AUTOMATIC GENERATION OF SEMANTIC ANNOTATIONS FOR SUPPORTING TECHNOLOGY MONITORING ON THE WEB

Tuan-Dung Cao, Rose Dieng-Kuntz
INRIA, Edelweiss Project-Team, 2004 route des Lucioles – B.P.93, 06902 Sophia Antipolis, France

Marc Bourdeau, Bruno Fiès
CSTB, 290 route des Lucioles – B.P.209, 06904 Sophia Antipolis, France

Keywords: Semantic Web Technologies, Semantic Annotation, Semantic Search, Ontology, Technology Monitoring.

Abstract: Automatic generation of semantic annotations of Web document can be useful to support technology monitoring on the Web. In this article, we present the main components of the OntoWatch technology monitoring system relying on a semantic Web approach: the O'Watch ontology dedicated to watch and an ontology-based algorithm for automatic search and annotation of Web documents.

1 INTRODUCTION

At CSTB, Technology Monitoring (TM) task is carried out by the archivists using classic information search tools, which sometimes do not give the expected results as they do not take into account context and semantics of information. OntoWatch system aims at supporting TM task by relying on semantic Web approach (Berners-Lee et al, 2001). When a document is needed, OntoWatch first tries to perform a semantic search relying on its local annotation base. It also tries to discover other Web resources relevant for the watch target but not yet annotated, so as to generate and store annotations on these Web resources in a local annotation base so that they become available for later queries of users. In this paper, after presenting lessons learnt from the building of the O'Watch ontology, we detail and evaluate the OntoWatch new algorithm for searching and annotating Web documents.

2 BUILDING THE O'WATCH ONTOLOGY

The O'Watch ontology aims at offering the vocabularies enabling to make the formulation of the watcher's query more accurate, and annotate the Web documents and information sources with watch topics. Therefore, O'Watch comprises: (a) a vocabulary dedicated to the corporate watch domain: building, water recycling, security…, (b) a vocabulary dedicated to the description of watch task: TM actors, monitoring phases, information sources and some document types.

For building O'Watch, we reused the O'CoMMA ontology since a part of O'CoMMA was related to the field of building and construction. It remained to add the concepts more specialized in the fields interesting for TM as well as concepts and relations dedicated for TM task: TM actors, monitoring phases, information sources and document types. Therefore, knowledge modeling work was required to add the vocabulary from the CSTB corporate thesauri (on *"security and fire"*, *"carcassing and second work"*, *"water reuse"*). Indeed, even though these thesauri are hierarchical, they are not ontologies since the hierarchical link in a thesaurus can be interpreted as *"If we search a document talking about* $topic_1$*, a document talking about* $topic_2$ *is useful"*. But it does not mean that $topic_2$ is necessarily a subconcept of $topic_1$. This hierarchical link is not always a subsumption link: for example, in the thesaurus of *"security and fire"*, a hierarchical link directly connects the term *"flame*

Cao T., Dieng-Kuntz R., Bourdeau M. and Fiès B. (2008).
AUTOMATIC GENERATION OF SEMANTIC ANNOTATIONS FOR SUPPORTING TECHNOLOGY MONITORING ON THE WEB.
In *Proceedings of the Tenth International Conference on Enterprise Information Systems - SAIC*, pages 234-238
DOI: 10.5220/0001698802340238

zone" with *"emissivity of flame"*, *"height of flame"* and "*temperature of flame"*. Therefore, a direct and automatic translation of terms of a corporate thesaurus into O'Watch concepts was impossible and a manual transformation was needed. The possible relations between thesaurus terms are: hierarchical relation, equivalence relation and association relation. Let us summarize the steps for integrating vocabularies from a thesaurus into an existing ontology. For each thesaurus descriptor term T, to be included in the ontology:

- Determine the location in the ontology for the new concept – called c(T) -corresponding to T (i.e. find its parent concept).
- Add in the ontology a concept c(T) having T as label.
- Examine all the relations of the descriptor T with the other terms in the thesaurus.
- If the relation *rel* between the thesaurus terms T and T' is the equivalence relation, T' is added in the ontology as a label of the concept c(T). It plays the role of a synonym for query formulation.
- If *rel* is the association relation: a new concept c(T') is added in the ontology with T' as label, if such a concept did not exist. A *«relatedTo»* relation between the concepts c(T) and c(T') is added in the ontology.
- If *rel* is the hierarchical relation :
  - If *rel* is interpreted as subsumption link, add a concept c(T') in the ontology if such a concept did not exist. Make c(T') child concept of c(T).
  - Otherwise, if *rel* is interpreted as a subpart relation, add a concept c(T') in the ontology and a *«subPartOf»* relation between c(T') and c(T).
- Continue the same steps for the term T'.

For example for the term *« bathtub»* appearing in the thesaurus *« carcassing and second work »* as follows:

```
TS Level 2 SANITARY EQUIPMENT
TS Level 3     BATHTUB
TS Level 3     BIDET
TS Level 3     SHOWER
TS Level 3     WASHBASIN
```

the following concept is added in the ontology O'Watch:

```
<rdfs:Class rdf:ID="BathTubTopic">
 <rdfs:subClassOf
rdf:resource="#SanitaryEquipmentTopic"/
>
 <rdfs:label
xml:lang="en">bathtub</rdfs:label>
 <rdfs:label
xml:lang="fr">baignoire</rdfs:label>
</rdfs:Class>
```

The resulting ontology, O'Watch, is bilingual (in French and English). It comprises 586 concepts and 86 relations and is represented in RDF(S). It is structured in three layers:

- A high layer stemming from the O'CoMMA ontology, and comprising abstract, generic concepts, useful for organizing the ontology but not aimed at the end-user.
- A middle layer comprising concepts related to documents, information sources, actors, and watch domain, and reusable by any company interested in TM.
- A specific layer, including corporate-specific concepts related to documents, information sources, actors and watch domain. This layer is very usable for the end-user but in general, not reusable by other companies.

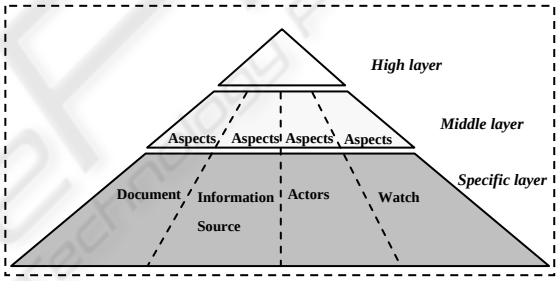

Figure 1: Structure of the O'Watch ontology.

O'Watch building shows how to reuse and extend an existing ontology by integrating vocabularies from existing corporate thesauri.

3 ONTOLOGY-BASED ALGORITHM FOR WEB DOCUMENT SEARCH AND ANNOTATION

When the OntoWatch system searches for new documents to annotate, the use of the O'Watch ontology is essential. Instead of keywords, concepts of O'Watch will be selected to form a query. As advantage, the child concepts of the initial concepts in the ontology can be used to constitute the request, so it helps to reduce some lacks in the search results obtained from a classic search engine. For example, when we search for documents about *"Sanitary equipment problem"*, it is useful to be able to find

documents evoking *"Bathtub overflow"*, *"Bidet plugging"*, *"Shower too-low-flow"* or *"Wash basin fissure"*. Thus the query should be able to include all these alternatives more precise than *"Sanitary equipment problem"*. The problem is how to form the query and send it to the search engine.

The best case of the algorithm is that a user's query, considered as a list of concepts in ontology, could be replaced by a system query. This query generated by the system contains all descendant concepts of all initial concepts in the user's query. But Google limits the number of keywords of a query and unfortunately, in most real situations the number of descendant concepts exceeds this limit. Thus, in our previous work (Cao et al, 2004), besides implementing the algorithm for ideal situation above, we developed two algorithms enabling to overcome this Google limit by using the branches of the domain ontology. Our evaluation on those previous algorithms shows that they are more appropriate for obtaining specialized documents than general documents. However they have the drawback to risk to privilege some branches and to sometimes retrieve documents not related to all the concepts of the user query. Therefore we needed another solution that would better ensure that the documents retrieved by Google are related to all the concepts of the user's query. More precisely, the system queries generated, which consist of a list of descendant concepts, should not privilege any initial concept of the user's query.

3.1 Balanced Concept Selection

Our new algorithm of the OntoWatch system allows us to retrieve and to annotate documents as much as possible related to initial concepts from user's query. As we mentioned before, while using the ontology for searching and annotating document, the major difficulty is the great number of descendant concepts. So our solution is not to take as much as possible descendant concepts of each initial concept, but rather take into account a balanced distribution between descendant concepts selected from different initial concepts in user query. As the number of concepts permitted in a Google query is too small in comparison with the number the descendant concepts possible to select, our algorithm will have to make multiple choices. In other words several queries corresponding to various selections of concepts will be generated.

To ensure the balanced distribution between descendant concepts selected, we rely on a criterion: the number of descendant concepts permitted depends on the weight of their initial concept in comparison with other concepts in user query.

Let Total_Desc (C) be the number of all descendant concepts of all initial concepts in the user query, and Local_Desc be the number of descendant concepts of an initial concept C.

The weight of C is the value of Local_Desc/Total_Desc (C).

The presence in the generated query of at least one descendant concept corresponding to each concept in user query avoids the drawbacks of the two previous algorithms. For each initial concept, we have a limit number of concepts to select in order to contribute to final query. The problem is the strategy of descendant concept selection. Contrary to our previous algorithms, which select concepts in the depth direction, for a initial concept we select its descendant concepts in the breadth direction. The result of this process is a set of partial queries, corresponding to each initial concept. The combination of these partial queries gives us the final set of system queries.

3.2 Detailed Description

BalancedOntologySearchAnnotation is the main module in charge of searching documents on the Internet by Google search engine with a set of queries generated by the algorithm. Then, in each document belonging to the results, the terms corresponding to the concepts in the ontology are extracted and stored in a hash table the key of which is the document URL, in order to generate an RDF annotation about this document. Based on the comparison of the URL of each document, all the redundancies are eliminated and the system aggregates the concept list when a same document was found by different queries. The module takes the user's query $Request_u$ (which is in fact a list of concepts) as input and produces annotations represented in RDF, as output.

```
Algorithm
BalancedOntologySearchAnnotation(Requestu)
Q = GenerateQueries (Requestu)
// Ann = hashtable the key of which is
the URL of each document retrieved.
Ann ← ∅
for each query q ∈ Q do
 Send q to Google
 for each document D in Google results do
    K ← ExtractConcept(D)
    if D.URL was not in Ann
      Add K and URL of D to Ann
```

```
    else Update the set of concepts to
annotate with D.URL
    endif
 endfor
endfor

Annotate all documents D in Ann with its
URL and set of concepts attached.

End
```

The most important part is the algorithm to generate system queries from a user's query using ontology. First, to ensure that a system query generated does not privilege only one or two branches of ontology, the algorithm calculates, for each concept in the original user's query, the limit number of descendant concepts to appear in the system query. This limit is based on the ratio between the number of descendant concepts of an original concept in the user's query and the number of descendant concepts of all original concepts in the user's query.

```
Algorithm NumberConceptLimited(Request_u)
Desc = Set of all descendant concepts of
all concepts from Request_u
S ← Card(Desc)
for each Ci ∈ Request_u do
if Card (Descendants (C_i)) ≤ 2
then Limit_i ← Card (Descendants (C_i))
else Limit_i ← Card(Descendants(C_i))/S)
*Google_limit
endfor
End
```

Notice that this algorithm takes into account the fact that if a concept has only one or two child concepts, these subconcepts often play a role as important as their parent, and they are very close semantically to the initial concept. So they should appear in the system query generated.

In this algorithm, the system query is composed of some partial queries. Each partial query is represented by a list of descendant concepts of an original concept in the user's query. For each original concept, there are several possible corresponding partial queries to be selected by an algorithm described in the module *SetofPartialConcepts*. Thus the result of module *GenerateQueries* is a set of all possible combinations of all partial queries of each concept in user's query.

```
Algorithm GenerateQueries(Request_u)
for each concept C_i ∈ Request_u do
Get the set of all partial queries
generated from Ci. Each partial query is
a set of concepts selected from
descendant concepts of C_i:
Q_i ← SelectPartialConcepts(C_i, Limit_i)
endfor

Q = Combine each partial query in all Q_i
to generate global set of queries using
combination algorithm.

End
```

As the number of descendant concepts of each concept often exceeds the limit permitted, how to select, among these concepts, the Limit best concepts to form the partial queries? The principle of algorithm `SelectPartialConcepts` is:

Calculate the maximum depth level of the initial concept in ontology. If the ontology is a graph, we define maximum depth level of concept C as the number of concepts (except C) in the path from C to its deepest descendant concept C_d. So we can say all the direct child concepts of C are at depth level of 1. And so on. So let K_i the maximum depth level of the initial concept Ci in the user's query; the number of generated queries is:

$$\prod_{i=1..n} K_i$$

At each level of depth from C, we choose a number of Limit concepts from its descendant concepts, to make a partial query. The inputs of this module are the original concept C in the user's query and the number of descendant concepts permitted to select. The output is the set of partial queries generated.

The module *SelectConceptFromLevel* is responsible for selecting a predefined number of concepts from a certain depth level of a concept. The algorithm starts by taking all concepts in the current depth level. If the number of concepts at that level is smaller than the limit specified, the algorithm steps to the next depth level and so on.

```
Algorithm SelectConceptFromLevel(C, level,
Limit)
Begin
SetC ← GetAllConceptAtLevel(C,level)
SelectConceptFromLevel(SetC,level,Limit)
End
Algorithm SelectConceptFromLevel(SetC,
level, Limit)
Begin
  Let k ← Card(SetC)
  if k ≥ Limit then
   Add a number of first Limit concepts in
SetC to OutC. Exit
  else if k < Limit then
   Add all concepts of SetC to OutC.
   NextSet← GetAllConceptsNextLevel(SetC)
SelectConceptFromLevel(NextSet,level+1,Limit
– k)
```

```
    endif
End
```

The balanced concept selection algorithm was implemented in Java, using Google API for searching Internet.

4 EVALUATION

The comparison of manual search by the CSTB archivist and automatic search by the OntoWatch system on the same watch target, enables to determine several evaluation measures:

- M the number of relevant documents (among the Max first ones retrieved by the watchers) obtained by manual search.
- A the number of documents retrieved by automatic search (among the Max first ones), and evaluated as relevant by the watchers.
- I the number of relevant documents found by both automatic and manual searches.

In order to carry out the evaluation, we interacted with an archivist (that usually performs TM for the company) and with corporate domain experts. We relied on the queries used in the watch process concerning a specific topic: *water reuse* (resp. *climate change*). Through these queries, we tested the algorithm using the O'Watch ontology to search and annotate Web documents. Table 1 shows the results of the evaluation for both queries.

Table 1: Evaluation results for the two queries.

Evaluation measure	Measure definition	Results for "Water reuse"	Results for "Climate change"
$Relevance_A$	A / Max	86%	60%
$Relevance_M$	M/ Max	38%	36,66%
$Covering_{A/M}$	I / M	94/7%	90,9%
$Covering_{M/A}$	I / A	41,86%	55,55%
$Newness_{A/M}$	A-I / A	58,13%	44,44%
$Newness_{M/A}$	M-I / M	5,26%	9,09%

So automatic search offered by OntoWatch, leads to better results than a manual search.

5 CONCLUSIONS

The main contributions of the OntoWatch system are: (a) the building of the O'Watch ontology (in particular, our original generic method for integrating vocabularies from existing thesauri in an existing ontology) and (b) an original algorithm for automatic searching and annotating Web document; while selecting concept from ontology to generate a query, this algorithm takes into account the balance in the distribution of descendant concept. The evaluation shows the interest of the automatic search w.r.t to the manual search by the watcher.

Few works based on semantic web tackle the TM problem: in (Maynard et al, 2005), the authors present the h-TechSight system that offers ontology-based information extraction used in the context of applications such as market monitoring and technology watch. OntoWatch search and annotation algorithm can also be compared to automatic annotation systems analyzed in (Uren et al, 2006). But such tools generally rely on Natural Language Processing, machine learning or information extraction techniques. OntoWatch algorithm relies on quite different principles in comparison with all these annotation tools.

Our work is generic: with an adaptation of the O'Watch ontology to the corporate watch characteristics (corporate watch process, corporate watch domain) in case of need, OntoWatch can be reused for TM task in any company in any domain.

REFERENCES

Berners-Lee, T, Hendler, J and Lassila, O., 2001. The Semantic Web. Scientific American. May, p. 29-37.

Cao, D., Dieng-Kuntz, R., Fiès, B., 2004. An Ontology-Guided Annotation System for Technology Monitoring, IADIS Int. WWW/Internet 2004 Conf.

Maynard, D., Yankova, M., Kourakis, A., Kokossis, A., 2005. Ontology-based information extraction for market monitoring and technology watch. ESWC'2005 Workshop on End User Aspects of the Semantic Web.

Uren, V., Cimiano, P., Iria J., Handschuh, S., Vargas-Vera, M., Motta, E. and Ciravegna, F., 2006. Semantic annotation for knowledge management: Requirements and a survey of the state of the art. In Web Semantics, Volume 4, Issue 1, 14-28.