Generating complex ontology instances from documents*

Roxana Danger[1] and Rafael Berlanga[2]

[1] Department of Information Systems and Computation,
at Technical University of Valencia, Spain,
rdanger@dsic.upv.es
[2] Departamento de Lenguajes y Sistemas Informáticos,
at Universitat Jaume I, Spain,
berlanga@uji.es

Abstract. A novel *Information Extraction* system able to generate complex instances from free texts available on the Web is presented in this paper. The approach is based on non monotonical processing over ontologies, and makes use of entity recognizers and disambiguators in order to adequately extract and combine instances and relations between them. Experiments conducted over the archaeological research domain provide satisfactory results and suggest that the tool is suitable for its application on Semantic Web resources.

1 Introduction

The Semantic Web is a form of web conceived for allowing human users and software tools to process and share the same sources of information. It builds on a set of standards which ensure syntactic consistency and semantic value. Large communities are participating in its development, producing as a result huge domain ontologies with very rich lexicons. Consequently, the problem of identifying possible instances of these ontologies, usually called *semantic annotation*, *ontology population* or *instance extraction*, has become crucial.

The formalization of a framework for the consistent generation of *complex instances* for the Semantic Web is the main contribution of this paper. A complex instance is an ontological instance involving several levels of aggregation between the entities mentioned in the document. Complex instances can also consist of summaries of sets of instances (i.e. generalized instances) and can include negative properties in their definition. To the best of our knowledge, no published work deals with this kind of instances, although many methods are available for extracting plain facts from text chunks.

Our system is able to collect complex instances and their relationships across the whole document, using the structure of the document and the relations between concepts expressed through the ontology. A non monotonic processing

*This work has been partially funded by the "Juan de la Cierva" program of the Ministry of Education and Science of Spain.

through an initial set of instances allows one to update the initial knowledge base according to the semantic descriptions of the ontology. Contradictory or inconsistent data can be instead removed from the knowledge base.

Section 2 of this paper reviews the current approaches for automatic ontology population. Section 3 provides a global description of our approach, which is formalized in Sections 4 and 5. Section 6 describes the practical process of instance extraction, which experimental results are presented in Section 7.

2 Related research

Semantic annotation methods can be classified into two fundamental types [1, 2]: based on *patterns* and based on *machine learning*. Pattern-based methods can be further divided into two subgroups: those in which extraction rules arise from an initial set of tagged entities [3–5], and those in which extraction rules are manually defined [6, 7]. On the other hand, probabilistic machine learning methods rely on statistical models to predict the location of entities in texts [9–12], while inductive machine learning methods deduce entity recognition rules from the syntactical analysis of texts [13–17].

Most annotation methods require a complete syntactic analysis, and in some cases a semantic analysis too, including co-reference and anaphora resolutions. Current syntactic parsers are error prone and their performance is often not satisfactory even for medium-size document collections. These methods are thus generally not suitable for the large scale analysis required by the Semantic Web (with the exception of probabilistic methods, which avoid syntactic analysis). Methods based on manual definition of rules require moreover a constant updating and a customized adjustment to each scenario.

The use of training corpora, such as in machine learning methods, is another source of complexity. In fact, corpora creation is a very resource consuming task, and learned models depend heavily on corpora, which are usually restricted to a few application domains. External linguistic tools and repositories (which consist of databases of named entities, dictionaries, thesauri and general purpose searchers like Google) are a good alternative. Regarding the nature of the information to extract, only a few methods [18, 10, 5] deal with aggregated instances and relations between entities, and the discovered associations only involve one relation, i.e., the extracted instances only have one aggregation level. The method proposed in this paper is designed to improve information extraction especially for what concerns the efficient identification of complex instances, as explained in the following section.

3 A new approach to multi-level semantic annotation of ontological instances

A schema of the approach we propose for Semantic Web population is shown in Figure 1. The information stored in natural language has to be extracted following strategies similar to those employed in information extraction systems. As

a first step, a (web) document is parsed by a *wrapper* in order to determine its syntactical structure (chapters, sections, paragraphs, etc.). At this stage, each text segment is associated to a scope definition, that indicates which other segments can be related to it according to the hierarchical document structure. The focus of this paper is on the following step, the *instance extractor* web service. The extractor makes use of OWL ontologies, which define concepts and relations between them. We assume the existence of lexicons, which can be expressed in OWL language, and describe lexical rules to identify concepts and relations in the ontologies. Considering such lexicons, a parsed document can be processed to extract the ontology entities mentioned in the document, by using similarity functions between text fragments and lexical descriptions.

Extracted entities are used to define an initial instance set. Then, by applying several inference rules (which take into account the knowledge contained in the ontology) and the segment scope definitions, new relations can be added to connect instances in the initial set, and instances representing a unique object are properly joined. This process is performed non monotonically, as new instances are formed but others are deleted from the knowledge base in case of contradiction. As a result, the system generates a set of complex instances that semantically describe the whole content of the document, according to the given domain ontology. This approach has the advantage of being independent from the ontology representation formalism (e.g. frames versus Description Logic).

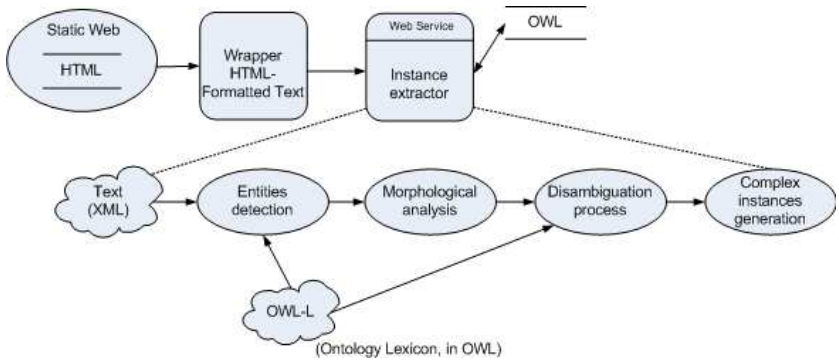

Fig. 1. Proposal for ontology population

A formal definition of our proposal for the process of instance extraction is provided in the next sections. In Section 4, the basic definitions concerning ontologies, lexicon, instances and operations over relations defined on ontologies are given. Operations for joining and aggregating instances and the transformations these operations imply over all the knowledge base are described in Section 5.

4 Formal conceptualization

Description Logic is considered as the logical framework that better adapts to the requirements of the Semantic Web, and OWL has been defined as the language for implementing ontologies. Different reasoners, which read the specifications in OWL format and infer the complete meaning of concepts, have been implemented. Any kind of reasoner (weak or strong, according to the set of tested semantic restrictions) allows the recovering of two kinds of relation between concepts: hierarchical and aggregational ones . This is the minimum useful information needed for an information extraction process. Here, we assume the availability of a reasoner to extract this essential information. An extended Tableau algorithm for extract such information for ontologies in SHOIQ(D) languages is described in [2].

The basic definitions required by our framework for operating over instances are introduced next. An ontology (Definition 4) is defined by: a) the semantic specification of a set of concepts and their relations, the abstract ontology (Definition 1); and b) the description of the valid data for each concept, the lexicon (Definition 3). These definitions are adapted from the work of Maedche [19].

Operations of specialization and abstraction over a relation of an ontology (Definition 2) allow to move through the relation hierarchy, recognizing the most specific relation associated to a given concept and the initial relation itself. In this way, relations $R|_c$ associated to a concept c can be derived. An instance (Definition 5) is defined by the set of relations and data associated to it.

Definition 1. *(Abstract ontology).* An abstract ontology is a structure $\mathcal{O}= (C, \leq_C, R, \sigma, card, \leq_R, IR)$ consisting of:

- two disjoint sets C and R whose elements are called concepts and relations, respectively,
- a partial order $\leq_C$ on C, called concept hierarchy or taxonomy,
- a function $\sigma : R \rightarrow C \times 2^C$, called signature,
- a cardinality function, $card : R \rightarrow N^0 \times N$, that represents the minimum and maximum cardinality of each relation, where N^0 and N are the sets of natural numbers including or not zero, respectively,
- a partial order $\leq_R$ on R, called relation hierarchy, where $r \leq_R r'$ implies that $\prod_1(\sigma(r)) \leq_C \prod_1(\sigma(r'))$, for $r, r' \in R$,

  The function $dom : R \rightarrow C$ with $dom(r) = \prod_1(\sigma(r))$ gives the domain of r, the function $range : R \rightarrow 2^C$ with $range(r) = \prod_2(\sigma(r))$ gives its range. We call the *set of datatypes* $TC = \{c/c \in C, \forall c', c' \leq_C c, \neg\exists r \in R, dom(r) = c'\}$, $TC \subseteq C$.

As an example, the central fragment of the archaeology ontology which will be used in the rest of the paper to explain concepts and experimental tests is represented in Figure 2.

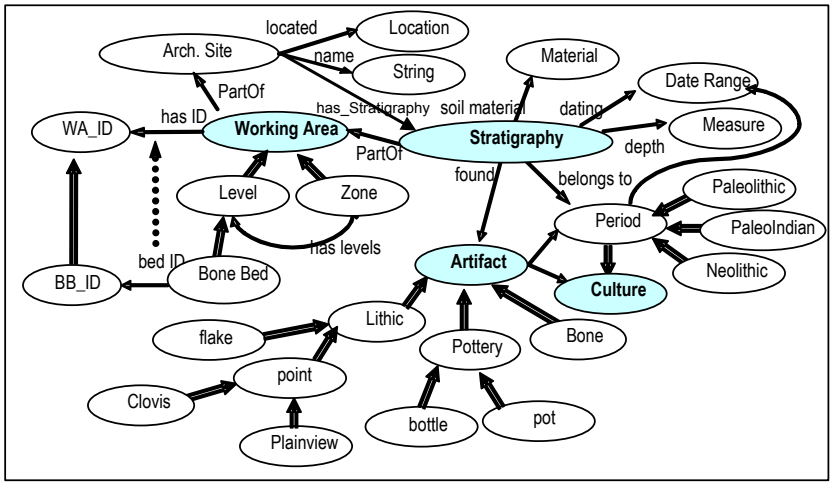

Fig. 2. Fragment of the application ontology. Arrows denote concept relationships, thick arrows denote is-a relationships ($\leq_C$), whereas dotted arrows denote the $\leq_R$ taxonomy. Shadow ovals are the entry concepts of the different parts of the ontology.

Definition 2. *(Specialization of a relation)* We define the specialization of a relation r with respect to a concept c', $c' \leq_C dom(r)$, denoted with $e\mid_{c'}(r)$, the relation:

$$e\mid_{c'}(r) = r' | r' \leq_R r, c' \leq_C dom(r') \leq_C dom(r), \neg\exists! r'', r'' <_R r' \leq_R r, \\ c' \leq_C dom(r'') \leq_C dom(r') \leq_C dom(r)$$

The specialization of a relation r with respect to a concept c', called r', is the unique and most specific relation for which the domain of r is an abstract concept of the domain of r', and the latter is an abstract concept of c'. Thanks to the reflexivity in partial orders, r' is relation r itself if no other more specific relations of r satisfy the conditions.

For example, given the ontology of Figure 2, the following is a specialization of a relation:

$e\mid_{Bone\_Bed}(Working\_Area.id) = Bone\_Bed.BB\_id$

In this way, we can define the set of relations associated to a class c as the set: $\cup_{c\leq_C c'}\{e\mid_c(r) | dom(r) = c'\}$, which is called *relation set of* c and it is denoted as $R|_c$. For example, $R|_{Bone\_Bed} = \{Bone\_Bed.BB\_id, Working\_Area.partOf\}$.

The abstraction of a relation r, $a\mid_{c'}(r)$, can be defined in a similar way: $a\mid_{Level}(Bone\_Bed.BB\_id) = Working\_Area.id$.

It is easy to prove that $r \in R|_c, r' \in R|_{c'} \Rightarrow (r' = e|_{c'}(r) \Leftrightarrow r = a|_c(r'))$.

In order to bridge an abstract ontology with its related lexical terms, we introduce the concept of lexicon.

Definition 3. *(Lexicon).* A lexicon for an abstract ontology $\mathcal{O}$ is the structure $\mathcal{L} = (S_C, S_R, Ref_C, Ref_R, Insts)$ consisting of:

– two sets S_C and S_R whose elements are called signs (lexical entities with specific semantics) for concepts and relations, respectively,
– two relations $Ref_C \subseteq S_C \times C$, $Ref_R \subseteq S_R \times R$ called lexical reference assignment for concepts and relations, respectively,
– a set of instances, $Insts : Literal \rightarrow C \times 2^{Literal \times R \times S_R \cup Literal}$, where $Literal$ denotes the set of possible strings that can be used to nominate instances (see Definition 5).

Besides, based on Ref_C and Ref_R, we define:
$Ref_C(s) = \{c \in C | (s, c) \in Ref_C\}$,
$Ref_C^{-1}(c) = \{s \in S_c | (s, c) \in Ref_C\}$,
$Ref_R(s)$ and $Ref_R^{-1}(r)$ are analogously defined and,
$AllRef_C^{-1} = \cup_{c' \leq_C c} Ref_C^{-1}(c')$ and $AllRef_R^{-1} = \cup_{r' \leq_R r} Ref_R^{-1}(r')$

In most cases, S_C and S_R are intentionally defined (i.e. through logical clauses); for example, "$BoneBed1''$ $\in$ $Ref_C^{-1}(BB\_id)$, "$South\ of\ Texas''$ $\in$ $Ref_C^{-1}(Location)$ and "$situated''$ $\in Ref_R^{-1}(Arch\_Site.located)$.

Definition 4. *(Ontology)* A (concrete) ontology is a pair $(\mathcal{O}, \mathcal{L})$ where O is an abstract ontology and $\mathcal{L}$ a lexicon for $\mathcal{O}$.

From now on, all definitions refer to the concept of concrete ontology. Besides, we will assume the existence of the function $NewLit$ that returns a new literal, not used on $Insts$, that can be used to nominate a new instance.

Definition 5. *(Instance)* We define an instance named o of class $c \in C - TC$ or simply instance, the set $I|_o^c = \{(o, r, o') | r \in R|_c\}$. Function $Inst(o) = (c, I|_o^c)$ defines the relation between the name of an instance and its most specific concept and description. We call $dc(o)$ the direct class associated to o: $dc(o) = \prod_1(Inst(o)) = c$. If $I|_o^c = \emptyset$, then o is an empty[3] instance of class c.

Using the previous example, the following sets are examples of instances:
$I|_{o_1}^{Arch\_site} = \{(o_1, Arch\_site.name, "Bonfire\ Shelter''),$
$(o_1, Arch\_site.located, "South\ of\ Texas'')\}$

$I|_{o_2}^{Bone\_Bed} = \{(o_2, Bone\_Bed.BB\_id, "Bone\ Bed\ 1''),$
$(o_2, Working\_Area.partOf, o_1)\}$

$I|_{o_3}^{Stratigraphy} = \{(o_3, Stratigraphy.partOf, o_2),$
$(o_3, Stratigraphy.found, o_4), (o_3, Stratigraphy.found, o_5)\}$

$I|_{o_4}^{Bone} = \emptyset$ $\quad$ $I|_{o_5}^{Clovis} = \{(o_5, Artifact.belongsTo, o_6)\}$ $\quad$ $I|_{o_6}^{paleolithic} = \emptyset$

Notice that o_4 and o_6 are empty instances.

[3] An empty instance is an anonymous instance of which nothing is known.

5 Instance Operations

In knowledge modeling, operations between concepts (or classes) like union, intersection, difference, complement, etc. are formally defined. However, this is not the case for the same set of operations between instances. In the Logic paradigm the description of all axioms associated with an instance is enough to represent it semantically, and a reasoning process has to be performed in order to retrieve a complete and compact description of an instance. In the Object-Oriented (OO) paradigm, the independence of instances based on their identifiers (name, physical direction, etc.) is assumed. Regarding the Logic paradigm, the proposed instance operations allow to maintain in the knowledge base a complete and compact description of all instances, avoiding repetitive simple reasoning for those cases in which the ontology can capture the structure of all possible instances (for example in most Description Logics Languages). For what concerns the OO paradigm, the proposal gives a formal definition for instance transformation.

In this section we introduce a set of definitions for transforming ontological instances of a concrete ontology through specialization, abstraction or combination operations, since they are the relevant ones for the present paper. These operations are used after concepts and relations mentioned in texts have been identified in order to construct complex instances (see Section 6).

Definition 6. *(Specialization of an instance)* We call specialization of instance $I|_o^c$ into class c', $c' \leq_C c$, denoted with $I|_{o\rightarrow(NewLit=o')}^{c\rightarrow c'}$, the instance $I|_{o'}^{c'}$ defined by the set:

$$\{(o', r', x') | (o, r, x) \in I|_o^c, e|_{c'}(r) = r', range(r') = \{c''\}, \exists I|_{x\rightarrow(NewLit=x')}^{dc(x)\rightarrow c''} = I|_{x'}^{c''}\} \cup$$
$$\cup \{(o', r', x) | (o, r, x) \in I|_o^c, e|_{c'}(r) = r', \exists c'' \in range(r'), dc(x) \in TC, x \in S_C(c'')\}.$$

The specialization of an instance is the replacement of each description (o, r, x) of o with the specialized description (o', r', x'), where r' is the specialized relation of r respect to c' and x' the specialized instance of x according to the range of r'. Notice that $o \rightarrow (NewLit = o')$ renames the instance $I|_o^c$ with the new literal o' obtained by using $NewLit$. Additionally, if $I|_o^c$ is an empty instance, $I|_{o'}^{c'}$ will be an empty instance too, the specialization process only modifies its name and the actual concept associated with it.

For example, the specialization of the instance:

$I|_o^{Working\_Area} = \{(o, Working\_Area.id, \text{“}Bone\ Bed\ 1\text{”}),$
$(o, Working\_Area.partOf, o_1)\}$ to the class $Bone\_Bed$ is:

$I|_{Bb\ 1}^{Bone\_Bed} = I|_{o\rightarrow Bb\ 1}^{Working\_Area\rightarrow Bone\_Bed} = \{(Bb\ 1, Bone\_Bed.BB\_id, \text{“}Bone\ Bed\ 1''),$
$(Bb\ 1, Working\_Area.partOf, o_1)\}.$

Definition 7. *(Specialization of an instance without missing information)* We call specialization of instance $I|_o^c$ to class c' without missing information, $c' \leq_C c$, denoted by $I|_{o\hat{\rightarrow}(NewLit=o')}^{c\hat{\rightarrow}c'}$ the instance:

$$I|_{o'}^{c'} = I|_{o\rightarrow(NewLit=o')}^{c\rightarrow c'} \cup \{(o', r, x) | (o, r, x) \in I|_o^c, e|_{c'}(r) = r', |range(r')| > 1\}.$$

This definition includes also those descriptions whose relations can not be specialized by a unique concept. Definitions concerning abstraction of an instance with or without missing information can be similarly derived.

Definition 8. *(Union of instances)* We call union of instances $I|_o^c$ and $I|_{o'}^{c'}$, denoted by $I|_o^c \hat{\cup} I|_{o'}^{c'} = I|_{NewLit=o''}^{c'}$, the instance $I|_{o''}^{c'}$ whose description can be computed by:

$$I|_{o\to o''}^{c\to c'} \cup I|_{o'\to o''}^{c'} - \cup\{d|(d,d')or(d',d) \in cd\} \cup \{Un(x_1,x_2)|((o'',r,x_1),(o'',r,x_2)) \in cd\}$$
$$\text{such that } \forall((o'',r,x_1),(o'',r,x_2)) \in cd, Un(x_1,x_2) \neq undef \ ,$$

$$cd = \{((o'',r,x_1),(o'',r,x_2))|(o'',r,x_1) \in I|_{o\to o''}^{c\to c'}, (o'',r,x_2) \in I|_{o'\to o''}^{c'}, \prod_2(card(r)) = 1\}$$

$$Un(x_1,x_2) = \begin{cases} x_1 & x_1,x_2 \in TC, x_1 = x_2 \\ undef & \in TC, x_1 \neq x_2 \\ I|_{x_1}^{dc(x_1)} \hat{\cup} I|_{x_2}^{dc(x_2)} & \text{otherwise} \end{cases}$$

cd represents the pairs of descriptions that should be unified in one value, employing the Un operation. In this way, the union of two instances is a new instance whose description is specialized to the most specific concept (between the joined instances), and where the pairs of functional relations are joined recursively by using the same union definition.

For example, instances: $I|_o^{Arch\_Site} = \{(o, Arch\_Site.name, \text{“}Bonfire\ Shelter''\text{)}\}$ and $I|_{o'}^{Arch\_Site} = \{(o, Arch\_Site.located, \text{“}South\ of\ Texas''\text{)}\}$ can be joined in instance:

$I|_{o''}^{Arch\_Site} = \{(o'', Arch\_Site.name, \text{“}Bonfire\ Shelter''\text{)}\}$, $(o'', Arch\_Site.located, \text{“}South\ of\ Texas''\text{)}\}$ supposing that o'' is the new literal returned by $NewLit$ function.

The last operation to formalize is the aggregation between instances. Its goal is to connect instances by using a set of relations and concepts semantically linked. For this reason, the notion of *path* between concepts is first introduced.

Definition 9. *(Path between concepts)* The list $(r_1,c_1),...,(r_n,c_n)$ is a path of an ontology $\mathcal{O}$ if:

- $\forall c_k, k \in \{1,...,n\}, c_k \leq_C c^*, c^* \in range(r_k)$
- $\forall c_k, k \in \{1,...,n-1\}, c_k \leq_C dom(r_{k+1})$

A path of an ontology is an ordered list of pairs $(relation, concept)$ through which a concept (c_n) can be reached from an other concept $(dom(r_1))$ by using the definitions of the ontology.

For example, $p = (Arch\_site.has\_Stratigraphy, Stratigraphy)$, $(Stratigraphy.found, bottle)$ is a path between *archaeological site* and *artifacts* concepts, whereas $p_1 = (Bone\_Bed.BB\_id, BB\_id)$ is a path between *Bone Bed* concept and the label used by archaeologists to indicate the exact place.

Definition 10. *(Aggregation of instances)* We call aggregation of instance $I|_{o'}^{c'}$ to instance $I|_o^c$, through path $p = (r_1,c_1),...,(r_n,c_n), c_0 = dom(r_1) \leq_C c, c_n \leq_C$

c', denoted by $I|_o^c \overset{p}{\leftarrow} I|_{o'}^{c'}$, the instance $I|_{o''}^{c_0}$ if its description can be computed by:

$$\begin{cases} \{(o'', r', x) \in I|_{o \to o''}^{c \to c_0} | r' \neq r\} \cup & \text{if } \prod_2 card(r_1) = 1, \exists (o, r, x) \in I|_o^c, r_1 \leq_R r, \\ \quad \{(o'', r_1, x_1^{'}) | \exists I|_{x_1^{'}}^{c_1} = I|_x^{dc(x)} \hat{\cup} I|_{x_1}^{c_1}\} & \quad \prod_2 card(r) = 1, \exists I|_x^{dc(x)} \hat{\cup} I|_{x_1}^{c_1} \\ \\ \{(o'', r', x) \in I|_{o \to o''}^{c \to c_0}\} \cup \{(o'', r_1, x_1)\} & \text{if } \prod_2 card(r_1) > 1, \forall r, r_1 \leq_R r \text{ where} \\ & \quad \exists (o'', r', x) \in I|_{o \to o''}^{c \to c_0}, \\ & \quad |\{(o'', r^*, v) | r^* \leq_R r\}| < \prod_2 card(r) \end{cases}$$

and where $x_1, x_2, ..., x_n$ are new instances described by:
$Inst(x_i) = (c_i, \{(x_i, r_i, x_{i+1})\}), i \in 1, ..., n-1$,
$Inst(x_n) = (c_n, \{(x_n, r_n, o^*)\})$ and $I|_{o^*}^{c_n} = I|_{o' \to o^*}^{c' \to c_n}$.

For example, instance $I|_{o'}^{bottle} = \emptyset$ can be aggregated to $I|_o^{Arch\_Site} = \emptyset$ by using path p of the previous example. $I|_{o''}^{Acrh\_Site} = \{(o'', has\_Stratigraphy, os)\}$, $I|_{os}^{Stratigraphy} = \{(os, Stratigraphy.found, o^*)\}$ and $I|_{o^*}^{bottle} = I|_{o' \to o^*}^{bottle}$ are new instances generated by the aggregation process.

Finally, the following two definitions establish the conditions to guarantee the consistence of the union and aggregation processes in a concrete ontology.

Definition 11. *(Complementary instances)* Two instances $I|_o^c$ and $I|_{o'}^{c'}$ of classes c and c' respectively, $c' \leq_C c$, are complementary, denoted with $I|_o^c \circ I|_{o'}^{c'}$, if they satisfy at least one of the following conditions:

- at least one of instances $I|_o^c$ or $I|_{o'}^{c'}$ is empty,
- for all relations in $I|_{o \hat{\to} o'}^{c \hat{\to} c'} \hat{\cup} I|_{o'}^{c'}$, i.e. the instance union of instances $I|_o^c$ and $I|_{o'}^{c'}$ specialized into c' without missing information, cardinality restrictions imposed by the ontology (through *card* function) are satisfied.

Two non-empty instances are complementary if their union, considering their non missing information specialization, maintains the cardinality property for all its relations. Notice that their union only contains the relations in $R|_{c'}$.

For example, $I|_o^{Arch\_Site} = \{(o, Arch\_Site.name, \text{“}Bonfire\ Shelter''\text{)}\}$ and $I|_{o'}^{Arch\_Site} = \{(o, Arch\_Site.located, \text{“}South\ of\ Texas''\text{)}\}$ are complementary, and can be joined as in the previous example. However, $I|_o^{Arch\_Site}$ is not complementary to $I|_{o''}^{Arch\_Site} = \{(o'', Arch\_Site.name, \text{“}Bolomor\ Cove''\text{)}\}$. In this case relation $Arch\_Site.name$ is biyective (so, its maximal cardinality is one). Instances $I|_{o_4}^{Bone}$ and $I|_{o_5}^{Colvis}$ are also not complementary because their concepts can not be ordered through the $\leq_C$ relationship.

Definition 12. *(Aggregable instances)* We say that instance $I|_{o'}^{c'}$ is aggregable to $I|_o^c$ if one and only one path, p, exists between concepts c^* and $c^{'*}$, $c^* \leq_C c$ and $c^{'*} \leq_C c'$ and $I|_o^c \overset{p}{\leftarrow} I|_{o'}^{c'}$ can be computed.

For example, instance $I|_{o_1}^{Arch\_Site}$ can be aggregated to instance $I|_{o}^{Stratigraphy} = \{(o, Stratigraphy.found, o_4), (o, Stratigraphy.found, o_5)\}$, producing:

$I|_{o}^{Stratigraphy} = I|_{o}^{Stratigraphy} \cup \{(o, Stratigraphy.PartOf, o_7)\}$ where
$I|_{o_7}^{Working\_Area} = \{(o_7, Working\_Area.PartOf, o_1)\}$

Definitions 11 and 12 allow us to introduce the following ***transformation rules of a concrete ontology***, which define the transformations on the set of $Insts$ functions during the union and aggregation operations. Given a concrete ontology, $(\mathcal{O}, \mathcal{L})$, it can be transformed to $(\mathcal{O}, \mathcal{L}')$ by the following:

TR-Union to join two instances $o, o' \in Insts$
if $I|_o^c \circ I|_{o'}^{c'}$ then $\mathcal{L}' = (S_C, S_R, Ref_C, Ref_R, Insts')$, where
$Insts' = Insts - \{(o, dc(o), I|_o^{dc(o)}), (o', dc(o'), I|_{o'}^{dc(o')})\} \cup \{(o'', c', I|_o^c \hat{\cup} I|_{o'}^{c'})\}$
and o'' is the new literal generated during the union process.
else *prevent Contradiction*.

TR-Aggreg to aggregate o' to o, $o, o' \in Insts$
if o' is aggregable to o then $\mathcal{L}' = (S_C, S_R, Ref_C, Ref_R, Insts')$, where
$Insts' = Insts - \{(o, dc(o), I|_o^{dc(o)}), (o', dc(o'), I|_{o'}^{dc(o')})\} \cup \{(o'', c', I|_o^c \overset{p}{\leftarrow} I|_{o'}^{c'}),$
$(o^*, dc(o^*), I|_{o' \rightarrow o^*}^{c' \rightarrow dc(o^*)}), (x_i, dc(x_i), I|_{x_i}^{dc(x_i)}) | i \in \{1, ..., n\}\}$, p is the unique aggregation path from concept c to c' ; $o'', x_1, ..., x_n$ are the new literals generated during the aggregation process as defined in Definition 10.
else *prevent Contradiction*.

The TR-Union and TR-Aggreg transformation rules allow to perform a non monotonic processing on the knowledge base. With *prevent Contradiction* the caller process verifies if the transformation results are as expected. Given a pair of instances of the $Insts$ set (the knowledge base) which have to be joined or aggregated, the caller process could delete both of them in case of contradiction, and/or adequately update the knowledge base. Therefore, incremental grow of the database is not guaranteed, but rather depends on the restrictions defined by the ontology used during the processes of union and aggregation. The whole instance extraction process, which uses the above rules to generate the set of complex instances described in a text, is explained next.

6 Extracting instances from texts

An initial instance set has to be obtained from a text before the transformation rules can be applied. Therefore, the text has to be analyzed and the fragment texts related to ontological entities have to be extracted. The solution we propose in [2] for executing this step considers entity name recognition, negation scope computing, generalized instance description identification, and a disambiguation step to recognize entities (fragments of the text) semantically associated to an ontology (called ontological entities), based on a morphological analysis of the text.

We consider texts with a syntactical division in chapters, sections, subsections, etc., and such division defines the scope (initial and final paragraphs)

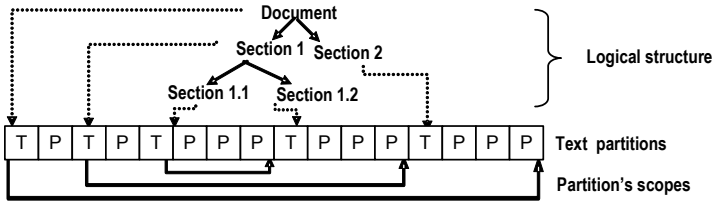

Fig. 3. Partitioning a structured document. T=Section title; P=paragraph.

of the subject of each particular part (as represented in Figure 3). Hence, the descriptive fragment texts (i.e. those fragments that describe ontology entities) are used to form the initial set of instances and each of these instances is also associated to the semantic scope of the text partition where the entity has been found. The scope of an instance delimits the instances that can be joined or aggregated with it: those whose scope overlap with it. Then, instances in the initial set are combined following the concepts for joining and aggregating instances introduced in Sections 4 and 5.

Algorithm 1 formally describes the process of complex instance extraction. The first step, the construction of the initial set of instances, considers all recognized entities and define a new instance for each of them: if the recognized entity is a concept or a relation and it is possible to infer an unique class having this relation, a new instance of the (inferred) concept is created and a scope is properly associated to it. During the second step, instances whose scope are overlapped are combined in the order in which these instances appears, while no contradictions are obtained with the application of the *TR-Union* and *TR-Aggreg* rules. The final result of all this process is a set of complex instances representing the information related to the ontology which is contained in the text.

7 Experimental results

In spite of the importance and the number of different approaches dedicated to information extraction, no universally recognized test datasets are available for comparing the results of different systems. Available test data are reduced to entities, or to very simple concepts with few levels of aggregability. These datasets are of very little use in this case, as our work is especially focused on the capability of using relevant entities to create complex instances, selecting in each case appropriate pair of instances to be joined or aggregated. Hence, until datasets suitable for comparison become available, we have to test our approach on custom test data. We present here the experimental results obtained applying our approach to a set of paragraphs extracted randomly from nine archaeological memories written in Spanish.

Figure 4 shows a screenshot of our information extractor system. A paragraph on an archaeological site memory has been selected, and the complex instances

Algorithm 1 Complex instance extraction

Require: $(\mathcal{O}, \mathcal{L})$, T
  {$(\mathcal{O}, \mathcal{L})$, ontology}
  {T, text of the ontology domain.}
Ensure: $Insts'$
  {$Insts'$, a set of new instances extracted from T.}

  Let EO be the ontological entities recognized in the text.
  Construction of initial set of instances.
  $cmbInsts = \emptyset$; $Insts' = \emptyset$
  for all $e \in EO$ **do**
    $o \leftarrow NULL$
    if $e \in C$ *(if e is a concept)* **then**
      $o \leftarrow NewLit$; $Insts' \leftarrow Inst' \cup \{(o, e, \emptyset)\}$
    else
      if $\exists! p, path(c, c')$ that contains the relation e (if e is a relation and exists a unique path that contain it) **then**
        $o \leftarrow NewLit$; $Inst' \leftarrow Inst' \cup \{(o, c, \emptyset)\}$; $o' \leftarrow NewLit$; $Inst' \leftarrow Inst' \cup \{(o', c', \emptyset)\}$
        Apply **TR-Aggreg** to aggregate o' to o
    if $o \neq NULL$ **then**
      $cmbInsts = cmbInsts \cup \{o\}$
      Associate to o the scope of e (as explained in the Section 6, the scope is defined considering the syntactical division of the text and the position where e has been found.)

  Combining entities.
  for all $\{o\} \in cmbInsts$ **do**
    if o has not been joined neither aggregated to other instance **then**
      for all $\{o'\} \in cmbInsts$ after e, where o and o' scopes are overlapped **do**
        if $dc(o) \leq_C dc(o') \vee dc(o') \leq_C dc(o)$ **then**
          Apply **TR-Union** to unify o and o'
        else
          Apply **TR-Aggreg** to aggregate o to o' or o' to o
          Interchange o and o' if o' has been aggregated to o.
        if *Contradiction* has been produced in this iteration **then**
          break (o can not be extended any more)

extracted from it are shown and highlighted on the text, and transcribed in the main window. In this example, the complexity of the process and the capacity of the system to infer instances with a high aggregations depth (in this example, five levels of aggregation have been inferred) can be appreciated.

The ontology is composed by 194 classes and 131 relations. It is not very large, but complex enough in order to obtain non-trivial specialization and aggregation depths (nine and five levels respectively). Current semantic annotation tools usually deal with much simpler ontologies. The lexicon is composed of synonyms and contexts for the concepts and relations of the ontology. Besides, a set of 28 named entities (e.g. archaeological sites and stratigraphic unit names, weight

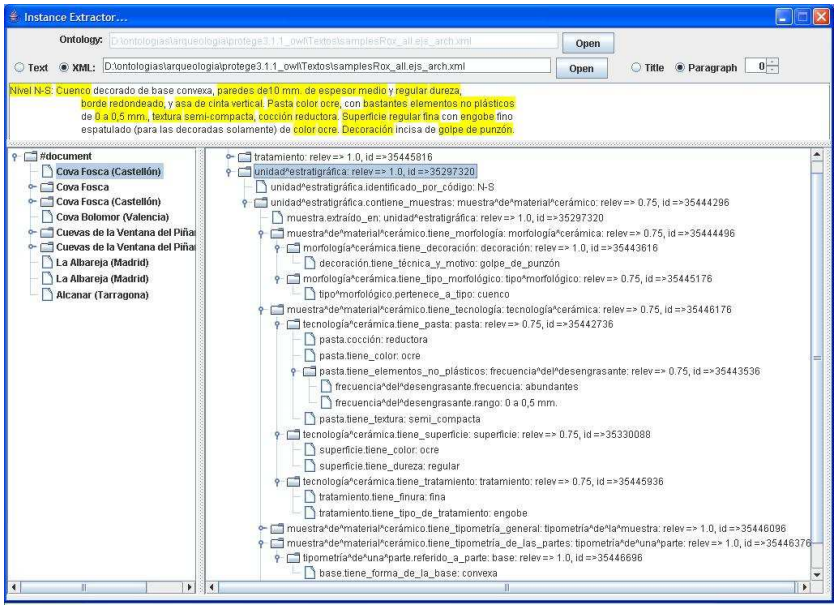

Fig. 4. Screenshot of the instance extractor system.

and dimensions of materials, percentages, etc.) were defined and the EEON (acronym of named entities extractor, in Spanish) module [2] allows their correct extraction, with over 90% of precision and recall[4] for all entities.

A characterization of the sample texts, according to the number of paragraphs, instances and relations in them, as well as the number of correct and incorrect extractions obtained by the system, the precision and recall for each sample text and for the overall dataset are shown in Table 1. The results allow to compare the correctness of: 1) the class associated to each part, 2) the relations associated to each identified instance, and 3) both elements in complex instances.A retrieved complex instance is correct if it belongs to the same class and describes an overlapped segment text of a target complex instance. A retrieved relation is correct if it belongs to a correct instance and the same relation (name and associated value) appears in the corresponding target instance.

The text samples contain 96% of the ontology entities, and a precision of around 99% was achieved in the extraction of these entities. This good result constitutes the ideal starting point for the manipulation and combination of initial instances into more complex ones.

All precision measures obtained for complex instances are over 90%, suggesting that the great majority of extracted instances are correct. It is important to

[4]Precision and recall are classical measures of Information Retrieval Systems. Precision is associated with the capacity of extracting entities correctly, while the recall is associated with the capacity of extracting entities as much as posible.

	Targets			Target Retrieval				Performance Indexes				Overall	
Sample text	Paragraphs	Instances (I)	Relations (R)	Correct Instances (CI)	Incorrect Instances (II)	Correct Relations (CR)	Incorrect Relations (IR)	Instance Precision (IP)	Instance Recall (IR)	Relation Precision (RP)	Relation Recall (RR)	Precision (CIP)	Recall (CIR)
1	15	26	222	25	2	199	6	0,93	0,96	0,97	0,90	0,97	0,90
2	5	8	56	6	0	19	0	1	0,75	1	0,34	1	0,39
3	6	9	26	8	0	25	0	1	0,89	1	0,96	1	0,94
4	1	1	2	1	0	2	0	1	1	1	1	1	1
5	55	68	85	54	6	75	19	0,9	0,79	0,79	0,88	0,84	0,84
6	13	28	105	22	0	82	4	1	0,78	0,95	0,78	0,96	0,78
7	5	5	3	5	0	2	1	1	1	0,67	0,67	0,875	0,875
8	2	4	21	3	0	20	3	1	0,75	0,87	0,95	0,88	0,92
9	2	4	18	3	0	18	1	1	0,75	0,95	1	0,95	0,95
Total	104	153	538	127	8	442	34	0,94	0,83	0,93	0,82	0,93	0,82

Table 1. Results for sample texts. IP = CI/(CI+II); IR = CI/I; RP = CR/(CR+IR); RR = CR/R; CIP = (CI+CR)/(CI+II+CR+IR); CIR = (CI+CR)/(I+R).

highligth that they are complex instances: this implies that not only the lexical definitions and the disambiguation process allow to retrieve the correct entities, but also that aggregation operations are correctly performed on the right instances. On the other hand, at least 75% of all complex instances in a sample text were correctly extracted. This relatively low value of recall is due to the use of lists and/or very complex explanations to describe instance sets. This kind of expressions lead the system to wrong decisions, creating a single complex instance instead of several ones. This error decreases the recall values for both instances and relations (see results for texts 2 and 6), and the same problem affects relation extraction precision, as can be observed for texts 5 and 7.

The overall precision of the system is around 93%, with a recall of 82%. These results are very satisfactory and promising in comparison with other systems available in the literature [18, 10, 5]. For a brief comparison with others methods, a few conclusions can be drawn:

1. The systems explained in [10], [9], [11] and [12] do not perform any syntactic analysis. In the first case, the precision is similar to ours, but in the second one, the precision and the recall are lower. The system in [12] obtains the best results, but the type of instances to be extracted are less complex.
2. The works in [15], [20], [7], [13] and [17] have better results than ours, but requires a complete syntactic analysis and a learning process. Besides, except in [13], in which instance of one level are generated, the other methods do not consider the problem of constructing complex instances.

3. Systems like [8], [13], [5] and [18], which can extract complex instances (the three first systems consider only one level of aggregation), require a complete syntactic analysis. In some cases, a semantic and/or learning processing is also needed. The third of these systems obtains precision and recall comparable to ours, the first has a precision nearly to ours but the recall is a 40% lower to the obtained by our system. In the case of [13] the results are visibly higher, but they use a complete syntactic and semantic analysis, even for reconstructing instances with a few aggregation levels. Finally, [18] does consider a very complex domain. They use a complete syntactic and semantic analysis, but results both in precision and recall are lower.
4. The analyzed methods do not consider OWL ontology formalism, and thus cannot infer aggregation paths that are no explicitly described.

Summarizing, our proposal provides good results in complex situations, but it is highly dependent on the quality of the entity recognition process, that is, the quality of the lexicon, and how well the text fragments can be associated to the correct ontological entities. In restricted knowledge domains (like archeology, bioinformatics, etc.) the use of the appropriate entity recognizers and a good, controlled vocabulary allow to discover complex instances with high values of precision and recall. On the other hand, in open domains a syntactic and semantic analysis of the texts might be required.

8 Conclusions

Information extraction systems are more and more essential for maintaining, using and interchanging information, especially in huge, unstructured and ever-growing environments such as the web. We presented here a novel system to extract complex ontological instances and relations. Experimental results, although not exhaustive yet, are very promising, and the system design considers some critical issues such as OWL-awareness and scalability, that make it a useful tool for populating and updating data in the Semantic Web. The approach is based on non monotonical processing, and makes use of an ontology and entity recognizers and disambiguators in order to combine adequately an initial set of instances. Exhaustive analysis and experimentation of the proposal is being performed in a variety of application scenarios.

References

1. L. Reeve and H. Han. Survey of semantic annotation platforms. In *SAC*, pages 1634–1638, 2005.
2. R. Danger. *Information Extraction and analysis from Semantic Web perspective (in Spanish).* PhD thesis, Universitat Jaime I, 2007.
3. A. Maedche, G. Neumann, and S. Staab. Bootstrapping an ontology-based information extraction system. In P. Szczepaniak, J. Segovia, J. Kacprzyk, and L. Zadeh, editors, *Intelligent Exploration of the Web.* Springer / Physica Verlag, Heidelberg, 2002.

4. B. Popov, A. Kiryakov, D. Ognyanoff, D. Manov, A. Kirilov, and M. Goranov. Towards Semantic Web information extraction. *Human Language Technologies Workshop at the 2nd International Semantic Web Conference (ISWC2003)*, 2003.
5. P. Buitelaar and S. Ramaka. Unsupervised ontology-based semantic tagging for knowledge markup. In Wray Buntine, Andreas Hotho, and Stephan Bloehdorn, editors, *Proc. of the Workshop on Learning in Web Search at the International Conference on Machine Learning*, 8 2005.
6. P. Kogut and W. Holmes. AeroDAML: applying information extraction to generate DAML annotations from web pages. In *Proc. of the Workshop on Knowledge Markup and Semantic Annotation at 1st International Conference on Knowledge Capture*, 2001.
7. D. Maynard, M. Yankova, N. Aswani, and H. Cunningham. Automatic creation and monitoring of semantic metadata in a dynamic knowledge portal. *Lecture Notes in Computer Science 3192*, pages 65 – 74, 2004.
8. H. Alani, S. Kim, D. E. Millard, M. J. Weal, W. Hall, P. H. Lewis and N. Shadbolt. Automatic ontology-based knowledge extraction from web documents. *IEEE Intelligent Systems*, 18(1):14–21, 2003.
9. P. Cimiano, S. Handschuh, and S. Staab. Towards the self-annotating web. In *Proc. of the 13th World Wide Web Conference*. 2004.
10. Stephen D., N. Eiron, D. Gibson, D. Gruhl, R. Guha, A. Jhingran, T. Kanungo, K. S. McCurley, S. Rajagopalan, A. Tomkins, J. A. Tomlin, and J. Y. Zien. A case for automated large-scale semantic annotation. *J. Web Sem.*, 1(1):115–132, 2003.
11. A. G. Valarakos, G. Paliouras, V. Karkaletsis, and G. A. Vouros. Enhancing ontological knowledge through ontology population and enrichment. In *Engineering Knowledge in the Age of the Semantic Web, EKAW*, pages 144–156, 2004.
12. J. Yang and J. Choi. Agents for intelligent information extraction by using domain knowledge and token-based morphological patterns. In *PRIMA*, pages 74–85, 2003.
13. F. Ciravegna, S. Chapman, A. Dingli, and Y. Wilks. Learning to harvest information for the Semantic Web. In *ESWS*, pages 312–326, 2004.
14. O. Etzioni, M. Cafarella, D. Downey, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates. Unsupervised named-entity extraction from the web: an experimental study. *Artif. Intell.*, 165(1):91–134, 2005.
15. M. Vargas-Vera, E. Motta, J. Domingue, M. Lanzoni, A. Stutt and F. Ciravegna. MnM: Ontology driven semi-automatic and automatic support for semantic markup. In *EKAW*, pages 379–391, 2002.
16. S. Handschuh, S. Staab, and R. Studer. Leveraging metadata creation for the Semantic Web with CREAM. In *Proc. of the Annual German Conference on AI*, volume 2821, pages 19–33. Springer, 2003.
17. D. Celjuska and M. Vargas-Vera. Semi-automatic population of ontologies from text. In Jan Paralic and Andreas Rauber, editors, *Workshop on Data Analysis WDA-2004*. 2004.
18. F. Amardeilh, P. Laublet and J.-L. Minel. Document annotation and ontology population from linguistic extractions. In *K-CAP '05: Proc. of the 3rd international conference on Knowledge capture*, pages 161–168. ACM Press, 2005.
19. A. Maedche and R. Volz. The ontology extraction maintenance framework text-to-onto, 2001.
20. M. Surdeanu, S. Harabagiu, J. Williams, and P. Aarseth. Using predicate-argument structures for information extraction. In *Proc. of the ACL, Sapporo, Japan*, 2003.