AN ARCHITECTURE FOR SEMI-AUTOMATIC ONTOLOGY MERGING SYSTEM.

Siham AMROUCH  
Computer Science Departement  
Med Cherif Messadia University  
Souk Ahraas, Algeria  
sihamamrouch@yahoo.fr

Sihem MOSTEFAI  
MISC Laboratory, Computer Science Departement  
Mentoury University  
Constantine, Algeria  
xmostefai@yahoo.com

ABSTRACT

In recent years, ontologies have played a key technology role for information sharing and agents interoperability in different information systems. But, it seems that there is always more than one conceptualization for the same domain or even for similar domains. In other words, it emerges every day, new different ontology to model the same domain. Therefore, to answer queries on the modeled domain, bridge the gaps between different ontologies is a key challenge for the researchers in the AI community by using ontology merging. In this paper, we propose an architecture for a semi-automatic ontology merging process. The semi-automatic character is handled by the human intervention where the knowledge engineer intervenes to validate the results provided by the similarity computation module. This later is self composed of two parallel sub-modules. Both of them are based on the combination of lexico-semantic similarity measure. But, in the first one, it is applied on the concept’s names while the second one, it is applied on their properties. If the first sub-module fails to recognize similarities, the second one will accomplish the task which minimizes the human intervention and enhances the performances of the algorithm. The identified similar concepts have to be merged into a single one in the merged ontology, after human validation. The judged different concepts are directly copied to the merged ontology after identifying the concept the most similar to it from the first ontology and checking their hypo(hyper)nymy relationship using wordNet.

Keywords- Semi-automatic ontology merging, semantic integration, wordNet, owl, lexico-semantic similarity.

1. INTRODUCTION

In the last decade, the aim of researchers in semantic web community was and still to bring the actual web to its full potential by considering ontologies as the best means to annotate the data on the web.[7]. In other words, ontologies are emerged as the best models for information storage and representation with preserving the semantics embedded in their application domains, in several areas such as semantic web and web services, industrial and e-technologies in general. According to T. Gruber[1], ontology is an explicit specification of a conceptualization. This structure can be cognitively semantic (ontology intended to be exploited by the user) or computationally semantic (ontology intended to be exploited by the machine), [2]. In general, an ontology is composed of a set of concepts described by a set of properties and related by a set of semantic relationships, to construct an hierarchy of classes, where each sub-class described a concept that is more specific then the concept described by the super-class. With several designers that appear every day, there is always more than one ontology that describes the same domain. In other words, it emerges every day, different ontologies developed by different developers with different viewpoints and in different goals of use. Hence, to create a common repository of knowledge base and to remove overlaps between existing ontologies, we go for ontology merging. This process is seen as the effort of building a single ontology from a set of source ontologies that cover a wider scope. Several tools and algorithms for ontology merging, based on different criteria, exist in the literature such as: Prompt, Chimaeira, ONION, FCA-Merge, etc. According to [4], these algorithms are generally based on: Names and descriptions of concepts in natural language, class hierarchy (the relations: subclass and superclass), setting properties (domain, co-domain and restrictions), classes’ instances and classes’ descriptions (Description Logic-based tools). The contribution outlined in this paper describes an architecture for a semi-automatic ontology merging system. In this later, and in order to identify similar concepts, the human intervention is necessary to validate the results obtained by combining the results of computation of lexical and semantic similarity measures. This paper is structured as follows: Section 2 briefly describes the
2. ONTOLOGY MERGING PROCESS

According to [3], ontology merging is seen as a complex process composed of three sub-processes: Firstly, ontology mapping and alignment. Then the normalization of source ontologies [5]. This later aims to reconcile the different choices of conceptual models and representation languages of the ontologies to be merged. Once these ontologies are sufficiently homogeneous, the final sub-process is to build the

3. SEMANTIC ENRICHMENT

This is a substantial research field that serves the semantic web by facilitating interoperability between different applications and/or knowledge sources such as ontologies. In this paper, we will opt for the semantic enrichment from an external resource, wordNet, to avoid the limitations of the lexical aspect in the ontology merging process after their possible extensions according to their application domain. So, it is at this stage where acts the semantic aspect to support the ontology merging process. Herein, the more the extension of the source ontologies is close to the same shared ontology, the easier will be the similarity identification process. In addition, reasoning and inference processes handled by the ontology representation languages contribute in specifying the constraints of similar concepts merging. However, semantic integration process may be altered by several types of mismatches [17]. The first one is the language level mismatches or syntactic mismatches caused by the different ontology representation languages. In addition, even with ontologies represented in the same representation language, we can have ontology level mismatches or semantic mismatches, such as using the same term to describe different concepts (homonyms), use different terms to describe the same concept (synonyms), use different levels of granularity, etc.

4. MAPPING DISCOVERING

This is the most important step in the ontology merging process. It identifies similar concepts that have to be merged into a single one in the last step of the ontology merging process. The resulting single concept includes all

5. RELATED WORKS

Several tools for ontology Merging (and even ontology Mapping or Alignment) exist in the literature. Most of these tools are semi-automatic and the design of fully automatic tools is usually a delicate issue. In this section, we outline the well known and recent ones:

FCA-Merge [12]. It’s a method for semi-automatic ontology merging. Its process is summarized as follows: First, from a set of input documents, popular ontologies (ontologies
equipped by their instances) are extracted. Once the instances are extracted and the concept lattice is constructed, FCA-techniques are used to generate the formal context of each ontology. Using lexical analysis, FCA-techniques retrieve specific information that combines a word or an expression to a concept if it has a similar concept in the other ontology. Then the two formal contexts are merged to generate the pruned concept lattice. Herein, the knowledge engineer may eventually intervene to resolve conflicts and eliminate duplications using his background about the domain. It should be mentioned that the major drawback of FCA-Merge is that it is based on instances to identify similar concepts, however, in most applications, there are no objects that are simultaneously instances in both source ontologies.

**PROMPT** [13] is an interactive ontology merging tool, it proposes a list of all possible merging actions (to-do list). After that, the knowledge engineer selects the appropriate proposals that go with his needs. Then, PROMPT automatically merges the selected pairs of concepts, provides the conflicts generated after merging (conflict-list) and proposes their appropriate solutions. Finally, the knowledge engineer selects the most suitable solutions.

**CHIMAERA** [14]. An interactive ontology merging tool, where the knowledge engineer is charged to make decisions that will affect the merging process. Chimaera analyzes the source ontologies and if it finds linguistic matches the Merging is performed automatically, otherwise, the user is prompted for further action. Like PROMPT, Chimaera is an ontology editor plugin, namely Ontolingua, but they differ in the suggestions they make to their users with regard to the merging steps.

**GLUE** [15]. To find mappings between two source ontologies $O'$ and $O''$, Glue uses machine learning techniques. So, for each concept of ontology $O'$, Glue finds its most similar concept in ontology $O''$ based on different practical similarity measures and several machine learning strategies. The authors also used a technique called "relaxation labeling" to map the two hierarchies of the two ontologies. This technique assigns a label to each node of a graph and uses a set of domain independent constraints, such as, two nodes of concepts $c'$ and $c''$ match if the nodes of their neighbourhood $v(c')$ and $v(c'')$ also match, and a set of domain dependent constraints, such as, if $X$ is an ascendent of $Y$ and $Y$ matches "direction" then $X$ does not match "sub-direction".

**ONION** [16]. According to the authors, ontology Merging is inefficient because it is costly and not scalable. So, ONtology compositION system provides an articulation generator for resolving mismatches between different ontologies. The rules in the articulation generator express the relationship between two (or more) concepts belonging to the

---

1 neighbourhood is defined to be the children, the parents or both.

---

6. **GENERAL PROPOSED SYSTEM**

The aim of our work is to propose an architecture for a semi-automatic ontology merging system where the human intervention must not be avoided to ensure good performances. First, we import the two source ontologies that cover the same application domain (or related domains). Next, we identify their similar concepts. Here we will use an Information Retrieval (IR) technique, where each concept of the first ontology is compared with all concepts of the second one. To avoid a lot of unnecessary comparisons, we begin from the top of one ontology and from the bottom of the second one. We will base our method on linguistic analysis of concepts’ names to compute the lexical and semantic similarities between them. But, sometimes, we may find two similar concepts that are described by the same or similar properties but which are labelled with different strings, for example, one concept can be labelled with the abbreviation of its name or even by a code, in such case, it is impossible to identify its synonyms from the used dictionaries. To resolve this problem, all the couples of concepts that are judged dissimilar by linguistic analyses of their names are passed through a module based on linguistic analysis of properties of both concepts. Both sub-modules of the similarity identification module combines the results of two similarity measure techniques used in string comparison (concepts or properties), one of them is lexical and the other one semantic. The obtained results are accepted or rejected by a knowledge engineer using his background on the domain and oriented by his own needs. After that, the concepts accepted as similar are merged into one concept whereas the concepts judged dissimilar are directly copied to the resulting ontology. The later step require an other referencing to wordNet to determine the hypo(hyper)nyn relationship between the current concept from the second ontology and the concept the most similar to it from the first ontology. Hence, the resulting ontology will be larger and more complete and will cover a wider application domain.

A. **LEXICAL SIMILARITY**

This technique is based on the computation of a distance between two strings describing the names of two concepts. Several measures of similarity or distances exist in the
literature such as Levenshtein distance [8], Hamming distance [11], Jaro distance [9], Jaro-winkler distance [10], etc. All of these measures are based on the same assumption described by [6] which states that two strings are similar if they share enough important elements. We have chosen to use the Jaro distance as a similarity measure because it yields a value which is consistent with the value given by the semantic similarity measure that we have proposed (a value between 0 and 1) and therefore their combination is easier. The lexical similarity between the two concepts c1 and c2 is given by:

\[ \text{SIMlex}(c1, c2) = D_j(s1, s2) \]  

(1)

where Dj(s1, s2) is the Jaro distance between the two strings s1 and s2 labelling the two concepts c1 and c2 and which is defined by the equation:

\[ D_j(s_1, s_2) = \frac{1}{3} \left( \frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m - t}{m} \right) \]  

(2)

Where: m: The number of matched characters. t = N / 2: the number of transpositions.
N: The number of pairs of matched characters that are not in the same order in their respective chains. Two identical characters of s1 and s2 (describing the concepts c1 and c2 respectively) are considered matched if their distance (i.e. the difference between their positions in their respective chains) does not exceed a certain value given by:

\[ \text{val} = \left[ \max \left( \frac{|r_1|}{2}, \frac{|r_2|}{2} \right) \right] - 1 \]  

(3)

The two concepts c1 and c2 are considered lexically similar if the distance between them exceeds a critical threshold to be determined empirically.

Example: Computation of lexical similarity between ‘auto and automobile’ and between ‘auto and car’:

\[ \text{SIMlex} (\text{auto, automobile}) = D_j(\text{auto, automobile}) = ? \]

\[
\begin{array}{cccccccccc}
| & a & u & t & o & m & o & b & i & l & c \\
\hline
a & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
u & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
t & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
o & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
\end{array}
\]

m=5 (number of 1 in the table), |s1|=10, |s2|=4, N=1, t=1/2,

\[ \text{SIMlex(\text{auto, automobile})} = \frac{1}{3} \left( \frac{5}{10} + \frac{5}{4} + \frac{5 - 0.5}{5} \right) = 0.883 \]

Assuming that the threshold = 0.5, SIMlex = 0.883 ≥ 0.5 then auto and automobile are lexically similar.

Now, let’s compare “car” and “auto”, SIMlex (car, auto) = D_j(car, auto) = ?

\[
\begin{array}{cccccccc}
| & a & u & t & o & m & o & b \\
\hline
a & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
u & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
t & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
o & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
\end{array}
\]

m=1, |s1|=4, |s2|=3, N=1, t=1/2,

\[ \text{SIMlex(\text{car, car})} = \frac{1}{3} \left( \frac{1}{4} + \frac{1}{3} + \frac{1 - 0.5}{1} \right) = 0.36. \]

SIMlex(car, auto) = 0.36 <0.5, then auto and car are lexically similar.

SIMlex(car, plane)=0.34 <0.5, then plane and car are lexically dissimilar.

B. SEMANTIC SIMILARITY

When the concepts are semantically similar but their names are different (synonyms) the null lexical similarity does not reflect the reality. To solve this problem, the integration of semantic similarity measure is crucial. To do this, we have begun with a semantic enrichment of the two source ontologies from wordNet2. It involves building a synonymy vector containing the synset elements for each concept. We recall that WordNet is a computerized english dictionary where the basic unit is the concept. It uses two different means to define the meaning of a word, the synsets and the lexical relations. A word is then defined by a set of synonyms (synset) and a definition. Example: Board: synset = {board, blank} Definition: A piece of wood.

For the computation of semantic similarity, we have used an information retrieval technique, which involves comparing each concept in the first ontology with all concepts of the second one to find out the most similar concept. We defined the semantic similarity between two concepts C1 and C2 as follows:

\[ \text{SIMsem} (c1, c2) = 2 \times \frac{\text{card} (\text{synset (c1)} \cap \text{synset (c2)})}{\text{card} (\text{synset (c1)}) + \text{card} (\text{synset (c2)})} \]  

(4)

\[ \text{SIMsem} (c1, c2) \quad [0,1] \]

The two concepts c1 and c2 are judged similar if SIMsem (c1, c2) is greater than a critical threshold which will be determined empirically. If the two concepts are exactly similar SIMsem (c1, c2) =1, in the opposite case SIMsem (c1, c2) =0.

Example: Computation of lexical similarity between ‘auto and car’ and between ‘car and plane’:

\[ \text{Synset (auto)}=\{ \text{car, auto, automobile, machine, motocar} \}, \text{synset (car)}=\{ \text{car, auto, automobile, machine, motocar} \}, \text{synset (plane)}=\{ \text{airplane, aeroplane, plane} \} \]

\[ \text{Note: } \]  

\[ \text{http://www.wordNet.princeton.edu/wordNet} \]
\[ \text{SIMsem}(\text{auto}, \text{car}) = 2 \times \frac{5}{10} = 1. \]

\[ \text{SIMsem}(\text{car}, \text{plane}) = 2 \times \frac{0}{8} = 0. \]

Then auto and car are semantically similar but plane and car are semantically dissimilar.

Once the two similarity measures are computed, we compute the \textit{lexico-semantic} similarity that combines the two results through the formula:

\[ \text{SIMlexSem}(c1, c2) = \frac{\text{SIMlex} + 2 \times \text{SIMsem}}{3} \quad (5) \]

The two concepts are considered similar if \( \text{SIMlexSem}(c1, c2) \) reaches a critical threshold which will be determined empirically.

Example:

\[ \text{SIMlexsem}(\text{auto}, \text{car}) = \frac{0.36 + 2 \times 1}{3} = 0.75 > 0.5 \]

So, if the knowledge engineer accept or validate this similarity, the two concepts auto and car are similar and then will be merged into a single concept autocar.

\[ \text{SIMlexsem}(\text{plane}, \text{car}) = \frac{0.34 + 2 \times 0}{3} = 0.113 < 0.5 \]

So, the two concepts plane and car are dissimilar and then will directly (without passing by the knowledge engineer) be separately copied in the resulting ontology.

**HOW THE MERGED ONTOLOGY IS CONSTRUCTED?**

First, the merged ontology is initialized by the first source ontology. (All the concepts with all their properties of the first ontology are copied in the initial merged ontology). Then, each concept of the second source ontology is compared with all the concepts of the first one. Such that, we begin from the top of the first ontology and from the bottom of the second one. This technique avoid a lot of unnecessary comparisons. We have composed this module on two sub-modules.

In the first one, the lexico-semantic similarity is simply and directly computed on the strings naming or labelling the discussed concepts. If this module fails to find similarities between them, the second module of similarity identification between their vectors of properties is launched. Herein, each one of the two concepts is identified by its array of properties. Then each property of the first array is compared with all the properties of the second array, using usually the lexico-semantic similarity measure presented previously. Each time two properties are found similar, a counter \( c \) is augmented by one.

Finally, the ratio \( R \) of similarity between the two concepts (described by their properties) is computed through the formula 6, Where \( p1 \) and \( p2 \) are the arrays of properties of \( c1 \) and \( c2 \) respectively. If \( R \) reaches a certain threshold (which will be determined empirically) the two concepts \( c1 \) and \( c2 \) are judged similar, else, they are dissimilar.

\[ R = \frac{2 \times c}{p1.length + p2.length} \quad (6) \]

At the end, if two concepts are judged similar, we compare their properties. The ones of the second concept that does not exist (or have not similar ones) in the first concept (which has been copied in the initial resulting ontology) are added to the properties of this first concept. Hence the two similar concepts are merged into a single one without any omission of information. Else, if the two concepts are judged dissimilar, the most related concept (class) from the first ontology to the current concept of the second ontology is identified. It corresponds to the concept with the highest similarity measure between their properties. Finally, the hyper(hypo)nym relationship between the two concepts is determined using wordNet. This step is achieved by checking one concept in the hierarchy structure of hypo(hyper)nymy of the second concept. The concept that figures between the hypernyms of the other one is the less specific, and hence will be copied (with all its properties) as the super-concept (super-class) of the other. This process is repeated for each concept of the second ontology. Hence, the whole merged ontology construction is accomplished. The whole proposed architecture of the semi-automated ontology merging system is depicted by figure 2.
Figure 2. The proposed architecture for the semi-automatic ontology merging system.
7. COMPARAISON WITH EXISTING ALGORITHMS

Finally, we compare the whole proposed system with the most known ones that exist in the literature such as: CHIMAERA, ONION, PROMPT, FCA-MERGE and GLUE, throw a set of critical properties. Join Table 1 above.

8. CONCLUSION

Ontology merging process is a prominent technique to overcome the restrictions and specifications of information and knowledge when the application covers more than one domain. In this work, we have proposed an architecture for a semi-automatic ontology merging system. In this later, the human intervention is prominent to validate the results of similarity computation module. To identify similar concepts, our algorithm is based on two parallel modules that are both based on linguistic analysis. The former deals with concepts’ names, while the later deals with their properties. Hence, if the first module fails to find similarities, the second one will accomplish the task. The linguistic analysis is based on the combination of a lexico-semantic similarity measure.

After that, the concepts, considered similar by combining the two previous results are handled by the knowledge engineer to validate them. If so, the two concepts under discussion are merged into a single concept. Whereas the concepts considered dissimilar or even similar (by the same combination) but their similarity is rejected by the knowledge engineer are directly copied in the resulting ontology. This provides an ontology that covers a hyper-domain of discourse. Our algorithm is far from complete, several improvements must be completed to make it more efficient.

In future work, we aim to enhance the mapping discovery results by using other information retrieval techniques and elaborate and use a thesaurus of synonymy specific to the application domain, to enhance the results of the semantic similarity measures. Then, we will choose and study an appropriate application domain, on which we will apply our approach.

REFERENCES


