Common Subsumbers in RDF

Simona Colucci[1], Francesco M. Donini[1], and Eugenio Di Sciascio[2]

1: DISUCOM, Università della Tuscia, Viterbo, Italy
2: DEI, Politecnico di Bari, Bari, Italy

Abstract. Since their definition in 1992, Least Common Subsumers (LCSs) have been identified as services supporting learning by examples. Nowadays, the Web of Data offers a hypothetically unlimited dataset of interlinked and machine-understandable examples modeled as **RDF** resources. Such an open and continuously evolving information source is then really worth investigation to learn significant facts. In order to support such a process, in this paper we give up to the subsumption minimality requirement of LCSs to meet the peculiarities of the dataset at hand and define Common Subsumers (CSs). We also propose an anytime algorithm to find CSs of pairs of **RDF** resources, according to a selection of such resources, which ensures computability.

1 Introduction

Since its definition and identification as fundamental layer over which building the Semantic Web [4], the Web of Data [22] has led to the availability of a huge amount of perfectly interconnected and machine-understandable data, modeled as **RDF** resources, usually addressed as Linked (Open) Data (LOD). Such an open and free (in most cases) dataset asks for new sorts of information management, which could further support the realization the Semantic Web principles.

One of the most challenging issues in **RDF** resources management is the identification of subsets of resources to some extent related to a common informative content. Finding commonalities in the information conveyed by different **RDF** descriptions may in fact turn out to be useful in several Semantic Web-related tasks. Intuitively, clustering of Web resources is one of these tasks: the need for restricting the search field in a space as wide as the Web has generated several research efforts in the literature.

Since 2001, learning from **RDF** graphs has been investigated [12] with the aim to cluster resources by incrementally constructing new concepts that partially describe all resources of interest, and to classify them according to subsumption. Fanizzi *et al.* [13] perform ontology clustering according to a metric-based approach, with reference to representation languages (namely, OWL-DL[1]) different from **RDF**; lists of so-called medoid are returned as clustering result.

Part of the literature is specifically devoted to the application of clustering algorithms to Semantic Web data. Grimnes *et al.* [16] evaluate, by applying both

[1] `http://www.w3.org/TR/2004/REC-owl-guide-20040210/`

supervised and unsupervised metrics, different methods for extracting instances from a large **RDF** graph and computing the distance between such instances. The evaluation shows how the behavior of extraction methods and similarity metric strictly depends on the data being clustered, suggesting the need for data-centric flexible solutions.

Mahmoud *et al.* [20] investigate on clustering for data integration from large numbers of structured data sources, suggesting the need for a pay-as-you-go approach [19] which adopts an initial data integration system (resulting from some fully automatic approximate data integration technique), refined during system use. The approach by Mahmoud *et al.* clusters data schema into domains only according to attribute names and manages uncertainty in assignment using a probabilistic model.

Also the simpler problem of clustering web pages has met the interest of researchers: a Fuzzy Logic-based representation for HTML document using Self-Organizing Maps has been proposed for clustering [15]. Zeng *et al.* [24] deal with the re-formalization of the clustering problem in terms of phrase ranking to produce candidate cluster names for organizing web search results.

Proposals taking somehow into account semantics conveyed in resources descriptions while performing clustering of search results have been presented. In particular, Lawrynowicz [18] proposes a method to cluster the results of conjunctive queries submitted to knowledge bases represented in the Web Ontology Language (OWL). Syntactic approaches show some limits in aggregating clustering results when the values instantiating a grouping criterion are all equal or are almost all different. d'Amato *et al.* . [10] overcome such limits by deductively grouping answers according to the subsumption hierarchy of the underlying knowledge base.

Far from being exhaustive, the above state of the art shows the deep interest for clustering (especially of Web content) and underlines how most of the clustering approaches introduced so far adopt induction to identify clusters according to some—sometimes semantic-based—distance between elements in the same cluster. Our approach instead supports the inference of such clusters, and may provide a description of the informative content associated to the common features of resources belonging to the same cluster.

In particular, we define *Common Subsumers* of pairs of **RDF** resources, in analogy to the *Least Common Subsumer*(LCS) service, well known in Description Logics. Since its proposal in 1992 [5], learning from examples was identified as one of the most important application fields for LCS computation.

We take the same assumption and adapt the original definition to the set of examples we aim at learning from: the Web of Data. The hypothetically unlimited size of the investigated dataset motivates the choice of giving up to the subsumption minimality requirement typical of LCSs and revert to Common Subsumers. Motivated by the chance to compute even rough Common Subsumers, still useful for learning in the Web of Data, we propose an anytime algorithm computing Common Subsumers of pairs of **RDF** resources. The algorithm may work as basis for finding commonalities in collections of **RDF** resources.

The paper is organized as follows: in the next section, we shortly recall related work on common subsumers computation and discuss peculiarities of **RDF** which make it different from other Web languages in terms of such an inference process. Section 3 provides a proof-theoretic definition of Common Subsumers in **RDF**, whose properties are shown in Section 4. The anytime algorithm computing Common Subsumers is given in Section 5, before closing the paper.

2 Common Issues

Coherently with the initial motivation of supporting inductive learning in the DL $\mathcal{LS}$ [5], several algorithms have been developed for computing LCS in different DLs, such as CoreClassic [6], $\mathcal{ALN}$ [7], Classic [14].

The idea of reverting to common subsumers which are not "least" has been already investigated to cope with practical applications [2].

The main issues of the problem we investigate here are on the one hand selecting a portion of the knowledge domain in order to ensure computability and, on the other hand, the peculiarities of the language in which resources are modeled: **RDF**. Aimed at learning from the Web of Data, our approach needs in fact to cope with **RDF/RDF-S**. Nevertheless, the **RDF/RDF-S** semantics is not trivial to be investigated, and even less trivial is determining its relationship with DLs [11] and/or other Web languages [21] in which the problem of finding (least or not) common subsumers has been defined and studied. For this reason, we provide in the following section an **RDF**-specific definition of Common Subsumers and clarify the choices about the adopted semantics.

3 Common Subsumers in RDF

We recall that in the Description Logics literature [5, 1], a concept L is a Least Common Subsumer of two concepts C_1 and C_2 if: (i) L subsumes both C_1 and C_2 (written as $C_1 \sqsubseteq L$, $C_2 \sqsubseteq L$) and moreover, (ii) L is a $\sqsubseteq$-minimal concept with such property, that is, for every concept D such that both $C_1 \sqsubseteq D$ and $C_2 \sqsubseteq D$ hold, $L \sqsubseteq D$ holds too.

We adopt the semantics based on entailment for **RDF**, that is, the meaning of a set of triples T is the set of triples (theorems) one can derive from T by using **RDF**-entailment rules. The **RDF**-entailment rules we consider are the 18 rules and the axiomatic triples of the official document regarding **RDF** semantics [17], modified according to ter Horst [23] as follows: In his paper, ter Horst proves that the original rules for entailment in **RDF** are incomplete, and that completeness is regained if (i) blank [2] nodes are allowed to stand as predicates in triples (called *generalized **RDF** triples*), and (ii) Rule rdfs7 is changed accordingly—namely, it can derive triples whose predicate is a blank node. In the following, when

[2] Recall that every blank node corresponds to an existentially quantified variable. A blank node without a name is denoted by `[]`, while named blank nodes are prefixed by `_:`, *e.g.*, `_:xxx`

we talk about **RDF** triples and **RDF** entailment, we always assume that the adjustments proposed by ter Horst have been made. To correctly embed **RDF** triples in text we use the notation $<<a\ p\ b>>$ to mean "$a\ p\ b$."

Clearly, the meaning of a resource r changes depending on which triples r is involved in, since different sets of triples derive (in general) different new triples. Hence, we always attach to a resource r the set of triples T_r we consider significant for its meaning and define such a pair as *rooted-graph* of r.

Definition 1 (Rooted Graph(r-graph)). *Let TW_r be the set of all triples with subject r in the Web. A* Rooted Graph(r-graph) *is a pair $\langle r, T_r \rangle$, where*

1. *r is either the URI of an* ***RDF*** *resource, or a blank node*
2. *$T_r = \{t \mid t = <<r\ p\ c>>\}$ is a subset of TW_r*

Observe that we implicitly define a set of resources, namely, all resources appearing in some relevant triple of some resource. The idea is to cut out a relevant portion of the Semantic Web, where the resources on the "frontier" of such a portion have no relevant triples—*i.e.*, they are treated as literals[3], even when they are not. In the r-graph $\langle r, \emptyset \rangle$, r has no "meaning" other than a generic resource, for which, *e.g.*, $<<r$ `rdf:type rdfs:Resource`$>>$ is always entailed (Rule rdfs4 in the official document about **RDF** entailment [17]).

We think that it is not realistic to assume that all relevant triples are known at once, and in advance: the usual way in which sets of triples are constructed, is that triples are discovered one at a time, while exploring the Web. In this setting, the question a crawler has to answer while surfing the Web is: "Is this triple t I have just found relevant for r or not?" The answer is provided by the *characteristic function* of T_r, namely, $\sigma_{T_r} : TW_r \rightarrow \{false, true\}$, that we will use to determine the set T_r of triples relevant for r (see Algorithm 2 for our current computation).

We clarify that when deciding the entailment of a triple involving r as subject, also triples not involving r at all must be considered. For example, let $\langle r, T_r \rangle$, $\langle a, T_a \rangle$ $\langle b, T_b \rangle$ be three r-graphs, where:

$$\begin{aligned} T_r &= \{<<r\ \texttt{rdf:type}\ a>>\} \\ T_a &= \{<<a\ \texttt{rdfs:SubClassOf}\ b>>\} \\ T_b &= \emptyset \end{aligned}$$

Observe that the triple $<<r$ `rdf:type` $b>>$, which enriches the "meaning" of r, is not entailed by T_r alone, while it is entailed by $T_r \cup T_a$ (Rule rdfs9 [17]). Hence, we assume that *entailment is always computed with respect to the union of all sets of relevant triples.* We call T such a union set. We are now ready to give our definition of Common Subsumer of two resources.

Definition 2 (Common Subsumer). *Let $\langle a, T_a \rangle$, $\langle b, T_b \rangle$ be two r-graphs and x, w, y be blank nodes. If $\langle a, T_a \rangle = \langle b, T_b \rangle$, then $\langle a, T_a \rangle$ is a Common Subsumer of $\langle a, T_a \rangle$, $\langle b, T_b \rangle$. Otherwise, if $T_a = \emptyset$ or $T_b = \emptyset$, the pair $\langle x, \emptyset \rangle$ is a Common*

[3] Recall that literals can never appear as subjects of triples.

Subsumer of $\langle a, T_a \rangle$, $\langle b, T_b \rangle$. Otherwise, a pair $\langle x, T \rangle$ is a Common Subsumer of $\langle a, T_a \rangle$, $\langle b, T_b \rangle$ iff:
$\exists t = <\!<x\ w\ y>\!>$ such that $(T$ entails $t)$

$$\Rightarrow \tag{1}$$

$\exists t_1 = <\!<a\ p\ c>\!>, t_2 = <\!<b\ q\ d>\!>$ such that $(T$ entails $t_1) \wedge (T$ entails $t_2)$ where $T_a \subseteq T$, $T_b \subseteq T$ and $\langle w, T \rangle$ is a Common Subsumer of $\langle p, T_p \rangle$ and $\langle q, T_q \rangle$, and $\langle y, T \rangle$ is a Common Subsumer of $\langle c, T_c \rangle$ and $\langle d, T_d \rangle$.

Observe that if we used "$\Leftrightarrow$" in (2) instead of "$\Rightarrow$", we would define Least Common Subsubers in **RDF**. However, we are more interested here in Common Subsumers, since they are easier to compute, while being still useful enough for our applications. Observe also that the reference to the relevant triples of each resource is crucial for defining the Common Subsumer; in order not to explore the entire Semantic Web, we will restrict the relevant triples either to some specific dataset such as *DBPedia*, or to triples "near" the initial resources—as a distance in the **RDF**-graph—or both.

We now provide an intuition of our definition through an example.

Example 1. Suppose that the following common prefixes have been defined:

```
@prefix dbpedia: <http://dbpedia.org/resource/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix dbpedia-owl: <http://dbpedia.org/ontology/> .
@prefix dbpprop: <http://dbpedia.org/property/> .
```

Then, consider the resource `dbpedia:Bicycle_Thieves` (a renowned Italian neo-realist movie), along with (what we decide to be) its relevant triples T_{BT} (written in Turtle [3] notation):

```
dbpedia:Bicycle_Thieves
    rdf:type dbpedia-owl:Film ;
    dbpprop:director   dbpedia:Vittorio_De_Sica ;
    dbpprop:language   dbpedia:Italian_language .
```

and the resource `dbpedia:The_Hawks_and_the_Sparrows` (a post-neorealist Italian movie), along with its relevant triples T_{UU}:

```
dbpedia:The_Hawks_and_the_Sparrows
    rdf:type            dbpedia-owl:Film ;
    dbpprop:director    dbpedia:Pier_Paolo_Pasolini ;
    dbpprop:language    dbpedia:Italian_language .
```

To make the example easy to follow, we consider that there are no relevant triples for all other resources—*i.e.*, `rdf:type`, `dbpedia-owl:Film`, `dbpprop:director`, etc.—involved in the above triples. This corresponds to choose only triples whose subject is either `dbpedia:Bicycle_Thieves` or `dbpedia:The_Hawks_and_the-_Sparrows`.

An intuitive Common Subsumer of $\langle$`dbpedia:Bicycle_Thieves`$, T_{BT} \rangle$ and $\langle$`dbpedia:The_Hawks_and_the_Sparrows`$, T_{UU} \rangle$ is $\langle$`_:cs1`$, T \rangle$, where `_:cs1` is a blank node and T contains, in addition to T_{BT} and T_{UU}, also the new triples

```
_:cs1
    rdf:type         dbpedia-owl:Film ;
    dbpprop:language dbpedia:Italian_language .
```

A more informative Common Subsumer—the one that would be computed by our algorithm in Sect. 5—is $\langle$`_:cs2`$, T\rangle$ where T contains, in addition to the previous example, also a triple referring to some director:

```
_:cs2
    rdf:type         dbpedia-owl:Film ;
    dbpprop:director _:cs3 ;
    dbpprop:language dbpedia:Italian_language .
```

where $\langle$`_:cs3`$, \emptyset\rangle$ is a Common Subsumer of $\langle$`dbpedia:Vittorio_De_Sica`$, \emptyset\rangle$ and $\langle$`dbpedia:Pier_Paolo_Pasolini`$, \emptyset\rangle$. Observe that since we did not consider any triple regarding the two directors—*e.g.*, they were both italian—we cannot add to $T_{\texttt{\_:cs3}}$ the triple

<<`_:cs3 dbpprop:director dbpedia:Italian_director`>>

This clearly exemplifies the tradeoff between, on one hand, how large is the portion of the linked data we want to consider, and on the other hand, how many computational resources we are willing to spend. Probably the most intuitive enlargement process is the one based on (**RDF** graph) distance: in that case, considering relevant also the triples whose subject has distance 1,2,... from the initial resources, one obtains progressively more accurate Common Subsumers at the price of a progressively heavier computation.

For sake of completeness, we note that one of the least informative Common Subsumers is $\langle$`_:cs4`$, T_{BT} \cup T_{UU}\rangle$, where `_:cs4` is a(nother) blank node. *(End of example).*

In general, a Common Subsumer of a, b is a blank node, except for the case $a = b$, in which the most informative Common Subsumer is the resource itself—see Idempotency in the next section. A Common Subsumer is a blank node because **RDF** can only express—by means of triples—necessary conditions for identifying a resource. For example, even if we state for some resource r exactly the same triples of `dbpedia:Bicycle_Thieves`, this does not allow us to derive in **RDF** that $r =$ `dbpedia:Bicycle_Thieves`—one can verify that equality of two resources is never the consequence of **RDF**-entailment Rules.

4 Properties of RDF Common Subsumers

We now explore some properties of Common Subsumers that already hold in Description Logics, and which we prove to hold also—suitably changed—for our definition of Common Subsumers in **RDF**, namely: Idempotency, Commutativity, and Associativity.

Idempotency. Clearly, a Common Subsumer of $\langle a, T_a\rangle$ and $\langle a, T_a\rangle$ is $\langle a, T_a\rangle$ itself. Differently from Description Logics, however, $\langle a, T_a\rangle$ is not the Least Common Subsumer (LCS) of itself—where "LCS" would be defined as in Def. 2, with "$\Leftrightarrow$" in place of "$\Rightarrow$". This is because when the two triples <<a p c>>, <<a q d>> are in T_a, with $p \neq q$, the Least Common Subsumer should contain the triple <<a `[]` `[]`>>. Such a triple is not derivable from **RDF** rules because of the blank node in the predicate position. Recall that from a triple <<a p c>>, one can derive both <<`[]` p c>> (Rule se2 in the W3C document [17]) and <<a p `[]`>> (Rule se1), but not <<a `[]` c>>. One would need a third rule se3, deriving exactly such a triple. The proposal of such an extension of **RDF** could be a subject of future research.

Commutativity. In Condition (1), it is evident that the role of a and b is completely interchangeable. Hence, a Common Subsumer of $\langle a, T_a\rangle$, $\langle b, T_b\rangle$ is always a Common Subsumer of $\langle b, T_b\rangle$, $\langle a, T_a\rangle$.

For the clustering applications we have in mind, it is necessary to extend the idea of Common Subsumer to a *group* of r-graphs $\langle a_1, T_{a_1}\rangle, \ldots, \langle a_n, T_{a_n}\rangle$. We can adapt the computation of a Common Subsumer from two to n r-graphs, by first computing a Common Subsumer $\langle x_1, T_1\rangle$ of, say, $\langle a_1, T_{a_1}\rangle$, $\langle a_2, T_{a_2}\rangle$, and then computing a Common Subsumer $\langle x_2, T_2\rangle$ of $\langle x_1, T_1\rangle$ and $\langle a_3, T_{a_3}\rangle$, etc., till computing a Common Subsumer $\langle x_{n-1}, T_{n-1}\rangle$ of $\langle x_{n-2}, T_{n-2}\rangle$ and $\langle a_n, T_{a_n}\rangle$.

In this case, it is important to ensure that no matter what pair of r-graphs we start from, and in what order we consider the rest of them, we always compute a Common Subsumer of all r-graphs. We refer to this property as Associativity, for historical reasons, although our computation does not define a proper algebraic operator. We state the property for just three r-graphs, since its inductive extension to n r-graphs is straightforward.

Theorem 1 (Associativity). *Given $\langle a_1, T_{a_1}\rangle$, $\langle a_2, T_{a_2}\rangle$, $\langle a_3, T_{a_3}\rangle$, let $\langle x_1, T_1\rangle$ be a Common Subsumer of $\langle a_1, T_{a_1}\rangle$, $\langle a_2, T_{a_2}\rangle$. Then, computing the Common Subsumer $\langle x_2, T_2\rangle$ of $\langle x_1, T_1\rangle$ and $\langle a_3, T_{a_3}\rangle$, one obtains a Common Subsumer of $\langle a_1, T_{a_1}\rangle$, $\langle a_2, T_{a_2}\rangle$, $\langle a_3, T_{a_3}\rangle$, pairwise considered.*

Proof. (Sketch.)We prove that $\langle x_2, T_2\rangle$ is a Common Subsumer of each of the three pairs of r-graphs $\langle a_1, T_{a_1}\rangle$–$\langle a_2, T_{a_2}\rangle$, $\langle a_2, T_{a_2}\rangle$–$\langle a_3, T_{a_3}\rangle$, $\langle a_1, T_{a_1}\rangle$–$\langle a_3, T_{a_3}\rangle$. Let G be the union of all r-graphs, G_1 be the union of G with the triples involving x_1 and attached resources, and let G_2 be the union of G_1 with the triples involving x_2 and all attached resources.

If $\exists t_2 =$ <<x_2 w_2 y_2>> entailed by G_2, then from Def. 2, there exist both a triple <<x_1 w_1 y_1>> entailed by G_1, and a triple <<a_3 p_3 c_3>> entailed by G, and such that $\langle w_2, W_2\rangle$ is a Common Subsumer of $\langle w_1, W_1\rangle$ and $\langle p_3, T_{p_3}\rangle$, and $\langle y_2, Y_2\rangle$ is a Common Subsumer of $\langle y_1, Y_1\rangle$ and $\langle c_3, T_{c_3}\rangle$. We now apply Def. 2 to $\langle x_1, T_1\rangle$: there exist two triples <<a_1 p_1 c_1>>, <<a_2 p_2 c_2>> which are both entailed by G, and such that $\langle w_1, W_1\rangle$ is a Common Subsumer of $\langle p_1, T_{p_1}\rangle$ and $\langle p_2, T_{p_2}\rangle$, and $\langle y_1, Y_1\rangle$ is a Common Subsumer of $\langle c_1, T_{c_1}\rangle$ and $\langle c_2, T_{c_2}\rangle$. This proves that $\langle x_2, T_2\rangle$ is also a Common Subsumer of $\langle a_1, T_{a_1}\rangle$ and $\langle a_2, T_{a_2}\rangle$, and since there exists also the triple <<a_3 p_3 c_3>>, $\langle x_2, T_2\rangle$ is also a Common Subsumer of the couples of r-graphs $\langle a_2, T_{a_2}\rangle$–$\langle a_3, T_{a_3}\rangle$, $\langle a_1, T_{a_1}\rangle$–$\langle a_3, T_{a_3}\rangle$. $\square$

Example 2. If we add to the two r-graphs of Example 1 also the r-graph $\langle$`The_Big_Sleep_(1946_film)`$, T_{\texttt{The\_Big\_Sleep\_(1946\_film)}}\rangle$, with the relevant triples:

```
dbpedia:The_Big_Sleep_(1946_film)
    rdf:type           dbpedia-owl:Film;
    dbpprop:director   dbpedia:Howard_Hawks;
    dbpprop:language  "English@en" .
```

one can get as a Common Subsumer $\langle$`_:cs5`$, T\rangle$, where T contains also:

```
_:cs5
    rdf:type           dbpedia-owl:Film;
    dbpprop:director   _:cs6;
    dbpprop:language   _:cs7 .
```

This result can be obtained either by comparing the above r-graph with each of the initial r-graphs of Example 1, or by comparing the above r-graph with the Common Subsumer `_:cs2` already computed in Example 1. *(End of example).*

5 Finding Common Subsumers in RDF

In order to find Common Subsumers of a pair of **RDF** resources, we propose Algorithm 1 below, which is an anytime algorithm.

As mentioned in Sect. 3, to define the r-graph of a resource r, we determine the subset $T_r \subseteq TW_r$ of triples relevant for r through its characteristic function $\sigma_{T_r} : TW_r \rightarrow \{false, true\}$:

$$\sigma_{T_r}(t) = \begin{cases} \text{true} & \text{if t } \in T_r \\ \text{false} & \text{if t } \notin T_r \end{cases}$$

Algorithm 1 takes as input two resources, a and b and the related maximum number of investigation calls, n_a and n_b ($n_a \geq 1$, $n_b \geq 1$). At any time, it can return a Common Subsumer x, together with the contextual set T of relevant triples inferred till the moment of its interruption. We notice that T does not only include triples $<<x\ y\ z>>$ involving x as subject, but also triples describing y and z and all resources needed to provide the required representation of x.

Algorithm 1 manages a global data structure S, collecting information about already computed Common Subsumers stored as tuples $[p, q, pq]$, where p and q are **RDF** resources and pq is (a self-evident name for) their Common Subsumer. S can be accessed through the pair p, q.

The adoption of variables n_a and n_b allows for limiting the recursive depth for computing the Common Subsumer of a and b, in presence of a hypothetically unlimited dataset. S supports, instead, the management of cycles in Common Subsumer computation which may occur during investigation.

In Rows 3 and 5, Algorithm 1 asks for the computation of σ_{T_a} and σ_{T_b}, in order to define the sets, T_a and T_b of relevant triples for the input resources.

The rationale for reverting to a subset of relevant triples defining a resource is, as hinted before, the trade-off between the need for a full resource representation

```
Find $CS(a, n_a, b, n_b)$;
Input  : $a, n_a, b, n_b$
Output: $x, T$
1 Let $S$ be a global data structure collecting computed Common Subsumers ;
2 Let $T$ be a global set of triples describing the Common Subsumer;
3 $\sigma_{T_a} = compute\_\sigma(n_a)$ ;
4 $T_a = \{t \mid \sigma_{T_a}(t) = true\}$;
5 $\sigma_{T_b} = compute\_\sigma(n_b)$;
6 $T_b = \{t \mid \sigma_{T_b}(t) = true\}$;
7 $T = T_a \cup T_b$;
8 $x = explore(a, \sigma_{T_a}, n_a, b, \sigma_{T_b}, n_b)$;
9 $return\ x, T$
```

Algorithm 1: Initialization and start of CS construction

and computability of Common Subsumers in a dataset as the Web of Data, too large to explore thoroughly.

In our current implementation we adopt Algorithm 2, *compute* $\sigma_{T_r}(n_r)$, for computing σ_{T_r} for a generic resource r, to be investigated through up to n_r calls.

Intuitively, Algorithm 2 considers relevant (see Row 5) only triples whose subject belongs to specific datasets of interest (collected in the set D) and returns an empty set of relevant triples when r has not to be further investigated, because the maximum recursive depth has been reached (Rows 2–3). In this way, Algorithm 2 ensures that at least one of the base cases in Definition 2 is reached (see Row 6 in Algorithm 3). We notice that more complex functions could be computed according to some heuristics without affecting Algorithm 1.

```
$compute\_\sigma(n_r)$;
Input  : $n_r$: maximum recursive depth in the RDF graph exploration;
Output: $\sigma_{T_r}$
1 let $D$ be set of RDF datasets of interest;
2 if  $n_r = 0$ then
3     $\sigma_{T_r} = function\ \sigma_{T_r}(t)$ {return false}
4 else
5     $\sigma_{T_r} = function\ \sigma_{T_r}(t)$ { if $t \in D$ then  return true else  return false };
6 return $\sigma_{T_r}$;
```

Algorithm 2: Computation of a simple σ_{T_r}

In order to start searching for a Common Subsumer of a and b, Algorithm 1 calls in Row 8 the recursive function *explore*, defined in Algorithm 3.

Algorithm 3 investigates on the sets (if both not empty – see Row 6) of relevant triples of the input resources a and b (Rows 7–26) and returns a resource x which is a blank node (or one of the input resources, if they are equal to each other – see Rows 3–5).

```
   $explore(a, \sigma_{T_a}, n_a, b, \sigma_{T_b}, n_b)$;
   **Input** : $a, \sigma_{T_a}, n_a, b, \sigma_{T_b}, n_b$
   **Output**: $x$
1  $T_a = \{t \mid \sigma_{T_a}(t) = true\}$;
2  $T_b = \{t \mid \sigma_{T_b}(t) = true\}$;
3  **if** $\langle a, T_a \rangle = \langle b, T_b \rangle$ **then**
4  |   add $T_a$ to $T$ ;
5  |   return $a$ ;
6  **if** $T_a = \emptyset$ $or$ $T_b = \emptyset$ **then** return $x$;
7  **foreach** $<<a\ p\ c>> \in T_a$ **do**
8  |   **foreach** $<<b\ q\ d>> \in T_b$ **do**
9  |   |   **if** $[p, q, pq] \in S$ **then**
10 |   |   |   $y = pq$;
11 |   |   **else**
12 |   |   |   $\sigma_{T_p} = compute\_\sigma(n_a - 1)$;
13 |   |   |   $\sigma_{T_q} = compute\_\sigma(n_b - 1)$;
14 |   |   |   $y = explore(p, \sigma_{T_p}, (n_a - 1), q, \sigma_{T_q}, (n_b - 1))$;
15 |   |   |   add [p, q, y] to S ;
16 |   |   |   add $T_y$ to T;
17 |   |   **if** $[c, d, cd] \in S$ **then**
18 |   |   |   $z = cd$;
19 |   |   **else**
20 |   |   |   $\sigma_{T_c} = compute\_\sigma(n_a - 1)$;
21 |   |   |   $\sigma_{T_d} = compute\_\sigma(n_b - 1)$;
22 |   |   |   $z = explore(c, \sigma_{T_c}, (n_a - 1), d, \sigma_{T_d}, (n_b - 1))$;
23 |   |   |   add [c, d, z ] to S ;
24 |   |   |   add $T_z$ to T;
25 |   |   **if** $y \neq$ [] $or$ $z \neq$ [] **then** add $<<x\ y\ z>>$ to $T_x$ ;
26 |   |   add $T_x$ to T;
27 return $x$;
```

Algorithm 3: Investigation on Relevant Graph Portion

For each pair of triples $<<a\ p\ c>> \in T_a$ and $<<b\ q\ d>> \in T_b$, Algorithm 3 performs a recursive call to investigate over the pairs of resources p and q (Row 14), and c and d (Row 22), unless such pairs have been already computed in any previous call (Rows 9–10 and 17–18). All sets of relevant triples inferred during the investigation are added to the global set T (Rows 4, 16, 24, 26).

The maximum number of recursive calls to be still performed, n_a and n_b, is updated at each call (Rows 12 and 22).

Algorithm 1 finally returns the result coming from Algorithm 3, together with the set T of triples required to fully describe x (Row 8 in Algorithm 1).

5.1 Computational Issues

Given $\langle a, T_a \rangle$ and $\langle b, T_b \rangle$ with $|T_a| = |T_b| = n$, we note that, in very artificial cases, a Common Subsumer $\langle x, T_x \rangle$ of them, can grow as large as $|T_x| \in O(n^2)$, and when generalized to k r-graphs, a Common Subsumer of all of them can grow as $O(n^k)$—*i.e.*, exponential in k. This is a very artificial worst case, since it presumes that every triple $<<a\ p\ c>>$ of T_a is "comparable" with every triple $<<b\ q\ d>>$ of T_b—*e.g.*, this case would occur if all triples use the same predicate $p = q$, and every pair of $\langle c, T_c\rangle$, $\langle d, T_d\rangle$ yields a non-trivial Common Subsumer.

A more realistic situation is that for each triple of T_a, only a constant number of triples of T_b, say $\alpha \geqslant 1$, yields a new triple added to T_x. Observe that this was the case for Example 1, with $\alpha = 1$. In this case, $|T_x| = \alpha n$, and for k resources, $O(\alpha^k n)$—still exponential in k for $\alpha > 1$. Even if $\alpha > 1$, a strategy to keep the Common Subsumer "small" could be, for each triple in T_a to choose one among the α triples of T_b which would add a triple to T_x—in practice, to force $\alpha = 1$. The result would be still a Common Subsumer anyway, even if less informative than the "original" one, whose size is $|T_x| = n$, independent of k.

6 Conclusion

Motivated by the need for greedily learning shared informative content in collections of **RDF** resources, we defined Common Subsumers. We decided not to handle Common Subsumers which are also subsumption minimal (known as Least Common Subsumer in DLs) because we need to refer to a selective representation of resources in terms of descriptive triples, in order to ensure computability in the reference dataset (the Web of Data), too large to be explored. The adoption of an anytime algorithm allows for using partial learned informative content for further processing, whenever the search for Common Subsumers is interrupted. Thanks to such distinguishing features, the proposed approach may support the clustering of collections of **RDF** resources, by exploiting associativity of Common Subsumers. Our future work will be devoted to the investigation on **RDF** clustering methods based on Common Subsumers computation, possibly adopting strategies proposed in our past research ([8],[9]).

Acknowledgments

We acknowledge support of project "Semantic Expert Finding for Service Portfolio Creation" funded by HP (Hewlett-Packard) Labs - Innovation Research Program Award.

References

1. Baader, F., Calvanese, D., Mc Guinness, D., Nardi, D., Patel-Schneider, P. (eds.): The Description Logic Handbook – 2nd edition. Cambridge University Press (2007)
2. Baader, F., Sertkaya, B., Turhan, A.Y.: Computing the least common subsumer w.r.t. a background terminology. J. Applied Logic 5(3), 392–420 (2007)
3. Beckett, D., Berners-Lee, T.: Turtle - Terse RDF Triple Language, W3C Team Submission. http://www.w3.org/TeamSubmission/turtle/ (2011)

4. Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web. Scientific American 248(4) (2001), (34-43)
5. Cohen, W., Borgida, A., Hirsh, H.: Computing least common subsumers in description logics. In: Rosenbloom, P., Szolovits, P. (eds.) Proc. of AAAI'92. pp. 754–761. AAAI Press (1992)
6. Cohen, W.W., Hirsh, H.: The learnability of description logics with equality constraints. Machine Learning 17(2-3), 169–199 (1994)
7. Cohen, W.W., Hirsh, H.: Learning the CLASSIC description logics: Theoretical and experimental results. In: Doyle, J., Sandewall, E., Torasso, P. (eds.) Proc. of KR'94. pp. 121–133 (1994)
8. Colucci, S., Di Noia, T., Di Sciascio, E., Donini, F., Piscitelli, G., Coppi, S.: Knowledge Based Approach to Semantic Composition of Teams in an Organization. In: Proc. of SAC '05. pp. 1314–1319. ACM (2005)
9. Colucci, S., Di Sciascio, E., Donini, F.M., Tinelli, E.: Finding informative commonalities in concept collections. In: Proc. of CIKM '08,. pp. 807–817. ACM (2008)
10. d'Amato, C., Fanizzi, N., Lawrynowicz, A.: Categorize by: Deductive aggregation of semantic web query results. In: Proc. of ESWC '10. pp. 91–105. Springer (2010)
11. De Giacomo, G., Lenzerini, M., Rosati, R.: Higher-order description logics for domain metamodeling. In: Proc. of AAAI '11 (2011)
12. Delteil, A., Faron-Zucker, C., Dieng, R.: Learning ontologies from RDF annotations. In: "Proc. of the Second Workshop on Ontology Learning (2001)
13. Fanizzi, N., d'Amato, C., Esposito, F.: Metric-based stochastic conceptual clustering for ontologies. Information Systems 34(8), 792–806 (2009)
14. Frazier, M., Pitt, L.: Classic learning. In: Proc. of the 7th Annual ACM Conference on Computational Learning Theory. pp. 23–34. ACM (1994)
15. Garcia-Plaza, A., Fresno, V., Martinez, R.: Web page clustering using a fuzzy logic based representation and self-organizing maps. In: Proc. of WI-IAT '08. vol. 1, pp. 851–854. ACM (2008)
16. Grimnes, G.A., Edwards, P., Preece, A.: Instance based clustering of semantic web resources. In: Proc. of the ESWC'08. pp. 303–317. Springer (2008)
17. Hayes, P.: RDF semantics, W3C recommendation. http://www.w3.org/TR/2004/REC-rdf-mt-20040210/ (2004)
18. Lawrynowicz, A.: Grouping results of queries to ontological knowledge bases by conceptual clustering. In: Proc. of ICCCI 2009. vol. 5796, pp. 504–515. Springer (2009)
19. Madhavan, J., Cohen, S., Dong, X.L., Halevy, A.Y., Jeffery, S.R., Ko, D., Yu, C.: Web-scale data integration: You can afford to pay as you go. In: Proc. of CIDR '07. pp. 342–350 (2007)
20. Mahmoud, H.A., Aboulnaga, A.: Schema clustering and retrieval for multi-domain pay-as-you-go data integration systems. In: Proc. of SIGMOD '10. pp. 411–422. ACM (2010)
21. Pan, J.Z., Horrocks, I.: RDFS(FA): Connecting RDF(S) and OWL DL. IEEE Trans. on Knowl. and Data Eng. 19(2), 192–206 (Feb 2007)
22. Shadbolt, N., Hall, W., Berners-Lee, T.: The semantic web revisited. Intelligent Systems, IEEE 21(3), 96–101 (2006)
23. ter Horst, H.J.: Completeness, decidability and complexity of entailment for RDF Schema and a semantic extension involving the OWL vocabulary. Journal of Web Semantics 3(2–3), 79–115 (2005)
24. Zeng, H.J., He, Q.C., Chen, Z., Ma, W.Y., Ma, J.: Learning to cluster web search results. In: Proc. of SIGIR '04. pp. 210–217. ACM (2004)