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 and Amedeo Napoli

LORIA (CNRS – Inria Nancy Grand Est – Université de Lorraine)
BP 239, Vandoeuvre-lès-Nancy, F-54506, France
{mehwish.alam,amedeo.napoli}

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 such as through SPARQL queries. 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 (FCA). 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.

Keywords: Formal Concept Analysis, SPARQL Query Views, Lattice-Based Views, SPARQL, Classification.

Introduction

A considerable amount of Semantic Web (SW) data is already available on the web. Thus many agents are looking for more and more data present in the form of ontologies and RDF triples. Linked Open Data (LOD) [2] is a huge source of RDF resources interlinked with each other to form a cloud. SPARQL queries are used in order to make these data usable by domain experts and software agents. Sometimes queries are executed which may generate huge amount of results giving rise to the problem of *information overload* [4]. A typical example is the answers retrieved by search engines, which may mix between several meanings of one keyword. In case of huge results, many results are navigated to find the interesting links, which may be overwhelming without any navigation tool. Same is the case with the answers obtained by SPARQL queries, which may be huge in number and it may be harder to extract the most interesting patterns. This

problem of information overload raises new challenges for data modeling and analysis and calls for improving data access, information retrieval and knowledge discovery w.r.t web querying.

In order to deal with the problem of *information overload*, this paper proposes a new approach based on Formal Concept Analysis (FCA [5]), which provides a lattice-based classification of the results obtained by SPARQL queries w.r.t user constraints. This new framework, namely Lattice Based View Access (LBVA), allows the classification of SPARQL query results into a concept lattice, referred to as views, for data analysis, knowledge discovery and information retrieval purposes. Based on one SPARQL query several views can be generated from different perspectives. In addition, LBVA allows for navigation over SPARQL query results. Here after, we describe how the *views* (a view corresponds to a concept lattice) can be designed from a SPARQL query and the result which is returned. Moreover, the analysis and the interpretation of the views is totally supported by the concept lattice. In case of large data only a part of the lattice can be considered for the analysis using the technique of iceberg lattices.

The intuition of classifying results obtained by querying LOD is inspired by web clustering engines [3] such as Carrot2[1]. The general idea behind web clustering engines is to group the results obtained by query posed by the user based on the different meanings of the terms related to a query. Such systems deal with unstructured textual data on web. However, there are some studies conducted to deal with structured RDF data. In [4], the authors target the problem of managing large amounts of results obtained by conjunctive queries with the help of subsumption hierarchy present in the knowledge base. On the other hand, lattice-based views provide classification based on the formal concepts and a partially ordered organization of the results. It also opens possibilities for navigation or information retrieval by traversing the concept lattice and for data analysis by allowing the extraction of association rules from the lattice. Additionally, unlike [4], LBVA also deals with data that has no schema (which is often the case with linked data).

The concept lattice provides a well founded structure on which a series of interpretations can be carried out. This framework is general and does not depend on any particular domain and may be used in addition with external resources, e.g. domain knowledge.

The paper is structured as follows: Section 1 gives a brief overview of Linked Open Data and gives the motivating example. Section 2 defines LBVA and gives the overall architecture of the framework. Section 3 briefly described the experimentation setting. Finally, Section 4 concludes the paper.

1 Linked Open Data

Linked Open Data (LOD) [2] is the way of publishing structured data which helps in the connection between several resources through their schema. LOD

[1] http://project.carrot2.org/index.html

represents data in the form of RDF graphs. 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. 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. V includes *subject* and *object* while E includes the *predicate*.

SPARQL[2] is the standard query language for RDF. In the current work we will focus more on the type of queries whose output performs value selection over the variables matching the patterns (queries containing $SELECT$ clause). 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. If a variable $?X \in V$ and $?X = c$ then $c \in U$. Given U, V and a triple pattern t a mapping $\mu(t)$ would be the triple obtained by replacing variables in t with U. $[\![.]\!]_G$ 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 SPARQL SELECT query is defined as follows:

Definition 1. *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:*

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

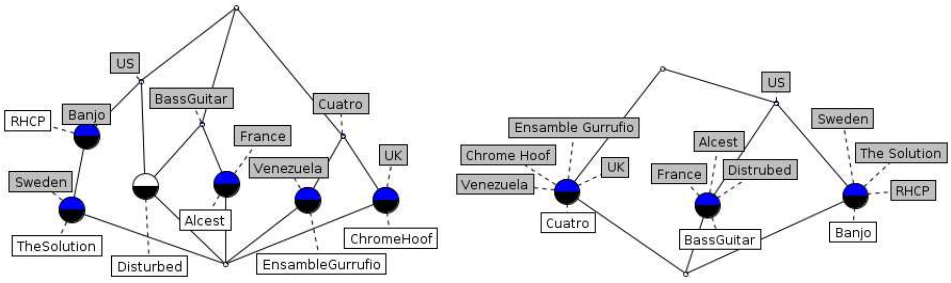
In the above definition $var(P)$ is the set of variables in pattern P and variables W in SELECT clause of SPARQL query[3]. Further details on the formalization and foundations of RDF databases are discussed in [1].

Example 1. Consider a query *all the bands which play different stringed instruments along with their origin.* This example will continue in the rest of this paper. 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 consider the SPARQL query in Listing 1.1 over DBpedia[4]. DBpedia is the central hub of LOD which extracts data from Wikipedia and makes it available in the structured format.

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

[3] In the rest of the paper we denote W as V to avoid overlap between the attribute values W in many-valued context.

[4] 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

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

Listing 1.1: SPARQL Query Q

```
SELECT ?band ?instrument ?origin WHERE {
  ?band rdf:type dbpedia-owl:Band.
  ?band dbpprop:origin ?origin.
  ?band dbpedia-owl:bandMember ?member .
  ?member dbpedia-owl:instrument ?instrument .
  ?instrument dcterms:subject Category:String_instruments .
GROUP BY ?instrument ?origin
```

The above SPARQL query returns a list of bands along with the instruments they play and their origin as an answer. An excerpt of the answers is shown below:

`dbpedia`[5]`:RHCP`[6] `dbpedia:Banjo dbpedia:US`

`dbpedia:Disturbed dbpedia:Bass_Guitar dbpedia:US,`

`dbpedia:The_Solution dbpedia:Banjo dbpedia:Sweden.`

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

[6] Red Hot Chilli Peppers

2 Lattice-Based View Access

In this paper, we propose an approach called Lattice-Based View Access which generates a concept lattice referred to as *view*. This view provides users with classification, navigation and analysis capabilities over these results. In the scenario of LOD, query processing procedure can not be controlled, so, in our algorithm we do not process the SPARQL query. The views are defined over RDF data by processing the set of tuples returned by the SPARQL query.

2.1 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. For example, we have a SPARQL SELECT query `Q = SELECT ?X ?Y ?Z WHERE {pattern P} VIEW BY ?X` then the set of variables $V = \{?X, ?Y, ?Z\}$. According to the definition 1 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`. 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 $\{\{?X \rightarrow X_i, ?Y \rightarrow Y_i, ?Z \rightarrow Z_i\}\}$.

Now, variables appearing in the `VIEW BY` clause are referred to as object variable[7] 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 $Ov \cap Av = \emptyset$.

Example 2. An alternate query for the query in Listing 1.1 with the `VIEW BY` clause can be given as:

```
SELECT ?band ?instrument ?origin WHERE {
   ?band rdf:type dbpedia-owl:Band.
   ?band dbpprop:origin ?origin.
   ?band dbpedia-owl:bandMember ?member .
   ?member dbpedia-owl:instrument ?instrument .
   ?instrument dcterms:subject dbpedia[8]:Category:String_instruments.}
VIEW BY ?band
```

Here, V={`?band, ?instrument, ?origin`} then the evaluation of the SELECT query $[\![(\{?band, ?instrument, ?origin\}, P)]\!]$ will generate the mappings shown in Table 1. 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; $Ov = \{?instrument\}$.

[7] The object here refers to the object in FCA.
[8] http://dbpedia.org/resource/

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

Table 1: Generated Mappings for SPARQL Query `Q`

2.2 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. If $Ov = \{?X\}$ 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 $\{\{?Y \rightarrow Y_i, ?Z \rightarrow Z_i\}\}$ 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 $?X(g_i) = w_i$, i.e., the value of g_i for attribute $?Y$ is w_i, $i \in \{1, \ldots, k\}$ as we have k values for $?Y$.

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 $G = \{X_i\}_{i\in\{1,\ldots,k\}}$, $M = \{Y_i\} \cup \{Z_i\}$ where $i \in \{1, \ldots, k\}$ for object variable $Ov = \{?X\}$.

Example 3. 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 provide $M = \{instrument, origin\}$, $W_1 = \{Banjo, BassGuitar, \ldots\}$ and $W_2 = \{US, UK, France \ldots\}$ in a many-valued context. The obtained many-valued context is shown in Table 2. Following the above defined procedure a many-valued context is conceptually scaled to obtain a binary context shown in Table 3. The corresponding concept lattice is shown in Figure 1(a).

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

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

Table 3: Formal Context $\mathcal{K}_{DBpedia}$.

2.3 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. In the current implementation we use AddIntent [8] which is an incremental concept lattice construction algorithm. In case of large data iceberg lattices can be considered [6]. The use of `VIEW BY` clause activates the process of LBVA, which transforms the SPARQL query answers (tuples) to a formal context $\mathcal{K}_{answers}$ through which a concept lattice is obtained which is referred to as a *Lattice-Based View*. A view on SPARQL query in section 1, i.e, a concept lattice corresponding to Table 3 is shown in Figure 1a. At the end of this step the concept lattice is built and the interpretation step can be considered.

2.4 Interpretation Operations over a Concept Lattice

Navigation Operation and Knowledge Discovery: The obtained concept lattice can be navigated for searching and accessing particular LOD elements. 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 `Bass 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.

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 3 state that all the bands which play `Banjo` are from `US` (rule: `Banjo` $\rightarrow$ `US`). Moreover, the rule `Venezuela` $\rightarrow$ `Cuatro` suggests that all the bands from `Venezuela` play `Cuatro`. This rule states that Cuatro is widely used in the folk music of Venezuela.

3 Experimentation

Several experiments were conducted on real datasets. These datasets include DBpedia, Yago [7], which is a knowledge base automatically extracted from

Wikipedia (infoboxes, categories), Wordnet and Geonames. The experiment was also tested on the biomedical data such as Sider[9] and Drugbank[10]. Sider keeps the information about the medicines along with their side effects. Drugbank keeps the detailed information about the drugs such as drug category and target proteins. These experiments provide qualitative and quantitative evaluation to our approach. These experiments are not discussed in the current paper due to lack of space. However, the software, a detailed technical report along with the visualization of the SPARQL query views can be accessed online[11].

4 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 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 variable* allowing to deal with more complex relations, with the help of Relational Concept Analysis (RCA). 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.

References

1. Marcelo Arenas, Claudio Gutierrez, and Jorge Pérez. Foundations of rdf databases. In *Reasoning Web*, pages 158–204, 2009.
2. Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data - the story so far. *Int. J. Semantic Web Inf. Syst.*, 5(3):1–22, 2009.
3. Claudio Carpineto, Stanislaw Osiński, Giovanni Romano, and Dawid Weiss. A survey of web clustering engines. *ACM Comput. Surv.*, 41(3):17:1–17:38, July 2009.
4. Claudia d'Amato, Nicola Fanizzi, and Agnieszka Lawrynowicz. Categorize by: Deductive aggregation of semantic web query results. In *ESWC (1)*, 2010.
5. Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer, Berlin/Heidelberg, 1999.
6. G. Stumme, R. Taouil, Y. Bastide, and L. Lakhal. Conceptual clustering with iceberg concept lattices. In *Proc. GI-Fachgruppentreffen Maschinelles Lernen (FGML'01)*, 2001.
7. Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: A core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 697–706, New York, NY, USA, 2007. ACM.
8. Dean van der Merwe, Sergei A. Obiedkov, and Derrick G. Kourie. Addintent: A new incremental algorithm for constructing concept lattices. In *ICFCA*, 2004.

[9] `http://sideeffects.embl.de/`
[10] `http://www.drugbank.ca/`
[11] `http://webloria.loria.fr/~alammehw/lbva/`