the form of (G, M, I). Finally, we have $G = \{X_i\}_{i \in \{1,\dots,k\}}$, $M = \{Y_i\} \cup \{Z_i\}$ where $i \in \{1, \dots, k\}$ for object variable $Ov = \{?X\}$.

Example: Following the above defined procedure a many-valued context is conceptually scaled to obtain a binary context shown in Table 5. Table 6 and Table 7 show the scales $S_{instrument}$ and S_{origin} for the attributes instrument and origin respectively. The corresponding concept lattice is shown in Figure 1(a).

Algorithm 1 gives the overall view of how a formal context is designed for the output tuples obtained by the SPARQL query `Q`. It takes the set of variables V, set of answer tuples μ (for the sake of simplicity we represent it as μ) and the index of object variable v in V appearing in the `VIEW BY` clause. Line 1 and 2 represent empty context $\mathcal{K}$ and empty set of objects G respectively. Line 3 extracts the index of the object variable in the set of variables V. Line 4 and 5 loop over the vector of answer tuples and extracts the set of objects from each answer tuple, the value at the same index as that of the object variable in W. Remember that each tuple keeps the vector of terms in the same order as that of the answer variable in V. Finally, G is obtained. Line 6 gets the set of attributes by removing the object variable from V and the remaining attribute variables serve as the set of attributes M. Lines 7 extracts and stores attribute values W for each of the variables from answer tuples and line 8 stores ternary relations from the answer tuples. Finally, line 9 and 10 complete and return a many-valued context.

Algorithm 2 details how the attribute values are extracted from the answer tuples. This takes

Instrument	Banjo	Bass Guitar	Cuatro
Banjo	×		
Bass Guitar		×	
Cuatro			×

Table 6: $S_{instrument}$.

Origin	US	Sweden	UK	France	Venezuela
US	×				
Sweden		×			
UK			×		
France				×	
Venezuela					×

Table 7: S_{origin}.

Algorithm 1: Designing Formal Context

procedure: FormalContextFromTuples(V,μ,Ov)
begin
  $\mathcal{K} = \emptyset$;
  $G = \emptyset$;
  $Index \leftarrow$ indexOfObjectVariable Ov in V;
  foreach $tup \in \mu$ **do**
    $G \leftarrow$ GetValueAtIndex($Index$) of tup;
  $M \leftarrow V \setminus Ov$;
  $W \leftarrow$ `AttributeValuesFromTuples(`M,μ,$Index$`)` ;
  $I \leftarrow$ `TernaryRelationFromTuples(`G,M,W,μ`)` ;
  $\mathcal{K} \leftarrow (G,M,W,I)$;
  return $\mathcal{K}$;

answer tuples and the index of the object variable in V as input. It removes the the term in each tuple on the index of the object variable. Intuitively, it will generate another tuple without the terms related to the object variables and the order of these terms would be the same as the order of M. This finally returns the attribute values W for each of the attribute i.e., attribute variable in M.

Algorithm 2: Extracting Attribute Values

procedure: AttributeValuesFromTuples(M,μ,$Index$)
begin
  $W = \emptyset$;
  foreach $tup \in \mu$ **do**
    $W \leftarrow tup \setminus$ TermAtIndex($Index$) of tup;
  return W;

Algorithm 3 takes the set of objects G, set of attributes M, set of attribute values W and the answer tuples as input. Initially I is empty, then the procedure iterates over each of the answer tuples and checks if an object $g \in G$ and an attribute value $w \in W$ occur together in a tuple (if condition). If it is the case, the index of that particular value in the tuple is extracted (line 6) and the attribute variable associated to that value (line 7) is obtained. Finally, it returns the ternary relations for the context.

5.3 Building a Concept Lattice

Once the context is designed, the concept lattice can be built using an FCA algorithm. This step is straight forward as soon as the context is provided. There are some very efficient algorithms that can be used [3, 11]. However, in the current implementation we use AddIntent [11] which is an incremental concept lattice construction algorithm. In case of large data iceberg lattices can be considered [4]. The use of `VIEW BY` clause activates the process of LBVA, which transforms the SPARQL query answers (tuples) to a formal context $\mathcal{K}_{tuples}$ through which a concept lattice is obtained which is referred to as a *Lattice-Based View*. A view on SPARQL query in section 4, i.e, a concept lattice corresponding to Table 5 is shown in Figure 1a. At the end of this step the

Algorithm 3: Extracting Ternary Relation

```
procedure: AttributeValuesFromTuples(G,M,W,μ)
begin
    I = ∅;
    foreach g ∈ G do
        foreach w ∈ W do
            foreach tup ∈ μ do
                if g ∈ tup && w ∈ tup then
                    k ← IndexOf(w) in tup;
                    m ← ValueAtIndex(k) of M;
                    I ← (g,m,w) ;
    return I;
```

concept lattice is built and the interpretation step can be considered.

Algorithm 4: LBVA Algorithm

```
procedure: CreateViews(Q,Ov)
begin
    V ← GetAnswerVariables(Q) ;
    μ ← ExecutingQuery(Q) ;
    K ← FormalContextFromTuples(V,μ,Ov) ;
    L ← AddIntent(K) ;
```

Algorithm 4 gives the overall intuition of the LBVA algorithm. It takes a SAPRQL query `Q`, and object variable v that appears in the `View By` clause. Line 1 extracts all the answer variables from `SELECT` clause, line 2 stores all the answer tuples obtained after executing the SPARQL query. Line 3 stores many-valued context which is then binarized and passed on to Add Intent for building a concept lattice. The algorithm of Add Intent has already been explained in [11]

5.4 Interpretation Operations over a Concept Lattice:

A formal context effectively takes into account the relations by keeping the inherent structure of the relationships present in LOD as object-attribute relation. When we build a concept lattice, each concept keeps a group of terms sharing some attribute (i.e., the relationship with other terms).

Navigation Operation: The obtained concept lattice can be navigated for searching and accessing particular LOD elements through the corresponding concepts within the lattice. It is possible to drill down from general to specific concepts according to some constraints.

For example, in order to search for bands in US playing Banjo, the concept lattice in Figure 1(a) is explored levelwise. First the broader concept contains all the bands from US, `RHCP, The Solution, Disturbed`. Then, the children concepts contain more specific concepts with the instruments `Banjo` and `Bass Guitar`. According to the initial constraint, the attribute concept of `Banjo` can be selected returning two objects namely `RHCP, The Solution`. Next, to check

which instruments are played in music originating from US, another concept lattice can be explored, where objects correspond to instruments shown in Figure 1(b). The results in this case is the set of objects `Bas Guitar, Banjo`.

FCA provides a powerful means for data analysis and knowledge discovery. Iceberg lattices provide the top most part of the lattice filtering out only general concepts. The concept lattice is still explored levelwise depending on a given threshold. Then, only concepts whose extent is sufficiently large are explored, i.e., the support of a concept corresponds to the cardinal of the extent. If further specific concepts are required the support threshold of the iceberg lattices can be lowered and the resulting concept lattice can be explored levelwise.

Knowledge Discovery: Another way of interpreting the data is provided by Duquenne-Guigues basis of implications which takes into account a minimal set of implications which represent all the association rules that can be generated for a given formal context.

For example, $\mathcal{DG}$-basis of implications according to the formal context in Table 5 state that all the bands which play `Banjo` are from `US` (rule: `Banjo` → `US`). Moreover, the rule `Venezuela` → `Cuatro` suggests that all the bands from `Venezuela` play `Cuatro`. This rule states that Cuatro is widely used in the folk music of Venezuela.

6 Experimentation

We have conducted our experiments on real dataset. Our algorithm is implemented in Java using Jena[8] platform and the experiments were conducted on a laptop with 2.60 GHz Intel core i5 processor, 3.7 GB RAM running Ubuntu 12.04. We extracted the information about the movie with their genre and location. The SPARQL query was enhanced with `View By` clause. Both qualitative and quantitative analysis were performed which are discussed here after. The qualitative evaluation shows how the concepts contained in a view obtained by LBVA can be interpreted for obtaining interesting information. Moreover, these views can also give the general overview and state of the underlying RDF data set. It may also help in evaluation of the RDF datasets generated by some automated algorithm and the quality of the information contained.

For each of the queries we tested how our method scales with growing number of results. The number of answers obtained by DBpedia were around 4000 and the answers obtained by YAGO were 100,000. The resulting view kept the classes of movies with respect to genre and location. Section 6.1 and section 6.2 give the qualitative analysis of DBpedia ad YAGO, while section 6.3 gives an idea of the knowledge discovery aspect over the views obtained by the two experiments. The $\mathcal{DG}$-Basis of implications obtained from both the views were compared. However, section 6.4 gives the qualitative evaluation and discusses the sparsity of semantic web data and the scalability of out approach over growing number of answers obtained by SPARQL query.

6.1 DBpedia

DBpedia is currently comprised of a huge amount of RDF triples in many different languages. It reflects the state of Wikipedia [14, 15]. Due to information extraction from crowd-sourced web site, triples present on DBpedia may contain incorrect information. Even if Wikipedia contains correct information, a parser may pick up wrong information [16]. Due to the above described reasons some of the properties may not be used uniformly. In the current experiment, we extracted the information about movies with their genre and location.

[8] https://jena.apache.org/

ID	Supp.	Intent
C#1	17	Hard Rock
C#2	15	Contemporary R&B
C#3	18	Jazz
C#4	732	United States en
C#5	2	United States
C#6	16	USA en
C#7	1225	India en
C#8	6	France

Table 8: Some Concepts from $\mathcal{L}_{DBpedia}$

```
SELECT ?movie ?genre ?country WHERE {
?movie rdf:type dbpedia-owl:Film .
?movie dbpprop:genre ?genre .
?movie dbpprop:country ?country .}
VIEW BY ?movie
```

The obtained concept lattice contained 1395 concepts. Out of which 201 concepts on the first level were evaluated manually for correctness of the information about the movie genre. 60 concepts contained the distinct classes related to the country of the movies. The other 141 concepts kept the genre information about the movie. Out of these 141 concepts 45% of the concepts contained wrong genre information as its intent. In Table 8, first three concepts contain wrong information about the music genre. In such a case, the generated lattice-based views helps in separating music genre from the movie genre and further guide in introducing a new relation such as `soundtrackGenre` and adding new triples to the knowledge base, for example, $dbpedia:The\_Scorpion\_King, dbpedia-owl:soundtrackGenre, dbpedia:Hard\_Rock$.

Moreover, If we observe the obtained view, it can be observed that there are too few movies from countries other than United States and India. For example, C#4,C#5,C#6 and C#7 are the classes for movies from United States and India, where there are 1225 movies from India in DBpedia and 750 movies from United States. In C#5 and C#6, the *en* suffix represents a literal for the country United States and in C#4 the intent united States represents a URI, which is a widely found error in DBpedia. Which means that such classes can be merged into one by taking the union of the extent of C#4,C#5 and C#6. Finally, it can be concluded that the information present on DBpedia still needs to be corrected and completed.

The concept lattice can help in obtaining classes of movies w.r.t countries also. As this approach provides an added value to the already existing `Group By` clause, it is possible to find movies which are made in collaboration with several countries. For example, `The Scorpion King` was made in collaboration with `United States, Germany` and `Belgium`. However, one of the problems encountered for obtaining such results is that the support and stability of such concepts is very low.

6.2 YAGO

The construction of YAGO ontology [17] is based on the extraction of instances and hierarchical information from Wikipedia and Wordnet. In the current experiment, we posed a query to YAGO the `VIEW BY` clause.

```
PREFIX rdf:  http://www.w3.org/1999/02/22-rdf-syntax-ns#
```

Impl. ID	Supp.	Implication
YAGO		
1.	96	wikicategory RKO Pictures films → United States
2.	46	wikicategory Oriya language films → India
3.	64	wikicategory Film remakes → wordnet remake
DBpedia		
4.	3	Historical fiction → United Kingdom@en
5.	3	Adventure fiction, Action fiction → Science fiction

Table 9: Some implications from $\mathcal{DG}$-Basis of Implication (YAGO, DBpedia)

```
PREFIX yago:  http://yago-knowledge.org/resource/
SELECT ?movie ?genre ?location WHERE {
?movie rdf:type yago:wordnet_movie_106613686 .
?movie yago:isLocatedIn ?location .
?movie rdf:type ?genre .  }
VIEW BY ?movie
```

While querying YAGO it was observed that the genre and location information were also given in the YAGO ontology. The classes related to Wordnet were more general than the one extracted from Wikipedia, so we filtered some of the very general Wordnet categories. In the resulting view, the first level of the obtained view kept the groups of movies with respect to their languages. e.g., the movies with genre `Spanish Language Films`. Drilling down in the concept lattice, more specific categories can be found from the location variable such as `Spain`, `Argentina` and `Mexico`. Separate classes obtained for movies based on novels which can be further specialized by the introduction of the country.

Finally with the help of lattice-based views, it can be concluded that YAGO provides a clean categorization of movies by making use of the partially ordered relation between the concepts present in the concept lattice. YAGO also learns instances from Wikipedia and contains many movies from several countries in the world. This observation through views gives the idea about the strong information extraction algorithm as it contains complete information.

6.3 $\mathcal{DG}$-Basis of Implications

$\mathcal{DG}$-Basis of Implications for YAGO and DBpedia were calculated. This section compares the information about movies present in DBpedia with information present in YAGO. The implications were filtered in three ways. Firstly, pruning was performed naively with respect to support threshold. For DBpedia, the number of rules obtained were 64 for a support threshold of 0.11%. However, for YAGO around 200 rules were extracted on support threshold of 0.2%. In order, to make the rules observable, the second type of filtering based on number of elements in the head of the rules was applied. All the implications which contained one item set in the head were selected. However, if there still are large number implications to be observed then a third type of pruning can be applied which involved the selection of implications with different attribute type in head and body, e.g., in rule#1 head contains United States which is of type country and body contains the wikicategory. Such kind of pruning helps in finding attribute-attribute relations.

Table 9 contains some of the implications. Calculating $\mathcal{DG} - Basis$ of implications is actually useful in finding regularities in the SPARQL query answers which can not be discovered from the raw tuples obtained. For example, rule#1 states that `RKO picture films` is an American film production and distribution company as all the movies produced and distributed by them are from United States. Moreover, rule#2 says that all the movies in `Oriya language` are from

No. of Tuples	$\|G\|$	$\|M\|$	No. of Concepts
20%	3657	2198	7885
40%	6783	3328	19019
60%	9830	4012	31264
80%	12960	4533	43510
100%	15272	4895	55357

Table 10: Characteristics of Datasets (YAGO)

No. of Tuples	$\|G\|$	$\|M\|$	No. of Concepts
20%	530	200	291
40%	1091	326	550
60%	1574	411	856
80%	2037	495	1149
100%	2619	584	1395

Table 11: Characteristics of Datasets (DBpedia)

`India`. This actually points to the fact that Oriya is one of many languages that is spoken in India. Rule#3 shows a link between a category from Wikipedia and Wordnet, which clearly says that the `wikicategory` is more specific than the `wordnet` category as `remake` is more general than `Film remakes`.

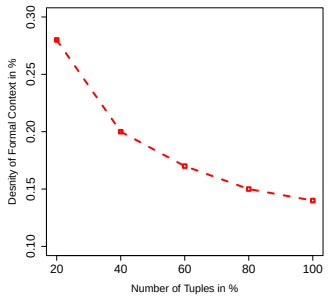
On the other hand, some of the rules obtained from DBpedia are incorrect. For example, rule#4 states the strange fact that all the `historical fiction` movies are from `United Kingdom`. Same is the case with rule#5 which states that all the movies which are `Adventure fiction` and `Action fiction` are also `Science Fiction`, which may not actually be the case. Through the comparison of the $\mathcal{DG}$-Basis for both the datasets it can be observed that the YAGO may be more appropriate for further use by the application development tools and knowledge discovery purposes.

6.4 Evaluation

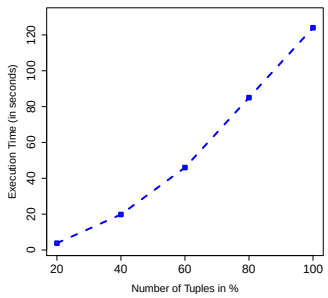
Besides the qualitative evaluation of LBVA, we performed an empirical evaluation. The characteristics of the dataset are shown in Table 10($|G|$ is the number of objects and $|M|$ is the number of attributes in the context). These concepts were pruned with the help of iceberg lattices and stability for qualitative analysis.

The plots for the experimentation are shown in Figure 2. Figure 2(a) shows a comparison between the number of tuples obtained and the density of the formal context. The density of the formal context is the proportion of pairs in I w.r.t the size $G \times M$. It has very low range for both the experiments, i.e., it ranges from 0.14% to 0.28%. This means in particular that the semantic web data is very sparse when considered in a formal context and deviates from the datasets usually considered for FCA (as they are dense). Here we can see that as the number of tuples increases the density of the formal context is decreasing which means that sparsity of the data also increases.

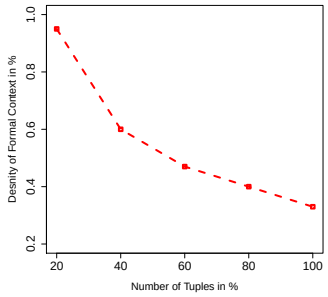
For the above query we tested how our method scales with growing number of results. The number of answers obtained by YAGO were 100,000. Figure 2(b) illustrate the execution time for building the concepts lattice w.r.t the number of tuples obtained. The execution time ranges



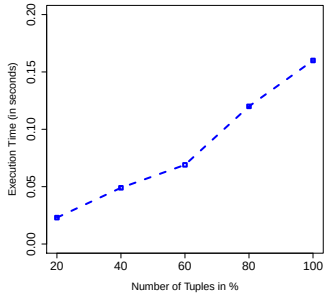
(a) Density of $\mathcal{K}_{YAGO}$



(b) Runtime for Building $\mathcal{L}_{YAGO}$



(a) Density of $\mathcal{K}_{DBpedia}$



(b) Runtime for Building $\mathcal{L}_{DBpedia}$

Figure 2: Experimental Results.

from 20 to 100 seconds, which means that the the concept lattices were built in an efficient way for the test datasets. Which means that large data can be considered for these kinds of experiments. Usually the computation time for building concept lattices depends on the density of the formal context but in the case of semantic web data, as the density is not more than 1%, the computation completely depends on the number of objects obtained which definitely increase with the increase in the number of tuples (see Table 10).

6.5 Application to Biomedical Data

This experiment is performed to evaluate that LBVA is an application independent framework i.e., it can be applied to any dataset. It also discusses how the view and the $\mathcal{DG}$-Basis of implications can be used for accessing useful knowledge. It also gives an overall idea of how LBVA can be used to extract knowledge from biomedical datasets present in the LOD cloud.

It considers a SPARQL query with four variables. The query is run on *Sider Database*[9] for extracting drugs with their side effects and on *Drug Bank*[10] for extracting drugs with their categories and the proteins they target for the largest drug set *Cardiovascular Agents (CVA)*. In this scenario, one objective is to check the validity of prescriptions drug-diseases and second objective is to check the side effects of some drugs. Following is the query:

```
SELECT ?drugname ?sideeffect ?protein ?category
WHERE {
```

[9] http://sideeffects.embl.de/
[10] http://www.drugbank.ca/

ID	Supp.	Intent
C#1	39	se:Jaundice
C#2	24	se:Infection, se:Shock
C#3	13	p08588
C#4	12	p07550, p08588
C#5	15	se:Acute coronary syndrome, cat:Antihypertensive Agents
C#6	12	se:Tachycardia, cat:Anti-Arrhythmia Agents

Table 12: Some Concepts from the Iceberg Lattice

ID	Supp.	Implication
1.	22	se:Stevens-Johnson syndrome → se:Erythema multiforme
2.	8	se:Acute coronary syndrome, se:Arthralgia, se:Infection, se:Pyrexia, se:Tachycardia → se:Cerebrovascular accident
3.	7	cat:Vasoconstrictor Agents → se:Acute coronary syndrome
4.	8	p43700 → se:Vision blurred

Table 13: Some implications from $\mathcal{DG}$-Basis of Implication

```
?drug rdf:type drugbank:drugs .
?drug rdfs:label ?drug_name .
?drug drugbank:sideEffect ?sideeffect .
?drug drugbank:hasCategory ?category .
?drug drugbank:target ?protein .
filter regex(?category, 'Cardiovascular Agents') }
VIEW BY ?drug
```

The obtained result set contain 4-tuples, *i.e., drug name, side effect, protein id (UniProt ID) and category*. In this experiment the total number of tuples obtained is 4843. with 89 drugs belonging to 10 distinct sub categories of CVA. These drugs target 161 proteins and have 72 distinct side effects. The objects count in the formal context is 89 and the attribute count is 243. The concept lattice includes some interesting concepts where sets of drugs constitute the extent while combination of side effects constitute the intents.

In the following, we discuss the possible interpretation of some concepts in collaboration with some domain experts.

Navigation and Information Retrieval: For this an Iceberg lattice with the support threshold of 12% gives 360 concepts (the maximum support is 46). Some of the concepts are shown in Table 12. The prefixes se: for side effect, cat: for category help in differentiating categories from side effects while interpreting. The maximum support 39 was for side effect Jaundice (C#1). This confirms that many drugs which are CVA cause Jaundice. The concept lattice generated also contain several interesting combinations of side effects caused by groups of drugs. For example, Infection along with Shock are caused by 24 drugs. $C\#3$ provides the following explanation: having some abnormality (mutation) in in the protein Beta-1 Adrenergic Receptor forbids the use of these 13 drugs. Moreover, 12 drugs target two proteins (p07550,p08588) (C#4). A subgroup of CVA (C#5) used for hypertension cause Acute Coronary Syndrome. The most eye catching result concern 12 drugs (in C#6) which are used for the treatment of Arrhythmia (irregular heart beat) and cause the same kind of side effect (as its indication) Tachycardia.

$\mathcal{DG}$-Basis of Implication: Some of the rules obtained in $\mathcal{DG}$-Basis of implications for the current formal context are shown in Table 13. The first rule states that all the drugs which have side effect Stevens-johnson Syndrome also have side effect Erythema multiforme (support of

22). This explains that if these drugs are prescribed and they cause `Stevens-johnson Syndrome` then a patient is most likely to have `Erythema multiforme`. Rule#2 depicts that all the drugs which have side effects `Acute coronary syndrome, Arthralgia, Infection, Pyrexia and Tachycardia` have the side effect `Cerebrovascular accident`. `Cerebrovascular accident` is the medical name for stroke. Thus if the drugs are causing the above mentioned side effects then it is most likely that a patient may have this side effect also. Similarly, rule#3 says that all the drugs which are used for the treatment of `Vasoconstriction` (narrowing blood vessels) have the side effect `Acute Coronary Syndrome`. Finally, rule#4 mentions that when the drugs target protein `p43700` they cause `blurred vision`.

All these results show that a careful interpretation of some concepts in the lattice may provide very useful explanation on some observations. These explanations can be reused by human agents and software agents as well.

7 Stability Based Faceted Browsing Over a Concept Lattice

Visualization and interactions are the two milestones for better analysis and search capabilities over a concept lattice. One of the ways is to enable analysis and browsing through faceted search. The objective of this faceted search is to guide the user in the analysis and navigation of the answers obtained by SPARQL queries.

Faceted browsing allows for exploration of the information based on restrictions. Facets are the information elements, the values of which are referred to as features. The facets are often derived by the existing collections which are represented as navigational facets. During navigation, a selection can be performed on a set of items included in features. After selection the restrictions are computed and displayed. There has already been some work related to faceted search based on FCA. One such system, namely Camelis [18], is built based on Logical Concept Analysis, an extension of FCA, which enables browsing and navigation of the documents. Another such study [19] combines the utilities of faceted search and expressive query language to allow navigation of the concept lattice by introducing a new query language LISQL (Logical Information System Query Language).

Lattice-Based Faceted Search: A lattice-based faceted search uses concept lattice as a collection over which navigational facets are defined. It not only guides FCA experts through the interpretation procedure but also helps the novice user to analyse the resulting lattice. In the current work, the concept lattice obtained for the results of each of the SPARQL query serves navigational purposes. We provide navigation over SPARQL query answers through navigational facets. As the views obtained over SPARQL query are large to be observed. Interpretation is a challenge even for an expert. This visualization mechanism makes the interpretation process more flexible. The intent of a concept is represented as a facet values, i.e., a feature. As we move from level 1 to level 2 the number of restrictions increase and more specific concepts are obtained.

Interface Description: Currently, we visualize the concept lattice with the help of the library d3js[11] which is a JavaScript library dealing with the documents containing data. For convenience of explanation in this report we generated concept lattice for the first 500 results of the query in section 6.2. A concept lattice is visualized as a tree where each branch acts can be clicked on and interacted with. Initially, there is only one node, which is the top concept in a concept lattice (see Figure 3). When this node is clicked it opens the first level of the lattice. Figure 4

[11] http://d3js.org/

shows the first level of the concept lattice. On mouseover on any node it shows the extent and intent of the selected concept in the panel on right hand side of the web page. For example, in Figure 4 shows the intent and extent of one of the selected concept. The panel named `Intent` on the top shows the intent `wikicategory 1970 films` and the panel names `Extent` shows the URI of two movies from 1970.

Each of the concepts can be clicked for navigating through the children nodes. Figure 5 shows the three clicked nodes which opens the subtree related to the selected node. On mouse over the intent and extent of the node are shown. The right panel shows that the movies in this class are English Language movies and are from United States. In case of large number of movies the extent panel can be scrolled. Finally the first two levels of the generated concept lattice along with intent and extent of the selected are shown in Figure 6.

Navigating and Interpreting a Concept Lattice: Figure 7 shows first two levels of a concept lattice of 500 answers obtained by the SPARQL query over YAGO (the concept lattice is named $\mathcal{L}_{Yago}$ accordingly) described in section 6.2. The labeled concept C_1, C_2 and C_3 create a branch attached to the top concept and hence is called a sub-tree corresponding to the intent `Documentary Film`. These sub-trees can be further navigated. Here C_1 represents a class of documentaries. After drilling down to the children concepts (going deeper into the branch), we have C_2 and C_3. These two concepts keep the combination of several dimensions of attributes and hence keep more specific information such as movie genre along with their country. C_2 keeps the group of `American Documentary Films` and C_3 keeps the group of `British Documentary Films`. After the concepts are selected the panel shows the links to the page of documentary films.

Restrictions Based on Stability: Now, the question arises why we need to visualize a concept lattice in the form of tree with nodes and edges and not as indented tree. The unique functionality introduced by our faceted search is the use of stability index for each of the concepts. In the concept lattice, the size of each node corresponds to the stability of the concept. Figure 7 shows that the concepts are a mix of smaller and bigger nodes. The smaller nodes correspond to the less stable concepts and the larger nodes correspond to more stable concepts.

Reducing the Navigational Space: Two types of reductions were applied over the navigation space created by the views. The first kind of reduction is with respect to the number of levels considered for navigation which is applied before hand. The second type of reduction uses stability index for each of the concepts. The introduction of intensional stability in the faceted browsing where the intent are referred to as facets gives the ability to reduce the number of concepts to be navigated on the run while browsing the concept lattice. In the example shown in Figure 7, it can be seen that C_3 is more stable than C_2 which means that if the objects from the concept containing documentary films from United States are removed than the concept will disappear. To this end, the C_2 can be ignored in and the navigation can be continued after C_3. Visualization for the experiments discussed in the previous section along with the implementation can be accessed online[12].

[12] `http://webloria.loria.fr/~alammehw/lbva/`

Figure 3: Parent Node.

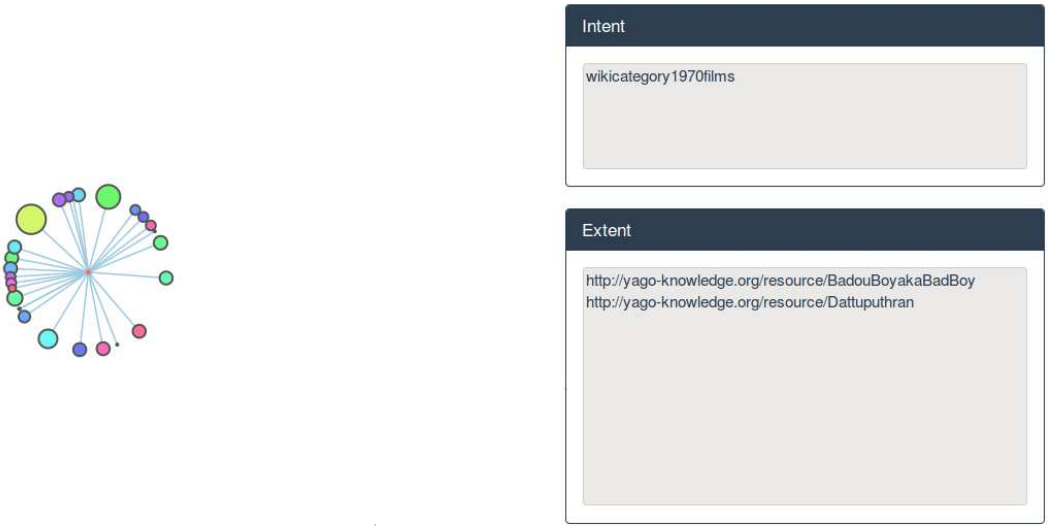


Figure 4: Level 1.

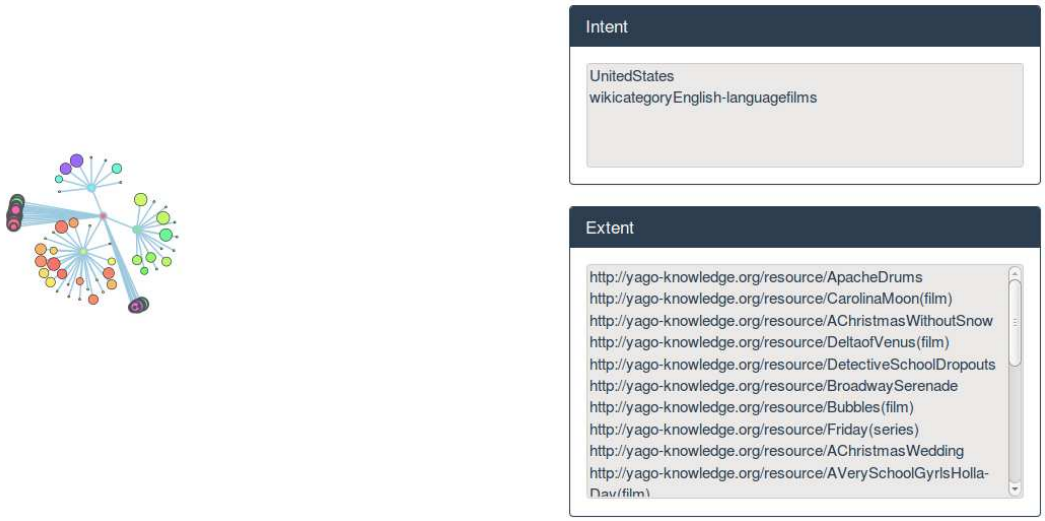


Figure 5: Opening children nodes on Click.

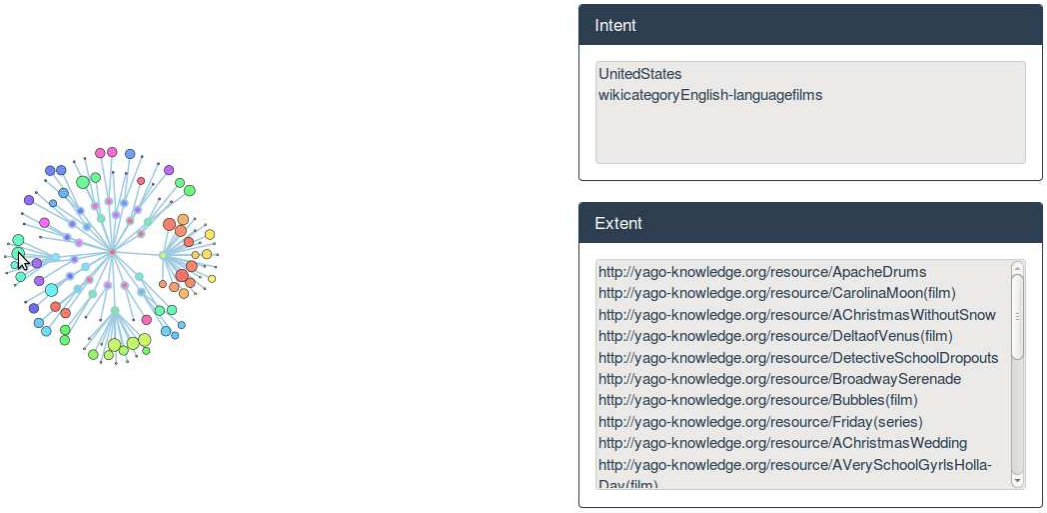


Figure 6: Level 2.

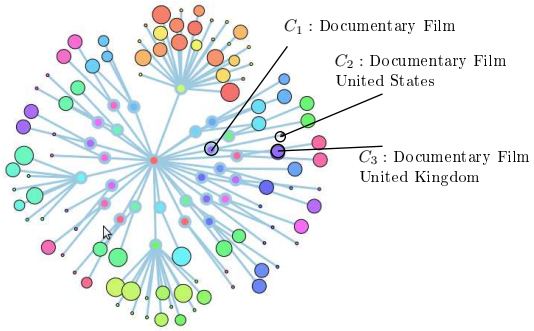


Figure 7: First two levels of the concept lattice $\mathcal{L}_{Yago}$ for the first five hundred results of the query in section 6.2.

8 Lattice-Based View Access (version 0.1) - How To?

The current version (version 0.1) of LBVA is implemented in Java using Jena platform. It can be downloaded from the download link[13]. The software Lattice-Based View Access(*lbva_v_*0.1), takes as input a SPARQL query along with the endpoint of the data resource present in Linked Open Data Cloud and performs transformations according to what is required by a data mining algorithm, i.e, Formal Concept Analysis. This data mining algorithm gives a classification of the SPARQL query answers with the help of concept lattices.

This module can be executed with the help of command line. The attached file is in `.zip` format. In order to run the software, extract the `.zip` folder. Open terminal (command line) and go to the folder containing the extracted folder. Execute `execute.sh` along with the necessary parameters. A sample command for running a SPARQL query over the DBpedia SPARQL endpoint is shown in Listing 2. This command takes five arguments:

- SPARQL endpoint e.g., `http://dbpedia.org/sparql`.
- SPARQL query `sample_query.sparql` with a SELECT and VIEW BY clause.
- Support threshold for the Iceberg concept lattices i.e. `3`.
- Name of the file containing the context (`test_new`).
- The name of the output lattice file (`output_lattice`).

Listing 2: Sample Command

```
./execute.sh http://dbpedia.org/sparql sample_query.sparql
3 test_new output_lattice
```

SPARQL query with VIEW BY clause: The sample query for extracting movie and it's genre information is given in Listing 3. Here the classification is introduced with the help of `VIEW BY` clause. This clause is a solution modifier so it is the last clause to appear in a SPARQL query. Moreover, the variable appearing in the `VIEW BY` clause should be one of the answer variables i.e., one of the variables provided in the `SELECT` clause. Currently this version works with two

[13] `http://webloria.loria.fr/~alammehw/lbva/`

or more than two variables in the `SELECT` clause and only one variable in the `VIEW BY` clause. In future, more than one variable will be considered in the `VIEW BY` clause to deal with relations. Moreover, we are also considering the case of single variable in the `SELECT` clause.

Listing 3: sample_query.sparql

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>
PREFIX dbpprop: <http://dbpedia.org/property/>
SELECT ?movie ?genre WHERE {
?movie rdf:type dbpedia-owl:Film .
?movie dbpprop:genre ?genre .
}
limit 500
VIEW BY ?movie
```

Generating a Concept Lattice: As soon as the command is executed the query is executed against the endpoint and the required results are generated. These results are then classified with the help of *FCA*. In the current version of this software there are two methods implemented for generating concept lattices, i.e., *AddIntent* and *Charm*. The software currently uses *AddIntent* for building concept lattices.

The *Charm algorithm* is already implemented by Coron[14]. Currently the code for Charm is not activated but if the user has knowledge of Java, it can be activated (uncommented) and used. In order to run Coron, it should be downloaded and kept in the same repository where `AddIntent` exists. The third argument of the command shown in Listing 2 i.e., 3 works only with the Charm algorithm.

Interpreting the Generated Files: The name of the output lattice file (`output _ lattice`) defined in the command shown in Listing 2 contains the coded information about the concept lattice generated by *AddIntent*. It means that it keeps the codes for each object and attribute in the extent and intent of the concept. It can be interpreted with the help of the helping files generated automatically by the program in the folder `AddIntent` listing down the object and attribute codes along with their labels (`object_list.txt` and `attribute_list.txt`).

Generating JSON Output: In order to further facilitate the interpretation process, the algorithm provides the faceted browsing over the answers of the SPARQL query. In order to do so, the software also generates the JSON file for the first two levels of the concept lattice which can further be visualized. This file is generated in `AddIntent` folder with the name `lattice.json`. The piece of code responsible for this part is currently deactivated which can be activated (uncommented) by the user and directly used.

9 Conclusion and Discussion

In LBVA, we introduce a classification framework for the set of tuples obtained as a result of SPARQL queries over LOD. We introduce a classification framework based on FCA for organizing a view, i.e., the set of tuples resulting from a SPARQL query. In this way, the view is organized

[14] `http://coron.loria.fr/site/index.php`

as a concept lattice that can be navigated where information retrieval and knowledge discovery can be performed.

For future work, we are interested in working with several *object variables* allowing to deal with more complex relations, with the help of Relational Concept Analysis (RCA)[20]. In addition, here only binary contexts are taken into account. It is possible to go beyond this limitation in using another variation of FCA which is the formalism of pattern structures [21] for dealing with heterogeneous data. Moreover, we also plan to further add provenance information to the SPARQL query answers with the help of pattern structures.

References

[1] C. Bizer, T. Heath, and T. Berners-Lee, "Linked data - the story so far," *Int. J. Semantic Web Inf. Syst.*, vol. 5, no. 3, pp. 1–22, 2009.

[2] C. d'Amato, N. Fanizzi, and A. Lawrynowicz, "Categorize by: Deductive aggregation of semantic web query results," in *ESWC (1)*, ser. Lecture Notes in Computer Science, L. Aroyo, G. Antoniou, E. Hyvönen, A. ten Teije, H. Stuckenschmidt, L. Cabral, and T. Tudorache, Eds., vol. 6088. Springer, 2010, pp. 91–105.

[3] B. Ganter and R. Wille, *Formal Concept Analysis: Mathematical Foundations*. Berlin/Heidelberg: Springer, 1999.

[4] G. Stumme, R. Taouil, Y. Bastide, and L. Lakhal, "Conceptual clustering with iceberg concept lattices," in *Proc. GI-Fachgruppentreffen Maschinelles Lernen (FGML'01)*, R. Klinkenberg, S. RÃ¼ping, A. Fick, N. Henze, C. Herzog, R. Molitor, and O. SchrÃ¶der, Eds., UniversitÃ¤t Dortmund 763, October 2001. [Online]. Available: http://www.kde.cs.uni-kassel.de/stumme/papers/2001/FGML01.pdf

[5] C. Carpineto, S. Osiński, G. Romano, and D. Weiss, "A survey of web clustering engines," *ACM Comput. Surv.*, vol. 41, no. 3, pp. 17:1–17:38, Jul. 2009. [Online]. Available: http://doi.acm.org/10.1145/1541880.1541884

[6] A. Lawrynowicz, J. Potoniec, L. Konieczny, M. Madziar, A. Nowak, and K. T. Pawlak, "Asparagus - a system for automatic sparql query results aggregation using semantics." in *ICCCI (1)*, ser. Lecture Notes in Computer Science, P. Jedrzejowicz, N. T. Nguyen, and K. Hoang, Eds., vol. 6922. Springer, 2011, pp. 304–313. [Online]. Available: http://dblp.uni-trier.de/db/conf/iccci/iccci2011-1.html#LawrynowiczPKMNP11

[7] M. Kirchberg, E. Leonardi, Y. S. Tan, S. Link, R. K. L. Ko, and B.-S. Lee, "Formal concept discovery in semantic web data," in *ICFCA*, ser. Lecture Notes in Computer Science, F. Domenach, D. I. Ignatov, and J. Poelmans, Eds., vol. 7278. Springer, 2012, pp. 164–179.

[8] M. Arenas, C. Gutierrez, and J. Pérez, "Foundations of rdf databases," in *Reasoning Web*, ser. Lecture Notes in Computer Science, S. Tessaris, E. Franconi, T. Eiter, C. Gutierrez, S. Handschuh, M.-C. Rousset, and R. A. Schmidt, Eds., vol. 5689. Springer, 2009, pp. 158–204.

[9] C. Carpineto and G. Romano, *Concept data analysis - theory and applications*. Wiley, 2005.

[10] P. Krajca, J. Outrata, and V. Vychodil, "Advances in algorithms based on cbo," in *CLA*, ser. CEUR Workshop Proceedings, M. Kryszkiewicz and S. A. Obiedkov, Eds., vol. 672. CEUR-WS.org, 2010, pp. 325–337.

[11] D. van der Merwe, S. A. Obiedkov, and D. G. Kourie, "Addintent: A new incremental algorithm for constructing concept lattices," in *ICFCA*, ser. Lecture Notes in Computer Science, P. W. Eklund, Ed., vol. 2961. Springer, 2004, pp. 372–385.

[12] S. O. Kuznetsov, "On stability of a Formal Concept," *Ann. Math. Artif. Intell.*, vol. 49, no. 1-4, pp. 101–115, 2007.

[13] J.-L. Guigues and V. Duquenne, "Familles minimales d'implications informatives rÃ©sultant d'un tableau de donnÃ©es binaires," *MathÃ©matiques et Sciences Humaines*, vol. 95, pp. 5–18, 1986.

[14] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann, "Dbpedia - a crystallization point for the web of data," *J. Web Sem.*, vol. 7, no. 3, pp. 154–165, 2009.

[15] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, and C. Bizer, "DBpedia - a large-scale, multilingual knowledge base extracted from wikipedia," *Semantic Web Journal*, 2014.

[16] D. Wienand and H. Paulheim, "Detecting incorrect numerical data in dbpedia," in *ESWC*, ser. Lecture Notes in Computer Science, V. Presutti, C. d'Amato, F. Gandon, M. d'Aquin, S. Staab, and A. Tordai, Eds., vol. 8465. Springer, 2014, pp. 504–518.

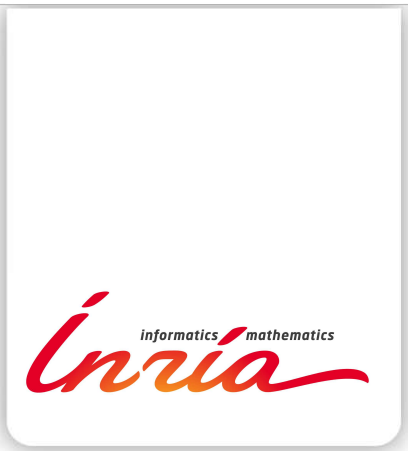
[17] F. M. Suchanek, G. Kasneci, and G. Weikum, "Yago: A core of semantic knowledge," in *Proceedings of the 16th International Conference on World Wide Web*, ser. WWW '07. New York, NY, USA: ACM, 2007, pp. 697–706. [Online]. Available: http://doi.acm.org/10.1145/1242572.1242667

[18] S. Ferré, "Camelis: a logical information system to organise and browse a collection of documents," *Int. J. General Systems*, vol. 38, no. 4, pp. 379–403, 2009.

[19] S. Ferré and A. Hermann, "Reconciling faceted search and query languages for the semantic web," *IJMSO*, vol. 7, no. 1, pp. 37–54, 2012.

[20] M. Rouane-Hacene, M. Huchard, A. Napoli, and P. Valtchev, "Relational Concept Analysis: Mining Concept Lattices From Multi-Relational Data," *Annals of Mathematics and Artificial Intelligence*, vol. 67, no. 1, pp. 81–108, 2013.

[21] B. Ganter and S. O. Kuznetsov, "Pattern structures and their projections," in *ICCS*, ser. Lecture Notes in Computer Science, H. S. Delugach and G. Stumme, Eds., vol. 2120. Springer, 2001, pp. 129–142.



RESEARCH CENTRE
NANCY – GRAND EST

615 rue du Jardin Botanique
CS20101
54603 Villers-lès-Nancy Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399