A proposal for annotation, semantic similarity and classification of textual documents

Emmanuel Nauer, Amedeo Napoli

A proposal for annotation, semantic similarity and classification of textual documents

Emmanuel Nauer and Amedeo Napoli

LORIA — UMR 7503
Bâtiment B, B.P. 239
F-54506 Vandœuvre-lès-Nancy cedex, France

{nauer,napoli}@loria.fr

Abstract. In this paper, we present an approach for classifying documents based on the notion of a semantic similarity and the effective representation of the content of the documents. The content of a document is annotated and the resulting annotation is represented by a labeled tree whose nodes and edges are represented by concepts lying within a domain ontology. A reasoning process may be carried out on annotation trees, allowing the comparison of documents between each others, for classification or information retrieval purposes. An algorithm for classifying documents with respect to semantic similarity and a discussion conclude the paper.
Keywords: content-based classification of documents, domain ontology, document annotation, semantic similarity.

1 Introduction and motivation

In this paper, we propose an approach for defining a semantic annotation of Web textual documents based on the content of the documents. The core of the approach relies on the notions of annotation tree and semantic similarity, allowing to manipulate documents with respect to their content, for, e.g. reasoning and information retrieval. An annotation is represented as a labeled tree according to a domain ontology, and is named annotation tree. Annotation trees can be compared and classified, and are the basis for evaluating a semantic similarity between documents.

Most of the information retrieval systems are based on keyword search, with a more or less sophisticated use of keywords, including for example normalized or weighted keywords, weighted or thesaurus-based relations between keywords for document matching and retrieval [16]. These information retrieval approaches are based on a rather rough and direct use of the set of –unrelated– keywords associated to a document, whereas it could be useful to take into account the semantic relations existing between keywords. These approaches are efficient for simple and standard tasks, but show their limits within more complex applications, e.g. applications for semantic Web, where an actual, explicit, and semantic access to the content of documents is needed. In addition, research

work on information retrieval and semantic Web is mainly based on the use of domain knowledge and reasoning, for improving document representation and query answering. Research work on semantic annotation of documents aims at making inferences using a knowledge-based annotation of documents, turning a human-understandable content into a machine understandable content [10, 9, 18].

In this research work, we present a content-based classification of textual Web documents based on a semantic annotation of the content of documents. A semantic annotation is represented as a labeled tree, where a node is typed by a concept of the domain ontology, while an edge is typed by a relation between two concepts. The ontology holds on the domain of the documents, and includes concepts and relations organized by a subsumption relation. This kind of labeled-tree representation can be likened to graphs of RDF statements or to the XML object model of documents (DOM), where the semantics associated to the elements of an annotation relies on a domain ontology.

This content-based annotation of documents supports a classification process using semantic similarity between document annotations. The classification process allows to organize documents in categories with respect to their contents and domain knowledge. Semantically similar documents are classified within a class describing their common characteristics according to the domain ontology, where an individual document is reified as an instance of a class. In this way, documents holding on some topics of interest may be classified according to the classes representing these topics in the domain ontology. For example, it can be useful for a researcher looking for specific scientific publications to navigate within a hierarchy of classes, where a class represents a set of documents holding on some selected topics, instead of consulting a flat and unordered list of documents. In this context, a query is represented as a class whose instances (representing individual documents) are considered as answers to the query. For answering a query, the class representing the query is built and then classified in the hierarchy of document categories. Once the query is classified, the instances of the classes subsumed by the query are returned as potential answers to the query.

The paper is organized as follows. In section 2, the notion of semantic annotation of documents is introduced. Then, in section 3, semantic similarity and document classification are detailed and illustrated. A discussion on the annotation and classification processes ends the paper.

2 The annotation of textual Web documents

2.1 Introducing the domain ontology

Given a reference domain $\mathcal{D}$, an ontology $\mathcal{O}_\mathcal{D}$ is considered as a hierarchy of classes and relation classes: classes represent the concepts of the domain $\mathcal{D}$ while relation classes represent relations between concepts [13, 17]. The ontology $\mathcal{O}_\mathcal{D}$ is used for studying a set of documents DOCS related to the domain $\mathcal{D}$. A class has

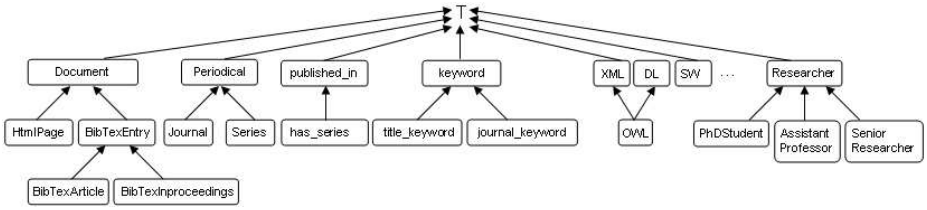

Fig. 1. A fragment of an ontology of classes and relation classes holding on computer science documents.

a name and is either primitive or defined by a set of attributes. As in description logics, the attributes of a defined class act as necessary and sufficient conditions for declaring an individual as an instance of a defined class [3]. In the same way, a relation class has a name, a domain, a range, and additional properties such as reflexivity, symmetry, transitivity... Classes are organized within a hierarchy, namely $\mathcal{O}_\mathcal{D}$, by a subsumption relation: whenever the class $\mathtt{C}_1$ is subsumed by the class $\mathtt{C}_2$, written $\mathtt{C}_1 \sqsubseteq \mathtt{C}_2$, then the individuals that are instances of $\mathtt{C}_1$ are also instances of $\mathtt{C}_2$. For relation classes, the relation $\mathtt{R}_1$ is subsumed by the relation $\mathtt{R}_2$, written $\mathtt{R}_1 \sqsubseteq \mathtt{R}_2$, when, given two individuals $\mathtt{a}$ and $\mathtt{b}$, $\mathtt{R}_1(\mathtt{a},\mathtt{b})$ implies $\mathtt{R}_2(\mathtt{a},\mathtt{b})$. All classes and relation classes are subsumed by $\top$ (`Top`), the root of the hierarchy $\mathcal{O}_\mathcal{D}$. A fragment of an ontology is given in figure 1, where the relation class `keyword` subsumes the relation class `title_keyword`, and the class `Document` subsumes the classes `HtmlPage` and `BibTexEntry`.

2.2 The annotation of documents

Definition 1. *An* annotation *associated to a document* $\mathtt{D}$*, denoted by* $\mathtt{A}(\mathtt{D})$*, is defined as a labeled rooted tree* $\mathtt{A}(\mathtt{D}) = (\mathtt{N},\mathtt{E})$ *where* $\mathtt{N}$ *is a set of nodes having a label and a type, and* $\mathtt{E}$ *is a set of edges having the form* $\mathtt{e} = (\mathtt{n},\mathtt{a},\mathtt{n}')$ *where* $\mathtt{n},\mathtt{n}' \in \mathtt{N}$ *and* $\mathtt{a}$ *is the label of the edge. The labeled tree* $\mathtt{A}(\mathtt{D}) = (\mathtt{N},\mathtt{E})$ *associated* $\mathtt{D}$ *is called the* annotation tree *of the document* $\mathtt{D}$*.*

In such a tree-based annotation, a node and an edge have a label and a type. The type makes reference either to a class of $\mathcal{O}_\mathcal{D}$ or to the special datatype `String` (not explicitly represented in $\mathcal{O}_\mathcal{D}$, and unique datatype considered in the present framework). For notational convenience, the type of an element $\mathtt{x}$ is supposed to be returned by the function $\mathtt{type}(\mathtt{x})$. In addition, the label of a node or an edge is derived from the associated type. Then, for an edge $\mathtt{e} = (\mathtt{n},\mathtt{a},\mathtt{n}')$, we will indifferently write $\mathtt{type}(\mathtt{e})$ or $\mathtt{type}(\mathtt{a})$ for denoting the type of the edge $\mathtt{e}$. A URI, i.e. a *Uniform Resource Identifier*, may be associated to a node for pointing to a specific resource (as in RDF statements). In case the node is a leaf in the annotation tree (i.e. the node has no descendant), a "value" whose type is `String` may be attached to that node. By analogy with types, the value or the URI attached to a node are supposed to be returned by the function $\mathtt{value}(\mathtt{x})$. It is assumed that a leaf can have a value or a URI, but not both (exclusive or).

At the moment, an annotation is represented as a tree and not as a graph (as it could be the case with RDF statements). One main reason is for keeping things more simple, i.e. a simple structure of an annotation allowing efficient comparison procedures and a simple conceptual representation (using description logics).

The figure 2 gives an example of two documents $\mathtt{D}_1$ and $\mathtt{D}_2$ with their associated annotation trees. The annotation tree associated to $\mathtt{D}_1$ describes an *HTML webpage* about a publication in the *ERCIM News journal*, where the authors, the title, the keywords, the journal keywords... are extracted from the HTML page. In the same way, the tree associated to $\mathtt{D}_2$ describes a *BibTex entry* using the main characteristics of this entry.

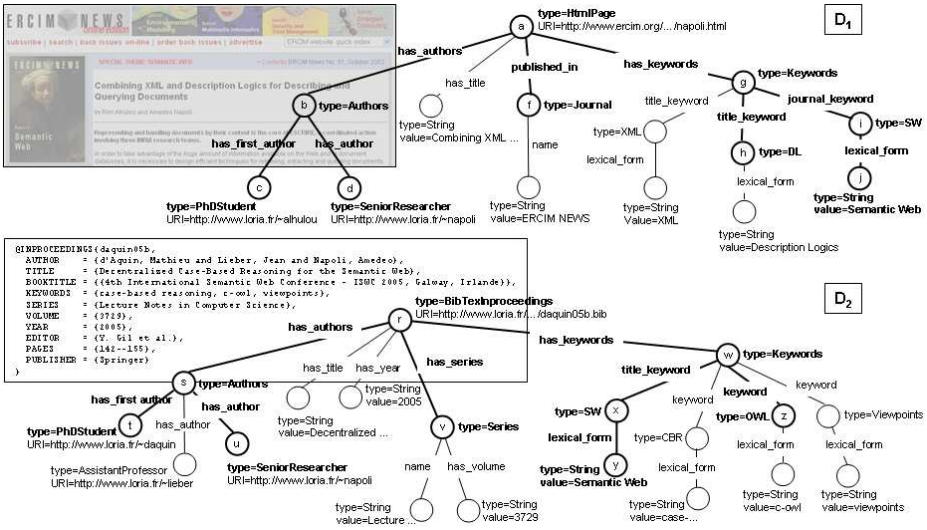

Fig. 2. Example of two documents with their annotation trees.

It can be useful to distinguish between *generic* and *specific* annotations. A generic annotation is associated to a set of different documents while a specific annotation is associated to only one particular document. Firstly, a node $\mathtt{n}$ is *specific* or *instantiated* whenever it has an associated URI or a value (whose type is `String`) ; otherwise the node is *generic*. Accordingly, an annotation $\mathtt{A}$ is *specific* whenever the root of $\mathtt{A}$, denoted by $\mathtt{root(A)}$, and the leaves of $\mathtt{A}$ are specific nodes ; otherwise, $\mathtt{A}$ is *generic*. For example, the annotation trees associated to the documents $\mathtt{D}_1$ and $\mathtt{D}_2$ in figure 2 are specific annotations. By contrast, the common subtree of $\mathtt{A}(\mathtt{D}_1)$ and $\mathtt{A}(\mathtt{D}_2)$ given in figure 3 is a generic annotation to which more than one document can be attached, here for example $\mathtt{D}_1$ and $\mathtt{D}_2$.

Let $\mathtt{D}_1$ and $\mathtt{D}_2$ be two documents, and $\mathtt{A}(\mathtt{D}_1) = (\mathtt{N}_1, \mathtt{E}_1)$ and $\mathtt{A}(\mathtt{D}_2) = (\mathtt{N}_2, \mathtt{E}_2)$ their respective specific annotation trees. When there is no ambiguity, $\mathtt{A}(\mathtt{D}_1) =$

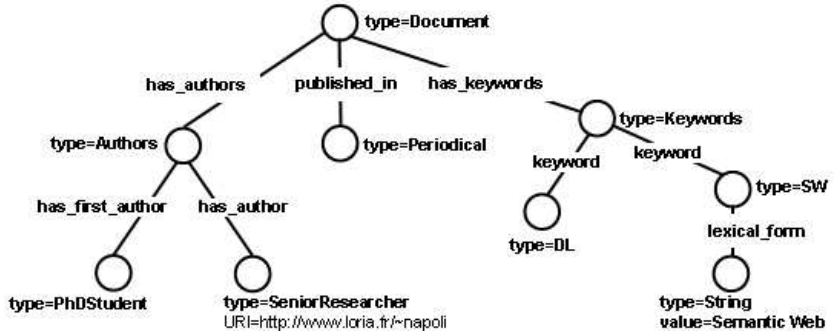

Fig. 3. The subtree generalizing the two annotation trees given in in figure 2.

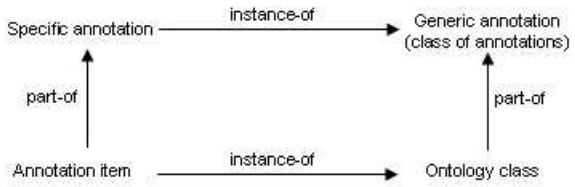

Fig. 4. The links existing between specific annotations, generic annotations and the domain ontology.

$(\mathtt{N}_1, \mathtt{E}_1)$ and $\mathtt{A}(\mathtt{D}_2) = (\mathtt{N}_2, \mathtt{E}_2)$ are written for short respectively $\mathtt{A}_1$ and $\mathtt{A}_2$. As introduced just above, a *subsumption* (a partial order relation) and an *instantiation* relations are defined on specific and generic annotations (these subsumption and instantiation relations are inspired from *subsumption* of molecular structures detailed in [15]).

Definition 2. *An annotation* $\mathtt{A}_1 = (\mathtt{N}_1, \mathtt{B}_1)$ *is* subsumed *by an annotation* $\mathtt{A}_2 = (\mathtt{N}_2, \mathtt{B}_2)$*, denoted by* $\mathtt{A}_1 \sqsubseteq \mathtt{A}_2$*, if there exists a subtree* $\mathtt{A}_1{}' = (\mathtt{N}_1{}', \mathtt{B}_1{}')$ *of* $\mathtt{A}_1$*, and an isomorphism* μ *between* $\mathtt{A}_1{}'$ *and* $\mathtt{A}_2$ *preserving types, i.e. for each edge* $\mathtt{e}_2 = (\mathtt{n}_2, \mathtt{a}_2, \mathtt{n}_2')$ *in* $\mathtt{A}_2$*, there exists an edge* $\mathtt{e}_1 = (\mathtt{n}_1, \mathtt{a}_1, \mathtt{n}_1')$ *in* $\mathtt{A}_1{}'$*, where* $\mathtt{n}_2 = \mu(\mathtt{n}_1)$ *and* $\mathtt{type}(\mathtt{n}_1) \sqsubseteq \mathtt{type}(\mathtt{n}_2)$*,* $\mathtt{a}_2 = \mu(\mathtt{a}_1)$ *and* $\mathtt{type}(\mathtt{a}_1) \sqsubseteq \mathtt{type}(\mathtt{a}_2)$*,* $\mathtt{n}_2{}' = \mu(\mathtt{n}_1{}')$ *and* $\mathtt{type}(\mathtt{n}_1{}') \sqsubseteq \mathtt{type}(\mathtt{n}_2{}')$*.*

A specific annotation $\mathtt{A}_1 = (\mathtt{N}_1, \mathtt{E}_1)$ *is an* instance *of a generic annotation* $\mathtt{A}_2 = (\mathtt{N}_2, \mathtt{E}_2)$ *if and only if* $\mathtt{A}_1 \sqsubseteq \mathtt{A}_2$*.*

Two remarks have to be made: (i) the most general annotation is supposed to be identified with $\top$, the root of the hierarchy of concepts and relations, (ii) subsumption on annotations is supposed to hold only between connected annotation trees.

Figure 4 illustrates how generic and specific annotations are related to each other. A specific annotation is associated to a particular document and is com-

posed of several elements of annotation (annotation items) that are typed by classes of the ontology $\mathcal{O}_\mathcal{D}$. Moreover, a specific annotation is an instance of a generic annotation.

The different elements composing semantic annotations may be represented using semantic Web languages such as RDF, RDF-SCHEMA or OWL [8, 2]. The annotation trees can be considered either as RDF graphs or XML document models. In this way, the present work can be appliqued to the comparison of XML documents (considered as trees). Moreover, based on annotation trees, it becomes possible to compare two documents on the basis of their content, by computing the "semantic similarity" of the two documents. For example, the documents $\mathtt{D}_1$ and $\mathtt{D}_2$ in figure 2 are "semantically similar" because they both describe a document written by an author named *Amedeo Napoli*, in collaboration with another author that is a *PhDStudent*, published in a *periodical*, and concerned with *Description Logics* and *Semantic Web*. Such a comparison is based on a subsumption test of the annotation trees of the considered documents. Moreover, the generalized common subtree of the annotation trees must be *sufficiently sized*, i.e. the size of the subtree must be greater than or equal to a given threshold. These constructions, based on annotation trees and subsumption of annotation trees, are made explicit in the next section.

3 Semantic similarity and document classification

The semantic similarity between two documents is based on the semantic similarity between their annotation trees. The similarity tree shared by two annotation trees is also defined, for being used in a classification process aimed at categorizing Web documents.

3.1 Semantic similarity and similarity tree

First of all, the notion of least common subsumer (LCS) within a class hierarchy is introduced, than can be likened to the LCS operation in description logics [4–6] or to the projection in the conceptual graph theory [7, 14].

Definition 3. *Given the ontology $\mathcal{O}_\mathcal{D}$ and two classes $\mathtt{C}_1$ and $\mathtt{C}_2$, the class $\mathtt{C}$ is the* least common subsumer (class) *of the classes $\mathtt{C}_1$ and $\mathtt{C}_2$, denoted by $\mathtt{C} = \mathtt{lcs}(\mathtt{C}_1, \mathtt{C}_2)$, if $\mathtt{C} \neq \top$, $\mathtt{C}_1 \sqsubseteq \mathtt{C}$ and $\mathtt{C}_2 \sqsubseteq \mathtt{C}$, and whenever $\mathtt{C}_1 \sqsubseteq \mathtt{C}'$ and $\mathtt{C}_2 \sqsubseteq \mathtt{C}'$ then $\mathtt{C} \sqsubseteq \mathtt{C}'$, i.e. $\mathtt{C}$ is minimal among the subsumers of the classes $\mathtt{C}_1$ and $\mathtt{C}_2$.*

Based on this definition, given two annotation trees $\mathtt{A}_1$ and $\mathtt{A}_2$, two nodes $\mathtt{n}_1 \in \mathtt{N}_1$ and $\mathtt{n}_2 \in \mathtt{N}_2$ are said to be *semantically similar* whenever one of the following conditions is verified:

- $\mathtt{type}(\mathtt{n}_1) = \mathtt{type}(\mathtt{n}_2) = \mathtt{String}$ and $\mathtt{value}(\mathtt{n}_1) = \mathtt{value}(\mathtt{n}_2)$.
- $\mathtt{type}(\mathtt{n}_1) \neq \mathtt{String}$, $\mathtt{type}(\mathtt{n}_2) \neq \mathtt{String}$, and $\mathtt{C} = \mathtt{lcs}(\mathtt{type}(\mathtt{n}_1), \mathtt{type}(\mathtt{n}_2))$ exists in $\mathcal{O}_\mathcal{D}$.

By convenience, the class $\mathtt{C}$ is said to be the least common subsumer of the two nodes $\mathtt{n}_1$ and $\mathtt{n}_2$, denoted by $\mathtt{C} = \mathtt{lcs}(\mathtt{n}_1, \mathtt{n}_2)$. In the same way, two edges $\mathtt{e}_1 = (\mathtt{n}_1, \mathtt{a}_1, \mathtt{n}'_1)$ and $\mathtt{e}_2 = (\mathtt{n}_2, \mathtt{a}_2, \mathtt{n}'_2)$ are *semantically similar* whenever the following is verified: $\mathtt{n}_1$ is semantically similar to $\mathtt{n}_2$, $\mathtt{n}'_1$ is semantically similar to $\mathtt{n}'_2$, and the class $\mathtt{A} = \mathtt{lcs}(\mathtt{type}(\mathtt{a}_1), \mathtt{type}(\mathtt{a}_2))$ exists in $\mathcal{O}_\mathcal{D}$. By analogy with nodes, the edge $\mathtt{e} = (\mathtt{n}, \mathtt{a}, \mathtt{n}')$ where $\mathtt{n} = \mathtt{lcs}(\mathtt{n}_1, \mathtt{n}_2)$, $\mathtt{n}' = \mathtt{lcs}(\mathtt{n}'_1, \mathtt{n}'_2)$, and $\mathtt{a} = \mathtt{lcs}(\mathtt{a}_1, \mathtt{a}_2)$, is said to be the least common subsumer of the edges $\mathtt{e}_1 = (\mathtt{n}_1, \mathtt{a}_1, \mathtt{n}'_1)$ and $\mathtt{e}_2 = (\mathtt{n}_2, \mathtt{a}_2, \mathtt{n}'_2)$, denoted by $\mathtt{e} = \mathtt{lcs}(\mathtt{e}_1, \mathtt{e}_2)$.

For example, considering the two annotation trees in figure 2, the following elements of semantic similarity may be extracted:

- The root of $\mathtt{A}_1$, labeled by $\mathtt{type} = \mathtt{HtmlPage}$, is similar to the root of $\mathtt{A}_2$, labeled by $\mathtt{type} = \mathtt{BibTexInproceedings}$, with $\mathtt{Document} = \mathtt{lcs}(\mathtt{HtmlPage}, \mathtt{BibTexInproceedings})$.
- The edges labeled in $\mathtt{A}_1$ by $\mathtt{has\_author}$, $\mathtt{author}$, and $\mathtt{has\_keywords}$, are identical to the corresponding edges in $\mathtt{A}_2$.
- The edges issued from the roots $\mathtt{root}(\mathtt{A}_1)$ and $\mathtt{root}(\mathtt{A}_2)$, labeled by $\mathtt{has\_series}$ and $\mathtt{published\_in}$, are similar to $\mathtt{published\_in} = \mathtt{lcs}(\mathtt{published\_in}, \mathtt{has\_series})$.
- The two leaves labeled by $\mathtt{type} = \mathtt{PhDStudent}$ are similar, even if their values are not equal.

The notion of semantic similarity can be generalized to annotation trees and thus to documents in the following way. Let $\mathtt{D}_1$ and $\mathtt{D}_2$ be two documents, and $\mathtt{A}_1$ and $\mathtt{A}_2$ their respective specific annotation trees. As $\mathtt{A}_1$ and $\mathtt{A}_2$ are specific annotation trees, they have an associated generic annotation tree, namely the generic annotation they are an instance of. Then, the annotation trees $\mathtt{A}_1$ and $\mathtt{A}_2$ are similar if there exists an annotation tree that is the least common subsumer of $\mathtt{A}_1$ and $\mathtt{A}_2$. For example, the similarity tree of the two annotation trees shown in figure 2 is given in figure 3. More precisely, the semantic similarity for annotation trees is defined as follows.

Definition 4. *The two annotation trees* $\mathtt{A}_1 = (\mathtt{N}_1, \mathtt{E}_1)$ *associated to the document* $\mathtt{D}_1$*, and* $\mathtt{A}_2 = (\mathtt{N}_2, \mathtt{E}_2)$ *associated to the document* $\mathtt{D}_2$*, are said to be* semantically similar *with* degree α*, if there exists an annotation tree denoted by* $\mathtt{A}_\mathtt{S}(\mathtt{A}_1, \mathtt{A}_2) = (\mathtt{N}, \mathtt{E})$ *that is the least common subsumer of* $\mathtt{A}_1$ *and* $\mathtt{A}_2$*. Moreover, the* degree α *is given by:*

$$\alpha((\mathtt{A}_1, \mathtt{A}_2)|\mathtt{A}_\mathtt{S}(\mathtt{A}_1, \mathtt{A}_2)) = \frac{|\mathtt{A}_\mathtt{S}(\mathtt{A}_1, \mathtt{A}_2)|^2}{|\mathtt{A}_1| \cdot |\mathtt{A}_2|}$$

where $|\mathtt{A}_\mathtt{i}|$ *denotes the cardinal of the node set of the tree* $\mathtt{A}_\mathtt{i}$*, and* $|\mathtt{A}_\mathtt{S}(\mathtt{A}_1, \mathtt{A}_2)|$ *the cardinal of the node set of the tree* $\mathtt{A}_\mathtt{S}(\mathtt{A}_1, \mathtt{A}_2)$*.*

A constraint can be set up on the similarity degree, such that the degree of the similarity tree $\mathtt{A}_\mathtt{S}(\mathtt{A}_1, \mathtt{A}_2)$ of $\mathtt{A}_1$ and $\mathtt{A}_2$ must be greater or equal to a fixed similarity threshold $\sigma_{\mathtt{sim}}$. For example, the two annotation trees $\mathtt{A}_1$ and $\mathtt{A}_2$ on figure 2 are semantically similar with the similarity tree shown on figure 3, and

a degree equal to $\alpha((\mathtt{A}_1, \mathtt{A}_2)|\mathtt{A}_\mathtt{S}(\mathtt{A}_1, \mathtt{A}_2)) = 9^2/(14.19) \approx 0.305$. This similarity is acceptable for a threshold $\sigma_{\mathtt{sim}}$ of 0.25 for example.

3.2 An algorithm for constructing a similarity tree

The semantic similarity between documents depends on their annotation trees, thus on the semantic similarity between nodes and edges of the annotation trees, on a similarity degree, on a similarity threshold $\sigma_{\mathtt{sim}}$, but also on isomorphism between trees (linked to subsumption between annotations). Indeed, the building of $\mathtt{A}_\mathtt{S}(\mathtt{A}_1, \mathtt{A}_2)$ is based on two *generalizations*, say γ_1 and γ_2, such that $\mathtt{A}_\mathtt{S}(\mathtt{A}_1, \mathtt{A}_2) = \gamma_1(\mathtt{A}_1) = \gamma_2(\mathtt{A}_2)$ (as in a unification process). Moreover, it has already been remarked that the building of the similarity tree can be likened to the building of the least common subsumers in description logics [4–6], or to the projection of conceptual graphs [7, 14].

The building of a similarity tree, given two annotation trees $\mathtt{A}_1$ and $\mathtt{A}_2$, may be achieved according to the following rules, the generalizations γ_1 and γ_2 being based on the definitions of the semantic similarity between nodes and edges.

- When two nodes $\mathtt{n}_1 \in \mathtt{N}_1$ and $\mathtt{n}_2 \in \mathtt{N}_2$ are semantically similar, then the node $\mathtt{n}$ in $\mathtt{A}_\mathtt{S}(\mathtt{A}_1, \mathtt{A}_2)$ results from the generalization of $\mathtt{n}_1$ and $\mathtt{n}_2$, with $\mathtt{n} = \gamma_1(\mathtt{n}_1) = \gamma_2(\mathtt{n}_2)$.
  Practically, a function $\mathtt{CreateNode}(\mathtt{n}_1, \mathtt{n}_2)$ takes the nodes $\mathtt{n}_1 \in \mathtt{N}_1$ and $\mathtt{n}_2 \in \mathtt{N}_2$ as inputs, and searches for the class $\mathtt{n} = \mathtt{lcs}(\mathtt{n}_1, \mathtt{n}_2)$ in $\mathcal{O}_\mathcal{D}$, and then builds the node labeled by $\mathtt{n}$. In the case $\mathtt{n}_1 = \mathtt{n}_2$, with $\mathtt{n}_1$ and $\mathtt{n}_2$ having the same value or URI, this value or URI is attached to the node labeled by $\mathtt{n}$.
- When a node $\mathtt{n}_1 \in \mathtt{N}_1$ does not have any similar node in $\mathtt{N}_2$, then the substitution cannot be computed and the node $\mathtt{n}_1$ is not taken into account in the building of the similarity tree. For example, this is the case of the node $\mathtt{type} = \mathtt{AssistantProfessor}, \mathtt{URI} = \mathtt{http://.../lieber}$ in figure 2, for $\mathtt{A}_2$.
- When two edges $\mathtt{e}_1 = (\mathtt{n}_1, \mathtt{a}_1, \mathtt{n}_1{}') \in \mathtt{E}_1$ and $\mathtt{e}_2 = (\mathtt{n}_2, \mathtt{a}_2, \mathtt{n}_2{}') \in \mathtt{E}_2$ are semantically similar, then the edge $\mathtt{e} = (\mathtt{n}, \mathtt{a}, \mathtt{n}')$ in $\mathtt{A}_\mathtt{S}(\mathtt{A}_1, \mathtt{A}_2)$ is obtained by generalizing $\mathtt{e}_1$ and $\mathtt{e}_2$, with $\mathtt{n} = \gamma_1(\mathtt{n}_1) = \gamma_2(\mathtt{n}_2)$, $\mathtt{a} = \gamma_1(\mathtt{a}_1) = \gamma_2(\mathtt{a}_2)$, and $\mathtt{n}' = \gamma_1(\mathtt{n}_1{}') = \gamma_2(\mathtt{n}_2{}')$.
  Practically, a function $\mathtt{CreateEdge}(\mathtt{e}_1, \mathtt{e}_2)$ takes the two edges $\mathtt{e}_1$ and $\mathtt{e}_2$ as inputs, and searches for the edge $\mathtt{e} = \mathtt{lcs}(\mathtt{e}_1, \mathtt{e}_2) = (\mathtt{n}, \mathtt{a}, \mathtt{n}')$ in $\mathcal{O}_\mathcal{D}$.
- When an edge $\mathtt{e}_1 \in \mathtt{E}_1$ does not have any similar edge in $\mathtt{E}_2$, then the substitution is not defined and the edge $\mathtt{e}_1$ is not taken into account in the building of the similarity tree.

The similarity tree is built by searching, starting from the roots $\mathtt{root}(\mathtt{A}_1)$ and $\mathtt{root}(\mathtt{A}_2)$, the largest subtree that can be matched according to the previous rules. The algorithm produces as output the correspondence between the nodes of $\mathtt{A}_1$ and the nodes of $\mathtt{A}_2$:

1. The set of node correspondence between $\mathtt{A}_1$ and $\mathtt{A}_2$ for semantically similar nodes, denoted by $\Pi_{\mathtt{node}}(\mathtt{A}_1, \mathtt{A}_2)$. For example, the correspondence for the annotation trees given in figure 2 is:
   $\Pi_{\mathtt{node}}(\mathtt{A}_1, \mathtt{A}_2) = \{(\mathtt{a}, \mathtt{r}), (\mathtt{b}, \mathtt{s}), (\mathtt{c}, \mathtt{t}), (\mathtt{d}, \mathtt{u}), (\mathtt{f}, \mathtt{v}), (\mathtt{g}, \mathtt{w}), (\mathtt{h}, \mathtt{z}), (\mathtt{i}, \mathtt{x}), (\mathtt{j}, \mathtt{y})\}$

2. The set of edge correspondence between $\mathtt{A}_1$ and $\mathtt{A}_2$ for semantically similar edges, is denoted by $\Pi_{\mathtt{edge}}(\mathtt{A}_1, \mathtt{A}_2)$. For example, $\Pi_{\mathtt{edge}}(\mathtt{A}_1, \mathtt{A}_2)$ contains the pairs of edges associated with the correspondence $\Pi_{\mathtt{node}}(\mathtt{A}_1, \mathtt{A}_2)$.
3. The set of nodes of $\mathtt{A}_1$ and the set of nodes of $\mathtt{A}_2$ that have not to be taken into account. For example, all nodes of $\mathtt{A}_1$ and $\mathtt{A}_2$ that are not in $\Pi_{\mathtt{node}}(\mathtt{A}_1, \mathtt{A}_2)$ are not taken into account for the construction of the similarity tree $\mathtt{A}_\mathtt{S}(\mathtt{A}_1, \mathtt{A}_2)$.

Two remarks have to be done. Firstly, the generalizations γ_1 and γ_2 are different only when the nodes $\mathtt{n}_1$ or $\mathtt{n}_2$, or the edges $\mathtt{e}_1$ or $\mathtt{e}_2$, do not have any similar node or edge. Secondly, the `lcs` operation here is much more simpler than the general building of a LCS in descriptions logics (see [5]): the LCS is actually not built but *searched for* within the $\mathcal{O}_\mathcal{D}$ concept hierarchy, and thus corresponds to an already existing concept.

3.3 An algorithm for the classification of annotation trees

Let $\mathtt{D}_1$ and $\mathtt{D}_2$ be two semantically similar documents with the similarity tree $\mathtt{A}_\mathtt{S}(\mathtt{A}_1, \mathtt{A}_2) = (\mathtt{N}, \mathtt{E})$, built from the annotation trees $\mathtt{A}_1 = \mathtt{A}(\mathtt{D}_1) = (\mathtt{N}_1, \mathtt{E}_1)$ and $\mathtt{A}_2 = \mathtt{A}(\mathtt{D}_2) = (\mathtt{N}_2, \mathtt{E}_2)$. A classification algorithm may be proposed, for retrieving the *best instantiation class* with respect to the similarity degree for the annotation tree associated to a given document.

Let DOCS be a set of documents, $\mathcal{O}_\mathcal{D}$ the ontology associated to DOCS, and $\mathcal{H}_\mathcal{A}$ a hierarchy of annotation trees associated to DOCS (see figure 5). The classes in $\mathcal{H}_\mathcal{A}$ represent generic annotations related to sets of documents in DOCS, with respect to the ontology $\mathcal{O}_\mathcal{D}$. Actually, the $\mathcal{H}_\mathcal{A}$ hierarchy may also be considered either as an ontology of generic annotations related to the documents in DOCS, or to a classification of the documents in DOCS.

Given the ontology $\mathcal{O}_\mathcal{D}$ and an annotation hierarchy $\mathcal{H}_\mathcal{A}$, the principle of the classification algorithm is the following. The classes subsuming $\mathtt{A}_1 = \mathtt{A}(\mathtt{D}_1) = (\mathtt{N}_1, \mathtt{E}_1)$ are searched in $\mathcal{H}_\mathcal{A}$ using a depth-first search. More precisely, there is a search for the classes $\mathtt{C}$ in $\mathcal{H}_\mathcal{A}$ subsuming the generic annotation associated to $\mathtt{A}_1$, i.e. $\mathtt{A}_1$ is an instance of $\mathtt{C}$ (as stated in definition 2). Zero or more subsuming classes may exist in $\mathcal{H}_\mathcal{A}$:

- When no subsuming class exists, then the type of the document $\mathtt{D}_1$ is not represented in $\mathcal{H}_\mathcal{A}$. A new class representing $\mathtt{D}_1$ can be created, as in an incremental classification algorithm [11]. Maybe, this means that the themes of the document $\mathtt{D}_1$ are out of the scope of those underlying the ontology $\mathcal{O}_\mathcal{D}$ and the documents in DOCS.
- If one or more classes subsume $\mathtt{A}_1$, then similar documents to $\mathtt{D}_1$ are found, namely the documents that are instances of classes subsuming $\mathtt{A}_1$ or the subsumers of $\mathtt{A}_1$. The subsumers can be sorted according to their similarity degree, i.e. more a class is close to $\mathtt{D}_1$ better it is ranked.

For example, the annotation tree $\mathtt{A}_2$ associated to the document $\mathtt{D}_2$ is classified in the hierarchy $\mathcal{H}_\mathcal{A}$ as follows (cf. figure 5). According to the ontology

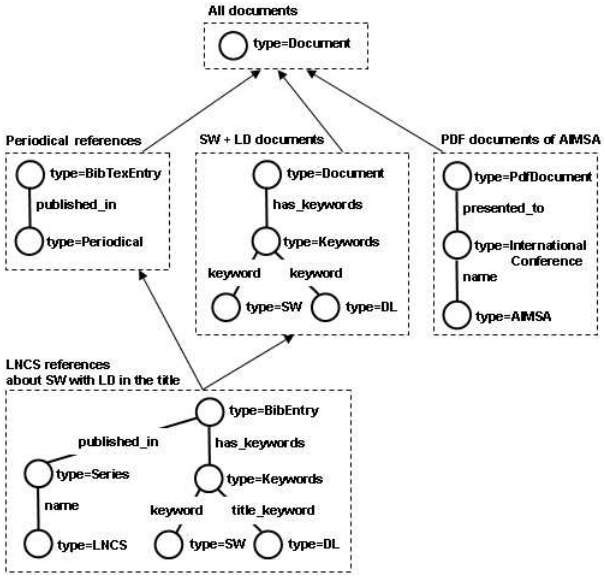

Fig. 5. A hierarchy $\mathcal{H}_{\mathcal{A}}$ of annotations

$\mathcal{O}_{\mathcal{D}}$ and to the hierarchy $\mathcal{H}_{\mathcal{A}}$, the class ‘‘`Periodical references`’’ may be selected (assimilating the series ‘‘`LNCS`’’ to a periodical publication), and then the class ‘‘`LNCS references about SW with DL in the title`’’ is also selected, as the most specific subsumer: both classes subsume the annotation tree $\mathtt{A}_2$. Then, the class ‘‘`SW + DL documents`’’ is selected, but the class ‘‘`PDF documents of AIMSA`’’ is discarded. Documents similar to $\mathtt{D}_2$ have been found, namely the instances of the classes ‘‘`Periodical references`’’ and ‘‘`LNCS references about SW with DL in the title`’’.

Such an algorithm can be used for classifying documents according to their content, for comparing documents and for finding similar documents. Moreover, the classification algorithm can be used for extending the $\mathcal{H}_{\mathcal{A}}$ hierarchy, with respect to the $\mathcal{O}_{\mathcal{D}}$ ontology, by proposing new annotations classes. Actually, these aspects of the present work are research perspectives and a straightforward continuation of this work.

In addition, a *semantic similarity measure* between the two documents $\mathtt{D}_1$ and $\mathtt{D}_2$ may be defined, taking into account the proportion of semantically similar nodes in $\mathtt{A}_1$ and $\mathtt{A}_2$ with respect to $\mathtt{A}_S(\mathtt{A}_1, \mathtt{A}_2)$, and the similarities between the types, URI and values of the leaves in $\mathtt{A}_1$ and $\mathtt{A}_2$. A first formalization is proposed in [1], and the definition of a more acceptable and efficient form is currently under investigation.

4 Discussion and conclusion

The present approach proposed for comparing documents represented by annotation trees shows a number of advantages. On the one hand, relations existing between terms describing the content of documents can be taken into account and used for having a better understanding of the documents, i.e. for knowing whether a document $\mathtt{D}_1$ is more general than a document $\mathtt{D}_2$, with respect to the ontology $\mathcal{O}_\mathcal{D}$ and to the content of documents. On the other hand, annotation trees can be manipulated, matched, and compared, using standard tools adapted to XML document manipulation. Moreover, the present approach takes explicitly advantage of the domain ontology $\mathcal{O}_\mathcal{D}$, allowing sophisticated reasoning, e.g. for finding analog documents, where two documents $\mathtt{D}_1$ and $\mathtt{D}_2$ are considered as analogs when there exists a "similarity path", i.e. a sequence of similar documents, between their annotation trees (see for example [12]).

This approach has been implemented, and an experiment has been carried out on bibliographic documents, for comparing the behavior of the present approach and more classical information retrieval approaches, based on vectors and a thesaurus (organized by a generic/specific relation). The annotations are composed of relations such as `written-by` (an author), `published-in` (a year), `edited-in` (publication-support), `talking-about` (a keyword), etc. Some documents are found to be similar with the vector model, while they are found to be not similar according to the annotation tree approach. For example, two references containing the same type of authors, or published in a same type of publication support, are found to be similar in the annotation tree approach, while they are not (because of the need of exact matching) in the vector approach. It would be interesting then to combine both approaches, the robust and efficient methods based on keyword vectors, and the approach based on the annotation trees and the domain ontology. Moreover, the ontology that has been used for the experiment remains to be improved, with more complex concept and relation descriptions.

Finally, it must be remarked that the present approach, based on annotation trees, may be well-suited for: (i) the detection or search of resources represented as RDF graphs (a format that is in accordance with annotation trees), (ii) the more general task of retrieval of XML documents according to a set of constraints, because of the tree-based representation that can be associated to an XML document.

References

1. R. Al-Hulou, A. Napoli, and E. Nauer. Une mesure de similarité sémantique pour raisonner sur des documents. In J. Euzenat and B. Carré, editors, *Langages et modèles à objets, Lille (LMO'04)*, pages 217–230. Hermès, L'objet 10(2–3), 2004.
2. G. Antoniou and F. van Harmelen. *A Semantic Web Primer*. The MIT Press, Cambridge, Massachusetts, 2004.
3. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, Cambridge, UK, 2003.

4. F. Baader and B. Sertkaya. Applying formal concept analysis to description logics. In P. Eklund, editor, *Second International Conference on Formal Concept Analysis, Sydney (ICFCA 2004)*, Lecture Notes in Artificial Intelligence 2961, pages 261–286. Springer, Berlin, 2004.
5. F. Baader, B. Sertkaya, and A.-Y. Turhan. Computing the least common subsumer w.r.t. a background terminology. In J.J. Alferes and J.A. Leite, editors, *Proceedings of the 9th European Conference on Logics in Artificial Intelligence (JELIA 2004), Lisbon, Portugal*, volume 3229 of *Lecture Notes in Computer Science*, pages 400–412. Springer-Verlag, 2004.
6. Franz Baader. Computing the least common subsumer in the description logic $\mathcal{EL}$ w.r.t. terminological cycles with descriptive semantics. In *Proceedings of the 11th International Conference on Conceptual Structures, ICCS 2003*, volume 2746 of *Lecture Notes in Artificial Intelligence*, pages 117–130. Springer-Verlag, 2003.
7. M. Chein and M.-L. Mugnier. Conceptual graphs: Fundamental notions. *Revue d'intelligence artificielle*, 6(4):365–406, 1992.
8. D. Fensel, J. Hendler, H. Lieberman, and W. Wahlster, editors. *Spinning the Semantic Web*. The MIT Press, Cambridge, Massachusetts, 2003.
9. S. Handschuh and S. Staab, editors. *Annotation for the Semantic Web*. Volume 96 Frontiers in Artificial Intelligence and Applications. IOS Press, Amsterdam, 2003.
10. J. Heflin, J.A. Hendler, and S. Luke. SHOE: A blueprint for the semantic web. In *Spinning the Semantic Web*, pages 29–63, 2003.
11. P. Langley. *Elements of Machine Learning*. Morgan Kaufmann Publishers, San Francisco, California, 1996.
12. J. Lieber and A. Napoli. Correct and Complete Retrieval for Case-Based Problem-Solving. In H. Prade, editor, *Proceedings of the 13th European Conference on Artificial Intelligence (ECAI'98), Brighton, UK*, pages 68–72. John Wiley & Sons Ltd, Chichester, 1998.
13. A. Maedche, S. Staab, N. Stojanovic, R. Studer, and Y. Sure. SEmantic portAL: the SEAL Approach. In D. Fensel, J. Hendler, H. Lieberman, and W. Wahlster, editors, *Spinning the Semantic Web*, pages 317–359. The MIT Press, Cambridge, Massachusetts, 2003.
14. M.L. Mugnier. On generalization/specialization for conceptual graphs. *Journal of Experimental & Theoretical Artificial Intelligence*, 6(3):325–344, 1995.
15. A. Napoli, C. Laurenço, and R. Ducournau. An object-based representation system for organic synthesis planning. *International Journal of Human-Computer Studies*, 41(1/2):5–32, 1994.
16. F. Sebastiani, editor. *Advances in Information Retrieval, 25th European Conference on IR Research, ECIR 2003, Pisa, Italy, April 14-16, 2003, Proceedings*, Lecture Notes in Computer Science 2633. Springer, 2003.
17. S. Staab and R. Studer, editors. *Handbook on Ontologies*. Springer, Berlin, 2004.
18. V. Uren, P. Cimiano, J. Iria, S. Handschuh, M. Vargas-Vera, E. Motta, and F. Ciravegna. Semantic annotation for knowledge management: Requirements and a survey of the state of the art. *Journal of Web Semantics*, 4(1), 2005.