Bridging DBpedia Categories and DL-Concept Definitions using Formal Concept Analysis

Mehwish Alam, Aleksey Buzmakov, Victor 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 data and its 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 the discovery of implications, which are used to automatically detect missing information and then to complete RDF data.

Keywords: Formal Concept Analysis, Linked Open Data, Data Completion.

1 Introduction

The World Wide Web has tried to overcome the barrier of data sharing by converging data publication into Linked Open Data (LOD) [3]. 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] [2] (infobox dataset) or mapped through a hand-crafted lookup table to what is called the DBPedia 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

[1] Resource Description Framework - http://www.w3.org/RDF/
[2] Emerging in the sense of "dynamic" or "in progress".

"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 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 [7]). The method considers the analysis of heterogeneous complex data (not necessarily binary data) through the use of "pattern structures" [6], 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. For more details read the complete version of this paper [1].

[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 an RDF triple.

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. Finally, Section 4 concludes the paper.

2 Preliminaries

Linked Open Data (LOD) [3] is a formalism for publishing structured data on-line using the resource description framework (RDF). RDF stores data in the form of statements 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 denotes a set of URI references, B refers to the blank node and L to literals. For the sake of simplicity, in the current study we do no take into account blank nodes (B). 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 object names as O. LOD can then be queried and accessed through SPARQL[4], which is a standard query language for RDF data. SPARQL is based on matching graph patterns (present in the *WHERE* clause of a query) against RDF graphs. For example, let us consider the SPARQL query given in Listing 1.1, for all the entities of type Automobile manufactured by *Lamborghini*, annotated as "Sport_cars" and as "Lamborghini_vehicles",

```
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 context in Figure 1. Prefixes are defined in Table 1.

Formal Concept Analysis (FCA) is a mathematical framework introduced in [7], but in the following we assume that the reader already has necessary background of FCA. We only explain it with the help of an example. For example, consider the formal context in Figure 1 where $G = U$, $M = (P \times O)$ and $(u, (p, o)) \in I \iff \langle u, p, o \rangle \in \mathcal{G}$, i.e. $\langle u, p, o \rangle$ is a triple built from different triples 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.

[4] 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.

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

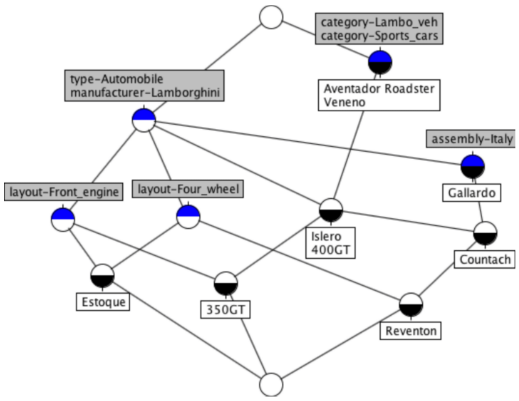

Fig. 1: The formal context shown on the left is built after scaling from DBpedia data given in Table 1. Each cross (×) corresponds to a triple subject-predicate-object. On the right the corresponding concept lattice is shown.

3 Improving DBpedia with FCA

3.1 Problem context

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 $\star$ on that book for future references. Then, in an empty page you will write ($\star$ - *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 ($\ominus$ - *Sports*), ($\diamond$ - *Football*) and ($\circ$ - *History*). Now you can even combine symbols when customers ask you for "Sport Cars" which you translate into $\star\ominus$. Actually, the demand for books 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 many 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 $\diamond$ and without $\ominus$. Finally, people do not get books marked with † when they look for "Cars", since they only search for the symbol $\ominus$.

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 "Films". Finally, DBpedia is not able to provide inferencing. For example, in Figure 1, 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 †.

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 specify 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 [5]. While these implications are trivial, they provide a good insight of the capabilities of our model. For instance, including a larger number of triples in our dataset would allow discovering that, while not all automobiles are manufactured by Lamborghini, they are manufactured by either a Company, an Organization or an Agent. These three *classes*[5] are types of the entity Lamborghini in DBpedia. Such a rule would allow providing a *domain* characterization to the otherwise empty description of the predicate "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.

[5] *In the OWL language sense.*

Rule	Confidence	Support	Meaning
d $\Longrightarrow$ c	100%	7	Every automobile is manufactured by Lamborghini.
c $\Longrightarrow$ d	100%	7	Everything manufactured by Lamborghini is an automobile.
e $\Longrightarrow$ b,c	100%	3	All the entities assembled in Italy are Lamborghini automobiles.
c,d $\Longrightarrow$ 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.

3.3 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 provide the necessary definitions to apply FCA to entities with complex descriptions. The basics of pattern structures are introduced in [6]. Below, we provide a brief introduction using interval pattern structures [8].

Let us consider Table 3 showing the predicate *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 $(D, \sqsubseteq)$[6]. 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 the convex hull of two descriptions as follows:

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

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

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

[6] It can be noticed that standard FCA uses a semi-lattice of set descriptions ordered by inclusion, i.e. $(M, \subseteq)$.

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. Some of them are presented in the lower part of 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.

Entity	*dbo:productionStartYear*
Reventon	2008
Countach	1974
350GT	1963
400GT	1965
Islero	1967
Veneno	2012
Aventador Roadster	-
Estoque	-
Gallardo	-
Interval Pattern Concepts	
Reventon, Veneno	$\langle[2008, 2012]\rangle$
Countach,	$\langle[1974, 1974]\rangle$
350GT,400GT,Islero	$\langle[1963, 1967]\rangle$

Table 3: Upper table shows values of predicate *dbp:productionStartYear* for entities in Figure 1. The symbol - indicates that there are no values present in DBpedia for those subjects. Lower table shows the derived interval pattern concepts .

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 [4] 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.

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. Regarding "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, considering the predicate

dbo:productionStartYear discussed in the previous section, $\mathcal{K}_{\text{dbo:productionStartYear}}$ should be modelled as an interval pattern structure. For the running example, the heterogeneous pattern structure is presented in Table 4. 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"*.

	$\mathcal{K}_A$ a	b	$\mathcal{K}_B$ c	$\mathcal{K}_C$ d	$\mathcal{K}_D$ e	$\mathcal{K}_E$ f	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 4: Heterogeneous pattern structure for the running example. Indexes for properties are shown in Table 1.

4 Conclusion

To conclude, in the current study we introduce a mechanism based on association rule mining for the completion of the RDF dataset. Moreover, we use heterogeneous pattern structures to deal with heterogeneity in LOD. This study shows the capabilities of FCA for completing complex RDF structures.

References

1. Mehwish Alam, Aleksey Buzmakov, Victor Codocedo, and Amedeo Napoli. Mining definitions from rdf annotations using formal concept analysis. In *IJCAI 2015, Proceedings of the 24th International Joint Conference on Artificial Intelligence, Buenos Aires, Argentina, July 25-31, 2015*, 2015.
2. 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*, 2010.
3. 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.
4. Víctor Codocedo and Amedeo Napoli. A Proposition for Combining Pattern Structures and Relational Concept Analysis. In *12th International Conference on Formal Concept Analysis*. 2014.
5. Christian Fürber and Martin Hepp. Swiqa - a semantic web information quality assessment framework. In *19th European Conference on Information Systems*, 2011.
6. 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.
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. *Information Sciences*, 181(10):1989–2001, 2011.