Contextual Itemset Mining in DBpedia

Julien Rabatel[1], Madalina Croitoru[1], Dino Ienco[2], Pascal Poncelet[1]

[1]LIRMM, University Montpellier 2, France
[2] IRSTEA UMR TETIS, LIRMM, France

Abstract. In this paper we show the potential of contextual itemset mining in the context of Linked Open Data. Contextual itemset mining extracts frequent associations among items considering background information. In the case of Linked Open Data, the background information is represented by an Ontology defined over the data. Each resulting itemset is specific to a particular context and contexts can be related each others following the ontological structure.
We use contextual mining on DBpedia data and show how the use of contextual information can refine the itemsets obtained by the knowledge discovery process.

1 Introduction

We place ourselves in a knowledge discovery setting where we are interested in mining frequent itemsets from a RDF knowledge base. Our approach takes into account contextual data about the itemsets that can impact on what itemsets are found frequent depending on their context [16]. This paper presents a proof of concept and shows the potential advantage of *contextual* itemset mining in the Linked Open Data (LOD) setting, with respect to other approaches in the literature that *do not consider contextual information* when mining LOD data. We make the work hypothesis that the context we consider in this paper is the class type (hypothesis justified by practical interests of such consideration such as data integration, alignment, key discovery, etc.). This work hypothesis can be lifted and explored according to other contexts such as predicates, pairs of subjects and objects, etc. [1] as further discussed in Section 5.

Here we are not interested in the use of how the mined itemsets can be relevant for ontological rule discovery [15], knowledge base compression [12] etc. We acknowledge these approaches and plan to investigate how, depending on the way contexts are considered, we can mine different kind of frequent itemsets that could be further used for reasoning. Therefore, against the state of the art our contribution is:

- Taking into account contextual information when mining frequent itemsets on the Linked Open Data cloud. This allows us to refine the kind of information the mining process can provide.
- Introducing the notion of frequent contextual pattern and show how it can be exploited for algorithmic considerations.

We evaluate our approach on the DBpedia [13] dataset. In Section 2 we give a short example of the intuition of our approach. Section 3 explains the theoretical foundations of our work. In Section 4 we briefly present the DBpedia dataset and explain the obtained results. Section 5 concludes the paper.

2 Paper in a Nutshell

A semantic knowledge base is typically composed of two parts. The first part is the ontological, general knowledge about the world. Depending on the subset of first order logic used to express it this part is also called TBox (in Description Logics [4]), support (in Conceptual Graphs [8]) or rules (in Conceptual Graphs and rule based languages such as Datalog and Datalog+- [6]).

The second part is the factual knowledge about the data defining how the instances are in relation with each other. In Description Logics the factual knowledge is called ABox. Usually in the Linked Open Data, the factual knowledge is stored using RDF (usually in a RDF Triple Store) as triples "Subject Predicate Object". Recently, within the Ontology Based Data Access [7] the data (factual information) can also be stored in a relational databases.

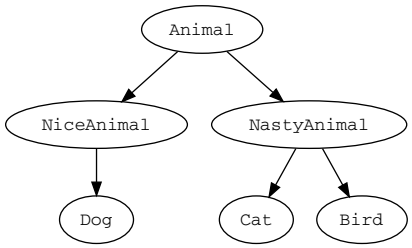
A study of the trade-off of using different storage systems (with equivalent expressivity) was recently done in [5]. DBpedia organises the data in three parts:

- The first part is the ontology (representing the TBox). The rules do not introduce existential variables in the conclusion (unlike existential rules as in Datalog+-) and they represent the Class Type hierarchy and the Predicate Type hierarchy. The ontology is guaranteed to be acyclic in the version of DBpedia we used. In the Directed Acyclic Graph (DAG) representing the ontology there are around 500 nodes, with around 400 being leaves. The maximal height is inferior to 10. The ontology we consider in the example in this section is depicted in Figure 1(b). We consider a six class ontology represented by a binary tree with height three. The algorithmic implications of the structure of the ontology are discussed in Section 5. Additionally, we consider the binary predicates "*playsWith*", "*eats*" and "*hates*" of signature "*(Animal, Animal)*".
- The second part is the mapping based types containing the instance type definition. In the example in this section, using the ontology defined in Figure 1(b), we consider the following mapping types (using a RDF triple notation): "*Bill hasType Dog*", "*Boule hasType Human*", "*Tom hasType Cat*", "*Garfield hasType Cat*" and "*Tweety hasType Bird*".
- The third part consists of the mapping based properties that correspond to the factual information. In the example here we consider the following facts: "*Bill eats Tweety*", "*Tweety hates Bill*", "*Bill playsWith Boule*", "*Tom eats Tweety*", "*Tom hates Bill*", "*Garfield hates Bill*", "*Garfield eats Tweety*", "*Garfield hates Boule*". These facts are summarized in Figure 1(a).

In this paper we make the choice of working with the data from the perspective of the Class of the Subject. As mentioned in the introduction we motivate

Subject	Predicate	Object
Bill	eats	Tweety
Tweety	hates	Bill
Bill	playsWith	Boule
Tom	eats	Tweety
Tom	hates	Bill
Garfield	hates	Bill
Garfield	eats	Tweety
Garfield	hates	Boule

(a) A fact base $\mathcal{F}$.



(b) An ontology $\mathcal{H}$.

Fig. 1: A knowledge base $KB = (\mathcal{F}, \mathcal{H})$.

tid	$\mathbf{I_{Animal}}$
$Bill$	$\{(eats, Tweety), (playsWith, Boule)\}$
$Tweety$	$\{(hates, Bill)\}$
Tom	$\{(eats, Tweety), (hates, Bill)\}$
$Garfield$	$\{(hates, Bill), (eats, Tweety), (hates, Boule)\}$

Fig. 2: The transactional database $\mathcal{T}_{KB,Animal}$ for the context *Animal* in the knowledge base KB depicted in Figure 1.

this initial choice from the perspective of various data integration tools on the Linked Open Data. One of the main challenges encountered by these tools is the mapping between various class instances. It is then not uncommon to consider the RDF database from a class type at a time.

If we consider this point of view then we model the couple "*(Predicate,Object)*" as an *item*, and the set of items associated with a given subject an *itemset*. The itemset corresponding to each distinct subject from $\mathcal{F}$ are depicted in Figure 2.

One may notice that 50% of the itemsets depicted in Figure 2 include the subset $\{(hates, Bill), (eats, Tweety)\}$, while 75% include {(hates,Bill)}.

But this is simply due to the fact that our knowledge base contains a lot of cats (that hate Bill). Actually, if we look closer, we notice that all cats hate Bill and all birds hate Bill but no dogs hate Bill. By considering this contextual information we could be more fine-tuned with respect to frequent itemsets.

3 Theoretical Foundations

The contextual frequent pattern (CFP) mining problem aims at discovering patterns whose property of being frequent is context-dependent.

This section explains the main concepts behind this notion for Linked Open Data.

We consider as input a knowledge base $KB = (\mathcal{F}, \mathcal{H})$ composed of an ontology $\mathcal{H}$ (viewed as a directed acyclic graph) and a set of facts $\mathcal{F}$.

The set of facts $\mathcal{F}$ is defined as the set of RDF triples of the form

$$(subject, predicate, object)$$

.

Each element of the triple is defined according to the ontology[1]. The ontology, also denoted the **context hierarchy** $\mathcal{H}$, is a directed acyclic graph (DAG), denoted by $\mathcal{H} = (V_{\mathcal{H}}, E_{\mathcal{H}})$, such that $V_{\mathcal{H}}$ is a set of vertices also called **contexts** and $E_{\mathcal{H}} \subseteq V_{\mathcal{H}} \times V_{\mathcal{H}}$ is a set of directed edges among contexts.

$\mathcal{H}$ is naturally associated with a partial order $<_{\mathcal{H}}$ on its vertices, defined as follows: given $c_1, c_2 \in V_{\mathcal{H}}$, $c_1 <_{\mathcal{H}} c_2$ if there exists a directed path from c_2 to c_1 in $\mathcal{H}$. This partial order describes a specialization relationship: c_1 is said to be more *specific* than c_2 if $c_1 <_{\mathcal{H}} c_2$, and more *general* than c_2 if $c_2 <_{\mathcal{H}} c_1$. In this case, c_2 is also called a subcontext of c_1.

A *minimal context* from $\mathcal{H}$ is a context such that no more specific context exists in $\mathcal{H}$, i.e., $c \in V_{\mathcal{H}}$ is minimal if and only if there is no context $c' \in V_{\mathcal{H}}$ such that $c' <_{\mathcal{H}} c$. The set of minimal contexts in $\mathcal{H}$ is denoted as $V_{\mathcal{H}}^{-}$.

Based on this knowledge base, we will build a **transactional database** such that each transaction corresponds to the set of predicates and objects of subjects of a given class. More precisely, given $KB = (\mathcal{F}, \mathcal{H})$ and $c \in V_{\mathcal{H}}$, the transactional database for c w.r.t. KB, denoted as $\mathcal{T}_{KB,c}$, is the set of transactions of the form $T = (tid, I_c)$ where $I_c = \{(pred, obj) | (s, pred, obj) \in \mathcal{F}$ *and* c is the class of $s\}$.

We define $\mathcal{I}_{\mathcal{H}}$ as the set $\{I_c | c \in V_{\mathcal{H}}\}$. In this paper, we are interested in itemset mining, thus a pattern p is defined as a subset of $\mathcal{I}_{\mathcal{H}}$.

Definition 1 (Pattern Frequency). *Let KB be a knowledge base, p be a pattern and c be a context, the frequency of p in $\mathcal{T}_{KB,c}$ is defined as $Freq(p, \mathcal{T}_{KB,c}) = \frac{|\{(tid,I) \in \mathcal{T}_{KB,c} | p \subseteq I\}|}{|\mathcal{T}_{KB,c}|}$.*

For the sake of readability, in the rest of the paper, $Freq(p, \mathcal{T}_{KB,c})$ is denoted by $Freq(p, c)$.

Definition 2 (Contextual Frequent Pattern). *Let KB be a knowledge base, p be a pattern, c be a context and σ a mininum frequency threshold. The couple (p, c) is a **contextual frequent pattern** (CFP) in KB if:*

- *p is frequent in c, i.e., $Freq(p, c) \geq \sigma$,*
- *p is frequent in every subcontext of c, i.e., for every context c' such that $c' <_{\mathcal{H}} c$, $Freq(p, c') \geq \sigma$,*

[1] Given the ontology we considered in *DBPedia*, the ontology is solely composed of a class hierarchy.

Additionally, (p, c) *is* ***context-maximal*** *if there does not exist a context* C *more general than* c *such that* (p, C) *is a contextual frequent pattern.*

Definition 3. *Given a user-specified mininum frequency threshold* σ *and a knowledge base* $KB = (\mathcal{F}, \mathcal{H})$*, the contextual frequent pattern mining problem consists in enumerating all the context-maximal contextual frequent patterns in* KB*.*

Th CFP mining problem is intrinsically different from the one addressed in [17, 14]. The CFP exploits a **context hierarchy** that define relationships over the contexts associated to each transaction while in [17, 14] the taxonomic information is employed to generalize the objects over which a transaction is defined.

3.1 Algorithm for computing CFPs

The above definitions provide us with a theoretical framework for CFP mining. In the rest of this section, we design the algorithm that extracts CFPs from *DBpedia*. This algorithm is inspired from the one that was proposed in [16] for mining contextual frequent sequential patterns (i.e., a variation of the frequent itemset mining problem where itemsets are ordered within a sequence [2]). We however handle itemsets in the current study and therefore have to propose an adapted algorithm. To this end, we propose to mine CFPs through post-processing the output of a regular frequent itemset miner.

Indeed, by considering the definition of a context-maximal CFP (*cf.* Definition 2), one could imagine how to extract them via the following easy steps: *(1)* extracting frequent patterns in every context of $\mathcal{H}$ by exploiting an existing frequent itemset miner, *(2)* for each context and each frequent itemset found in this context, check whether it satisfies the requirements of a CFP (i.e., checking whether it was also found frequent in the subcontexts, and whether it is context-maximal. This approach, while convenient for its straightforwardness, is inefficient in practice. Mining every context of the hierarchy can quickly become impractical because of the number of such elements. In addition, mining all the contexts of the hierarchy is redundant, as more general contexts contain the same elements as their subcontexts.

In order to tackle these problems, we propose to remove this redundancy by mining frequent patterns in minimal contexts of $\mathcal{H}$ only and building CFPs from the patterns found frequent in those only. In consequence, we define the *decomposition* notions, by exploiting the fact that a context can be described by its minimal subcontexts in $\mathcal{H}$. To this end, we consider the *decomposition* of a context c in $\mathcal{H}$ as the set of minimal contexts in $\mathcal{H}$ being more specific than c, i.e., $decomp(c, \mathcal{H}) = \{c' \in V_{\mathcal{H}}^{-} | (c' <_{\mathcal{H}} c) \vee (c' = c)\}$. Please notice that given this definition, the decomposition of a minimal context c is the singleton $\{c\}$.

Proposition 1. *Let* KB *be a knowledge base,* p *be a pattern and* c *be a context.* (p, c) *is a contextual frequent pattern in* KB *if and only if* p *is frequent in every element of* $decomp(c)$*.*

This proposition (whose proof can be found in [16] and adapted to the current framework) is essential by allowing the reformulation of the CFP definition w.r.t. minimal contexts only: a couple (p, c) is a CFP if and only if the set of minimal contexts where p is frequent includes the decomposition of c.

Extending this property to context-maximal CFPs is straightforward. The algorithm we use to extract context-maximal CFPs in *DBpedia* data can be decomposed into the following consecutive steps:

1. **Mining.** Frequent patterns are extracted from each minimal context. At this step, by relying on Proposition 1, we do not mine non-minimal contexts. The frequent itemset miner employed to perform this step is an implementation of the *APriori* algorithm [3] provided in [10].

2. **Reading.** Output files from the previous step are read. The patterns p are indexed by the set of minimal contexts where they are frequent, denoted by l_p. Then, we initialize a hash table K as follows. The hash table keys are the sets of minimal contexts and the hash table values are the sets of patterns such that $K[l]$ contains the patterns p such that $l_p = l$. The hash table K, at the end of this step, thus stores all the patterns found frequent in at least one minimal context during the previous step. The patterns are indexed by the set of minimal contexts where they are frequent.

3. **CFP Generation.** During this step, each key l of K is passed to a routine called *maxContexts* which performs a bottom-up traversal of the vertices of $\mathcal{H}$ in order to return the set of maximal contexts among $\{c \in V_{\mathcal{H}} \mid decomp(c) \subseteq l\}$. Such contexts satisfy the Proposition 1. Then, for each pattern p such that $l = l_p$ and each context returned by the *maxContexts* routine, one context-maximal CFP is generated and stored. Two patterns p and p' frequent in the same minimal contexts (i.e., $l_p = l_{p'}$) are general in the same contexts. They will generate the same result via the *maxContexts* routine. By using a hash table K to store the patterns that are frequent in the same minimal contexts, the number of calls to *maxContexts* is greatly reduced to the number of keys in K rather than the number of distinct patterns discovered during the *mining* step.

4 Experimental Results

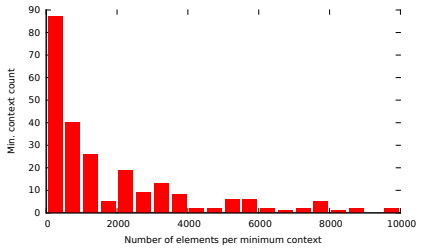
In this section, we describe the results obtained through discovering contextual frequent patterns in the *DBpedia* dataset. All experiments have been conducted on an Intel i7-3520M 2.90GHz CPU with 16 GB memory. The rest of the section is organized as follows. First, we comment the quantitative aspects of the evaluation. Second, we show and explain some examples of contextual frequent patterns found in the data.

Quantitative evaluation. In order to apply the mining algorithm to the *DBpedia* data, we pre-process them by removing all the contexts associated to less than 10

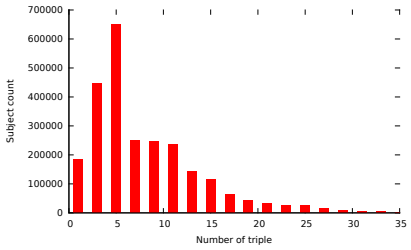
elements. The obtained contextual hierarchy contains 331 contexts, out of whic 278 are minimal. The intuition behind this pre-processing step is that extracting frequent patterns from contexts that contain a very small amount of elements is statistically insignificant and can lead to noisy results.

After the above-mentioned pre-processing, the data has the following features. The whole database contains a total of $2,501,023$ transactions. The number of elements per minimal context (i.e., the partition of the subjects over the classes) is naturally unbalanced in the *DBpedia* data. Indeed, a minimal context contains in average 8996 ± 40383 elements, with a minimum of 12 and a maximum of 577196. Figure 3(a) depicts how subjects are distributed over the minimal contexts and shows that the majority of minimal contexts contains less than 2000 transactions.

Similarly, the repartition of triples regarding their associated subjects is unbalanced. A subject in the *DBpedia* data we considered is associated in average with 7.43 ± 6.54 triples, with a maximum amount of 821 triples per subject. Please notice that this is equivalent to the average count of items per itemset in our contextual database. Figure 3(b) shows the repartition of the subjects in the data according to the number of triples they are asociated with.



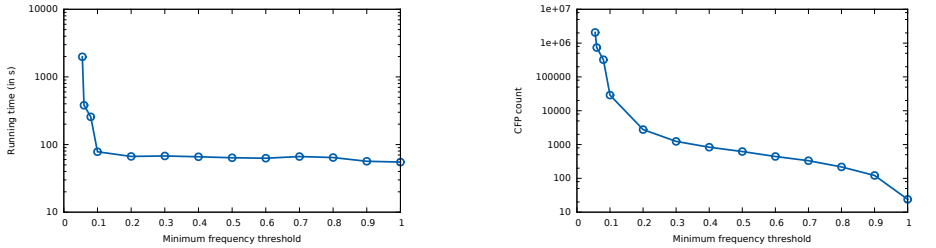
(a) Histogram depicting the repartition of elements per minimum context.



(b) Histogram depicting the repartition of triples associated to a given subject.

Fig. 3: Elements of data repartition in *DBpedia*.

Figure 4(a) shows the runtime required to discover contextual frequent patterns in the whole database according to the minimum frequency threshold. The proposed approach is shown to be scalable regarding the *DBpedia* data. The runtimes are indeed lower than 100 seconds for minimum frequency thresholds lower than 0.1. Unsurprisingly, the required time becomes much higher with low minimum frequency thresholds (around 5%). As shown in Figure 4(b), decreasing the minimum frequency threshold also provokes a higher number of discovered CFPs (more than $1,000,000$ for a minimum frequency threshold of 5%). This global behavior regarding the user-specified minimum frequency threshold is typical of frequent pattern miners.



(a) Running time w.r.t. the minimum frequency threshold.

(b) Contextual frequent pattern count w.r.t. the minimum frequency threshold.

Fig. 4: Elements of data repartition in the considered subset of *DBpedia*.

CFP examples and interpretation. CFPs have the ability to describe how (predicate,object) couples can be correlated to a class. Some examples can be found in minimal contexts of the hierarchy $\mathcal{H}$. For instance, the CFP $(\{(location, UnitedStates)\}, WineRegion)$ with a frequency of 72% in the minimal context $WineRegion$ means that "*72% of wine regions described in DBpedia are located in the United States*". Similarly, the CFP $(\{(BirthPlace, England)\}, DartsPlayer)$ with a frequency of 33% in the minimal context $DartsPlayer$ shows that "*33% of the darts players in DBpedia were born in England*". Hence, mining CFPs has the ability to describe frequent patterns in every minimal context of $\mathcal{H}$. Such CFPs, because they are associated to minimal contexts, bring other information. All extracted CFPs are context-maximal (*cf.* Definition 3). As a consequence, they also bring an additional piece of information to help an expert interpretating the results. For instance, $(\{(BirthPlace, England)\}, DartsPlayer)$ being context-maximal also indicates that $(\{(BirthPlace, England)\}, Athlete)$ is not a CFP. In other terms, the fact that the itemset $\{(BirthPlace, England)\}$ is frequent in the context $DartsPlayer$ does not hold in all the other subcontexts of $Athlete$ (such subcontexts include $TennisPlayer$, $Wrestler$, $Cyclist$, etc.).

Previous examples describe facts associated to minimal contexts only. However, the contextual frequent pattern mining problem also aims at describing how such facts can be lifted to more general contexts. A simple example can be found in the context $MusicGroup$, which has two subcontexts $Band$ and $MusicalArtist$. With a minimum frequency threshold of 10%, the context-maximal CFP $(\{hometown, UnitedStates\}, MusicGroup)$ is discovered. This pattern brings several pieces of information:

- more than 10% of $MusicGroup$ from *DBpedia*, i.e., bands or musical artists, have their hometown in the United States. More precisely, the algorithm also provides us with the exact frequency of this pattern in $MusicGroup$, i.e., 15.3%;
- this fact also holds for subcontexts of $MusicGroup$: more than 10% of bands and more than 10% of musical artists have their hometown in the United States;

– no context in $\mathcal{H}$ more general than $MusicGroup$ can be associated with the itemset $\{hometown, UnitedStates\}$ to form a CFP.

CFPs hence have the ability to describe the facts contained in *DBpedia* regarding their frequency, but also how the property of being frequent can be generalized or not in the whole context hierarchy.

5 Discussion

RDF data constitutes a rich source of information and, recently, data mining community starts to adapt its methods to extract knowledge from such kind of data [18]. In particular, preliminary works in this direction employ pattern mining techniques in order to extract frequent correlation and meta information from RDF dataset. In [9] the authors propose to use association rule mining (not using any contextual information) in order to enrich RDF schema with property axioms. The properties axioms are automatically induced by means of a pattern mining step. These axioms can be directly used as meta-data to index the RDF dataset.

[12] exploits pattern mining in order to compress RDF data. In this work the authors apply well known association rule mining algorithms to induce rules that cover the information in the RDF dataset. Once the rules are extracted, the RDF dataset is compressed considering the set of rules plus the not covered RDF triples. Both previous approaches did not employ any additional background knowledge (such as taxonomy). This information can be useful in order to exploit existing relationships among the data.

A first attempt in this direction is presented in [11]. In this work an RDF triple is an item and taxonomical information is directly employed to generalise subjects or objects at item level. Differently from this strategy, our approach allows to characterise (by means of the extracted itemsets) a node of the taxonomy supplying cues about how the extracted knowledge can be organised for its analysis. Since the taxonomy nodes we exploit (the contexts) are classes we could have presented our contribution directly from an RDF class view point. This choice would have meant that the conceptual link with the notion of context in the data mining setting was lost. Thus we preferred to keep the context terminology. Let us mention here that the shape of the ontology changes the efficiency of the algorithm. Due to the nature of our algorithm and pruning method we have better results if the DAG is not large but has a big height. We plan to apply our method to more specialised ontologies where such property is satisfied.

As explained above the mined itemsets could be used for inference rules. For instance, if in the context Cat the itemset $\{hates, Bill\}$ is frequent we can view this as a rule $\forall x Cat(x) \rightarrow hates(x, Bill)$ that hold in the knowledge base at a given time with a given confidence. We are currently working towards providing such itemsets with logical semantics and see what the frequency means in this case.

Another issue when considering the itemsets as rules is how to deal with their number, and how to order them in order to be validated by an expert.

One possible avenue we can explore is to generalise instances to concepts and try to extract less rules but more meaningful. For example we can generalise Bill to Dog and mine the rule $\forall x Cat(x) \rightarrow \exists y Dog(y) \wedge hates(x, y)$. The question is where to put the tradeoff between expressivity and number.

Finally let us mention that changing the context (not considering the class as a context but the couple (subject, object)) we could mine interesting rules about predicates. For instance we could get that $\forall x \forall y Cat(x) \wedge Dog(y) \rightarrow hates(x, y)$.

References

1. Z. Abedjan and F. Naumann. Context and target configurations for mining rdf data. In *Workshop on Search and mining entity-relationship data*, pages 23–24. ACM, 2011.
2. R. Agrawal and R. Srikant. Mining sequential patterns. In *ICDE*, pages 3–14. IEEE, 1995.
3. R. Agrawal, R. Srikant, et al. Fast algorithms for mining association rules. In *VLDB*, volume 1215, pages 487–499, 1994.
4. F. Baader. *The description logic handbook: theory, implementation, and applications*. Cambridge university press, 2003.
5. J. Baget, M. Croitoru, and B. P. L. Da Silva. Alaska for ontology based data access. In *ESWC (Satellite Events)*, pages 157–161. Springer, 2013.
6. A. Cali, G. Gottlob, T. Lukasiewicz, B. Marnette, and A. Pieris. Datalog+/-: A family of logical knowledge representation and query languages for new applications. In *LICS*, pages 228–242. IEEE, 2010.
7. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, and R. Rosati. Ontology-based database access. In *SEBD*, pages 324–331, 2007.
8. M. Chein and M. Mugnier. *Graph-based knowledge representation: computational foundations of conceptual graphs*. Springer, 2008.
9. D. Fleischhacker, J. Völker, and H. Stuckenschmidt. Mining rdf data for property axioms. In *OTM Conferences (2)*, pages 718–735, 2012.
10. P. Fournier-Viger, A. Gomariz, A. Soltani, H. Lam, and T. Gueniche. Spmf: Open-source data mining platform. http://www.philippe-fournier-viger.com/spmf, 2014.
11. T. Jiang and A. Tan. Mining rdf metadata for generalized association rules: knowledge discovery in the semantic web era. In *WWW*, pages 951–952, 2006.
12. A. K. Joshi, P. Hitzler, and G. Dong. Logical linked data compression. In *ESWC*, pages 170–184, 2013.
13. J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, and C. Bizer. DBpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web Journal*, 2014.
14. R. A. Lisi and D. Malerba. Inducing multi-level association rules from multiple relations. *Machine Learning Journal*, 2(55):175–210, 2004.
15. A. Maedche and S. Staab. Ontology learning for the semantic web. *IEEE Intelligent systems*, 16(2):72–79, 2001.
16. J. Rabatel, S. Bringay, and P. Poncelet. Contextual sequential pattern mining. In *(ICDMW), ICDM*, pages 981–988. IEEE, 2010.
17. R. Srikant and R. Agrawal. Mining generalized association rules. In *VLDB*, pages 407–419, 1995.
18. G. Stumme, A. Hotho, and B. Berendt. Semantic web mining: State of the art and future directions. *J. Web Sem.*, 4(2):124–143, 2006.