Completing RDF Data in Linked Open Data Cloud using Formal Concept Analysis

Mehwish Alam, Aleksey Buzmakov, Víctor Codocedo, and Amedeo Napoli

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

Abstract. In the last years there has been a huge increase in the amount of information published as Linked Open Data (LOD). Its popularization and quick growth has led to challenging aspects regarding quality assessment and data exploration of the RDF triples that shape the LOD cloud. Particularly, we are interested in two important aspects of these challenges, namely how to deal with different data schemas and how to process heterogeneous resource descriptions. In this work we propose a novel technique to overcome these issues by the implementation of a knowledge discovery process based on Formal Concept Analysis which automatically detects incomplete information. We propose a structure for the organization of the RDF triples through a concept lattice of graph patterns, providing a powerful navigation mechanism while allowing for the discovery of implication rules that can be used to complete the LOD cloud. Such an organization also provides a way to help users in formulating specific SPARQL queries.

Keywords: Lattice-Based Organization, RDF Data, Formal Concept Analysis, Pattern Structures.

1 Introduction

World Wide Web has tried to overcome the barrier of data sharing by converging data publication into Linked Open Data (LOD) [1]. The LOD cloud stores data in the form of *subject-predicate-object* triples in RDF[1], a standard framework for information description of web resources. While LOD has made the task of data sharing easier and more accessible, it also led to the proliferation of knowledge bases and information sources rendering the task of knowledge discovery more difficult. At the same time, assessing the quality of the data available has become a major challenge given the complexity of LOD, i.e. several repositories with different descriptions with redundant resources, etc. Consider the example of DBPedia [2], one of the main (if not the principal) knowledge bases supporting the LOD cloud. DBpedia, in its version 3.9, contains over 470 million triples extracted in an automated fashion from Wikipedia. In this scenario, it is easy

[1] Resource Description Framework

to observe the challenges of quality assessment and data discovery. In the first aspect, it is not only a matter of designing the right schemas of descriptions, but also of discovering what are those currently in use and how can they be improved and completed. For the second aspect, it is necessary to provide a robust structure that is able to support the user in its search for information.

In this paper, we propose a novel approach based on Formal Concept Analysis [7] (FCA) to construct a data structure (more specifically, a concept lattice) to support both, data completion and exploration. Our model considers the analysis of heterogeneous data, i.e. mining patterns of entities described by numerical or binary data through an implementation of the pattern structure framework [6], an extension of FCA to process complex data descriptions. We show how, through the use of a concept lattice, we are able to discover *implication rules* which can be used for quality assessment and data completion in the knowledge base. Furthermore, we describe how the concept lattice is able to provide a SPARQL query map which allows the user to navigate and discover triples based on a query refinement/expansion process, i.e. adding or removing restrictions from a query.

The remainder of this article is structured as follows: Section 3 gives a brief introduction to the theoretical background necessary to sustain the rest of the paper. Section 4 describes the approach used for data completion in the DBpedia knowledge base. Section 5 describes the process of data discovery in RDF graphs using a concept lattice. Section 6 provides experimental results in four datasets created from DBpedia and a brief discussion over our findings. Finally, Section 7 concludes the paper offering some perspectives over our approach.

2 Related Work

There have already been few studies discussing the incompletion of RDF data. The *type* information plays an important role in the knowledge bases. In [11] authors use type inference mechanism which takes into account the links between instances to finally infer the types of these instances assuming that some relations occur only with certain types. Moreover, there are some studies which focus on the correction of numerical data present in DBpedia [13] using *Outlier detection method*, which detects those facts which deviate from the other members of the sample. By contrast, this paper focuses on completing the RDF data with the help of association rule mining. It also provides a lattice-based navigation over the RDF triples which further guides the user in building specific SPARQL queries. In [5], authors use FCA for obtaining a hierarchy of meaningful sets of entities which is then used for generating more specific questions. However, the questions generated in this case are in the form of natural language. Another such study [12] uses multi-context for browsing the knowledge base. The authors represent concept lattice as a materialized view over a knowledge base which can further be navigated and browsed. In our approach we not only provide lattice navigation we consider each concept in the concept lattice as a blue print of one SPARQL query and suggest it to the user.

3 Preliminaries

3.1 Linked Open Data

Linked Open Data (LOD) [1] has become the *de facto* standard for publishing data on-line. LOD uses standard semantic web technologies such as RDF which can further be linked to other data sources published in the LOD cloud. The idea of RDF is based on storing data in the form of directed node-arc-labeled graphs. An RDF graph consists of a set of vertices V and a set of labeled edges E. 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*. Finally all the assertions present in an RDF graph are given as follows $A \subseteq V \times V \times E$. An alternate representation of an RDF triple is given as $U \cup B \times U \cup B \times U \cup B \cup L$, where U is the URI reference, B refers to blank node and L is the literal. The first component $(U \cup B)$ is the *subject* in an RDF triple, the second component $(U \cup B)$ is the *predicate* while the third component $(U \cup B \cup L)$ is the *object*. In the current study we do not take into account blank nodes (B). So, an RDF triple is represented as $U \times U \times U \cup L$. In the rest of the paper, we use also use *relation or property* for the predicate. These data can then be queried and accessed through SPARQL[2], which is a standard query language for RDF data. SPARQL is based on matching graph patterns against RDF graphs called Basic Graph Pattern (BGP). A graph pattern is a set of triple patterns which allows variables. Let us consider that a user is looking for information about the Lamborghini Sports cars in DBpedia. The related SPARQL query for extracting information about the Sports Cars more specifically Lamborghini is given in Listing 1.1.

```
SELECT ?s WHERE {
?s dc:subject dbpc:Sports_cars .
?s dc:subject dbpc:Lamborghini_vehicles .
?s rdf:type dbo:Automobile .
?s dbo:manufacturer dbp:Lamborghini }
```

Listing 1.1: SPARQL for the formal concept in Table 2. Prefixes are defined in Table 1.

3.2 Formal Concept Analysis

Formal Concept Analysis (FCA) is a mathematical framework used for classification, data analysis, information retrieval and knowledge discovery [3] among other tasks. The basics of FCA can be found in [7], but in the following we recall

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

Predicates		Objects	
Index	URI	Index	URI
A	dc:subject	a	dbpc:Sport_Cars
		b	dbpc:Lamborghini_vehicles
B	dbp:manufacturer	c	dbp:Lamborghini
C	rdf:type	d	dbo:Automobile
D	dbp:assembly	e	dbp:Italy
E	dbo:layout	f	dbp:Four-wheel_drive
		g	dbp:Front-engine

Namespaces:	
dc:	`http://purl.org/dc/terms/`
dbo:	`http://dbpedia.org/ontology/`
rdf:	`http://www.w3.org/1999/02/22-rdf-syntax-ns\#`
dbp:	`http://dbpedia.org/resource/`
dbpc:	`http://dbpedia.org/resource/Category:`

Table 1: Index of pairs predicate-object and namespaces.

some important definitions for the remainder of this. Let G be a set of entities[3], M a set of attributes, and $I \subseteq G \times M$ an incidence relation indicating by gIm that the entity $g \in G$ has the attribute $m \in M$. The triple $\mathcal{K} = (G, M, I)$ is called a "formal context" in which for $A \subseteq G$ and $B \subseteq M$, two derivation operators (both denoted by $'$) can be formalized as follows:

$$A' = \{m \in M \mid gIm \ \forall \ g \in A\}$$
$$B' = \{g \in G \mid gIm \ \forall \ m \in B\}.$$

Where A' denotes the set of attributes shared by the entities in A and B' denotes the set of entities having *all* the attributes in B. Both derivation operators $'$ form a *Galois connection*[4] between the powersets $\wp(G)$ and $\wp(M)$. We say that the pair (A, B) is a formal concept of $\mathcal{K}$ iff $A' = B$ and $B' = A$, where A is called the "extent" and B is called the "intent" of (A, B). Given two formal concepts (A_1, B_1) and (A_2, B_2), a hierarchical relation is defined between them as follows:

$$(A_1, B_1) \leqslant_{\mathcal{K}} (A_2, B_2) \iff A_1 \subseteq A_2 (or B_2 \subseteq B_1)$$

In such a case, (A_1, B_1) is called a subconcept of (A_2, B_2) ((A_2, B_2) a superconcept (A_1, B_1)). The set of all formal concepts in $\mathcal{K}$ (denoted by $\langle \mathcal{C}_{\mathcal{K}}$) together with the order $\leqslant_{\mathcal{K}}$ forms the concept lattice denoted by $\mathcal{L}_{\mathcal{K}}$. For a given object $g \in G$, the formal concept (g'', g') is called the *object concept* of g and it is denoted by γg (the attribute concept of $m \in M$ is $\mu m = (m', m'')$). A concept lattice for which only the object and attributes concepts are labelled with their

[3] Actually, in FCA elements in G are called "objects". In this article we call them "entities" to avoid confusions with "objects" as defined for RDF triples.

[4] A Galois connection is based on a dual adjunction between partially ordered sets.

respective objects and attributes is called a concept lattice in reduced notation. Several algorithms have been proposed to calculate the set of all formal concepts from a formal context and its associated concept lattice. A comprehensive study and comparison of them can be found in [9].

For example, consider the formal context in Table 2 build from 35 RDF triples manually extracted from DBpedia about nine different Lamborghini cars represented in rows. Columns represent the pairs predicate-object and each cross in the table represents the relation *subject* has the pair *predicate-attribute*. Figure 1 depicts the concept lattice in reduced notation calculated for this formal context and contains 12 formal concepts. Consider the first five cars (*subjects*) in the table for which the maximal set of attributes they share is given by the first five *predicate-object* pairs. Actually, they conform a formal concept depicted by the gray cells in Table 2 and labeled as "Islero, 400GT" in Figure 1.

	A-a	A-b	B-c	C-d	D-e	D-f	E-g
Reventon	×	×	×	×	×	×	
Countach	×	×	×	×	×		
350GT	×	×	×	×			×
400GT	×	×	×	×			
Islero	×	×	×	×			
Veneno	×	×					
Aventador Roadster	×	×					
Estoque			×	×		×	×
Gallardo			×	×	×		

Table 2: Formal context manually built from DBPedia. It contains 9 entities (subjects) and 7 attributes composed by pairs property-object indexed in Table 1. Each cross (×) in the context corresponds to a triple subject-predicate-object.

Given a concept lattice, rules can be extracted from the intents of concepts which are comparable. For two sets $X, Y \subseteq M$, a rule has the form $X \implies Y$ where X and Y are subsets of attributes. A rule $X \implies Y$ has a support given by the proportion of entities having the set of attributes $X \cap Y$ w.r.t. the whole set of entities, and a confidence which is the proportion of entities having the same set of attributes $X \cap Y$ but w.r.t. X. A rule $X \implies Y$ of confidence 1 verifies that $X \cap Y = X$ or that $X \subseteq Y$ and is called an *implication rule*. Otherwise, the rule confidence is less than 1 and is called an *association rule*.

4 Linked data and heterogeneous pattern structures

Consider the example in Table 2. All the cars are actually of the brand "Lamborghini", however none of the attributes reflects this fact. It can be observed that 7 out of 9 entities are "Automobiles" and "manufactured by Lamborghini"

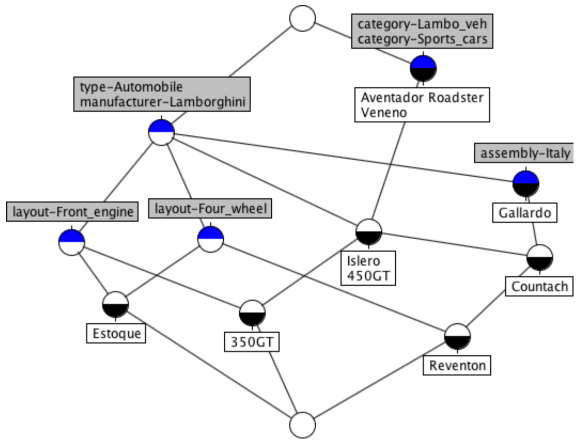

Fig. 1: Concept lattice for formal context in Table 2.

while also 7 out of 9 cars are categorized as "Sports cars" (sic) and "Lamborghini vehicles". Two of the entities in the dataset are not *typed* as "Automobiles" even though they are. Moreover, two entities which correspond to Lamborghini manufactured automobiles are not categorized as "Lamborghini vehicles".

Under this evidence, it is useful to understand the semantics behind these relations. DBpedia "rdf:type" objects come from the infobox template used to describe a wikipage. Infoboxes are not fully controlled, but its creation is more difficult than other descriptors (e.g. category names) and its use is encouraged by the Wikipedia guide [5] and the DBpedia project has created infobox mappings that allow to find the *type* assigned to a given wikipage [2]. Knowing this, we can make the fairly sure inference (supposing there are no data errors) that in *?s rdf:type dbo:Automobile*,*?s* is a car. Furthermore, we can assume that when *?s dbo:manufactured dbp:Lamborghini* also occurs, the subject *?s* corresponds to a Lamborghini car. In such a case, "rdf:type" and "dbo:manufactured" correspond to a level of "semantics" to which a query can be understood as a matching of meanings entailing a deeper level of description. For example, the DBpedia schema indicates that "Automobiles" have engine, layout, wheelbase, etc.

In DBpedia, the objects related to the predicate "dc:subject" come from Wikipedia categories, a semi-controlled hierarchical folksonomy which users may or may not use. "dc:subject" objects are usually more noisy in DBpedia than "rdf:type" objects as in Wikipedia, categories are used as markers of the content of a given wikipage. Categories are rich in semantics, sadly they are only understandable by human readers and not by semantic systems. Consider for example the subject "Lamborghini vehicles". The semantics implied for a human reader are that elements in this category are "automobiles" and furthermore, they were manufactured by "Lamborghini". For a semantic system, the subject

[5] `http://en.wikipedia.org/wiki/Help:Infobox#Adding_an_infobox_to_an_article`

is just a symbol and the category “Lamborghini engines” is as different from “Lamborghini vehicles” as the category “Semantic Web”. From the perspective of RDF, “dc:subject” objects correspond to a kind of “syntactic expression” to which a query can be only understood as a matching of symbols (even when a certain level of inference is allowed through the categorical hierarchy).

Our main concern in this case lies in two aspects. First, are we able to complete data using logic inferences? For example, can we *complete* the information in the dataset by indicating that the entities “Estoque” and “Gallardo” should be categorized as “Lamborghini vehicles” and “Sport cars”? Second, are we able to *complete* the descriptions of a given type? For example, DBpedia do not specifies that an “Automobile” should have a “manufacturer”. For both of these tasks we use implication and association rules. Consider those provided in Table 3.

Rule	Confidence	Support	Meaning
d $\implies$ c	100%	7	Every automobile is manufactured by Lamborghini.
c $\implies$ d	100%	7	Everything manufactured by Lamborghini is an automobile.
e $\implies$ b,c	100%	3	All the entities assembled in Italy are Lamborghini automobiles.
c,d $\implies$ a,b	71%	7	71% of the Lamborghini automobiles are categorized as “sport cars” and “Lamborghini vehicles”

Table 3: Association rules extracted from formal context in Table 2.

Of course, the first three implications are only true in our dataset. This is due to the fact that we use the “closed world” assumption, meaning that our rules only apply in “our world of data” where all cars of the brand “Lamborghini”. While this information is hardly useful, it provide a good insight of the capabilities of implication rules. For instance, including a larger number of triples in our dataset would allow to discover that, while not all automobiles are manufactured by Lamborghini, they are manufactured by either a Company, an Organization or an Agent. These three *classes*[6] are types of the entity Lamborghini in DBpedia. Such a rule would allow to provide a *domain* characterization to the otherwise empty description of the property “dbo:manufacturer” in the DBpedia schema.

The association rule given in the fourth row of Table 3 shows the fact that 29% of the subjects of type “Automobile” and manufactured by “Lamborghini” should be categorized by “Sports cars” and “Lamborghini vehicles” to complete the data. This actually corresponds to the entities “Estoque” and “Gallardo” in Table 2. Using this notion, we can use association rules also to create new triples that allow the completion of the information included in DBpedia.

[6] *owl:class*

4.1 Supporting complex data using pattern structures

The aforementioned models to support linked data using FCA are adequate for small datasets as the example provided. In reality, LOD do not always consists of triples of resources (identified by their Universal Resource Identifiers or URIs) but contains a diversity of *datatypes* including dates, numbers, lists, strings and others.

Pattern structures are an extension of FCA which enables the analysis of complex data, such as numeric values, graphs, partitions, etc. In a nutshell, pattern structures provides the necessary definitions to apply FCA to entities with complex descriptions. The basics of pattern structures are introduced in [6], in here we provide a brief introduction using the instance of interval pattern structures [8] which is necessary to understand the remainder of this article.

Consider Table 4 showing the property *dbo:productionStartYear* for the subjects in Table 2. In this case we would like to extract, not a graph pattern, but a pattern in the year of production of a subset of cars. Different from a formal context as introduced in Section 3.2, instead of having a set M of attributes, interval pattern structures use a semi-lattice of interval descriptions ordered by inclusion (denoted by $(D, \sqsubseteq)$) (we can say that standard FCA uses a semi-lattice of set descriptions ordered by inclusion $(M, \subseteq)$). Furthermore, instead of having an incidence relation set I, pattern structures use a mapping function $\delta : G \to D$ which assigns to any instance $g \in G$ with its correspondent interval description $\delta(g) \in D$. For example, the entity “350GT” in Table 4 has the description $\delta(350GT) = \langle[1963, 1963]\rangle$. For any two given descriptions $\delta(g_1) = \langle[l_i^1, r_i^1]\rangle$ and $\delta(g_2) = \langle[l_i^2, r_i^2]\rangle$, with $i \in [1..n]$ where n is the number of intervals used for the description of entities, the similarity operator $\sqcap$ and the order $\sqsubseteq$ between them are defined as follows:

$$\begin{aligned}
\delta(g_1) \sqcap \delta(g_2) &= \langle[min(l_i^1, l_i^2), max(r_i^1, r_i^2)]\rangle \\
\delta(g_1) \sqsubseteq \delta(g_2) &\iff \delta(g_1) \sqcap \delta(g_2) = \delta(g_1) \\
e.g. \quad \delta(350GT) \sqcap \delta(Islero) &= \langle[1963, 1967]\rangle \\
(\delta(350GT) \sqcap \delta(Islero)) &\sqsubseteq \delta(400GT)
\end{aligned}$$

Finally, a pattern structure is denoted as $(G, (D, \sqsubseteq), \delta)$ where the derivation operators $(\cdot)^\square$ creating a Galois connection between $\wp(G)$ and $(D, \sqsubseteq)$ are defined for an interval pattern concept (A, d) such as $A \subseteq G$, $d \in D$, $A = d^\square$, $d = A^\square$ and:

$$A^\square := \bigsqcap_{g \in A} \delta(g) \qquad\qquad d^\square := \{g \in G \mid d \sqsubseteq \delta(g)\}$$

Using interval pattern concepts we can extract the actual patterns regarding the years of production of the cars. Table 5 summarizes the most important patterns extracted from Table 4. We can appreciate that cars can be divided in three main periods of time of production given by the intent of the interval pattern concepts.

Entitiy	*dbo:productionStartYear*
Reventon	2008
Countach	1974
350GT	1963
400GT	1965
Islero	1967
Veneno	2012
Aventador Roadster	-
Estoque	-
Gallardo	-

Table 4: Values of property *dbp:productionStartYear* for entities in Table 2. The symbol - indicates that there are no values present in DBpedia for those subjects.

Extent	Intent
Reventon, Veneno	$\langle[2008, 2012]\rangle$
Countach,	$\langle[1974, 1974]\rangle$
350GT,400GT,Islero	$\langle[1963, 1965]\rangle$

Table 5: Example of interval pattern concepts extracted from Table 4.

Supporting heterogeneous descriptions Different instances of the pattern structure framework have been proposed to deal with different kinds of data (graph, sequences, interval, partitions, etc.). For linked data we propose to use the heterogeneous pattern structure framework originally introduced in [4] as a way to describe objects in a heterogeneous space. For the sake of simplicity we do not provide all the details in the full model used for linked data.

Let G be the set of all entities, R the set of all properties (hereafter referred as relations), O_L be the set of all literals and O_U the set of all URI-objects such as $O = O_L \cup O_U$ is the set of all objects extracted from a RDF dataset defined by triples *subject-relation-object* or *subject-relation-literal.* Let us define that when the range of a relation holds for $range(r) \subseteq O_U$, we call r and object relation. Analogously, we call r a literal relation when $range(r) \subseteq O_L$. For any given relation (object or literal), we can define the pattern structure $\mathcal{K}_r = (G, (D_r, \sqcap), \delta_r)$ where $(D_r, \sqsubseteq)$ is an arbitrary order defined for the elements in $range(r)$ and δ_r is the mapping of entities $g \in G$ to its descriptions in D_r. Then, the triple (G, H, Δ) is called a heterogeneous pattern structure when H is the Cartesian product of all the descriptions sets D_r and Δ maps an entity $g \in G$ to a tuple where each component corresponds to a description in a set D_r.

For an "object relation", the order in $(D_r, \sqsubseteq)$ is given by standard set inclusion and thus, the pattern structure $\mathcal{K}_\nabla$ is just a formal context. For "literal relations" such as numerical properties the kind of pattern structure may vary according to what is more appropriate to deal with that specific kind of data. For example, for the property *dbo:productionStartYear* discussed in the previous section, $\mathcal{K}_{\text{dbo:productionStartYear}}$ should be modelled as an interval pattern structure. Finally, for the running example, the heterogeneous pattern structure is presented in Table 6. Cells in grey mark a *heterogeneous concept lattice* the extent of which contains cars "350GT, 400GT and Islero". The heterogeneous pattern is given by the tuple $(\{a, b\}, \{c\}, \{d\}, \langle[1963, 1967]\rangle)$.

	$\mathcal{K}_A$		$\mathcal{K}_B$	$\mathcal{K}_C$	$\mathcal{K}_D$	$\mathcal{K}_E$		$\mathcal{K}_{\text{dbo:productionStartYear}}$
	a	b	c	d	e	f	g	
Reventon	×	×	×	×	×	×		$\langle[2008, 2008]\rangle$
Countach	×	×	×	×	×			$\langle[1974, 1974]\rangle$
350GT	×	×	×	×			×	$\langle[1963, 1963]\rangle$
400GT	×	×	×	×				$\langle[1965, 1965]\rangle$
Islero	×	×	×	×				$\langle[1967, 1967]\rangle$
Veneno	×	×						$\langle[2012, 2012]\rangle$
Aventador Roadster	×	×						-
Estoque			×	×		×	×	-
Gallardo			×	×	×			-

Table 6: Heterogeneous pattern structure for the running example. Indexes for properties are shown in Table 1

5 Concept lattice as an index for the RDF graph

Intuitively, the intent of a formal concept represents a pattern in the RDF graph which, in terms of SPARQL, can be expressed as the SPARQL query in Listing 1.1. The concept lattice defines an order among the "graph patterns" found in the set of 35 triples of the example in Table 2 and, moreover it generates a map of the SPARQL queries that could be answered by this dataset. For example, the concept lattice in Figure 1 indicates two possible *refinements* (adding restrictions to the query to reduce the result yielded) and two other possible *expansions* (removing restrictions from the query to enlarge the result yielded) of the query in Listing 1.1. Namely, by adding the condition *?s dbo:layout dbp:Front-engine*, we refine the query to get a unique answer *350GT*. On the other hand, by losing the conditions *?s dc:subject dbp:Sports_cars* and *?s dc:subject dbp:Lamborghini_vehicles*, we expand the query to include the subjects *Estoque* and *Gallardo*.

Query refinements/expansions are of particular interest in the case of linked open data. In an environment such as DBPedia which contains more than 400 million triples, it is unlikely that a user will know a-priori the availability of data and the schema in which triples has been created. We can make the analogy of trying to create a SQL query in a database in which the data schema is previously unknown. For this reason, the creation of a SPARQL query usually considers a previous step of data exploration to assess data availability and how should it be queried. A concept lattice automatizes this task by proposing a "map of queries" which induces a constant refinement/expansion process given the patterns extracted from the set of RDF triples. Given that the formal concept extents are sets of related RDF nodes (particularly, objects), the concept lattice becomes a graph index which allows to navigate it through its patterns rather than its nodes.

Navigation can be used to focus on specific information related to some queries or to the interest of a user. The extent of a concept corresponds to

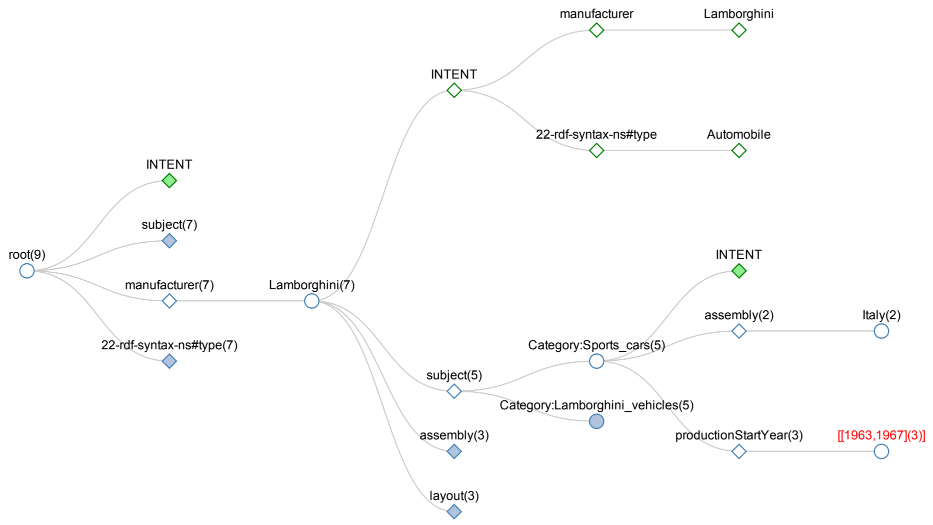

Fig. 2: A possible branch of the concept lattice visualisation for the running example.

the set of entities, e.g. a set of Lamborghini cars, and the intent of a concept corresponds to the common description of that set of entities, e.g. cars with a four-wheel drive. However, a user cannot in general analyze the whole concept lattice as this structure is often very large. Moreover, the lattice substructure with a high number of interconnections can be misguiding. Thus, we propose to visualize a concept lattice with help of "embedded subtrees" and only immediate descendants in a subtree are shown to a user.

In the visualization, we use the following notations. Circles represents the nodes of the concept lattice or the concepts. Diamonds corresponds to predicates and represents the transitions between nodes. If a concept $\mathtt{C}_1$ is an ancestor of a concept $\mathtt{C}_2$ in the lattice and they have intents with different values w.r.t. a predicate P, then we have a transition (a diamond) with the name of P from $\mathtt{C}_1$ to $\mathtt{C}_2$. In this case the intent of the concept $\mathtt{C}_2$ has a new value V of predicate P w.r.t. the concept $\mathtt{C}_1$, the pair $P : V$ represents an attribute of a pattern description. The value V is marked at the node representing $\mathtt{C}_2$. There is a special case of diamond-nodes. Such a node marked as "INTENT" represents a technical node of the tree visualization. This node shows the difference between descriptions of a concept connected to this node and the parent of the concept. Finally the text at every node can be either selected or not. It allows a user to select interesting concepts and to check the join of the extents of selected concepts – the set of entities within selected concepts.

Imagine the following scenario. A user wants to buy a Lamborghini car and he also knows which kinds of parameters he is interested in. Having this in mind he creates a SPARQL query providing the context from Table 2. In the resulting lattice, the descendant concepts of the top concept are different from the top concept w.r.t. the predicates "subject", "manufacturer", and "type".

Dataset	# Subjects	# Objects	# Triples	# Formal concepts	Exec. time [s]
Cars	529	1,291	12,519	14,657	17.32
Videogames	655	3,265	20,146	31,031	17.14
Smartphones	363	495	4,710	1,232	0.7
Countries	3,153	8,315	50,000	13,754	59.82

Table 7: Dataset characteristics.

The user can make precise the fact that he is interested in cars manufactured by Lamborghini, following the branch opened in Figure 2.

The user can check the modification in the description w.r.t. the parent node using the special node "INTENT". For example, the concept "Lamborghini(7)" is different from the concept "root(9)" in "manufacturer:Lamborghini" (the one he has explicitly made precise) and in "type:Automobile". It can be noticed that the properties attached to a node "INTENT" are inherited by all subconcepts in the navigation tree. The user can also check the entities attached to a node. For example, he can check the set of sport-cars manufactured my Lamborghini in 60th. There are 3 cars, "350GT" , "400GT", and "Islero".

This tree-based visualisation is gradual and allows one to navigate the concept lattice level by level, even without any precise objective in mind. This is a very practical way of navigating a concept lattice based on complex structures such as RDF triples. All data sets considered in this paper are visualized using this kind of navigation tree (see `http://www.loria.fr/~abuzmako/EKAW2014`).

6 Experimentation

To evaluate our model four datasets were created from DBpedia, namely Cars, Videogames, Smartphones and Countries. Each of them was created using a single SPARQL query with a unique restriction (either a fixed subject or a fixed type) while a different list of properties was extracted for each of them (see Table 9). Each dataset consists of a set of triples with predicate given by the properties in Table 9. In order to evaluate the heterogeneous aspect of our approach, for two of the four datasets those properties were selected whose range were numerical values. Table 7 shows the characteristics of each dataset.

For each dataset we calculated the $\mathcal{DG}-$Basis of implications derived from the heterogeneous pattern concept lattice. Each rule of the form $a \implies b$ was ranked according to the confidence of the rule $b \implies a$. It is worth noticing that when $a \implies b$ and $b \implies a$ we have an equivalence. Thus, given that implication rules have always a confidence of 100%, the confidence of the inverted rule tells us how close we are from defining an equivalence, which leads to the decision whether a set of RDF triples should be modified or not. For example, consider the following implication rule from the Countries dataset:

$$\textit{dc:subject-dbpc:Member\_states\_of\_the\_European\_union} \implies \\ \textit{rdf:type-dbo:EuropeanUnionMemberEconomies}$$

The inverted rule has a 92% of confidence. In this case we establish a definition and state that all the subjects with type "European Union Member Economies" should be annotated with the category "Member states of the European Union". It is important to notice here that we assume that all the elements with type "European Union Member Economies" are *correctly annotated with the type information* (e.g. the entity Mexico is not among these elements). While there are some measures that could be considered in order to overcome this issue, we left them outside the scope of this article.

Considering that there are no ground truth for any of the datasets, the reported results are rather for assessing the feasibility of our approach than for comparison purposes. For each of the ranked basis of implications in the experimentation we performed human evaluation. With the aid of DBpedia, it was evaluated if the implication rule was likely to become a definition. The answer provided for each of the rule was binary (yes or no). Afterwards, we measured the precision for the first 20 ranked implication rules (P@20) as the proportion of the rules that were likely to become a definition (those evaluated as yes) over 20 (the total number of rules taken). In our case, the precision value works as an indicator of how likely are implication rules to become useful for RDF data completion (see Table 8). Sadly, since we do not have a ground truth of all the triples that should be added to each dataset, we are not able to report on recall. Nevertheless, to complement this evaluation we provide the values of the precision at 11 points (P@11p) [10] where we consider each human evaluation as the ground truth for its respective dataset and thus, each list of implication rules has a 100% recall. Precision at 11 points provides a notion on how well distributed are the answers in the ranking. Figure 3 contains the curves for each of the datasets. Values for precision at 20 points are high for all datasets and particularly for the dataset "Countries". This may be due to the fact that Countries was the only dataset built for resources with a fixed type. A precision of 0.9 indicates that 9 out of 10 implication rules can be transformed into definitions by creating RDF triples that would complete the entities descriptions. Precision at 11 points shows that confidence is a good indicator on the usefulness of the implication rules for data completion. For example, for the worst result, it shows that in the Videogames dataset, when the evaluator provides the last "yes" answer for an implication rule, he has also given a "yes" to 6 out 10 (from a total of 46). For our best result (Countries dataset) it is over 8 out of 10.

Results show that confidence is a good indicator for the selection of implication rules in terms of data completion. Further experimentation should be performed to assess if the triples being created are "correct" or not. As we have previously described, we start from the assumption that resources being completed are correctly linked to the implication rule. While this may not always be true, to our understanding, our approach is still useful under those circumstances given that it would allow to discover such "incorrectly" annotated entities. Finally, regarding execution times, Table 7 shows that even for the larger dataset, the execution time is less than a minute. While this time is perfectly acceptable for the analysis of implication rules, it is hardly the case for online navigation of the RDF graph as the one proposed in Section 5. This is due to the fact that we are calculating the whole concept lattice for the user to navigate in while, in a real world scenario we would only calculate the part of the lattice that the user is currently seen using different FCA algorithms.

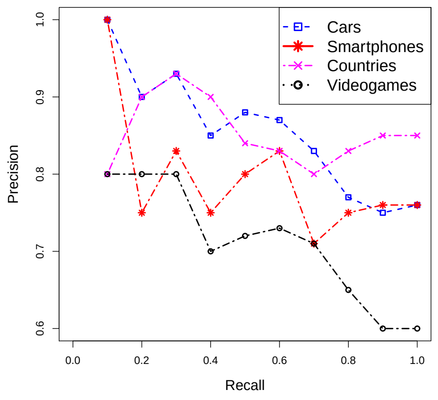

Fig. 3: Precision at 11 points for each dataset.

Dataset	Rules evaluated	P@20
Cars	19	0.85
Videogames	46	0.7
Smartphone	47	0.79
Countries	50	0.9

Table 8: Precision at the first 20 implication rules for each dataset.

7 Conclusion and Discussion

In the current study, we introduce a mechanism based on association rule mining for the completion of RDF dataset. Moreover, we use heterogeneous pattern structures to deal with heterogeneity in data present in LOD. Based on the concept lattice obtained by heterogeneous patterns structures, a navigation mechanism over the RDF triples is provided which also takes into account the suggestion of SPARQL queries. Several experiments have been conducted over the targeted dataset and the evaluation has been conducted for each of the experiments. This study shows the capabilities of FCA to deal with the complex RDF structure and how the data mining algorithms can help in completing and understanding the underlying RDF data. In future, we plan to build a tool which allows the user to define the domain of knowledge (or part of knowledge base) he wants to explore and provide automated selection of the important information that can be extracted and presented to the user.

Dataset	Cars	Videogames	Smartphones	Countries
Restriction	dc:subject dbpc:Sports_cars	dc:subject dbpc:First-person_shoothers	dc:subject dbpc:Smartphones	rdf:type dbo:Country
Properties	rdf:type dc:subject dbo:bodyStyle dbo:transmission dbo:assembly dbo:designer dbo:layout	rdf:type dc:subject dbo:computingPlatform dbo:developer dbo:requirement dbo:genre dbo:releaseDate	rdf:type dc:subject dbo:manufacturer dbo:operatingSystem dbo:developer dbo:cpu	rdf:type dc:subject dbo:language dbo:governmentType dbo:leaderType dbo:foundingDate dbo:gdpPppRank

Table 9: Dataset construction properties. Underlined properties with a range as numerical value.

References

1. 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.
2. Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. DBpedia - A Crystallization Point for the Web of Data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7(3):154–165, 2009.
3. Claudio Carpineto and Giovanni Romano. *Concept data analysis - theory and applications.* Wiley, 2005.
4. Víctor Codocedo and Amedeo Napoli. A Proposition for Combining Pattern Structures and Relational Concept Analysis. In CynthiaVera Glodeanu, Mehdi Kaytoue, and Christian Sacarea, editors, *Formal Concept Analysis*, volume 8478 of *Lecture Notes in Computer Science*, pages 96–111. Springer International Publishing, 2014.
5. Mathieu d'Aquin and Enrico Motta. Extracting relevant questions to an rdf dataset using formal concept analysis. In Mark A. Musen and Óscar Corcho, editors, *K-CAP*, pages 121–128. ACM, 2011.
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. Mehdi Kaytoue, Sergei O. Kuznetsov, Amedeo Napoli, and Sébastien Duplessis. Mining gene expression data with pattern structures in formal concept analysis. *Inf. Sci.*, 181(10):1989–2001, 2011.
9. Sergei O Kuznetsov and Sergei A Obiedkov. Comparing Performance of Algorithms for Generating Concept Lattices. *Journal of Experimental and Theoretical Artificial Intelligence*, 14:189–216, 2002.
10. Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schtze. *Introduction to Information Retrieval.* July 2008.
11. Heiko Paulheim and Christian Bizer. Type inference on noisy rdf data. In Harith Alani, Lalana Kagal, Achille Fokoue, Paul T. Groth, Chris Biemann, Josiane Xavier Parreira, Lora Aroyo, Natasha F. Noy, Chris Welty, and Krzysztof Janowicz, editors, *International Semantic Web Conference (1)*, volume 8218 of *Lecture Notes in Computer Science*, pages 510–525. Springer, 2013.
12. Julien Tane, Philipp Cimiano, and Pascal Hitzler. Query-based multicontexts for knowledge base browsing: An evaluation. In *ICCS*, pages 413–426, 2006.
13. Dominik Wienand and Heiko Paulheim. Detecting incorrect numerical data in dbpedia. In Valentina Presutti, Claudia d'Amato, Fabien Gandon, Mathieu d'Aquin, Steffen Staab, and Anna Tordai, editors, *ESWC*, volume 8465 of *Lecture Notes in Computer Science*, pages 504–518. Springer, 2014.