Defining Views with Formal Concept Analysis for Understanding SPARQL Query Results

Mehwish Alam, Amedeo Napoli

Defining Views with Formal Concept Analysis for Understanding SPARQL Query Results

Mehwish Alam[2,3] and Amedeo Napoli[1,2]

[1] CNRS, LORIA, UMR 7503, Vandoeuvre-lès-Nancy, F-54506, France
[2] Inria, Villers-lès-Nancy, F-54600, France
[3] Université de Lorraine, LORIA, UMR 7503, Vandoeuvre-lès-Nancy, F-54506, France
{mehwish.alam,amedeo.napoli@loria.fr}

Abstract. SPARQL queries over semantic web data usually produce list of tuples as answers that may be hard to understand and interpret. Accordingly, this paper focuses on Lattice-Based View Access (LBVA), a framework based on FCA. This framework 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. 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.

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) [2] 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 to use but sometimes 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 [5]. 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 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.

Accordingly, this paper proposes a new approach based on Formal Concept Analysis (FCA [7])s. 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. This 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 [10] can be considered for the analysis. In this way, this paper investigates also the capabilities of FCA to deal with semantic web data.

The intuition of classifying results obtained by SPARQL queries is inspired by web clustering engines [3] such as Carrot2[4]. 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. By contrast, there are some studies conducted to deal with structured RDF data. In [5], 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. Moreover, it also paves way for navigation or information retrieval by traversing the concept lattice and for data analysis 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 iceberg lattices.

The paper is structured as follows: Section 2 introduces a motivating example. Section 3 gives a brief introduction of the state of the art while Section 4 defines LBVA and gives the overall architecture of the framework. Section 5 discusses some experiments conducted using LBVA. Finally, Section 6 concludes the paper.

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

2 Motivation

In this section we introduce a motivating example focusing on why LOD should be queried and why the SPARQL query results need classification. This scenario will continue in the rest of the paper. Let us consider that a query `Q` searching for *museums where the exhibition of some famous artists is taking place along with the location of the museum*. Here, we do not discuss the interface aspects and we will assume that SPARQL queries are provided. A standard query engine is not adequate for answering such kind of questions and a direct query over LOD will give better results. One of the ways to obtain such an information is to query LOD through its SPARQL endpoint. This query will generate a huge amount of results, which will need further manual work to group the interesting links.

3 Background

3.1 Linked Open Data

Linked Open Data (LOD) [2] is the way of publishing structured data in the form of RDF graphs. Given a set of URIs $\mathbf{U}$, blank nodes $\mathbf{B}$ and literals $\mathbf{L}$, 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[5] is the standard query language for RDF. In the current work we will focus on the queries containing `SELECT` clause. Let us assume that there exists a set of variables $\mathbf{V}$ disjoint from $\mathbf{U}$ in the above definition of RDF, then $(\mathbf{U} \cup \mathbf{V}) \times (\mathbf{U} \cup \mathbf{V}) \times (\mathbf{U} \cup \mathbf{V})$ 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:*

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

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

In Definition 1, $var(P)$ is the set of variables in pattern P and W is the set of variables in SELECT clause. Here, P includes the triple patterns containing variables. This triple pattern is then evaluated against the RDF Graph G given as $[\![P]\!]_G$. It returns a set of mappings with respect to the variables in $var(P)$. Finally a projection over μ is done w.r.t. the variables in W. The projected set of mappings obtained as represented as $\mu_{|W}$. Further details on the formalization and foundations of RDF databases are discussed in [1].

Example 1. Continuing the scenario in section 2, following is the SPARQL query:

```
1 SELECT ?museum ?country ?artist WHERE {
   2 ?museum rdf:type dbpedia-owl:Museum .
   3 ?museum dbpedia-owl:location ?city .
   4 ?city dbpedia-owl:country ?country .
   5 ?painting dbpedia-owl:museum ?museum .
   6 ?painting dbpprop:artist ?artist}
7 GROUP BY ?country ?artist
```

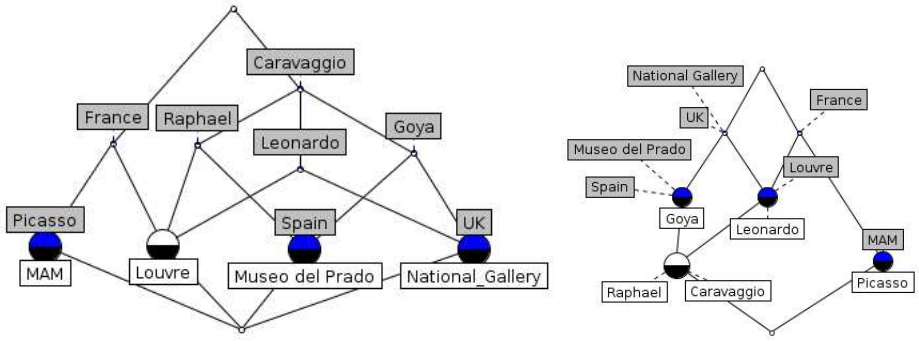
This query retrieves the list of museums along with the artists whose work is exhibited in a museum along with the location of a museum. `Lines 5 and 6` retrieve information about the artists whose work is displayed in some museum. More precisely, the page containing the information on a museum (`?museum`) is connected to the page of the artists (`?artist`) through a page on the work of artist (`?painting`) displayed in the museum. In order to integrate these three resources, two predicates were used `dbpedia-owl:museum` and `dbpprop:artist`. An excerpt of the answers obtained by `Group by` clause is shown below:

```
Pablo_Picasso     Musee_d'Art_Moderne France
Leonardo_Da_Vinci Musee_du_Louvre     France
Raphael           Museo_del_Prado     Spain
```

The problem encountered while browsing such an answer is that there are too many statements to navigate through. Even after using the `GROUP BY` clause the answers are not organized in any ordered structure. By contrast, the clause `VIEW BY` activates the LBVA framework, where the user will obtain a classification of the statements as a concept lattice where statements are partially ordered (see Figure 1a). To obtain the museums in `UK` displaying the work of `Goya`, all the museums displaying the work of `Goya` can be retrieved and then the specific concept containing `Goya and UK` is obtained by navigation. The answer obtained is `National Gallery` in the example.

3.2 Formal Concept Analysis (FCA)

As the basics of Formal Concept Analysis (FCA) [7] are well known, we only introduce some of the concepts which are necessary to understand this paper. FCA is a mathematical framework used for a number of purposes, among which classification and data analysis, information retrieval and knowledge discovery [4]. In some cases we obtain a huge number of concepts. In order to restrict the



(a) Classes of Museums w.r.t Artists and Countries, e.g., the concept on the top left corner with the attribute `France` contains all the French Museums, i.e., Musee du Louvre (Louvre) and Musee d'Art Moderne (MAM). (`VIEW BY ?museum`)

(b) Classes of Artists w.r.t Museums and Countries. (`VIEW BY ?artist`)

Fig. 1: Lattice-Based Views w.r.t Museum's and Artist's Perspective .

number of concepts, iceberg concept lattices can be used [10]. Iceberg concept lattices contain only the top most part of the lattice. Along with iceberg lattices a stability index [9] is also used for filtering the concepts. The stability index shows how much the concept intent depends on particular objects of the extent.

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 $(A_1, B_1) \leq (A_2, B_2)$ in the lattice, we have that $B_1 \rightarrow B_2$. Duquenne-Guigues ($\mathcal{DG}$) basis for implications [8] 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 contains all information lying in the concept lattice.

4 Lattice-Based View Access

4.1 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 query of the form `Q = SELECT ?X ?Y ?Z WHERE {pattern P} VIEW BY ?X` then the set of variables $V = \{?X, ?Y, ?Z\}$ [6]. 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`. For the sake of simplicity, $\mu_{|W}$ is given as μ. Here, $dom(\mu_i) = \{?X, ?Y, ?Z\}$ which means that $\mu(?X) = X_i$,

[6] 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.

$\mu(?Y) = Y_i$ and $\mu(?Z) = Z_i$. Finally, a complete set of mappings can be given as $\{\{?X \rightarrow X_i, ?Y \rightarrow Y_i, ?Z \rightarrow Z_i\}\}$.

The variable appearing in the `VIEW BY` clause is 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 $Ov \cap Av = \emptyset$, so, $Av = \{?Y, ?Z\}$.

Example 2. Following the example in section 2, an alternate query with the `VIEW BY` clause can be given as:

```
SELECT ?museum ?artist ?country WHERE {
   ?museum rdf:type dbpedia-owl:Museum .
   ?museum dbpedia-owl:location ?city .
   ?city dbpedia-owl:country ?country .
   ?painting dbpedia-owl:museum ?museum .
   ?painting dbpprop:artist ?artist}
VIEW BY ?museum
```

	?museum	?artist	?country
μ_1	Musee_d'Art_Moderne	Pablo_Picasso	France
μ_2	Museo_del_Prado	Raphael	Spain
⋮	⋮	⋮	⋮

Table 1: Generated Mappings for SPARQL Query `Q`

Here, V=`{?museum, ?artist, ?country}` and P is the conjunction of patterns in the `WHERE` clause then the evaluation of $[\![(\{?museum, ?artist, ?country\}, P)]\!]$ will generate the mappings shown in Table 1. Accordingly, $dom(\mu_i) = \{?museum, ?artist, ?country\}$. Here, $\mu_1(?museum) = Musee\_d'Art\_Moderne$, $\mu_1(?artist) = Pablo\_Picasso$ and $\mu_1(?country) = France$. We have Ov = {?`museum`} because it appears in the `VIEW BY` clause and Av = {?`artist`, ?`country`}. Figure 1a shows the generated view when Ov = {?`museum`} and in Figure 1b, we have; Ov = {?`artist`} and Av = {?`museum`, ?`country`}.

4.2 Designing a Formal Context of Answer Tuples

The results obtained by the query are in the form of set of tuples, which are then organized as a many-valued context.

Obtaining a Many-Valued Context (G, M, W, I)*:* 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 $\{\{?Y \rightarrow Y_i, ?Z \rightarrow Z_i\}\}$.

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

In order to obtain a ternary relation, let us 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 $?Y$.

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 $Ov = \{?X\}$. The binary context obtained after applying the above transformations to the SPARQL query answers w.r.t to object variable is called the *formal context of answer tuples* and is denoted by $\mathcal{K}_{tuple}$.

Example 3. In the example $Ov = \{?\texttt{museum}\}$, $Av = \{?\texttt{artist}, ?\texttt{country}\}$. The answers obtained by this query are organized into a many-valued context as follows: the distinct values of the object variable `?museum` are kept as a set of objects, so $G = \{MuseeduLouvre,\ MuseodelPrado, \ldots\}$, attribute variables provide $M = \{artist, country\}$, $W_1 = \{Raphael,\ LeonardoDaVinci, \ldots\}$ and $W_2 = \{France, Spain, UK, \ldots\}$ in a many-valued context. The obtained many-valued context is shown in Table 2. Finally, the obtained many-valued context is conceptually scaled to obtain a binary context shown in Table 3.

Museum	*Artist*	*Country*
Musee du Louvre	{Raphael, Leonardo Da Vinci, Caravaggio}	{France}
Musee d'Art Moderne	{Pablo Picasso}	{France}
Museo del Prado	{Raphael, Caravaggio, Francisco Goya}	{Spain}
National Gallery	{Leonardo Da Vinci, Caravaggio, Francisco Goya}	{UK}

Table 2: Many-Valued Context (Museum).

	Artist					*Country*		
Museum	Raphael	Da Vinci	Picasso	Caravaggio	Goya	France	Spain	UK
Musee du Louvre	×	×		×		×		
Musee d'Art Moderne			×			×		
Museo del Prado	×			×	×		×	
National Gallery		×		×	×			×

Table 3: Formal Context $\mathcal{K}_{tuple}$ w.r.t $?museum$.

The organization of the concept lattice is depending on the choice of object variable and the attribute variables. Then, to group the artists w.r.t the museums where their work is displayed and the location of the museums, the object variable would be `?artist` and the attribute variables will be `?museum` and `?country`. Then, the scaling can be performed for obtaining a formal context. In order to complete the set of attribute, domain knowledge can also be taken into account, such as the the ontology related to the type of artists or museums. This domain knowledge can be added with the help of pattern structures, an approach linked to FCA, on top of many-valued context without having to perform scaling. For the sake of simplicity, we do not discuss it in this paper.

4.3 Building a Concept Lattice

Once the context is designed, the concept lattice can be built using an FCA algorithm.There are some very efficient algorithms that can be used [7, 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 [10]. 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 2, i.e, a concept lattice corresponding to Table 3 is shown in Figure 1a.

4.4 Interpretation Operations over Lattice-Based Views

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). This concept lattice can be navigated for searching and accessing particular LOD elements through the corresponding concepts within the lattice. It can be drilled down from general to specific concepts or rolled up to obtain the general ones which can be further interpreted by the domain experts. For example, in order to search for the museums where there is an exhibition of the paintings of `Caravaggio`, the concept lattice in Figure 1(a) is explored levelwise. It can be seen that the paintings of `Caravaggio` are displayed in `Musee du Louvre, Museo del Prado and National Gallery`. Now it can be further filtered by country, i.e., look for `French` museums displaying `Caravaggio`. The same lattice can be drilled down and `Musee du Louvre` as an answer can be retrieved. Next, to check the museums located in `France` and `Spain`, the roll up operation from the French Museums to the general concept containing all the museums with Caravaggio's painting can be applied and then the drill down operation to Museums in `France` or `Spain` displaying `Caravaggio` can be performed. The answer obtained will be `Musee du Louvre` and `Museo del Prado`.

A different perspective on the same set of answers can also be retrieved, meaning that the group of artists w.r.t museums and country. For selecting French museums according to the artists they display, the object variable will be $Ov = \{?artist\}$ and attribute variables will be $Av = \{?museum, ?country\}$. The lattice obtained in this case will be from Artist's perspective (see Figure 1b). Now, it is possible to retrieve `Musee du Louvre` and `Musee d'Art Moderne`, which are the French museums and to obtain a specific French museum displaying the work of `Leonardo Da Vinci` a specific concept can be selected which gives the answer `Musee du Louvre`.

FCA provides a powerful means for data analysis and knowledge discovery. `VIEW BY` can be seen as a clause that engulfs the original SPARQL query and enhances it's capabilities by providing *views* which can be reduced using iceberg concept lattices. 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: Among the means provided by FCA for knowledge discovery, the Duquenne-Guigues basis of implications takes into account a minimal set of implications which represent all the implications (i.e., association rules with confidence 1) that can be obtained by accessing the *view* i.e., a concept lattice. For example, implications according to Figure 1(a) state that all the museums in the current context which display `Leonardo Da Vinci` also display `Caravaggio` (`rule: Leonardo Da Vinci` $\rightarrow$ `Caravaggio`). It also says that only the museums which display the work of `Caravaggio` display the work of `Leonardo Da Vinci` Such a rule can be interesting if the museums which display the work of both `Leonardo Da Vinci` and `Caravaggio` are to be retrieved. The rule `Goya, Raphael, Caravaggio` $\rightarrow$ `Spain` suggests that there exists a museum which have works of `Goya, Raphael, Caravaggio` only in Spain, more precisely `Museo Del Prado`. (These rules are generated from only the part of SPARQL query answers shown as a context in Table 3).

5 Experimentation

The experiments were conducted 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 using SPARQL query enhanced with `VIEW BY` clause. The experiment shows that even though the background knowledge (ontological information) was not extracted the views reveal the hidden hierarchical information contained in the SPARQL query answers and can be navigated accordingly. Moreover, it also shows that useful knowledge is extracted from the answers through the the views using $\mathcal{DG}-$Basis of implications. We also performed quantitative analysis where we discussed about the sparsity of the semantic web data. We also tested how our method scales with growing number of results. The number of answers obtained by YAGO were 100,000. The resulting view kept the classes of movies with respect to genre and location.

5.1 YAGO

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

```
PREFIX rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
PREFIX yago: http://yago-knowledge.org/resource/
SELECT ?movie ?genre ?location WHERE {
```

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

```
   ?movie rdf:type yago:wordnet_movie_106613686 .
   ?movie yago:isLocatedIn ?location .
   ?movie rdf:type ?genre . }
VIEW BY ?movie
```

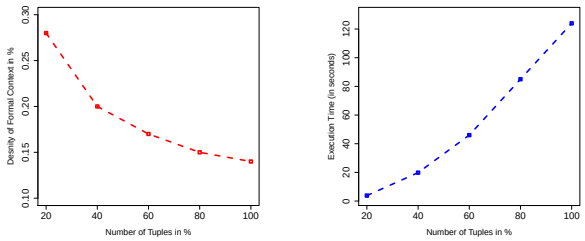
While querying YAGO it was observed that the genre and location information was also given in the ontology. The first level of the obtained view over the SPARQL query results over YAGO kept the groups of movies with respect to their languages. e.g., the movies with genre `Spanish Language Films`. However, as we further drill down in the concept lattice we get more specific categories which include the values from the location variable such as `Spain, Argentina` and `Mexico`. There were separate classes obtained for movies based on novels which were then further specialized by the introduction of the country attribute as we drill down the concept lattice. Finally with the help of lattice-based views, it can be concluded that the answers obtained by querying YAGO provides a clean categorization of movies by making use of the partially ordered relation between the concepts present in the concept lattice.

$\mathcal{DG}$-Basis of Implications: $\mathcal{DG}$-Basis of Implications for YAGO were calculated. The implications were filtered in three ways. Firstly, pruning was performed naively with respect to support threshold. 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 body of the rules was applied. All the implications which contained one item set in the body were selected. However, if there still are large number of 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 4 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 `India`. This actually points to the fact that Oriya is one of many languages that is spoken in India. This rule also tells that Oriya language is only 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`.

Impl. ID	Supp.	Implication
1.	96	`wikicategory RKO Pictures films` → `United States`
2.	46	`wikicategory Oriya language films` → `India`
3.	64	`wikicategory Film remakes` → `wordnet remake`

Table 4: Some implications from $\mathcal{DG}$-Basis of Implication (YAGO)



(a) Density of $\mathcal{K}_{YAGO}$ (b) Runtime for Building $\mathcal{L}_{YAGO}$

Fig. 2: Experimental Results.

5.2 Evaluation

Besides the qualitative evaluation of LBVA, we performed an empirical evaluation. The characteristics of the dataset are shown in Table 5. 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.

We also 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 concept lattice w.r.t the number of tuples obtained. The execution time ranges from 20 to 100 seconds, it means that the the concept lattices were built in an efficient way and 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 5).

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 5: Characteristics of Datasets (YAGO)

6 Conclusion and Discussion

In LBVA, we introduce a classification framework based on FCA for the set of tuples obtained as a result of SPARQL queries over LOD. In this way, a view

is organized as a concept lattice built through the use of `VIEW BY` clause that can be navigated where information retrieval and knowledge discovery can be performed. Several experiments show that LBVA is rather tractable and can be applied to large data.

For future work, we are interested in extending the `VIEW BY` clause by including the available background knowledge of the resources using the formalism of pattern structures [6]. Moreover, we intend to use implications for completing the background knowledge. We also intend to use pattern structures with a graph description for each considered object, where the graph is the set of all triples accessible w.r.t reference object.

References

1. Marcelo Arenas, Claudio Gutierrez, and Jorge Pérez. Foundations of rdf databases. In Sergio Tessaris, Enrico Franconi, Thomas Eiter, Claudio Gutierrez, Siegfried Handschuh, Marie-Christine Rousset, and Renate A. Schmidt, editors, *Reasoning Web*, volume 5689 of *Lecture Notes in Computer Science*, pages 158–204. Springer, 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, 2009.
4. Claudio Carpineto and Giovanni Romano. *Concept data analysis - theory and applications.* Wiley, 2005.
5. Claudia d'Amato, Nicola Fanizzi, and Agnieszka Lawrynowicz. Categorize by: Deductive aggregation of semantic web query results. In Lora Aroyo, Grigoris Antoniou, Eero Hyvönen, Annette ten Teije, Heiner Stuckenschmidt, Liliana Cabral, and Tania Tudorache, editors, *ESWC (1)*, volume 6088 of *Lecture Notes in Computer Science*, pages 91–105. Springer, 2010.
6. Bernhard Ganter and Sergei O. Kuznetsov. Pattern structures and their projections. In Harry S. Delugach and Gerd Stumme, editors, *ICCS*, volume 2120 of *Lecture Notes in Computer Science*, pages 129–142. Springer, 2001.
7. Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis: Mathematical Foundations.* Springer, Berlin/Heidelberg, 1999.
8. J.-L. Guigues and V. Duquenne. Familles minimales d'implications informatives résultant d'un tableau de données binaires. *Mathématiques et Sciences Humaines*, 95:5–18, 1986.
9. Sergei O. Kuznetsov. On stability of a Formal Concept. *Ann. Math. Artif. Intell.*, 49(1-4):101–115, 2007.
10. Gerd Stumme, Rafik Taouil, Yves Bastide, and Lotfi Lakhal. Conceptual clustering with iceberg concept lattices. In R. Klinkenberg, S. Rüping, A. Fick, N. Henze, C. Herzog, R. Molitor, and O. Schröder, editors, *Proc. GI-Fachgruppentreffen Maschinelles Lernen (FGML'01)*, Universität Dortmund 763, October 2001.
11. Dean van der Merwe, Sergei A. Obiedkov, and Derrick G. Kourie. Addintent: A new incremental algorithm for constructing concept lattices. In Peter W. Eklund, editor, *ICFCA*, Lecture Notes in Computer Science, pages 372–385. Springer, 2004.