Lattice-Based Views over SPARQL Query Results

Mehwish Alam, Amedeo Napoli

Lattice-Based Views over SPARQL Query Results

Mehwish Alam and Amedeo Napoli

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

Abstract. SPARQL queries over semantic web data usually produce a huge list of tuples as answers that may be hard to understand and interpret. Accordingly, this paper focuses on Lattice Based View Access (LBVA), a framework based on Formal Concept Analysis, to provide a view using `View By` clause based on a concept lattice. This lattice can be navigated for retrieving or mining specific patterns in query results.

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

1 Introduction

At present, the Web has become a potentially large repository of knowledge, which is becoming main stream for querying and extracting useful information. In particular, Linked Open Data (LOD) [1] provides a method for publishing structured data in the form of RDF. These RDF resources are interlinked with each other to form a cloud. SPARQL queries are used in order to make these resources usable, i.e., queried. Queries in natural language against standard search engines sometimes may require integration of data sources. The standard search engines will not be able to easily answer these queries, e.g., *Currencies of all G8 countries*. Such a query can be formalized as a SPARQL query over data sources present in LOD cloud through SPARQL endpoints for retrieving answers. Moreover, these queries may generate huge amount of results giving rise to the problem of information overload [4]. A typical example is given by the answers retrieved by search engines, which mix between several meanings of one keyword. In case of huge results, user will have to go through a lot of results to find the interesting ones, which can be overwhelming without any specific navigation tool. Same is the case with the answers obtained by SPARQL queries, which are huge in number and it may be harder for the user to extract the most interesting patterns. This problem of information overload raises new challenges for data access, information retrieval and knowledge discovery w.r.t web querying.

This paper proposes a new approach based on Formal Concept Analysis (FCA [5]). It describes a lattice-based classification of the results obtained by SPARQL queries by introducing a new clause "`View By`" in SPARQL query.

This framework, called Lattice Based View Access (LBVA), allows the classification of SPARQL query results into a concept lattice, referred to as a *view*, for data analysis, navigation, knowledge discovery and information retrieval purposes. The `View By` clause enhances the functionality of already existing `Group By` clause in SPARQL query by adding sophisticated classification and Knowledge Discovery aspects. Here after, we describe how a lattice-based view can be designed from a SPARQL query. Afterwards, a view is accessed for analysis and interpretation purposes which are totally supported by the concept lattice. In case of large data only a part of the lattice [8] can be considered for the analysis.

The intuition of classifying results obtained by SPARQL queries is inspired by web clustering engines [2] such as Carrot2[1]. The general idea behind web clustering engines is to group the results obtained by query posed by the user based on the different meanings of the terms related to a query. Such systems deal with unstructured textual data on web. By contrast, there are some studies conducted to deal with structured RDF data. In [4], the authors introduce a clause `Categorize By` to target the problem of managing large amounts of results obtained by conjunctive queries with the help of subsumption hierarchy present in the knowledge base. By contrast, the `View By` clause generates lattice-based views which provide a mathematically well-founded classification based on formal concepts and an associated concept lattice. Moreover, it also paves way for navigation or information retrieval by traversing the concept lattice and for data analysis by allowing the extraction of association rules from the lattice. Such data analysis operations allow discovery of new knowledge. Additionally, unlike `Categorize By`, `View By` can deal with data that has no schema (which is often the case with linked data). Moreover, `View By` has been evaluated over very large set of answers (roughly 100,000 results) obtained over real datasets. In case of larger number of answers, `Categorize By` does not provide any pruning mechanism while this paper describes how the views can be pruned using icerberg lattices.

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

2 Motivation

In this section we introduce a motivating example focusing on why LOD should be queried and why the SPARQL query results need classification. Let us consider a query searching for museums where the exhibition of some artists is taking place along with their locations. A standard query engine is not adequate for answering such kind of questions as it will produce a separate list of museums with the artists whose work is displayed there and a separate list of museums

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

with their locations. However, a direct query over LOD will perform resource integration to provide answers to the query. This query generates a huge amount of results, which further needs manual work to group the interesting links.

Linked Open Data represents data as RDF (Resource Description Framework) graphs. An RDF graph is a set of RDF triples, i.e., $\langle subject, predicate, object \rangle$, which is represented as node-and-arc-labeled directed graphs. SPARQL[2] is the standard query language for querying RDF graphs, which is based on matching graph patterns against RDF graphs. According to the scenario described above, the SPARQL query is shown in Listing 1.1.

Listing 1.1: SPARQL Query Museum

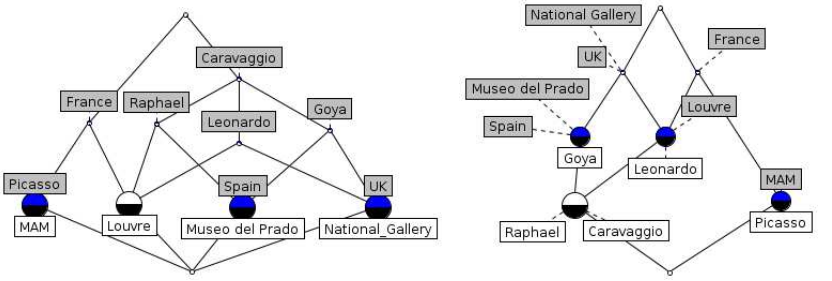
```
1 SELECT ?museum ?country ?artist WHERE {
2       ?museum rdf:type dbpedia-owl:Museum .
3       ?museum dbpedia-owl:location ?city .
4       ?city dbpedia-owl:country ?country .
5       ?painting dbpedia-owl:museum ?museum .
6       ?painting dbpprop:artist ?artist }
7       GROUP BY ?country ?artist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

This query retrieves the list of museums along with the artists whose work is exhibited in a museum along with the location of a museum. `Lines 5 and 6` of this query retrieve information about the artists whose work is displayed in some museum. More precisely, the page containing the information on a museum (`?museum`) is connected to the page of the artists (`?artist`) through a page on the work of artist (`?painting`) displayed in the museum. An excerpt of the answers obtained by `Group by` clause is shown below:

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

The problem encountered while browsing such an answer is that there are thousands of results to navigate through. Even after using the `Group By` clause the answers are arranged into several small groups because first there is more than one grouping criteria and second, there are many values of the variables in the `Group By` clause. By contrast, the clause `View By` activates the LBVA framework, where a classification of the statements is obtained as a concept lattice (see Figure 1a). The concept lattice shown in Figure 1a is labeled in a reduced format (reduced labeling), meaning that if a parent class contains an attribute then this attribute is also inherited by its children concepts. Let us consider the parent concept has the attribute `France` then its two children concepts also have the attribute `France`. Now if the museums in `UK` displaying the work of `Goya` are to be retrieved, first the concept containing all the museums displaying the work of `Goya` is obtained and then the specific concept `Goya and UK` is retrieved by drilling down. Finally, the answer is `National Gallery`.

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



(a) Classes of Museums w.r.t Artists and Countries, e.g., the concept on the top left corner with the attribute `France` contains all the French Museums. (`VIEW BY ?museum`)

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

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

3 Background

Formal Concept Analysis (FCA): FCA [5] is a mathematical framework used for a number of purposes, among which classification and data analysis, information retrieval and knowledge discovery [3]. Let G be a set of objects and M a set of attributes, and $I \subseteq G \times M$ a relation where gIm is true iff an object $g \in G$ has an attribute $m \in M$. The triple $\mathcal{K} = (G, M, I)$ is called a "formal context". Given $A \subseteq G$ and $B \subseteq M$, two derivation operators, both denoted by $'$, formalize the sharing of attributes for objects, and, in a dual way, the sharing of objects for attributes: $A' = \{m \in M \mid gIm \ for\ all\ g \in A\}$, $B' = \{g \in G \mid gIm \ for\ all\ m \in B\}$. The two derivation operators $'$ form a *Galois connection* between the powersets $\wp(G)$ and $\wp(M)$. Maximal sets of objects related to maximal set of attributes correspond to closed sets of the composition of both operators $'$ (denoted by $''$). Then a pair (A, B) is a formal concept iff $A' = B$ and $B' = A$. The set A is the "extent" and the set B is the "intent" of the formal concept (A, B). The set $\mathcal{C}_\mathcal{K}$ of all concepts from $\mathcal{K}$ is partially ordered by extent inclusion (or dually intent inclusion), denoted by $\leq_\mathcal{K}$ as $(A_1, B_1) \leq (A_2, B_2) \Leftrightarrow A_1 \subseteq A_2 (\Leftrightarrow B_2 \subseteq B_1)$. Consequently, $\mathcal{L}_\mathcal{K} = \langle \mathcal{C}_\mathcal{K}, \leq_\mathcal{K} \rangle$ forms the *concept lattice* of $\mathcal{K}$. There exist several algorithms [9] to build a concept lattice which also focus on efficiency of building the lattices for large number of objects.

In order to restrict the number of concepts in some cases iceberg concept lattices can be used [8], which contain only the top most part of the lattice. Formally, let $B \subseteq M$ and let minimum support, denoted by *minsupp*, be an integer representing a support threshold value. For a given concept (A,B), the support of B is the cardinality of A denoted by $|A|$. Relative support is given by $|A|/|G|$ and belongs to the interval $[0, 1]$. An intent B in concept (A,B) is said to be frequent as soon as $supp(B) = |A|/|G| \geq minsupp$. Likewise, a concept is called a frequent concept if its intent is frequent. The set of all frequent concepts of $\mathcal{K}$, for a given threshold, is called an iceberg concept lattice of $\mathcal{K}$. Along with iceberg lattices a stability index is also used for filtering the concepts. The stability index shows how much the concept intent depends on particular objects of the extent.

In some cases, a many-valued context is obtained instead of a formal context. A many-valued context is denoted by (G, M, W, I), where G is the set of objects, M is the set of attributes, W is the set of attribute values for each attribute and I represents a ternary relation between G, M, W, denoted as $I \subseteq G \times M \times W$. However, in order to obtain a one-valued binary context from the many valued context, scaling procedure is adopted. A scale S_m of an attribute m of a many-valued context is a one-valued context (G_m, M_m, I_m) with $m(G) = S_m$ for $m \in M$ and then the new set of attributes is $M_s = \bigcup_{m \in M} S_m$. During plain scaling the object set G remains unchanged, every many-valued attribute m is replaced by the scale attributes of scale S_m.

FCA also allows knowledge discovery using association rules. Duquenne-Guigues ($\mathcal{DG}$) basis for implications [6] is the minimal set of implications equivalent to the set of all valid implications for a formal context $\mathcal{K} = (G, M, I)$. An implication over the attribute set M in a formal context is of the form $B_1 \rightarrow B_2$, where $B_1, B_2 \subseteq M$. The implication holds iff every object in the context with an attribute in B_1 also has all the attributes in B_2. For example, when $(A_1, B_1) \leq (A_2, B_2)$ in the lattice, we have that $B_1 \rightarrow B_2$. $\mathcal{DG}$-basis represents all information lying in the concept lattice.

4 Lattice Based View Access

In this paper, we propose an approach called Lattice Based View Access for classification of SPARQL query results in the form of a concept lattice referred to as view. In the scenario of LOD, the RDF data and query processing procedure can not be controlled. Here we define views over RDF data by processing the set of tuples returned by the SPARQL query as answers.

SPARQL Queries with Classification Capabilities: The idea of introducing a `View By` clause is to provide classification of the results and add a knowledge discovery aspect to the results w.r.t the variables appearing in `View By` clause. Initially, the user poses SPARQL query of the form `SELECT` $?v_1$ $?v_2$ $\ldots$ $?v_n$ `WHERE { condition/pattern } VIEW BY` $?v_l$. More formally, "$view\_by(v_l) : q(\overrightarrow{v})$" where $\overrightarrow{v}$ is a vector of variables containing free variables called answer variables and v_l is a variable in the `SELECT` clause of SPARQL query providing the viewing criteria. The evaluation of query q over RDF triple store generates answers in the form of tuples. A tuple is a vector of terms (set of constants) mapped to the answer variables in query q. The processing of a SPARQL query $q(\overrightarrow{v}) = q(v_1, v_2, \ldots, v_n)$ yields a set of tuples $R = \{(X_1^i, X_2^i, \ldots, X_n^i)\}$, where $i = \{1, \ldots, k\}$ where each tuple provides an elementary answer to the query $q(\overrightarrow{v})$.

Following the example in section 2, let us consider the user gives " `VIEW BY ?artist`" instead of `Group By` clause for the query in Listing 1.1. Then, $v_1 = artist$ is the object variable, $v_2 =$ `museum` and $v_3 =$ `country` are the attribute variables and Figure 1a shows the generated view. In Figure 1b, we have; $v_1 =$ `artist` is the object variable, $v_2 =$ `museum` and $v_3 =$ `country` are attribute variables.

Designing a Formal Context (G, M, W, I)***:*** The results obtained by the query are in the form of set of tuples, which are then organized as a many-valued context. Among the variables one variable appears in the `View By` clause and is considered as the *object variable*. All the other variables are considered as *attribute variables*. Let v_l be the object variable in $\overrightarrow{v} = (v_1, v_2, \ldots, v_n)$, X_l^i be the answers obtained for v_l, then attribute variables will be $v_1, \ldots, v_{l-1}, v_{l+1}, \ldots, v_n$. The answers associated to attribute variables can be given as $\{X_1^i, X_2^i, \ldots, X_{l-1}^i, X_{l+1}^i, \ldots, X_n^i\}$. Then, $G = \{X_l^i, i = 1, \ldots, k\}$ and M is the set of many valued attributes, given as $M = \{v_1, v_2, \ldots, v_{l-1}, v_{l+1}, \ldots, v_n\}$ and W represents the attribute values, i.e., $W = \{X_1^i, X_2^i, \ldots, X_{l-1}^i, X_{l+1}^i, \ldots, X_n^i\}$. The occurrence of an object and an attribute together in $R = \{(X_1^i, X_2^i, \ldots, X_n^i)\}$ gives the ternary relation I.

Let us continue the example discussed in section 2. The answers are organized into many-valued context as follows. The distinct values of the variable `?museum` are kept as a set of objects, so $G = \{MuseeduLouvre, MuseodelPrado\}$. The attribute variables `artist,country` provide the set of attributes $M = \{artist, country\}$ and the tuples related to the variables provide attribute values, $w_1 = \{Raphael, LeonardoDaVinci\}$ and $w_2 = \{France, Spain\ UK\}$. The obtained many-valued context is shown in Table 1. The corresponding nominally scaled one-valued context is shown in Table 2.

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

Table 1: Many-Valued Context (Museum).

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

Table 2: One-Valued Context $\mathcal{K}_{Museum}$.

Building a Concept Lattice: Once the context is designed, a concept lattice (view) can be built using an FCA algorithm. This step is straight forward as soon as the context is provided. In the current study, we used *AddIntent*, which is an efficient implementation for building a concept lattice [9]. At the end of this step the concept lattice is built and the interpretation step can be considered. However, one limitation of the systems based on FCA is that they may encounter exponential time and space complexity in the worst case scenario for generating a concept lattice [7]. A view on SPARQL query in section 2, i.e, a concept lattice corresponding to Table 2 is shown in Figure 1a.

Interpretation Operations over a Lattice-Based Views: A formal context effectively takes into account the relations by keeping the inherent structure of

the relationships present in LOD as object-attribute relation. When a concept lattice is built, each concept keeps a group of terms sharing some attribute. This concept lattice can be navigated for searching and accessing particular LOD elements through the corresponding concepts within the lattice. This lattice can be drilled down from general to specific concepts or rolled up to find the general ones. For example, for retrieving the museums where there is an exhibition of `Caravaggio`'s paintings, it can be seen in the concept lattice shown in Figure 1a that the paintings of `Caravaggio` are displayed in `Musee du Louvre, Museo del Prado and National Gallery`. Now, in order to filter it by country, i.e., obtain `French` museums displaying `Caravaggio`. `Musee du Louvre` can be retrieved by navigating the same lattice. To retrieve museums located in `France` and `Spain`, a general concept containing all the French Museums with Caravaggio's painting is retrieved and then a specific concept containing the museums in `France` or `Spain` displaying `Caravaggio` can be accessed by navigation. The answer obtained will be `Musee du Louvre` and `Museo del Prado`.

After obtaining the view, i.e., the concept lattice, it can be accessed to obtain the $\mathcal{DG}$-basis of implications. For example, the rule `Goya, Raphael, Caravaggio` $\rightarrow$ `Spain` suggests that in Spain there exists a museum displaying the work of `Goya, Raphael, Caravaggio` (this implication is obtained from the view in Figure 1a). In order to get more specific answer, the user can browse through lattice and obtain `Museo Del Prado`.

5 Experimentation

5.1 DBpedia

DBpedia is currently comprised of a huge amount of RDF triples in many different languages which reflects the state of Wikipedia. Due to information extraction from crowd-sourced web site, triples present on DBpedia may contain incorrect information. Even if Wikipedia contains correct information, a parser may pick up wrong information [10]. Due to the above described reasons some of the properties may not be used uniformly. In the current experiment, we extracted the information about movies with their genre and location.

```
SELECT ?movie ?genre ?country WHERE {
?movie rdf:type dbpedia-owl:Film .
?movie dbpprop:genre ?genre .
?movie dbpprop:country ?country .}
VIEW BY ?movie
```

The obtained concept lattice contained 1395 concepts. Out of which 201 concepts on the first level were evaluated manually for correctness of the information about the movie genre. 141 concepts kept the genre information about the movie. 45% of these concepts contained wrong genre information as its intent (see first three concepts in Table 3). In such a case, the generated lattice-based view helps in separating music genre from the movie genre and further guide in introducing a new relation such as `soundtrackGenre` and adding

new triples to the knowledge base, for example, `dbpedia:The_Scorpion_King, dbpedia-owl:soundtrackGenre, dbpedia:Hard_Rock`.

ID	Supp.	Intent
C#1	17	`Hard Rock`
C#2	15	`Contemporary R&B`
C#3	18	`Jazz`
C#4	750	`United States`
C#5	1225	`India`
C#6	6	`France`

Table 3: Some Concepts from $\mathcal{L}_{DBpedia}$ (Concept Lattice for DBpedia)

Moreover, If we observe the obtained view, it can be seen that there are too few movies from countries other than United States and India. For example, C#4 and C#5 are the classes for movies from United States and India, where there are 1225 movies from India in DBpedia and 750 movies from United States. Finally, it can be concluded that the information present on DBpedia still needs to be corrected and completed.

The concept lattice can help in obtaining classes of movies w.r.t countries also. As this approach provides an added value to the already existing `Group By` clause, it is possible to find movies which are made in collaboration with several countries. For example, `The Scorpion King` was made in collaboration with `United States, Germany` and `Belgium`.

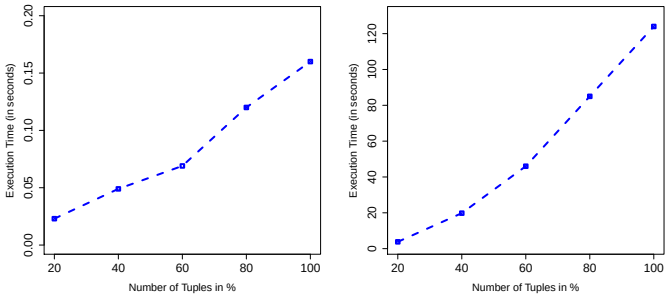
5.2 YAGO

The construction of YAGO ontology is based on the extraction of instances and hierarchical information from Wikipedia and Wordnet. In the current experiment, we posed a similar query to YAGO with the `View By` clause. While querying YAGO it was observed that the genre and location information was also given in the subsumption hierarchy of ontology. The first level of the obtained view kept the groups of movies with respect to their languages. e.g., the movies with genre `Spanish Language Films`. However, as we further drill down in the concept lattice we get more specific categories which include the values from the location variable such as `Spain, Argentina` and `Mexico`. Finally, it can be concluded that YAGO provides a clean categorization of movies by making use of the partially ordered relation between the concepts present in the concept lattice. YAGO also learns instances from Wikipedia and it contains many movies from all over the world. This observation gives the idea about the strong information extraction algorithm as it contains more complete information.

$\mathcal{DG}$-Basis of Implications for YAGO and DBpedia were computed. The implications were naively pruned with respect to support threshold. For DBpedia, the number of rules obtained were 64 for a support threshold of 0.11%. In case of YAGO, around 1000 rules were extracted on support threshold of 0.2%. Table 4 contains some of the implications obtained for both the datasets. It can be clearly observed that the support for the implications is much larger in YAGO

Impl. ID	Supp.	Implication
YAGO		
1.	96	`wikicategory RKO Pictures films` → `United States`
2.	46	`wikicategory Oriya language films` → `India`
DBpedia		
3.	3	`Historical fiction` → `United Kingdom@en`
4.	3	`Adventure fiction, Action fiction` → `Science fiction`

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



(a) Runtime for DBpedia (b) Runtime for YAGO

Fig. 2: Experimental Results.

than DBpedia, which points towards the completion of YAGO. This fact is actually useful in finding regularities in the SPARQL query answers which can not be discovered from the raw tuples obtained. For example, rule#1 states that `RKO picture films` is an American film production and distribution company as all the movies produced and distributed by them are from United States. Moreover, rule#2 says that all the movies in `Oriya language` are from `India`. Which actually points to the fact that Oriya is one of many languages that is spoken in India. On the other hand, some of the rules obtained from DBpedia are incorrect. For example, rule#3 states the strange fact that all the `historical fiction` movies are from `United Kingdom`. Same is the case with rule#4 which states that all the movies which are `Adventure fiction` and `Action fiction` are also `Science Fiction`, which may not actually be the case. Through the comparison of the $\mathcal{DG}$-Basis for both the datasets it can be observed that the YAGO may be more appropriate for further use by the application development tools and knowledge discovery purposes.

For each of the above queries we tested how our method scales with growing number of results. The number of answers obtained by DBpedia were around 4000 and the answers obtained by YAGO were 100,000. The experimental results of the runtime for the building the concept lattice are shown in Figure 2. Visualization of these experiments along with the implementation can be accessed online[3].

[3] `http://webloria.loria.fr/~alammehw/lbva/`

6 Conclusion and Discussion

In LBVA, we introduce a classification framework based on FCA for the set of tuples obtained as a result of SPARQL queries over LOD. In this way, a view is organized as a concept lattice built through the use of `View By` clause that can be navigated where information retrieval and knowledge discovery can be performed. Several experiments show that LBVA is rather tractable and can be applied to large data. For future work, we intend to use pattern structures with a graph description for each considered object, where the graph is the set of all triples accessible w.r.t reference object.

References

1. 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. Claudio Carpineto, Stanislaw Osiński, Giovanni Romano, and Dawid Weiss. A survey of web clustering engines. *ACM Comput. Surv.*, 41(3):17:1–17:38, July 2009.
3. Claudio Carpineto and Giovanni Romano. *Concept data analysis - theory and applications.* Wiley, 2005.
4. Claudia d'Amato, Nicola Fanizzi, and Agnieszka Lawrynowicz. Categorize by: Deductive aggregation of semantic web query results. In Lora Aroyo, Grigoris Antoniou, Eero Hyvönen, Annette ten Teije, Heiner Stuckenschmidt, Liliana Cabral, and Tania Tudorache, editors, *ESWC (1)*, volume 6088 of *Lecture Notes in Computer Science*, pages 91–105. Springer, 2010.
5. Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis: Mathematical Foundations.* Springer, Berlin/Heidelberg, 1999.
6. J.-L. Guigues and V. Duquenne. Familles minimales d'implications informatives résultant d'un tableau de données binaires. *Mathématiques et Sciences Humaines*, 95:5–18, 1986.
7. Sergei O. Kuznetsov and Sergei A. Obiedkov. Comparing performance of algorithms for generating concept lattices. *J. Exp. Theor. Artif. Intell.*, 14(2-3):189–216, 2002.
8. Gerd Stumme, Rafik Taouil, Yves Bastide, and Lotfi Lakhal. Conceptual clustering with iceberg concept lattices. In R. Klinkenberg, S. Rüping, A. Fick, N. Henze, C. Herzog, R. Molitor, and O. Schröder, editors, *Proc. GI-Fachgruppentreffen Maschinelles Lernen (FGML'01)*, Universität Dortmund 763, October 2001.
9. Dean van der Merwe, Sergei A. Obiedkov, and Derrick G. Kourie. Addintent: A new incremental algorithm for constructing concept lattices. In Peter W. Eklund, editor, *ICFCA*, volume 2961 of *Lecture Notes in Computer Science*, pages 372–385. Springer, 2004.
10. 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.