Learning Ontologies from RDF annotations

Alexandre Delteil, Catherine Faron Zucker, Rose Dieng-Kuntz

Learning ontologies from RDF annotations

Alexandre Delteil, Catherine Faron-Zucker, Rose Dieng
ACACIA project, INRIA, 2004, route des Lucioles, B.P. 93, 06902 Sophia Antipolis, France
{Alexandre.Delteil, Catherine.Faron, Rose.Dieng}@sophia.inria.fr

Abstract

In this paper, we present a method for learning ontologies from RDF annotations of Web resources by systematically generating the most specific generalization of all the possible sets of resources. The preliminary step of our method consists in extracting (partial) resource descriptions from the whole RDF graph gathering all the annotations. In order to deal with algorithmic complexity, we incrementally build the ontology by gradually increasing the size of the resource descriptions we consider.

1 Introduction

The Semantic Web, expected to be the next step that will lead the Web to its full potential, will be based on semantic metadata describing all kinds of Web resources. Resource Description Framework [RDF, 1999] seems to be the emerging standard allowing to semantically annotate Web resources. These annotations are related to ontologies, for example declared in RDF Schema [RDFS, 2000], as proposed by W3C, or in RDF-compatible languages like OIL [Fensel *et al.*, 2000]. Methods for building hierarchies of classes from RDF annotations will appear to be useful to classify resources, organize knowledge and finally learn ontologies.

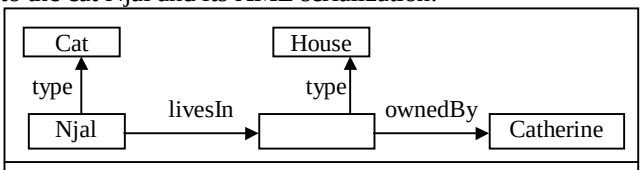
The building of hierarchical structures has been extensively studied in machine learning, especially in concept formation. Most approaches of concept formation are dedicated to the prediction of unknown features of new objects [Fischer *et al.*, 1987; Gennari *et al.*, 1989]. The clusters of *similar* objects are then privileged, the learned conceptual hierarchy does not comprise all the possible sets of objects, but only the best ones according to some heuristic criteria.

For learning ontologies, we adopt a particular approach of concept formation. An ontology is viewed as a concept hierarchy, where each concept is defined in extension by a cluster of resources and in intension by the most specific common description of these resources. This approach leads to the systematic generation of all the possible clusters of objects, as in [Mineau, 1990; Carpineto and Romano, 1993; Bournaud *et al.*, 2000].

Since all RDF annotations are gathered inside a common RDF graph, the problem which arises is the extraction of a description for a given resource from the whole RDF graph. After a brief description of the RDF data model (Section 2) and of RDF Schema (Section 3), Section 4 presents several criteria for extracting partial resource descriptions. In order to deal 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. The principle of the approach is explained in Section 5 and more deeply detailed in Section 6.

2 The RDF data model

RDF is the emerging Web standard for annotating resources with semantic metadata [RDF, 1999] [Decker *et al.*, 2000]. An RDF annotation consists of a set of statements, each one specifying a value of a property of a resource. A statement is thus a triple (resource, property, value), a value being either a resource or a literal. 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. RDF is provided with an XML syntax. Figure 1 presents an example of an RDF graph describing the Web page relative to the cat Njal and its XML serialization.




```
<rdf:Description about='#Njal'>
    <rdf:type resource=`#Cat' />
        <livesIn>
            <rdf:Description>
                <rdf:type resource=`#House' />
                    <ownedBy rdf:resource=`#Catherine' />
                </rdf:Description>
            </livesIn>
</rdf:Description>
```

Figure 1. An RDF annotation and its XML serialization

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.

3 Representation of Ontologies in RDF Schemas

RDF Schema (RDFS) is a schema specification language [RDFS, 2000]. 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.

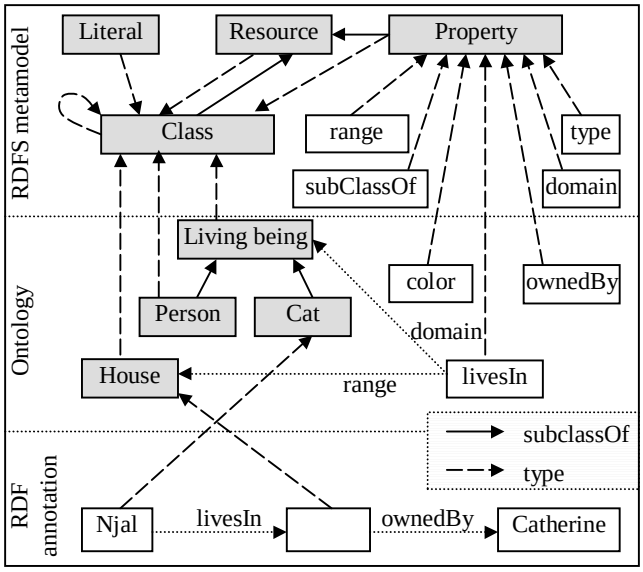


Figure 2. The RDFS metamodel and an RDFS ontology.

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

4 Extracting resource descriptions

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" [RDF, 1999]. The RDF model does not handle the delimitation of the subgraph of G describing a resource. We thus propose different criteria for extracting a description of a resource R from G.

Complete description: We define the complete description of a resource R as follows. A resource is completely described by the subgraph of G containing all the resources reachable from R through properties. Formally, the complete description of R is the largest connected subgraph of G containing R; it is inductively defined as the join of the complete descriptions of the resources adjacent to R in G. Such a complete description may be very large; potentially it may be the graph G representing the whole knowledge base. This is why we define ways of extracting partial descriptions of a resource R from G.

Piece of knowledge: We define the piece of knowledge relative to a resource R as the largest connected subgraph of G whose all internal nodes excepted R are anonymous resources. External nodes are either identified resources, or literals or anonymous resources connected to the only resources belonging to the piece.

Description of length n: We define the description of length n of a resource R as the largest connected subgraph of G containing all possible paths of length smaller or equal to n, starting from or ending to R. The description $D_n(R)$ of length n of a resource R 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 $D_{n-1}(R)$.

Partial description: We define a partial description of a resource R as either the piece of knowledge relative to R or a description of length n of R.

Figure 3 presents the extraction of three possible partial descriptions of Njal from the whole RDF graph which the RDF annotation of Figure 1 participates to: the piece of knowledge relative to Njal, the description of length 1 of Njal and the description of length 2 of Njal. $D_1(Njal)$ is a subgraph of $D_2(Njal)$ which is made of paths of length 1 and of length 2 starting from or ending to the resource Njal.

Given the whole RDF graph G, by choosing a description extraction criterion, we can now be provided with a set of partial descriptions for all the resources that are nodes of G.

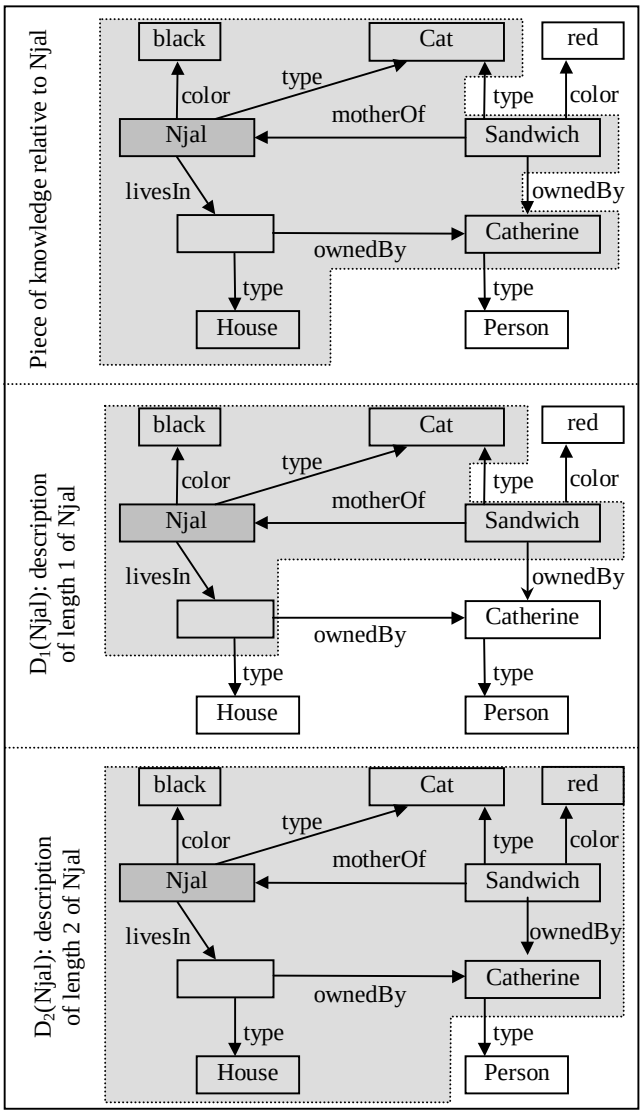


Figure 3. Partial RDF descriptions of the resource Njal.

5 Our approach of ontology learning

Our general aim is to learn from the whole RDF graph G comprising the resources we are interested by, new domain specific concepts to enrich the ontology from which the RDF annotations participating to G have been built.

5.1 Concept formation

Our approach of ontology learning consists of systematically considering all the concepts covering a set of resources nodes of G. Each of these concepts may then be defined in extension as a cluster of resources; its definition in intension must generalize the descriptions of the resources belonging to its extension. The definition of criteria to extract resource descriptions from an RDF graph G was thus a preliminary step to concept.

Now, 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. We can now further describe our approach of ontology learning. It consists in associating to this set of resource descriptions the hierarchy of the concepts whose extensions correspond to all the possible resource clusters.

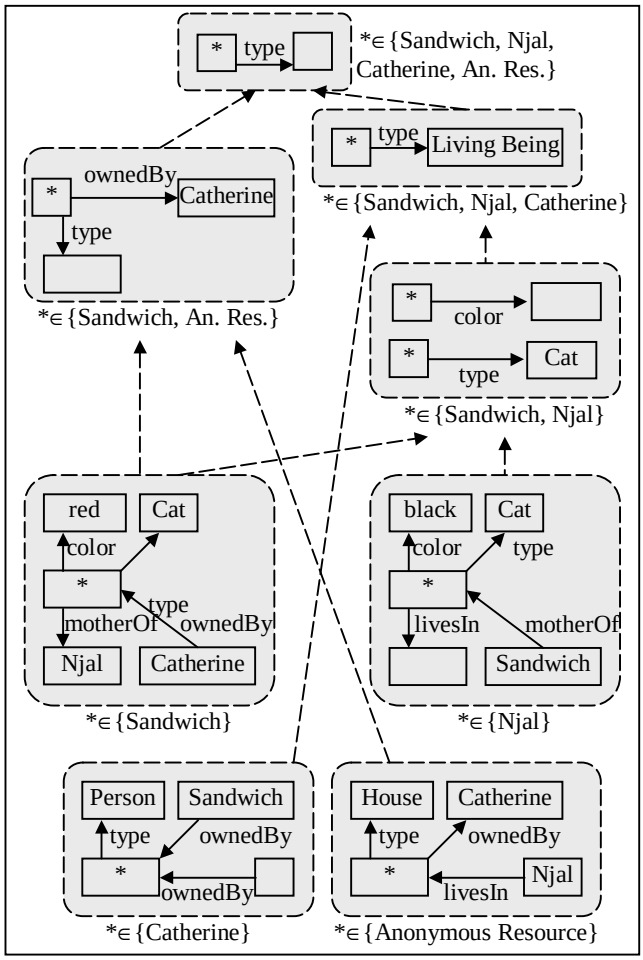


Figure 4. The concept hierarchy associated to descriptions of length 1 extracted from the RDF graph of Figure 3.

In this hierarchy, each concept c_i is labeled by a pair (ext_i, int_i), where ext_i is the extension of c_i and int_i is the intension of c_i. Int_i is the most specific generalization of the descriptions of the resources belonging to ext_i. We thus call this concept hierarchy a generalization hierarchy. The generalization of a resource description is based on the subsumptions relations between classes and properties declared in the RDF schema representing the ontology which the RDF graph G we consider is relative to. Such a generalization hierarchy is a lattice: its nodes are partially ordered by the subsumption relation on their intensions as well as the inclusion relation on their extensions.

Figure 4 presents the generalization hierarchy built from descriptions of length 1 of four resources nodes of the RDF

graph depicted in Figure 3: Njal, Sandwich, Catherine and the anonymous resource of type 'House'.

If several concepts share the same intension, a single concept is added to the generalization hierarchy: the one with the largest extension. Therefore, if the size of the generalization hierarchy may theoretically reach 2^N concepts for N resources in the RDF graph G, in practice it is much lower. For instance, the size of the hierarchy of Figure 4 is 8 concepts instead of 16 (2^4).

5.2 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-significant the concepts will be. On the other hand, graph matching and lattice building both have a well-known intrinsic exponential complexity. As a result, we adopt an incremental approach for the construction of the generalization hierarchy.

To be precise, we gradually increase the length of the partial resource descriptions we consider. We first build a generalization hierarchy S_1 from resource descriptions of length 1. The concepts of S_1 thus have intensions of length 1. S_n is then inductively built from S_{n-1} and S_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 $D_{n-1}(R)$.

6 Incremental building of a generalization hierarchy

6.1 Building of a generalization hierarchy based on resource descriptions of length 1

The principle for building S_1 is as follows:

1. Extraction of resource descriptions of length 1 from the whole RDF graph. $D_1(R)$ is the set of RDF triples beginning or ending by R.
2. Iterative generalization of all the possible pairs of triples. The generalizations of two triples (R_1, P_1, V_1) and (R_2, P_2, V_2) are the most specific triples (R_G, P_G, V_G) subsuming them. This is based on the ontological knowledge represented in the RDF Schema relative to the RDF annotations we consider. P_G is one of the most specific properties generalizing P_1 and P_2. If V_1 and V_2 are classes, then V_G is one of the most specific classes generalizing V_1 and V_2; else either $V_G=V_1=V_2$ or V_G is anonymous. We call L_1 the set of all generalized triples.
3. Construction of the intensions of length 1 of the S_1 nodes. The triples sharing a same extension are grouped together. An intension may include redundant triples, one being more general than another. It is cleaned up by deleting triples subsuming another one. Such an intension is the most specific description of the node.
4. Building of S_1 based on the inclusion relations between the node extensions. Several nodes may share the same intension. In this case, the node we preserve corresponds to the largest extension.

Let us apply this principle on the RDF graph depicted on Figure 3. The consecutive steps are illustrated in Figure 5.

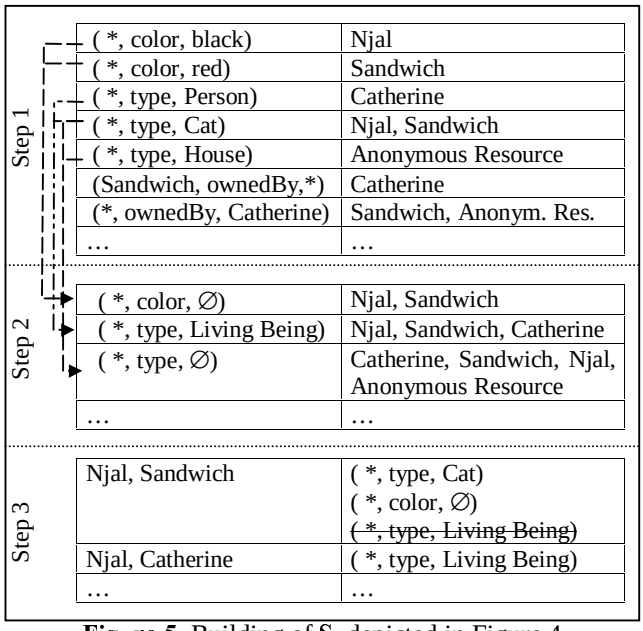


Figure 5. Building of S_1 depicted in Figure 4.

In the first step, the descriptions of length 1 of all the resources are extracted from the RDF graph of Figure 4. For each RDF triple (R, P, V), both the triples (*, P,V) and (R, P, *) are generated; the resources R matching * in (*, P, V) and the resources V matching * in (R, P, *) are indexed to these triples. For instance, the triples ('Njal', type, cat) and ('Sandwich', type, cat) both match the triple (*, type, cat): the resources 'Njal' and 'Sandwich' are thus indexed by the triple (*, type, Cat). The result of step 1 is thus an inverted file where resources are indexed by the triples they are the extensions of. 'Njal' and 'Sandwich' belong to the extension of the intension represented by (*, type, Cat).

In the second step, the triples are matched two by two; for each pair of triples, the most specific generalization of both triples is computed, according to the ontological knowledge expressed in the RDF schema the RDF graph is relative to. The extension of a generalized triple is the union of the extensions of the two initial triples it generalizes. For instance, the triples (*, color, black) and (*, color, red) are generalized into the triple (*, color, ∅), black and red being incomparable; (*, type, Person) and (*, type, Cat) are generalized into the triple (*, type, Living Being), 'Living Being' being the most specific class subsuming 'Cat' and 'Person'. 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.

In the third step, each possible set of resources is considered as the extension of a concept whose intension is to be found: for each extension, the common triples the resources are indexed by are grouped together to build the intension of the concept. For instance, the two resources 'Njal' and 'Sandwich' viewed as a concept extension lead to the construction of the intension of this concept from the set of triples they are both indexed by: {(*, type, Cat), (*, color, ∅), (*, type, Living Being)}. Since the triple (*, type, Living Being) subsumes the triple (*, type, Cat), it is discarded from the final intension. Finally, a new concept will be added to the generalization hierarchy, with {'Njal', 'Sandwich'} as extension and {(*, type, Cat), (*, color, ∅), (*, type, Living Being)} as intension.

The last step is dedicated to the building of the generalization hierarchy based on the inclusion relations between the concept extensions. The concepts sharing their intensions with concepts whose extensions include their own ones are discarded: for instance, the concept ({Njal, Catherine}, {(*, type, Living Being)}) is discarded since the concept whose extension is the set {Njal, Sandwich, Catherine} shares the same intension. The generalization hierarchy depicted in Figure 4 is the final result S_1 obtained from the index file depicted in Figure 5.

6.2 Building of a generalization hierarchy based on resource descriptions of length n

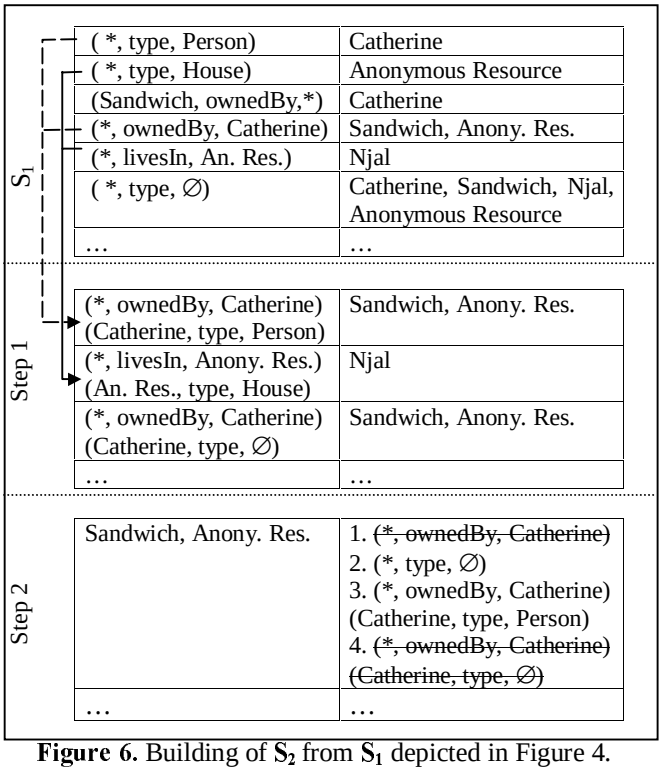


Figure 6. Building of S_2 from S_1 depicted in Figure 4.

The principle for building S_n from S_{n-1} and S_1 is as follows:

1. Iterative construction of L_n by join of all the possible pairs of one triple of L_1 and one triple path of length n-1 of L_{n-1}. Two triples can be joined if the value in the first triple is equal to the resource described in the second triple. A triple path of length n is thus the result of n-1 joins between n triples. This iterative construction of L_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..k, R_i being the external nodes of $D_{n-1}(R)$.
2. Construction of the intensions of length n of the S_n nodes (this step is similar to step 3 of S_1 building).
3. Building of S_n based on the inclusion relations between the node extensions (similar to step 4 of S_1 building).

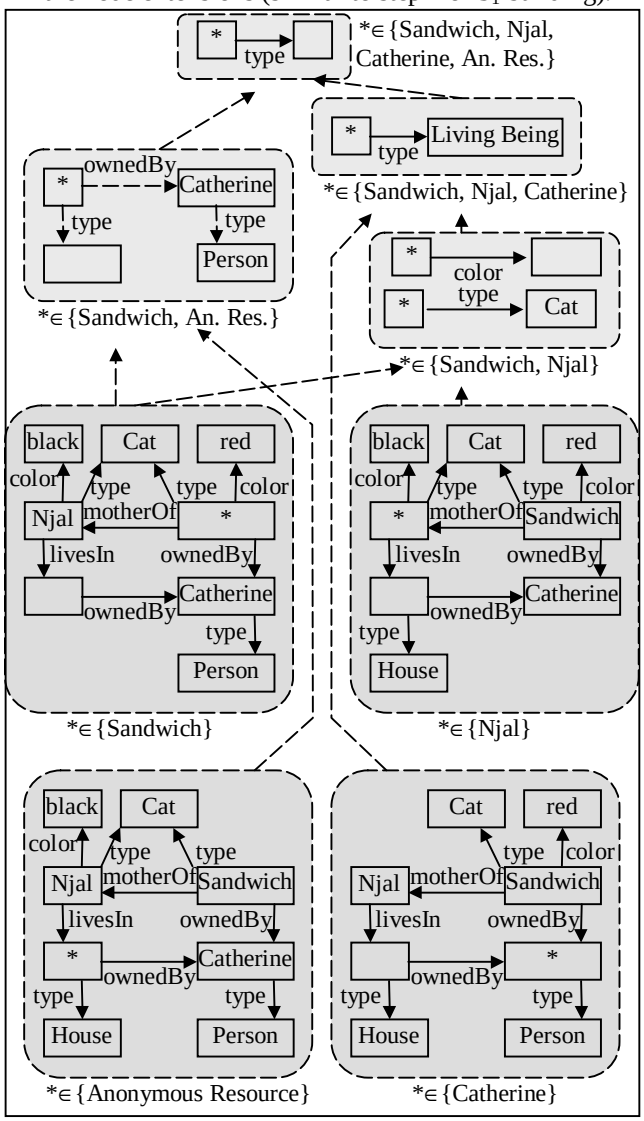


Figure 7. S_2 resulting from the index file of Figure 6.

Figure 6 describes the consecutive steps of the building of S_2 (shown in Figure 7) from S_1 (depicted in Figure 4). In the first step, the triples of S_1 are joined one with another (in the general case, the triples of S_{n-1} would be joined with the ones of S_1). For instance, the triple (*, ownedBy, Catherine)

can be joined with the triple (*, type, Person) 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 (*, ownedBy, Catherine) (Catherine, type, Person), whose extension is equal to the extension of the former triple.

In the second step, each possible set of resources is considered as the extension of a concept whose intension is to be found: the common triples the resources of a concept extension are indexed by are grouped together to build the intension of the concept. It is thus cleaned up in order to obtain the most specific description. For instance, the triple path (*, ownedBy, Catherine) (Catherine, type, ∅) is discarded from the intension of the extension {Sandwich, Anonymous Resource}, since it subsumes the triple path (*, ownedBy, Catherine) (Catherine, type, Person).

The last step is dedicated to the building of the generalization hierarchy based on the inclusion relations between the node extensions. Figure 7 presents the generalization hierarchy S_2 built from the generalization hierarchy S_1 depicted in Figure 4. S_2 has the same number of concepts than S_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 {Sandwich, An. Res.}.

7 Related Work

Conceptual clustering aims at building hierarchies to cluster similar objects and classify object descriptions [Fischer *et al.*, 1987]; a single class hierarchy is built, the best according to a certain criterion. Our approach of concept formation for ontology learning 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 [Wille, 1982], on knowledge organization [Mineau *et al.*, 1990] and in inductive logic programming [Kietz and Morik, 1994; Schlobach, 2000]. Another particularity of our method is the gradual increase of the size of the resource descriptions to deal with the intrinsic complexity of description matching and ontology building. A similar approach is adopted in [Bournaud *et al.*, 2000]; 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 descriptions. Our method differs in that the resource descriptions are not given in an RDF graph and its incrementallity is based on the gradual increase of the *size* and not of the structure of matching. Finally compared to the approaches of ontology learning based on the analysis of textual corpus (e.g. [Maedche and Staab, 2000]), RDF annotations may give a better starting point than HTML documents.

8 Conclusion

We have presented a method to learn ontologies from RDF annotations by systematically generating the most specific generalization of all the possible sets of resources. In order to deal with the intrinsic exponential complexity of such a task, we incrementally build the hierarchy by increasing at each step the maximum size of the resource descriptions we extract from the RDF graph gathering all the annotations.

Our algorithm is currently under implementation and will be tested inside of the European IST Comma Project. The so learned ontologies will be used to improve the efficiency of a query engine over a set of annotations.

We are currently exploring the possible improvements of our algorithms for the construction of the first class hierarchy [Baader *et al.*, 1999].

References

[Baader *et al.*, 1999] F. Baader, R. Kusters, R. Molitor. Computing Least Common Subsumers in Descriptions Logics with Existential Restrictions. In 7th IJCAI, Morgan Kaufmann, 1999.

[Bournaud *et al.*, 2000] I. Bournaud, M. Courtine and J-D Zucker. KIDS: An iterative algorithm to organize relational knowledge. In 12th EKAW, LNAI 1937, Springer-Verlag, p. 217-232, Juan-Les-Pins, France, 2000.

[Carpineto and Romano, 1993] C. Carpineto and G. Romano. GALOIS: an Order Theoretic Approach to Conceptual Clustering. In 10th ICML, Morgan Kaufmann, p. 33-40, Amherst, Massachusetts, 1993.

[Decker *et al.*, 2000] S. Decker, P. Mitra, S. Melnik. Framework for the Semantic Web – An RDF Tutorial. In *IEEE Internet Computing*, December 2000.

[Fischer *et al.*, 1987] D.H. Fisher, M.J. Pazzani, and P. Langley. Concept Formation: Knowledge and Experience in Unsupervised Learning. Morgan Kaufmann, 1991.

[Fensel *et al.*, 2000] D. Fensel, I. Horrocks, F. Van Harmelen, S. Decker, M. Erdmann and M. Klein. OIL in a nutshell. In 12th EKAW, Juan-Les-Pins, France, 2000.

[Gennari *et al.*, 1989] J. H. Gennari, P. Langley and D.H. Fisher. Models of incremental concept formation. *Artificial Intelligence*, 40: 11-61, 1989.

[Kietz and Morik, 1994] J.U. Kietz and K. Morik. A Polynomial approach to the constructive induction of structural knowledge. In *Machine Learning*, 1994.

[Maedche and Staab, 2000] A. Maedche and S. Staab. Mining ontologies from text. In 12th EKAW, 2000.

[Mineau *et al.*, 1990] G. Mineau, J. Gecsei and R. Godin. Structuring knowledge bases using automatic learning. In 6th ICDE, p. 274-280, Los Angeles, CA, February 1990.

[RDF, 1999] http://www.w3.org/TR/REC-rdf-syntax/, 1999.

[RDFS, 2000] http://www.w3.org/TR/rdf-schema/, 2000.

[Schlobach, 2000] S. Schlobach. Assertional Mining in Description Logics. In proc. of DL'2000.

[Wille, 1982] R. Wille. Restructuring lattice theory: an approach based on hierarchies of concepts. In: I. Rival (ed): Ordered Sets, Reidel, Dordrecht-Boston, 1982.