An Algorithm and a Tool for Comparing Ontology Versions

Nora Taleb

1Research laboratory in data processing, Annaba, Algeria
Nora.taleb@univ-annaba.org

Abstract

In this paper we developed and implemented COMP algorithm. An algorithm that provides an approach to compare ontologies versions. COMP performs all tasks automatically and the intervention of the user is not required. The objective is to underline the interest of having a tool for the analysis of the differences between versions of ontology of the same domain. We present a tool named CVO based on COMP algorithm that helps handling and comparing ontologies in OWL. The domain of study relates to the medical field of the breast cancer disease.

Keywords: Ontologies, comparison of ontologies, alignment

1. Introduction

Considering the increase in the number of ontologies specifying the same field, the research in ontological engineering on the comparison and the alignment of ontologies in order to establish correspondences, to trace the evolution of an ontology through the comparison of the various versions, to create a set of axioms or rules to bring ontologies together, to generate mediators for queries, or in order to constitute a target base of knowledge which will be used as reference for the field, starting from a base of knowledge sources [1] Moreover, the distributed environments which develop ontologies require, among other functionalities, the possibility of integrating and of amalgamating extracts of existing ontologies. Within this framework, the comparison of the various entities defined in such ontologies is a central spot.

This task must be carried out in a completely automatic way as it is not realistic to perform the mapping of ontologies by hand, a fortiori when these ontologies exceed a certain size or complexity.

Generally, the existence of several versions of an ontology can be considered in two principal situations:

1. Several versions can result, when a distributed ontology is developed in an independent way [2].
2. An ontology resulting from the application of a change can be considered as a new version.

The ideal would be to preserve the various versions of ontology and to keep all the information concerning the differences and compatibilities between them. This requires methods of identification of versions, of differentiation of versions (based on the same principles as the methods of measurement of semantic similarities in ontology alignment), and of specification of relations between versions; procedures of update of ontology; and of the mechanisms of access to the various versions of an ontology [3].
In this work, we propose a framework both logical and probabilistic allowing the automatic comparison of ontologies. In particular, we present a software tool using the semantics of the definitions and axioms of OWL in order to establish the journal of the changes between two ontological versions. The field of experimentation is the medical field of the disease cancer.

2. State Of The Art

2.1. Classification of the Changes

The objective of the classification of the changes is to define, for a given language of representation of ontologies, a taxonomy of changes specifying the classes of changes and their properties. The principal classifications defined in the literature were proposed for languages KAON2 [4, 5, 6] and OWL3 [7].

The KAON ontology of changes classifies the changes according to three levels of abstraction [6]:
- Elementary changes, applying modifications to only one entity of an ontology.
- Composite changes, applying modifications in the direct neighborhood of an entity of an ontology.
- Complex changes, applying modifications to an arbitrary set of entities of an ontology.

Moreover, the KAON changes were also classified according to their effect [6]:
- Additive changes, adding new entities to ontology without deteriorating the existing entities.
- Subtractive changes, removing certain entities or parts of entities. Thought like a minimal and complete unit, the ontology of changes KAON does not take into account the modifications of entities. Klein differentiates from the basic operations of changes and complex changes [7]:
- The basic changes are simple and atomic changes which can be specified while being based only on the structure of ontology and which modify only one characteristic of the model of knowledge OWL, i.e., only one entity of the ontology (like an operation of addition of a class or suppression of a relation “ is-a ») ;
- The complex changes correspond to composite and rich changes grouping logical sequences of basic changes and incorporating information on their impact on the logical model of ontology (as for example moving up subclasses, widening the same domain of a property object to its super classes, amalgamating classes, etc.). In addition to their specification, complexity also appears in the effects of these changes. If the effects of the basic changes remain relatively minor, the cumulated effects of all the intermediate changes carrying out change complexes can be important.

2.2. Methods of ontological comparison

There are four methods of comparison of ontologies:

1) Comparison of the internal structures: compare the internal structure of the entities (e.g., value interval, cardinality of attributes, etc.);
2) Comparison of the external structures: compare the relations entities with others. It is composed of methods of comparison of the entities within their taxonomies and methods of comparison of the external structures by taking the cycles into account;
3) Comparison of the instances: compare the extensions of the entities, i.e., it compares the set of the other entities which are attached to them (concepts of ontology);

4) Semantic method: compare interpretations (or more exactly the Models) of the entities.

2.3. Tools to manage Ontology changes

Within the framework of management of change suggested for ontologies, several specialized prototypes have been developed [7, 8, 9, and 10]:

- The OntoView tool implements a procedure of detection of changes for ontologies in RDF. Its principle consists in observing rules in order to discover specific operations of change and to produce sets of transformation between the versions of ontology [12];

- Two extensions of the PROMPTdiff tool - one plug-in developed for Protected to research mappings between frames while basing itself on heuristics [13] - were proposed in [13] their role is to define the relations of evolution between the elements of two versions of ontology. The user interface makes it possible to visualize certain complex changes between versions of ontology.

    A more complete system of ontology evolution is described in [13, 14]. The core of the system is based on the CHAO ontology of changes and annotations (Exchange and Ontology Annotation). The instances of the CHAO ontology represent the changes between two versions of the ontology and the annotations users related to these changes. The system is implemented in the form of two Protégé plug-ins:

    - Plug-in management of change giving the access to a list of changes and allowing the users to add individual or grouped annotations of change and to consult the history of the concepts;

    - PROMPT Plug-in providing comparisons between two versions of an ontology and information on the users having made these changes, and facilitating the acceptance or the rejection of the changes [13].

3. Proposed Approach for The Comparison Between Versions Of Ontologies

The comparison of two versions (V1 and V2) of ontology consists in identifying the various types of change:

- Addition: a concept (or relation) is added if it does not appear in V1 but it is in V2 or the reverse.

- Deletion: the concept (or relation) is removed if it appears in V1 but not in V2

- Modification: the concept (or relation) is modified if it appears in the two versions V1 and V2, but in two different forms.

- Stability: the concept (or relation) is not changed if it appears in V1 and V2 and that it with the same form in the two versions.
Consequently, the various types of comparisons being able to be drawn are as follows:

A- Exact Comparison:

This comparison consists in finding the concepts (or the relations) with the same name which exists in the two versions of the ontologies.

B- Approximate comparison:

This comparison is more interesting because it consists in finding synonymous terms for concept names or concepts with similar names. The method is based on hybridization of the algorithms of fusion with a dictionary of the synonyms.

C- Similar comparison:

The similar comparison is a manual comparison, the user selects the names of concepts which he considers similar.

D- Comparison different:

This comparison consists in finding the concepts (or relations) which exist in the first version and do not exist in the second one.

Figure 1. Two versions of the cancer ontology
4. CVO for The Management Of The Ontological Comparison

Our project of Ontology Versions Comparison (CVO) makes it possible to compare two versions selected by the user. As follows is the COMP Algorithm implemented for CVO tool.

4.1. The COMP algorithm

Figure 3 illustrates the COMP Algorithm; it takes two ontologies as input in the form of OWL files as follows:

• Traverse the two versions of an ontology,
• For each version, extract the concepts and the relations,
• Compare the two versions,
• Post the comparison result in the form of a journal changes (concepts, relations) The following architecture explains in detail the CVO tool:

The CVO system is a tool made up of several complementary components where each one has a precise task to achieve.

Among the major functionalities which these components offer:

➢ Analysis of two input versions.
➢ Localization of the differences between these two versions through the posting of the journal of the changes of concepts (or relations) for each one.
The COMP algorithm is composed of two principal modules: analyze and comparison.

4.1 Analyze

To analyze the two versions of the program written in OWL, we benefited from the capacities of Java; in extracting the concepts and the relations of these two versions, it returns as result the journal of the changes for these concepts and relations for each version.

4.2 Comparison

The entities of the two versions which we want to compare are the concepts and the relations. To compare these entities, we distinguish the following classes:

➢ Class 1

This class is the main class. It is responsible for the initialization of COMP, it manages the graphic interface and the interaction with the user. It offers a window with various menus specific to precise functionalities (to extract the concepts (relations) of an OWL file², to analyze the program, comparison of two ontology versions).

➢ Class 2

This class is regarded as one of the most important classes in COMP. It provides the means and the structures necessary to extract the concepts (relations) of an ontology which exist in the form of an OWL file, and visualizes them in a journal of changes.

➢ Class 3

This class deals with the detection of the differences (to make the comparison itself).

5. Results

The result of the execution of COMP algorithm is the journal of the changes between the two versions. The experiment was carried out on two versions of the ontology of the cancer disease. See Figure 4, 5.

Figure 4. analyse Of CVO
- The first column contains the concepts which exist in the version 1 and does not exist in the second version.

- The second column contains the concepts which exist in the version and does not exist in the first version.

- The third column contains the intersection of the concepts between the two versions of ontologies (common concepts).

When one selects an item in the journal of the changes of the relations, the window Figure 6 is presented.

---

**Figure 5. Relations changes**

**Figure 6. Journal of the relations changes**
- The first column presents the relations which exist in the version 1 and does not exist in the second version.
- The second column shows the relations which exist in the version and does not exist in the first version.
- The third column shows the intersection of the relations between the two versions of ontology.

6. Conclusion

Any our application was developed in Java; it can be integrated in other resource supporting the Java virtual machine.

For the implementation of our ontology, we chose Protégé 3.4; several reasons justified our choice:

Protégé 3.4 source is a free open editor, it allows to import and to export ontologies in various implementation languages (RDF-Schema, OWL, DAML, OIL, ..., etc.), it has a modular interface, which allows its enrichment by additional modules (plug-in), Protégé 3.4.1 allows the edition and the visualization of ontologies. Finally, it is provided with API written in Java, which makes it possible to develop applications being able to access Protégé ontologies and to handle them.

Our ontology is implemented in OWL (ontology Web Language). However, OWL files are not easily exploitable in their rough form because of their complex structure. In order to be able to exploit it we needed a “translator” able to translate the mark-up tags and the semantics conveyed by OWL files into objects easy to handle by programs. For this purpose, we used the JENA API.

An experimentation of the tool was made on the ontology of the cancer disease, and satisfactory results were obtained. Us minks we aim to use the tool to make the comparison in more complicated fields such as sociology in order to be able to find the differences between two companies and to trace the common points and the points in disjunction between them.

References
