Mining Definitions from RDF Annotations Using Formal Concept Analysis

Mehwish Alam, Aleksey Buzmakov, Victor Codocedo, Amedeo Napoli

An approach to improving RDF Data with Formal Concept Analysis

Mehwish Alam, Aleksey Buzmakov, Víctor Codocedo, Amedeo Napoli
LORIA (CNRS – Inria Nancy Grand Est – Université de Lorraine)
BP 239, Vandoeuvre-lès-Nancy, F-54506, France
{firstname.lastname@loria.fr}

Abstract

The popularization and quick growth of Linked Open Data (LOD) 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 the completeness of the data and the their potential to provide concept definitions in terms of necessary and sufficient conditions. In this work we propose a novel technique based on Formal Concept Analysis which organizes RDF data into a concept lattice. This allows data exploration as well as the discovery of implication rules which are used to automatically detect missing information and then to complete RDF data. Moreover, this is a way of reconciling syntax and semantics in the LOD cloud. Finally experiments on the DBPedia knowledge base show that the approach is well-founded and effective.

1 Introduction

World Wide Web has tried to overcome the barrier of data sharing by converging data publication into Linked Open Data (LOD) [Bizer *et al.*, 2009]. The LOD cloud stores data in the form of *subject-predicate-object* triples based on the RDF language[1], a standard formalism for information description of web resources. In this context, DBpedia is the largest reservoir of linked data in the world currently containing more than 4 million triples. All of the information stored in DBpedia is obtained by parsing Wikipedia, the largest open Encyclopedia created by the collaborative effort of thousands of people with different levels of knowledge in several and diverse domains.

More specifically, DBpedia content is obtained from semi-structured sources of information in Wikipedia, namely *infoboxes* and *categories*. Infoboxes are used to standardize entries of a given type in Wikipedia. For example, the infobox for "automobile" has entries for an image depicting the car, the name of the car, the manufacturer, the engine, etc. These *attributes* are mapped by the DBpedia parser to a set of "properties" defined in an emerging ontology[2] [Benz *et al.*, 2010] (infobox dataset) or mapped through a hand-crafted lookup table to what is called the DBPedia Ontology (mapped-based ontology). Categories are another important tool in Wikipedia used to organize information. Users can freely assign a category name to an article relating it to other articles in the same category. Example of categories for cars are "Category:2010s automobiles", "Category:Sports cars" or "Category:Flagship vehicles". While we can see categories in Wikipedia as an emerging "folksonomy", the fact that they are curated and "edited" make them closer to a controlled vocabulary. DBpedia exploits the Wikipedia category system to "annotate"[3] objects using a taxonomy-like notation. Thus, it is possible to query DBpedia by using *annotations* (e.g. all cars annotated as "Sport cars"). While categorical information in DBpedia is very valuable, it is not possible to use a category as one could expect, i.e. as a definition of a class of elements that are instances of the class or, alternatively, that are "described" by the category. In this sense, such a category violates the actual spirit of semantic Web.

Let us explain this with an example. The Web site of DBpedia in its section of "Online access" contains some query examples using the SPARQL query language. The first query has the description "People who were born in Berlin before 1900" which actually translates into a graph-based search of entities of the type "Person", which have the property "birthPlace" pointing to the entity representing the "city of Berlin" and another property named "birthDate" with a value less than 1900. We can see here linked data working at "its purest", i.e. the form of the query provides the right-hand side of a definition for "People who were born in Berlin before 1900". Nevertheless, the fourth query named "French films" does not work in the same way. While we could expect also a graph-based search of objects of the type "Film" with maybe a property called "hasCountry" pointing to the entity representing "France", we have a much rougher approach. The actual SPARQL query asks for objects (of any type) annotated as "French films".

In general, categorization systems express "information needs" allowing human entities to quickly access data. French films are annotated as such because there is a need

[1]Resource Description Framework

[2]Emerging in the sense of "dynamic" or "in progress".

[3]Notice that in DBPedia the property used to link entities and categories is called "subject". We use "annotation" instead of "subject" to avoid confusions with the "subject" in a triple subject-predicate-object.

to find them by these keywords. However, for a machine agent this information need is better expressed through a *definition*, like that provided for the first query (i.e. "People who were born in Berlin before 1900"). Currently, DBPedia mixes these two paradigms of data access in an effort to profit from the structured nature of categories, nevertheless further steps have to be developed to ensure coherence and completeness in data.

Accordingly, in this work we describe an approach to bridge the gap between the current syntactic nature of categorical annotations with their semantic correspondent in the form of a concept definition. We achieve this by mining patterns derived from entities annotated by a given category, e.g. All entities annotated as "Lamborghini cars" are of "type automobile" and "manufactured by Lamborghini", or all entities annotated as "French films" are of "type film" and of "French nationality". We describe how these category-pattern equivalences can be described as "definitions" according to *implication rules* among attributes which can be mined using Formal Concept Analysis (FCA [Ganter and Wille, 1999]). The method considers the analysis of heterogeneous complex data (not necessarily binary data) through the use of "pattern structures" [Ganter and Kuznetsov, 2001], which is an extension of FCA able to process complex data descriptions. A concept lattice can be built from the data and then used for discovering *implication rules* (i.e. association rules whose confidence is 100%) which provide a basis for "subject definition" in terms of necessary and sufficient conditions.

The remainder of this article is structured as follows: Section 2 gives a brief introduction to the theoretical background necessary to sustain the rest of the paper. Section 3 describes the approach used for data completion in the DBpedia knowledge base. Section 4 provides experimental results in four datasets created from DBpedia and a brief discussion over our findings. Finally, Section 5 concludes the paper offering some perspectives over our approach.

2 Preliminaries

Linked Open Data (LOD): [Bizer *et al.*, 2009] is a formalism for publishing structured data on-line in the form Resource Description Framework (RDF)[4]. RDF stores data in the form of statements, where each statement is referred to as an RDF triple and is represented as $\langle subject, predicate, object\rangle$. The profile of an RDF triple $\langle s, p, o\rangle$ is given by $(U \cup B) \times (U \cup B) \times (U \cup B \cup L)$ where a set of RDF triples is an RDF graph, denoted by $\mathcal{G}$. Here, U is the URI reference, B refers to blank node and L is the literal. In the current study we do not take into account blank nodes (B) because DBpedia does not contain any blank nodes. So, an RDF triple is represented as $U \times U \times U \cup L$. For convenience, in the following we denote the set of predicate names as P and the set of objects as O. LOD can then be queried and accessed through SPARQL[5], 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. For example, let us consider the query for all entities of type film annotated as French films and as Psychological thrillers shown in Listing 1.

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

[5]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.

```
SELECT ?s WHERE {
?s rdf:type dbo:Film ;
   dc1:subject dbpc:French_films ;
   dc1:subject dbpc:Psychological_thriller_films }
```

Listing 1: Example of a SPARQL query for films annotated as French Films and Psychological thrillers. Prefixes are defined in Table 1.

Formal Concept Analysis (FCA): is a mathematical framework used for classification, data analysis, information retrieval and knowledge discovery [Carpineto and Romano, 2005] among other tasks. The basics of FCA can be found in [Ganter and Wille, 1999], but in the following we recall some important definitions. Let G be a set of entities, M a set of attributes, and $I \subseteq G \times M$ an incidence relation. Actually, in FCA elements of G are called "objects". In this article we call them "entities" to avoid confusions with "objects" as defined for RDF triples. Then, the relation gIm means that entity $g \in G$ has attribute $m \in M$. The triple $\mathcal{K} = (G, M, I)$ is called a "formal context". Two derivation operators, both denoted by $'$, formalize the sharing of attributes for entities, and, dually, the sharing of entities for attributes:

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

A' denotes the set of attributes shared by *all* the entities in A and B' denotes the set of entities having *all* the attributes in B. 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 partial-ordering is defined between them as follows:

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

Then, (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 $\mathcal{C}_{\mathcal{K}}$) together with the order $\leqslant_{\mathcal{K}}$ forms the concept lattice denoted by $\mathcal{L}_{\mathcal{K}}$.

For example, consider the formal context in Figure 1 where $G = U$, $M = (P \times O)$ and $(u, (p, o)) \in I \iff \langle u, p, o\rangle \in$

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

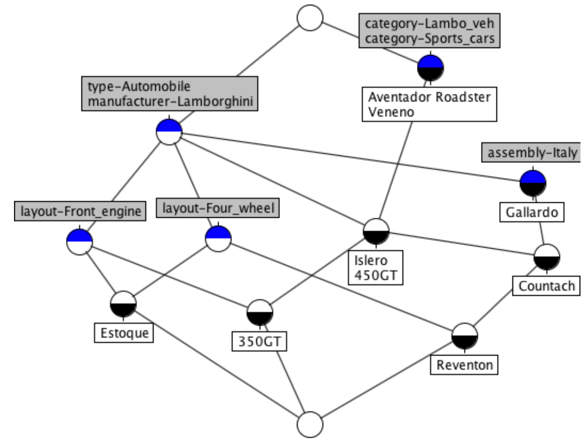

Figure 1: The formal context shown on the left is built after scaling from DBpedia data given in Table 1. Each cross (×) corresponds to a triple subject-predicate-object. On the right the corresponding concept lattice is shown.

$\mathcal{G}$, i.e. $\langle u, p, o\rangle$ is triple built from different statements manually extracted from DBpedia about nine different Lamborghini cars (35 RDF triples in total). Given a subject-predicate-object triple, the formal context contains subjects in rows, the pairs predicate-object in columns and a cross in the cell where the triple subject in row and predicate-object in column exists.

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 four *predicate-object* pairs. Actually, they form a formal concept depicted by the gray cells in Figure 1 and labelled as "Islero, 400GT" in Figure 1 (actually, the extent of this concept is "Islero, 400GT, 350GT, Reventon").

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 in X and Y (i.e. $|X' \cap Y'|$[6]) w.r.t. the whole set of entities, and a confidence which is the proportion of entities having the (same) set of attributes in X and Y, w.r.t. X (i.e. $|X' \cap Y'|/|X|'$). A rule $X \implies Y$ of confidence 1 verifying 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*. When $X \implies Y$ and $Y \implies X$ are implication rules we say that $X \iff Y$ is an equivalence or a definition. This can happen when two attributes have the same "attribute concept", e.g. *type-Automobile* and *manufacturer-Lamborghini* in the concept lattice of Figure 1.

3 Completing DBpedia data with pattern structures

3.1 Rationale

Consider the following fictional scenario. You are a bookkeeper in a library of books written in a language you do not understand. A customer arrives and asks you for a book about "Cars". Since you do not know what the books are about (because you cannot read them), you ask the customer to browse the collection on his own. After he finds a book he is interested to read, you will mark the symbol ⋆ on that book for future references. Then, in an empty page you will write (⋆ - *Cars*). After several cases like this, you will probably end up with a page full of symbols representing different topics or categories of your books, among them (⊖ - *Sports*), (⋄ - *Football*) and (○ - *History*). Now you can even combine symbols when customers ask you for "Sport Cars" which you translate into ⋆⊖. Actually, the demand for book about "Sport Cars" is so high that you create a new symbol just for it †. So doing, you have created your own categorization system of a collection of books you do not understand.

In general, given a topic, you are able to retrieve books without much troubles, however since you do not understand the books, you are restricted to the set of symbols you have for doing this. Furthermore, if you are not careful some problems start to arise, such as books marked with ⋄ and without ⊖. Finally, people do not get books marked with † when they look for "Cars", since they only search for the symbol ⊖.

It is easy to stablish an analogy on how DBPedia profits from Wikipedia's categorization system and the above scenario. DBPedia is able to retrieve entities when queried with an annotation (as the example of "French films"), however any information need not initially provided as a category is unavailable for retrieval (such as "French films about the Art Nouveau era"). Incoherences in categorical annotations are quite frequent in DBPedia, for example there are over 200 entities annotated as "French films" which are not typed as "Fims". Finally, DBPedia is not able to provide inferencing. For example, in Figure 2, the entities Veneno and Aventador, even though they are annotated as "Lamborghini vehicles", cannot be retrieved when queried simply by "vehicles". In such a way, it is exactly as if they were marked with a symbol such as †.

[6] $|\cdot|$ denotes set cardinality.

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 2: Association rules extracted from formal context in Figure 1.

3.2 The completion of DBpedia data

Our main concern in this case lies in two aspects. Firstly, are we able to complete data using logical 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"? Secondly, are we able to *complete* the descriptions of a given type? For example, DBpedia does not specifies that an "Automobile" should have a "manufacturer". In the following, we try to answer these two questions using implications and association rules.

Consider rules provided 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 are of "Lamborghini" brand, i.e. all other information about cars that we do not know can be assumed as false [Fürber and Hepp, 2011]. While this implication is trivial, it provides a good insight of the capabilities of implications. 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*[7] 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 in Table 2 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 Figure 1. Based on this fact, we can use association rules also to create new triples that allow the completion of the information included in DBpedia.

3.3 Pattern structures for the completion process

The aforementioned models to support linked data using FCA are adequate for small datasets as the example provided. Actually, LOD do not always consists of triples of resources (identified by their URIs) but contains a diversity of *data types* and structures including dates, numbers, collections, strings and others making the process of data processing much more complex. This calls for a formalism able to deal with this diversity of complex and heterogeneous data.

Accordingly, pattern structures are an extension of FCA which enables the analysis of complex data, such as numerical 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 [Ganter and Kuznetsov, 2001]. Below, we provide a brief introduction using interval pattern structures [Kaytoue *et al.*, 2011].

[7] *In the OWL language sense.*

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

Table 3: Values of property *dbp:productionStartYear* for entities in Figure 1. The symbol - indicates that there are no values present in DBpedia for those subjects.

Let us consider Table 3 showing the property *dbo:productionStartYear* for the subjects in Figure 1. In such a case we would like to extract a pattern in the year of production of a subset of cars. Contrasting a formal context as introduced in Section 2, instead of having a set M of attributes, interval pattern structures use a semi-lattice of interval descriptions ordered by a subsumption relation and denoted by $(D, \sqsubseteq)$[8]. Furthermore, instead of having an incidence relation set I, pattern structures use a mapping function $\delta : G \rightarrow D$ which assigns to any $g \in G$ the corresponding interval description $\delta(g) \in D$. For example, the entity "350GT" in Table 3 has the description $\delta(350GT) = \langle[1963, 1963]\rangle$.

Let us consider two 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 operation $\sqcap$ and the associated subsumption relation $\sqsubseteq$ between descriptions are defined as:

$$\delta(g_1) \sqcap \delta(g_2) = \langle[min(l_i^1, l_i^2), max(r_i^1, r_i^2)]\rangle$$
$$\delta(g_1) \sqsubseteq \delta(g_2) \iff \delta(g_1) \sqcap \delta(g_2) = \delta(g_1)$$
$$\delta(350GT) \sqcap \delta(Islero) = \langle[1963, 1967]\rangle$$
$$(\delta(350GT) \sqcap \delta(Islero)) \sqsubseteq \delta(400GT)$$

Finally, a pattern structure is denoted as $(G, (D, \sqsubseteq), \delta)$ where the derivation operators $(\cdot)^\square$ between $\wp(G)$ and $(D, \sqsubseteq)$ are given below:

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

An interval pattern concept (A, d) is such as $A \subseteq G$, $d \in D$, $A = d^\square$, $d = A^\square$. Using interval pattern concepts, we can extract and classify the actual pattern (and pattern concepts) representing the years of production of the cars. Table 4 summarizes the most important pattern concepts extracted from Table 3. 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.

[8] It can be noticed that standard FCA uses a semi-lattice of set

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

Table 4: Example of interval pattern concepts extracted from Table 3.

3.4 Heterogeneous pattern structures

Different instances of the pattern structure framework have been proposed to deal with different kinds of data, e.g. graph, sequences, interval, partitions, etc. For linked data we propose to use the approach called "heterogeneous pattern structure" framework introduced in [Codocedo and Napoli, 2014] as a way to describe objects in a heterogeneous space, i.e. where there are relational, multi-valued and binary attributes. It is easy to observe that this is actually the case for linked data where the set of literals L greatly varies in nature depending on the predicate. For the sake of simplicity we provide only the most important details of the model used for working with linked data.

When the range of a predicate (hereafter referred to as "relation") $p \in P$ is such that $range(p) \subseteq U$, we call p an "object relation". Analogously, when the range is such that $range(p) \subseteq L$, p is a "literal relation". For any given relation (object or literal), we define the pattern structure $\mathcal{K}_p = (G, (D_p, \sqcap), \delta_p)$, where $(D_p, \sqsubseteq)$ is an ordered set of descriptions defined for the elements in $range(p)$, and δ_p maps entities $g \in G$ to their descriptions in D_p. Based on that, the triple (G, H, Δ) is called a "heterogeneous pattern structure", where $H = \times D_p(p \in P)$ is the Cartesian product of all the descriptions sets D_p, and Δ maps an entity $g \in G$ to a tuple where each component corresponds to a description in a set D_p.

Then for an "object relation", the order in $(D_p, \sqsubseteq)$ is given by standard set inclusion and thus, the pattern structure $\mathcal{K}_p$ is just a formal context. For "literal relations", such as numerical properties, the 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 modeled as an interval pattern structure. For the running example, the heterogeneous pattern structure is presented in Table 5. Cells in grey mark a *heterogeneous pattern concept* the extent of which contains cars "350GT, 400GT, Islero". The intent of this heterogeneous pattern concept is given by the tuple $(\{a, b\}, \{c\}, \{d\}, \langle[1963, 1967]\rangle)$, i.e. *"Automobiles manufactured by Lamborghini between 1963 and 1967"*. The model of heterogeneous pattern structures is the basis of the experiments which are presented in the next section.

4 Experimentation

To evaluate our model, four datasets were created from DBpedia, namely "Cars", "Videogames", "Smartphones" and "Countries" (see characteristics of the datasets in Table 6).

	$\mathcal{K}_A$ a	$\mathcal{K}_A$ b	$\mathcal{K}_B$ c	$\mathcal{K}_C$ d	$\mathcal{K}_D$ e	$\mathcal{K}_E$ f	$\mathcal{K}_E$ g	$\mathcal{K}_{\text{dbo:productionStartYear}}$
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 5: Heterogeneous pattern structure for the running example. Indexes for properties are shown in Table 1.

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 6: Dataset characteristics.

Each dataset was created using a single SPARQL query with a unique restriction (either a fixed subject or a fixed type). A dataset consists of a set of triples whose predicate is given by the properties in Table 7. The heterogeneous aspect of data is illustrated by the fact that in two of the four datasets there are properties with numerical ranges.

For each dataset we calculated the set of all implication rules derived from the heterogeneous pattern concept lattice. Each rule of the form $X \implies Y$ was ranked according to the confidence of the rule $Y \implies X$. Thus, given that implication rules have always a confidence of 100%, the confidence of the inverted rule tells us how close we are from a definition, i.e. $a \iff b$ or both rules are implications. Having an implication or not leads to the decision whether a set of RDF triples should be completed or not. For example, the following implication rule from the Cars dataset has an inverted rule of 92% of confidence:

rdf:type-dbo:MaseratiVehicles $\implies$
dbo:manufacturer-dbp:Maserati

Accordingly, we can make of this implication a definition stating that the remainder 8% of the entities manufactured by *Maserati* should also be "typed" as *MaseratiVehicles* (recall in here that we have constructed our "world of data" by taking all "Sport Cars" from DBPedia, thus things built by Maserati which are not vehicles do not belong in our data). Of course, there are cases in which this will not be true. For example with a 90% of confidence in the opposite direction we have the implication rule:

dbo:layout-dbp:Quattro $\implies$ *dbo:manufacturer-dbp:Audi*

The creation of a definition from this rule (i.e. making the

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 7: The properties in the datasets. Underlined properties have numerical ranges.

descriptions ordered by inclusion, i.e. $(M, \subseteq)$.

remainder 10% of the cars manufactured by Audi have a layout 4x4) would be wrong. While we expect that the high confidence of the opposite association rule distinguish the case when a definition should be made, a human ruling to include background information will always be needed.

Considering that there is no ground truth for any of the datasets, the reported results are given for assessing the feasibility of our approach. For each of the ranked basis of implications in the experimentation we performed a human evaluation. With the help of DBpedia, it was evaluated if an implication rule was likely to become a definition. The answer provided for each of the rule was binary (yes or no). For instance, in the previous examples the first implication would render a "yes" answer, while the second, a "no". Afterward, 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). Actually, the precision value works as an indicator of how likely implication rules are useful for RDF data completion (see Table 8).

By contrast, 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) [Manning *et al.*, 2008]. 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 2 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 i.e. the Videogames dataset, when the evaluator provides the last "yes" answer for an implication rule, he/she 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 already mentioned, we assume that resources being completed are correctly linked to the implication rule. While this may not always be true, our approach is still useful under those circumstances given that it would allow to discover such "incorrectly" annotated entities. Finally, regarding execution times, Table 6 shows that even for the larger dataset, the execution time is less than a minute, and this time is perfectly acceptable for the analysis of implication rules.

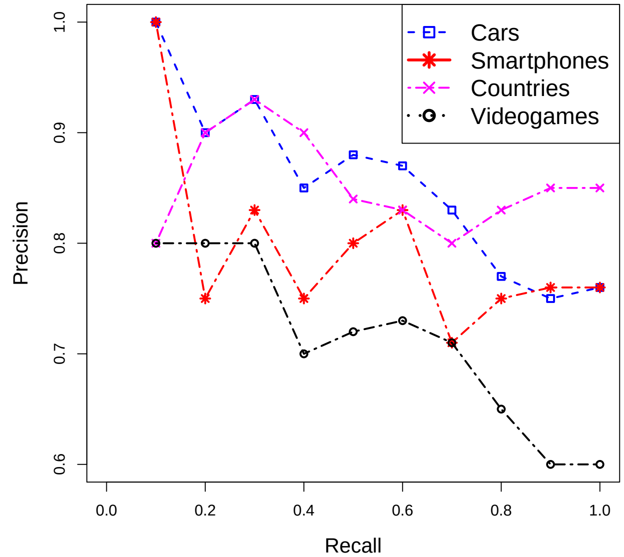

Figure 2: Precision at 11 points for each dataset.

5 Related work, discussion and conclusion

In [Paulheim and Bizer, 2013] 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 [Wienand and Paulheim, 2014] 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. In [Zaveri *et al.*, 2013], authors propose a manual and semi-automatic methodology for evaluating the quality of LOD resources w.r.t. classified common quality problems. Quality assessment is based on user input (crowd-sourcing) and tries to measure the correctness of schema axioms in DBpedia. In [Yu and Heflin, 2011a; 2011b], authors try to detect triples which are regarded as erroneous w.r.t. similar triples. The detection is based on the learning of probabilistic rules and on the discovery of generalized functional dependencies that are used to characterize the abnormality of the considered triples.

Different interesting perspectives are opened following this work. As we have discussed, categories represent some preloaded information needs in Wikipedia, i.e. a pre-answered

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.

question the answer of which is relevant for a group of people. Thus, the task would be to translate these information needs into description logics definitions, instead of attributes. It is possible to think that instead of annotating each French film with a "FrenchFilm" tag, we could define the category as $\mathtt{FrenchFilm} \equiv \mathtt{Film} \sqcap \mathtt{hasCountry}.\{\mathtt{FRANCE}\}$, or $\mathtt{LamborghiniCars} \equiv \mathtt{Automobile} \sqcap \mathtt{manufacturedBy}$ $.\{\mathtt{LAMBORGHINI}\}$. Given that these definitions are more restrictive than typing (*rdf:type*), our work should be adapted to deal with "near-definitions" in which both directions ($X \implies Y$ and $Y \implies X$) are association rules with high confidence.

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.

References

[Benz *et al.*, 2010] Dominik Benz, Andreas Hotho, and Gerd Stumme. Semantics made by you and me: Self-emerging ontologies can capture the diversity of shared knowledge. In *Proceedings of the 2nd Web Science Conference (WebSci10)*, Raleigh, NC, USA, 2010.

[Bizer *et al.*, 2009] 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.

[Carpineto and Romano, 2005] Claudio Carpineto and Giovanni Romano. *Concept data analysis - theory and applications.* Wiley, 2005.

[Codocedo and Napoli, 2014] Víctor Codocedo and Amedeo Napoli. A Proposition for Combining Pattern Structures and Relational Concept Analysis. In *Formal Concept Analysis*, volume 8478 of *Lecture Notes in Computer Science*, pages 96–111. Springer International Publishing, 2014.

[Fürber and Hepp, 2011] Christian Fürber and Martin Hepp. Swiqa - a semantic web information quality assessment framework. In Virpi Kristiina Tuunainen, Matti Rossi, and Joe Nandhakumar, editors, *19th European Conference on Information Systems, ECIS 2011, Helsinki, Finland, June 9-11, 2011*, 2011.

[Ganter and Kuznetsov, 2001] Bernhard Ganter and Sergei O. Kuznetsov. Pattern structures and their projections. In *ICCS*, volume 2120 of *Lecture Notes in Computer Science*, pages 129–142. Springer, 2001.

[Ganter and Wille, 1999] Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer, Berlin/Heidelberg, 1999.

[Kaytoue *et al.*, 2011] 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.

[Manning *et al.*, 2008] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schtze. *Introduction to Information Retrieval*. July 2008.

[Paulheim and Bizer, 2013] Heiko Paulheim and Christian Bizer. Type inference on noisy rdf data. In *International Semantic Web Conference (1)*, volume 8218 of *Lecture Notes in Computer Science*, pages 510–525. Springer, 2013.

[Wienand and Paulheim, 2014] Dominik Wienand and Heiko Paulheim. Detecting incorrect numerical data in dbpedia. In *ESWC*, volume 8465 of *Lecture Notes in Computer Science*, pages 504–518. Springer, 2014.

[Yu and Heflin, 2011a] Yang Yu and Jeff Heflin. Detecting abnormal data for ontology based information integration. In Waleed W. Smari and Geoffrey Fox, editors, *2011 International Conference on Collaboration Technologies and Systems, CTS 2011, Philadelphia, Pennsylvania, USA, May 23-27, 2011*, pages 431–438. IEEE, 2011.

[Yu and Heflin, 2011b] Yang Yu and Jeff Heflin. Extending functional dependency to detect abnormal data in RDF graphs. In Lora Aroyo, Chris Welty, Harith Alani, Jamie Taylor, Abraham Bernstein, Lalana Kagal, Natasha Fridman Noy, and Eva Blomqvist, editors, *The Semantic Web - ISWC 2011 - 10th International Semantic Web Conference, Bonn, Germany, October 23-27, 2011, Proceedings, Part I*, volume 7031 of *Lecture Notes in Computer Science*, pages 794–809. Springer, 2011.

[Zaveri *et al.*, 2013] Amrapali Zaveri, Dimitris Kontokostas, Mohamed Ahmed Sherif, Lorenz Bühmann, Mohamed Morsey, Sören Auer, and Jens Lehmann. User-driven quality evaluation of dbpedia. In Marta Sabou, Eva Blomqvist, Tommaso Di Noia, Harald Sack, and Tassilo Pellegrini, editors, *I-SEMANTICS 2013 - 9th International Conference on Semantic Systems, ISEM '13, Graz, Austria, September 4-6, 2013*, pages 97–104. ACM, 2013.