Building Concept Lattices by Learning Concepts from RDF Graphs Annotating Web Documents

Alexandre Delteil, Catherine Faron, Rose Dieng

Building Concept Lattices by Learning Concepts from RDF Graphs Annotating Web Documents

Alexandre Delteil[1], Catherine Faron[2], and Rose Dieng[1]

[1] INRIA, Acacia project
2004 route des Lucioles, BP93, 06902 Sophia Antipolis cedex, France
{Alexandre.Delteil,Rose.Dieng}@sophia.inria.fr
[2] I3S, Mainline project
930 route des Colles, BP145, 06903 Sophia Antipolis cedex, France
faron@essi.fr

Abstract. This paper presents a method for building concept lattices by learning concepts from RDF annotations of Web documents. It consists in extracting conceptual descriptions of the Web resources from the RDF graph gathering all the resource annotations and then forming concepts from all possible subsets of resources - each such subset being associated with a set of descriptions shared by the resources belonging to it. The concept hierarchy is the concept lattice built upon a context built from the power context family representing the RDF graph. In the framework of the CoMMA European IST project dedicated to ontology-guided Information Retrieval in a corporate memory, the hierarchy of the so learned concepts will enrich the ontology of primitive concepts, organize the documents of the organization's Intranet and then improve Information Retrieval. The RDF Model is close to the Simple Conceptual Graph Model; our method can be thus generalized to Simple Conceptual Graphs.

1 Introduction

The Semantic Web is expected to be the next step that will lead the Web to its full potential [2]. It is based on the description of all kinds of Web resources with semantic metadata. The Resource Description Framework (RDF) [12] is the emerging standard to annotate Web documents with such metadata. These annotations are related to ontologies, declared in RDF Schema [13]. RDF(S) is very close to the Simple Conceptual Graph Model, and the work on one formalism can be easily generalized to the other.
The research presented in this paper takes place in the framework of the CoMMA European IST project dedicated to ontology-guided Information Retrieval in a corporate memory. This corporate memory is constituted by documents semantically described by RDF annotations. We propose a method for learning concepts and extracting knowledge to manage the amount of information available in the documents of the memory.
The building of hierarchical structures from structured data has been extensively

studied in machine learning, especially in concept formation. Most approaches of concept formation are dedicated to the prediction of unknown features of new objects [7] [10]. The clusters of similar objects are then privileged, the learned concept hierarchy does not comprise all the possible sets of objects, but only the best ones according to some heuristic criteria.
We adopt a particular approach of concept formation, where each concept is defined in extension by a subset of resources and in intension by a set of descriptions shared by these resources. In this approach, all the possible subsets of objects are systematically considered, as in [16] [4] [3]. Given the RDF graph gathering all the annotations we consider, we build a concept lattice upon a context built from the power context family representing this RDF graph.
In the following section, we briefly describe the RDF data model and the RDF Schema and we present several criteria for extracting partial resource descriptions from RDF annotations. We then present the principles of our approach of concept formation that deals with the intrinsic complexity of the building of a generalization hierarchy: we propose an incremental approach by gradually increasing the size of the descriptions we consider. We then formally describe the building of a concept lattice from a context built upon the power context family representing the RDF graph we consider. Finally, we show how the learned concept hierarchy will be exploited in the framework of the CoMMA project.

2 From Document Annotations to Conceptual Descriptions

In the framework of the CoMMA project, the documents building up the corporate memory are annotated by semantic metadata. These document annotations are based on domain ontologies and then enable knowledge-based Information Retrieval. With the growth of the Semantic Web, the development of methods to exploit the document annotations will become of prime importance. We address the problem of learning concepts from the semantic annotations of documents to organize the documents of a corporate memory into a conceptual hierarchy, to enrich the ontology on which the annotations are constructed with this concept hierarchy, and finally to improve Information Retrieval on the corporate memory.

2.1 The RDF(S) Data Model

The RDF annotation of a Web resource consists of a set of statements, each one specifying a value of a property of the resource. A statement is thus a triple (resource, property, value), a value being either a resource or a literal. Resources are either identified or anonymous but are uniformly handled by RDF parsers which generate new identifiers for anonymous resources. The RDF data model is close to semantic nets. A set of statements is viewed as a directed labeled graph: a vertex is either a resource or a literal; an arc between two vertices is labeled by a property.

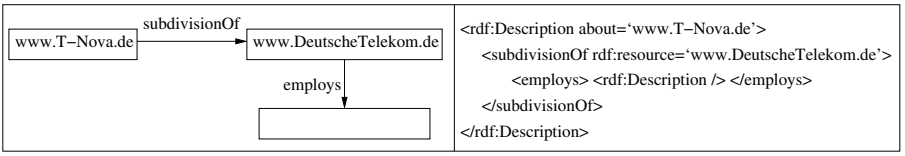

Fig. 1. An example of an RDF annotation

Figure 1 presents an example of an RDF graph with its corresponding XML syntax. This annotation describes the Web page relative to the company Deutsche Telekom. All the examples illustrating our article stem from the O'CoMMA ontology [8].

An RDF annotation is a set of RDF triples. It can thus be viewed as a graph, which is a subgraph of the complete RDF graph representing the whole set of annotations on the Semantic Web.

RDF Schema (RDFS) is a schema specification language [13]. It is dedicated to the specification of schemas representing the ontological knowledge used in RDF statements: a schema consists of a set of declarations of classes and properties. Multi-inheritance is allowed for both classes and properties. A property is declared with a signature allowing several domains and a single range. The RDFS metamodel is presented in Figure 2 and is itself defined as a set of statements by using the core RDFS properties: rdfs:subclassOf and rdf:type which denote respectively the subsumption relation between classes and the instantiation relation between an instance and a class.

As shown in Figure 2, an ontology embedding domain-specific knowledge is represented by a schema defined by refining the core RDFS. Domain-specific classes are declared as instances of the 'Class' resource, and domain-specific properties as instances of the 'Property' resource. The 'subclassOf' and 'subPropertyOf' properties enable to define class hierarchies and property hierarchies. The resources appearing in an RDF annotation are then typed by the

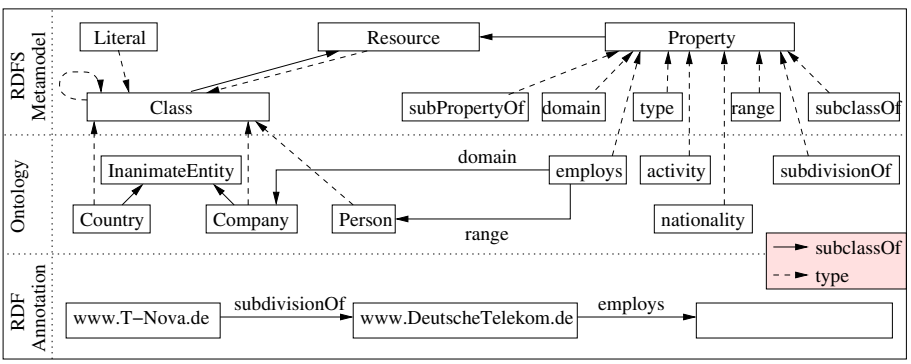

Fig. 2. The RDFS metamodel and an RDFS ontology

classes declared in the RDF schema the annotation is relative to; the properties between the resources are those declared in the RDF schema.

2.2 The RDF(S) Model and the Conceptual Graphs Model

The RDF(S) Model and the Simple Conceptual Graph Model have similar expressiveness: both correspond to the positive, existential and conjunctive subfragment of first order logic, and both enable to use sentences as objects of the language. In [5], a way to translate RDFS Schemas into CG supports and RDF triples into CGs is presented. In [6], a detailed description of similarities and differences between both formalisms is presented. The main differences between RDF(S) and CGs concern the way constraints on property domains are handled (a signature for a relation in the CG Model, several domains but only one range for a property in the RDF Model) and the way membership to a class is expressed (by a concept type in the CG Model and by a specific property called *type* in the RDF Model. However these differences are not fundamental, and our method described in this paper can be generalized easily to Simple Conceptual Graphs.

2.3 Extracting Conceptual Descriptions of Web Resources

Regarding the RDF model, the knowledge base representing the resource annotations consists of a single graph G. There is no difference between stating a resource description in one annotation and stating it in several pieces in separate annotations: 'there is no distinction between the statements made in a single sentence and the statements made in separate sentences' [12].
Learning concepts from RDF annotations requires resource descriptions to be given. As the RDF model does not handle the delimitation of a subgraph of G describing a resource, we introduce the notion of description of length n of a resource.

Definition 1 (Description of Length n). *The description of length n of a resource R is the largest connected subgraph of G containing all possible paths of length smaller or equal to n, starting from or ending to R. It is noted $D_n(R)$. It is inductively obtained by joining $D_{n-1}(R)$ with the descriptions D_1 of length 1 of the resources which are external nodes of $D_{n-1}(R)$.*

Figure 3 presents the extraction of two possible descriptions of the resource Deutsche Telekom from the whole RDF graph which the RDF annotation of Figure 1 participates to: the description of length 1 of Deutsche Telekom and the description of length 2 of Deutsche Telekom. D_1 (Deutsche Telekom) is a subgraph of D_2 (Deutsche Telekom) which is made of paths of length 1 and of length 2 starting from or ending to the resource Deutsche Telekom.
Given the whole RDF graph G, we can now be provided with a set of partial descriptions for all the resources that are nodes of G (in the example Deutsche Telekom, TNova, Germany and an anonymous resource of type 'Person').

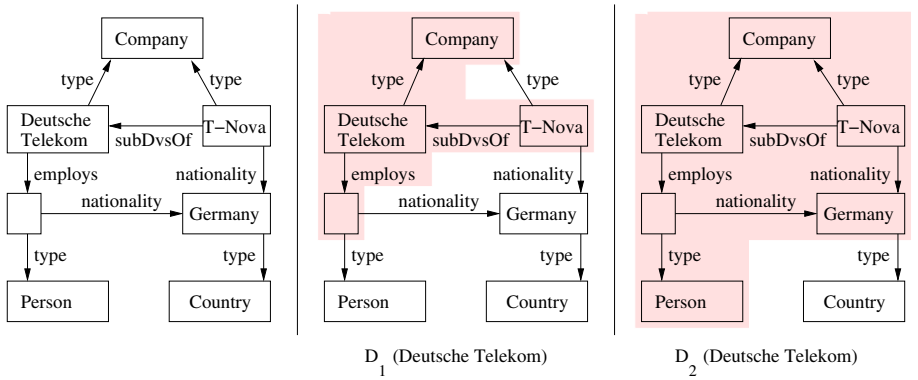

Fig. 3. The RDFS metamodel and an RDFS ontology

3 Learning Concepts from Conceptual Descriptions

Our approach of knowledge capture consists in learning new domain-specific concepts from the whole RDF graph G comprising the resources participating to a given corporate memory.

3.1 Systematic Conceptual Clustering

To learn concepts from RDF metadata, we adopt an approach of concept formation. Concept formation or conceptual clustering aims at building hierarchies to cluster similar objects and classify object descriptions. However in these approaches a single particular hierarchy of classes is built, the best according to a given criterion. Our approach of concept formation is slightly different since it aims at systematically generating a class for each possible set of objects. This systematic approach is shared by researches in formal concept analysis [17] and on knowledge organization [16] [3].

Given an RDF graph G and a resource description extraction criterion, let us consider the set of the descriptions of all the resources nodes of G. Our approach consists in associating to this set of descriptions a hierarchy of concepts whose extensions correspond to all the possible subsets of the set of resources of G. All the concepts covering a set of resources of G are systematically considered. A concept is defined in extension as a set of resources; its definition in intension is the set of all the descriptions satistied by all the resources in its extension; this concept description language is presented in the following section. Each concept c_i of the hierarchy is thus a pair (ext_i, int_i), where ext_i is the extension of c_i and int_i is its intension. This concept hierarchy is a lattice: its nodes are partially ordered by the inclusion relation on their extensions, as well as on their intensions.

3.2 The Concept Description Language

The concept description language is close to the object description language. A description is a path of RDF triples and a concept intension is a set of such triple pathes of length less or equal to the length chosen for the object descriptions considered. For readability, a concept intension can be presented as a graph in normal and non redundant form built by join of the descriptions belonging to this intension. Let us note that, regarding subsumption on RDF graphs relying upon the subsumptions relations between classes and properties declared in the associated RDF schema, such a graph representing a concept intension subsumes the object descriptions of all the resources belonging to the extension of the concept considered. Moreover, since anonymous resources are handled like identified ones in resource descriptions, in case of a concept extension is a singleton, the graph built from the concept intension is equivalent (under subsumption) to the description of the single resource.

Figure 4 presents the concept hierarchy built from descriptions of length 1 of four resources nodes of the RDF graph depicted in Figure 3: Deutsche Telekom, TNova, Germany and the anonymous resource of type 'Person'. The graph representing the intension of each concept is built by join of all the triples of the concept intension, satisfied by the resources in its extension. The intension of the bottom concept is the set of all the triples that describe at least one resource; it is not depicted in the Figure.

3.3 Incremental Principle

The question which now arises is the choice of a resource description extraction criterion: starting from an RDF graph, we must choose from which partial resource descriptions the concept hierarchy will be built. On the one hand, the larger the extracted resource descriptions will be, the more domain-specific the concepts will be. On the other hand, graph matching has a well-known intrinsic exponential complexity.

As a result, we adopt an incremental approach for the construction of the concept hierarchy to deal with the intrinsic complexity of description matching. A similar approach of incremental building of a concept hierarchy is adopted in [3]. It is based on a gradual increase of the structure of matching. Object descriptions are given and the concept description language is made more expressive at each step to gradually take into account the complexity of the object descriptions. In our approach, the incrementality is based on the gradual increase of the size of the structure of matching -and not its structure: the resource descriptions are not given in the RDF graph, they are partial and their length is gradually increased. To be precise, we first build a concept hierarchy H_1 from resource descriptions of length 1. The concepts of H_1 thus have intensions of length 1. H_n is then inductively built from H_{n-1} and H_1 by incrementally increasing the maximum length of the resource descriptions we consider. The description $D_n(R)$ of length n of a resource R is inductively increased by joining $D_{n-1}(R)$ with the descriptions of length 1 of the resources which are external nodes of $D_{n-1}(R)$.

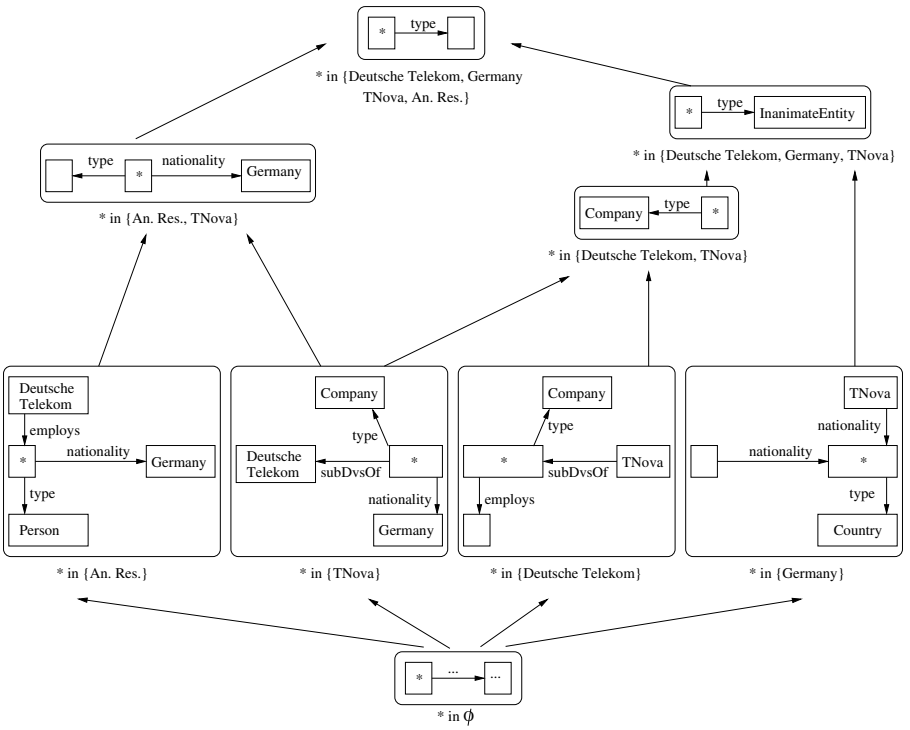

Fig. 4. The concept hierarchy associated to descriptions of length 1 extracted from the RDF graph of Figure 3

3.4 Resource Exploration and Size of the Concept Hierarchy

If several sets of resources share the same intension, a single concept is added to the hierarchy: the one having for extension the largest set of resources. Therefore, if the size of the concept hierarchy may theoretically reach 2^N concepts for N resources in the RDF graph G, it is in practice much lower. For instance, the size of the hierarchy of Figure 4 is 9 concepts instead of 16 (2^4).

We avoid the computation of concept descriptions for all the subsets of the set of resources of G that do not lead to concepts: those that would share a same set of descriptions with a larger subset of resources. To do this, the subsets of resources are considered according to a total order that enables to memorize those which do not correspond to maximal subsets: for each of the non maximal subsets, the complementary subset of resources necessary to build a maximal set are memorized. This is adapted from an algorithm proposed in formal concept analysis [17] for attribute exploration [9] [1].

Once a concept is created, it is inserted in the concept hierarchy under construction. To deal with the intrinsic complexity of the classification of a concept into a hierarchy, we take advantage of the order according to which the resources are

considered to limit the comparison of the concept to be classified. This order ensures that when inserting a concept, there will be no concept in the hierarchy that subsumes it.

4 Incremental Building of a Concept Hierarchy

4.1 Building of a Concept Hierarchy Based on Resource Descriptions of Length 1

In this section we formally describe the principle for building a concept hierarchy H_1 of concepts with sets of triples as intension as the building of a concept lattice from a context, the RDF graph from which the resource descriptions are extracted being viewed as a power context family.

Given an RDF graph G, let O be the set of resources in G which are not classes in the RDF schema upon which G is built, C and P the set of classes and the set of properties in the RDF schema of G, and P^{-1} the set of property inverses associated to P. We represent G by a power context family $(\mathcal{C}, \mathcal{P})$ where $\mathcal{C} = (O, C, I_{\mathcal{C}})$ and $\mathcal{P} = (O^2, P \cup P^{-1}, I_{\mathcal{P}})$.
In our concept description language, a triple t is a triplet (r, p, v) with $r \in O \cup \{*\} \cup \{\text{`}\emptyset\text{'}\}$, $p \in \mathcal{P} \cup \mathcal{P}^{-1}$ and $v \in O \cup \{\text{`}\emptyset\text{'}\} \cup C$; `$\emptyset$' denotes a resource that is unidentified (whereas in the object descriptions anonymous resources are provided with an identifier generated by the RDF parser). A triple path is a sequence of triples whose first triple $(*, p, v)$ has for first resource a star $*$, that designates any resource the triple path is a description of, and such that for all consecutive triples t_i and t_{i+1}, $v_i = r_{i+1}$.
To build a concept lattice from $(\mathcal{C}, \mathcal{P})$, we first build a set of triple pathes T_1 defined as follows:

- if (r_1, p, r_2) is a triple of G, then $(*, p, r_2) \in T_1$ and $(*, p^{-1}, r_1) \in T_1$,
- if $(*, type, c) \in T_1$ and $(*, type, c') \in T_1$, then $(*, type, c'') \in T_1$, for all $c'' \in C$ most specific subsumer of c and c',
- if $(*, p, r) \in T_1$ and $(*, p, r') \in T_1$ with $r \neq r'$, then $(*, p, \text{`}\emptyset\text{'}) \in T_1$,
- if $(*, p, r) \in T_1$ and $(*, p', r) \in T_1$ with $r \in O \cup \{\text{`}\emptyset\text{'}\}$, then $(*, p'', r) \in T_1$ for all p'' most specific subsumer of p and p',

We then build two contexts $\mathcal{C}_1 = (O, T_1, I_1)$ and $\mathcal{C}'_1 = (O^2, T_1, I'_1)$ with I_1 and I'_1 defined as follows:
$I_1 = \{(o, t),\ t = (*, type, c) \text{ and } (o, c) \in I_C\} \cup \{(o, t),\ t = (*, p, v) \text{ and } ((o, v), p) \in I_P\} \cup \{(o, t),\ t = (*, p, \text{`}\emptyset\text{'}) \text{ and } \exists o' \in O\ ((o, o'), p) \in I_P\}$,
$I'_1 = \{((o, o'), t),\ t = (*, p, \text{`}\emptyset\text{'}) \text{ and } (o, (*, p, o')) \in I_1\}$.

Finally, H_1 is the concept lattice built from $\mathcal{C}_1$.

Let us apply this principle on the RDF graph depicted on Figure 3. $O = \{DeutscheTelekom, TNova, An.Res., Germany\}$, $C = \{Country, Person, Company\}$, and $P = \{nationality, employs, subDivisionOf, type\}$. $\mathcal{C}_1$ is represented in Figure 5.

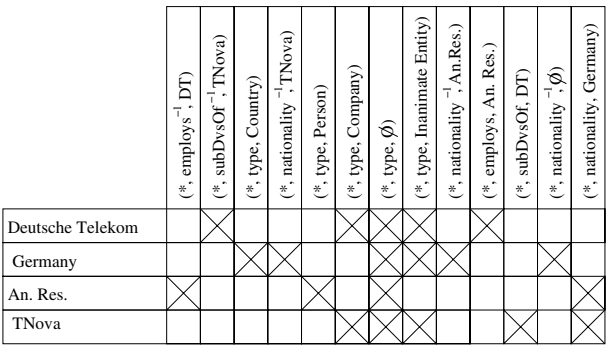

Fig. 5. The context C_1 upon which H_1 represented in Figure 4 is built

T_1 is built from the triples extracted from G and matched two by two. For instance, the triples (An. Res., nationality, *) and (TNova, nationality, *) of G lead to the triple (∅, nationality, *), TNova being incomparable with the identifier generated by the RDF parser for 'An. Res.'; (*, type, Company) and (*, type, Country) lead to the triple (*, type, Inanimate Entity), 'Inanimate Entity' being one of the most specific classes subsuming 'Company' and 'Country'. Note that RDFS allows for multi-inheritance on class and property hierarchies. Therefore two classes or two properties may have several most specific subsumers; in such cases, the generalization of two triples may lead to several triples.

The concept hierarchy H_1 in Figure 4 represents the concept lattice built from the context C_1 depicted in Figure 5. Concept intensions are represented as graph in normal and non redundant form built by join of the triples in the intensions. Let us note that the non maximal subsets of resources have been discarded: for instance, the concept ({Deutsche Telekom, TNova}, {(*, type, Inanimate Entity)}) is not created since the concept whose extension is the set {Deutsche Telekom, TNova, Germany} shares the same intension. A naive algorithm for the construction of H_1 from C_1 would be to consider every possible extensions, for each one of them to compute its intension and then to discard those of the learned concepts that are not of maximal extension. However this may be very unefficient in most practical cases where a lot of concepts are expected to be discarded. Among several algorithms to build a lattice from propositional data, we chose the one proposed by [9] that is much more efficient in the general case.

4.2 Building of a Concept Hierarchy Based on Resource Descriptions of Length n

The principle for building a concept hierarchy H_n of length n from H_{n-1} and H_1 consists in an iterative construction of a set T_n of triple pathes of length n by join of all the possible pairs of one triple path of length $n-1$ of T_{n-1} and one triple (triple path of length 1) of T_1. Two triple pathes can be joined if the value in the

last triple of the first path is equal to the resource described in the first triple of the second path. This iterative building of T_n is equivalent to considering resource descriptions $D_n(R)$ of length n by joining $D_{n-1}(R)$ and $D_1(R_i)$, with $i = 1 \ldots k$, R_i being the external nodes of $D_{n-1}(R)$.

Formally, T_n is defined as follows: $T_n = \{t | (\rho(t), p, r)$ with $t \in T_{n-1}$, $p \in P \cup P^{-1}$ and $r \in O \cup C$ and ρ the function which associates to a triple path t the value of its last triple$\}$.

We then inductively build two contexts $\mathcal{C}_n = (O, T_n, I_n)$ and $\mathcal{C}'_n = (O^2, T_n, I'_n)$ whose attributes are in T_n from the two contexts $\mathcal{C}_{n-1} = (O, T_{n-1}, I_{n-1})$ and $\mathcal{C}'_{n-1} = (O^2, T_{n-1}, I'_{n-1})$. I_n and I'_n are defined as follows:
$I_n = \{(o, t_{n-1} | (\rho(t_{n-1}), type, c)), (o, t_{n-1}) \in I_{n-1}$ and $\exists o' \in O$, $(o', (*, type, c)) \in I_1$ and $((o, o'), t_{n-1}) \in I'_{n-1}\} \cup \{(o, t_{n-1} | (\rho(t_{n-1}), p, r)), (o, t_{n-1}) \in I_{n-1}$ and $\exists o' \in O$, $(o', (*, p, r)) \in I_1$ and $((o, o'), t_{n-1}) \in I'_{n-1}\}$,
$I'_n = \{((o, o'), t_{n-1} | (\rho(t_{n-1}), p, \text{`}\emptyset\text{'})), (o, t_{n-1} | (\rho(t_{n-1}), p, o')) \in I_n\}$.

Finally, H_n is the concept lattice built from the context $\mathcal{C}_1 + \ldots + \mathcal{C}_n$.

Let us apply this principle to build the concept hierarchy H_2 depicted in Figure 7. The building of the context C_2 used to build H_2 is represented in Figure 6. To build C_2, the triples of the context C_1 depicted in Figure 5 are joined one with another (in the general case, the triple pathes of C_{n-1} would be joined with the ones of C_1). For instance, the triple (*, nationality, Germany) is joined with the triple (*, type, Country) since the value of the former triple is equal to a resource belonging to the extension of the second one. The join results in a triple path of length 2 (*, nationality, Germany) (Germany, type, Country), whose extension is equal to the extension of the former triple.

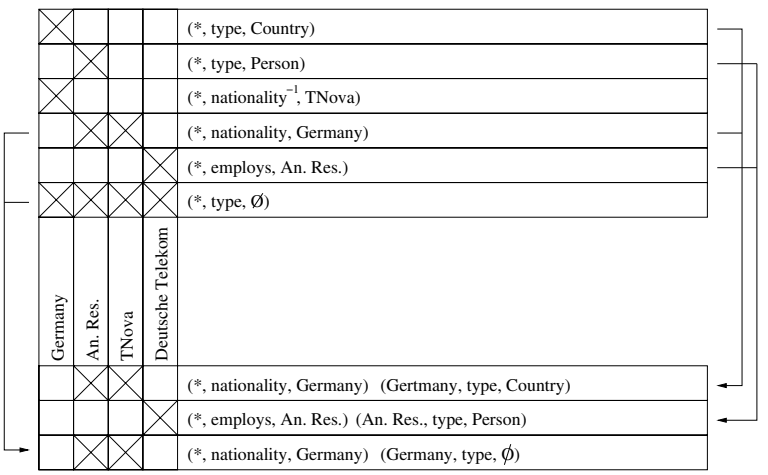

Fig. 6. Building of the context C_2 from C_1 depicted in Figure 5

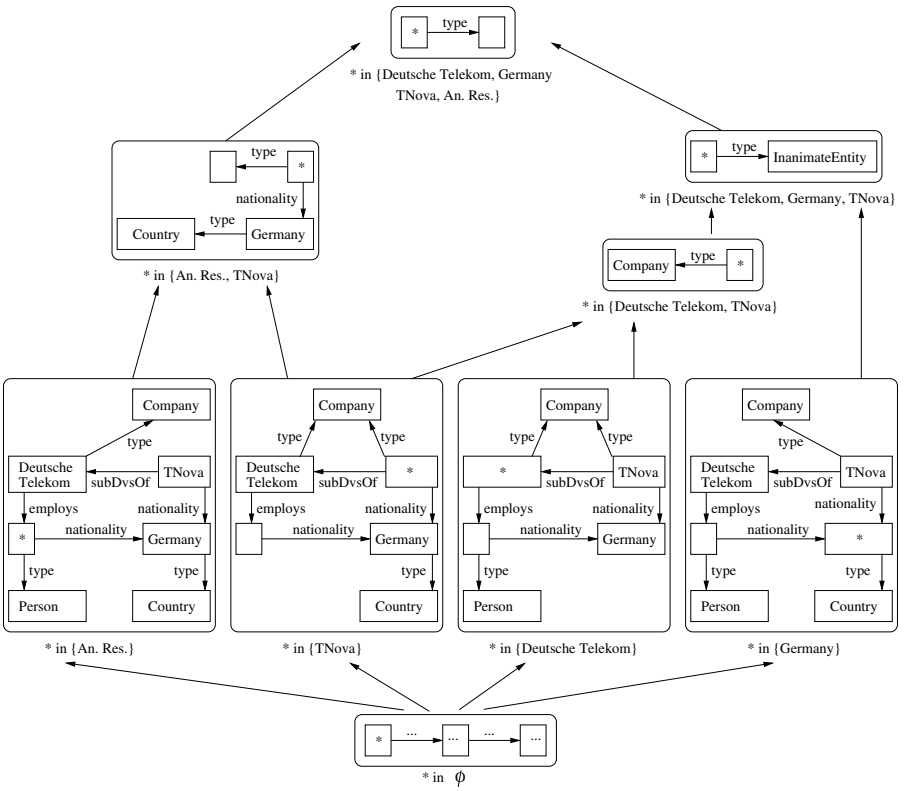

Fig. 7. The concept hierarchy H_2 built upon the context $C_2 + C_1$ of Figures 6 and 5

Figure 7 presents the concept hierarchy H_2 built upon C_2. H_2 has the same number of concepts than H_1 but five of its concepts have more complex intensions: the four concepts whose extensions are reduced to a single resource and whose intensions correspond to the descriptions of length 2 of these resources, and the concept of extension {TNova, An. Res.}.

5 Experiments

5.1 Preliminary Results and Discussion

Our algorithm has been tested in the framework of the European IST CoMMA Project where the so learned ontologies will be used to organize the documents of the corporate memory, to improve the Information Retrieval process on the corporate memory, and provide feedback to the ontology designer to refine and enrich the domain ontology. On a set of CoMMA annotations, with an ontology of height 6 and containing 50 classes and properties, the results of applying the

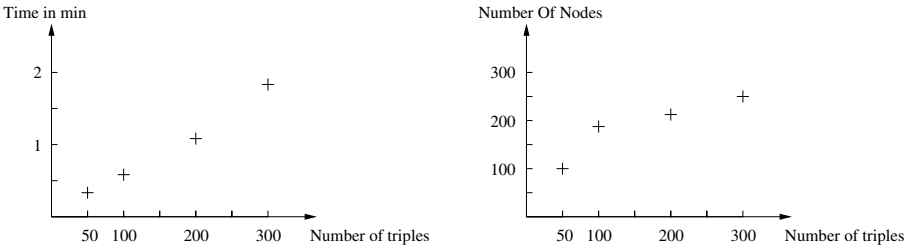

Fig. 8. Results obtained at level 5

algorithm is shown in Figure 8. Time seems to grow in a linear way and number of concepts seems to grow first rapidly and then slower. This depends of course a lot on the shape of the ontology (height, number of classes and properties). Althougth in the worst case time and number of nodes could be exponential, it shows that in practical applications it is definitely not.

5.2 Exploitation of the Conceptual Hierarchy

The learned concept hierarchy is expected to be exploited in the CoMMA project for three purposes. It first will be helpful in refining the domain ontology design. The learned concepts whose intensions are judged particularly relevant and interesting enough by the ontology designer will be integrated in the ontology. Some of the learned concepts will correspond to primitive concepts already present in the ontology; the definitions of these concepts will then be provided. Moreover the learned concepts may be useful to detect regularities in the use of the classes and properties of the primitive ontology that betray a misuse or a misconception of the ontology. A further work will be the development of heuristics and the choice of domain-specific criteria to extract particularly interesting classes from the learned concept hierarchy.

Second, the learned concept hierarchy is dedicated to the organization of the corporate memory. By indexing the documents of the intranet to the concepts they belong to, the concept hierarchy builds up a classification of the documents. This is of prime importance to support the navigation of the users in the corporate memory and help them access to the documents by browsing the concept hierarchy [16].

Finally, the concept hierarchy will be used to improve Information Retrieval on the corporate memory. To answer a query, it will be classified in the concept hierarchy instead of being matched with the descriptions of all the documents of the memory. Moreover, the concept hierarchy will enable to sort and organize the answers to a query to help the user access them with a classificatory structure.

6 Conclusion

We presented a method to capture knowledge from Web documents. More precisely, we build a concept lattice from a context built upon a power context family representing the RDF graph gathering the annotations of the Web documents we consider. In order to deal with the intrinsic exponential complexity of such a task, the concept hierarchy is incrementally built by increasing at each step the maximum size of the RDF resource descriptions we consider.
Our further work deals with the specialization of the general principle presented in this paper to classify all the resources in an RDF graph. In many applications, we may identify peculiar subsets of the resources of the RDF graph to classify, e.g. those sharing a particular type. Next, we intend to explore a more expressive description language and compute for a set of resources the set of patterns satisfied by all these resources. At each step of our inductive process, patterns would be refined into patterns with one more triple. As it may lead to too numerous patterns, language bias (e.g. trees or graphs with only unidentified nodes) should be explored to reduce their number. Finally, conditions on the refinement operation should be found to refine, at each step of the process, the only patterns that will lead to further interesting patterns, i.e. patterns whose some refinement is more specific than any of the refinements of the other patterns.

References

1. Baader, F., Molitor, R. Building and Structuring Description Logic Knowledge Bases Using Least Common Subsumers and Concept Analysis. In Proceedings of ICCS 2000 (Darmstadt, Germany, 2000), LNAI 1867, Springer-Verlag, 292-305. 197
2. Berners Lee, T.: Weaving the Web, Harper San Francisco, 1999. 191
3. Bournaud, I., Courtine, M. and Zucker, J-D. Kids: An Iterative Algorithm to Organize Relational Knowledge. In Proceedings of 12th EKAW (Juan-Les-Pins, France, 2000), LNAI 1937, Springer-Verlag, 217-232. 192, 195, 196
4. Carpineto, C., Romano, G. Galois: An Order Theoretic Approach to Conceptual Clustering. In Proceedings of 10th ICML (Amherst, Massachusetts, 1993), Morgan Kaufmann, 33-40. 192
5. Corby, O., Dieng, R., Hebert, C.: A Conceptual Graph Model for W3C Resource Description Framework. In Proceedings of ICCS'00, Darmstadt, Germany, LNAI 1867, Springer-Verlag, 2000. 194
6. Delteil, A., Faron, C., Dieng, R. Extension of RDF(S) based on the CGs Formalisms, in Proceedings of 9th ICCS, Stanford, CA, USA, August, 2001, Springer-Verlag, LNAI 2120, p. 275 - 389. 194
7. Fischer, D. H., Pazzani, M. J., and Langley, P. Concept Formation: Knowledge and Experience, Unsupervised Learning, Morgan Kaufmann, 1991. 192
8. Gandon, F. Ontology Engineering: a Survey and a Return on Experience. Research Report of Inria, RR4396, France, March 2002. 193
9. Ganter, B. Finding all Closed Sets: A General Approach. Order, 8, 1991. 197, 199
10. Gennari, J. H., Langley, P., and Fisher, D. H. Models of Incremental Concept Formation. Artificial Intelligence, 40: 11-61, 1989. 192

11. Mineau, G., Gecsei, J., and Godin, R. Structuring Knowledge Bases using Automatic Learning. In Proceedings of 6th ICDE (Los Angeles, CA, 1990), 274-280.
12. RDF: http://www.w3.org/TR/REC-rdf-syntax/, 1999. 191, 194
13. RDFS: http://www.w3.org/TR/2000/CR-rdf-schema-20000327/, 2000. 191, 193
14. Sowa, J. F.: Conceptual Graphs, Conceptual Structures: Information Processing in Mind and Machine, Addison-Wesley, Reading, MA, 1984.
15. Sowa, J. F.: Conceptual Graphs: DpANS. In Proceedings of ICCS'99, Blacksburg, VA, USA, LNAI 1640, p.1-65, Springer-Verlag, 1999.
16. Stumme, G. Hierarchies of Conceptual Scales. In Proceedings of 12th KAW (Banff, Canada, 1999). 192, 195, 202
17. Wille, R. Restructuring Lattice Theory: an Approach Based on Hierarchies of Concepts. In: I. Rival (ed): Ordered Sets, Reidel, Dordrecht-Boston, 1982. 195, 197