OM fuses Ontologies without Human Intervention

Alma-Delia Cuevas, Adolfo Guzman-Arenas

Centro de Investigación en Computación, Instituto Politécnico Nacional, Mexico City
almadeliacuevas@gmail.com, a.guzman@acm.org

Abstract. A person builds his knowledge from different sources of information. In school he learns that *Juárez was born in Oaxaca* and at home they tell him that the name of the neighbor's dog is *Fido*. In order to know more, he combines information from many sources. But this multi-source information can contain repetitions, references to facts formulated in different way, different level of details, and contradictions. These problems are not easy to solve by computers. Nevertheless, the enormous masses of accumulated knowledge (only in the Web they exist more than billion different pages) demand computer efforts to combine them, since merging manually this information in a consistent way is outside the human capacities. In this paper, a method is explained to combine multi-source information in a manner that is complete, automatic, consistent and robust; contradictions are detected and sometimes solved. The method combines two source ontologies; through iteration, any number can be combined.

Keywords: Language, combination, sources of information, ontology merging, ontology fusion

1. Introduction

Knowledge accumulation is important. A person accumulates knowledge gradually, as it adds concepts to his previous knowledge. Initial knowledge is not zero, even for animals. How can a machine do the same?

1.1 Fusion of new knowledge by computers

Learning occurs by adding new concepts, associating them to the information already learnt. New information can contradict or confuse a human being, or be simply redundant (already known, said with more words) or less accuracte (more vague). A person somehow solves these tasks, and keeps a consistent knowledge base.

This paper is centered in the fusion of ontologies (arising from different sources) between computers. During this fusion the same problems (redundancy, repetition, inconsistency…) arise; the difference is that the machines have no common sense [8] and the challenge is to make them understand that *beneficial* is the same as *generous*, and that *triangle* represents: • *a three-sided polygon*; • *a musical percussion instrument*; or • *a social situation involving three parties.* We want the computer solution to fusion to be very close to people's solution.

Works exist [8, 9, 19, 21] that perform the union of ontologies in a semiautomatic way (requiring user's assistance). Others [5, 12] require ontologies to be organized in formal ways, and to be consistent with each other. In real life, ontologies coming from different sources are not likely to be similarly organized, not to be consistent among them. The automation of fusion needs to solve these problems.

This paper explains a process of union of ontologies in *automatic* and *robust* form. *Automatic* because the computer (unaided) detects and solves the problems appearing during the union, and *robust* because it performs the union in spite of different organization (taxonomies) and when the sources are jointly inconsistent.

The fusion is demonstrated by taking samples of real documents (from the Web), and converting them by hand to ontologies. These are then fed to the computer, which produces a third ontology as result. This result is checked (by hand) with the result obtained by a person. Mistakes are low (§3.2, Table 1).

The problem to solve: To merge two data sources into a result containing its common knowledge, without inconsistencies or contradictions.

OM (Ontology Merging) automatically merges two ontologies into a third one containing the joint knowledge at the sources, without contradictions or redundancies. OM is based in • the theory of Confusion (§2.2); • the use of COM (§2.4), to map a concept into the closest concept of another ontology; • the use of the OM notation (§2.5) to better represent ontologies. These are briefly explained in Section 2, whereas section 3 explains the OM Algorithm, and gives examples of its use.

2. Definitions

2.1 Hierarchy [3]

A hierarchy is a tree where each node is a concept (a symbolic value) or, if it is a set, its descendants must form a partition of it. Example: see Figure 1.

Hierarchies code a taxonomy of related terms, and are used to measure confusion (§2.2), which OM uses for synonym detection and to solve inconsistencies.

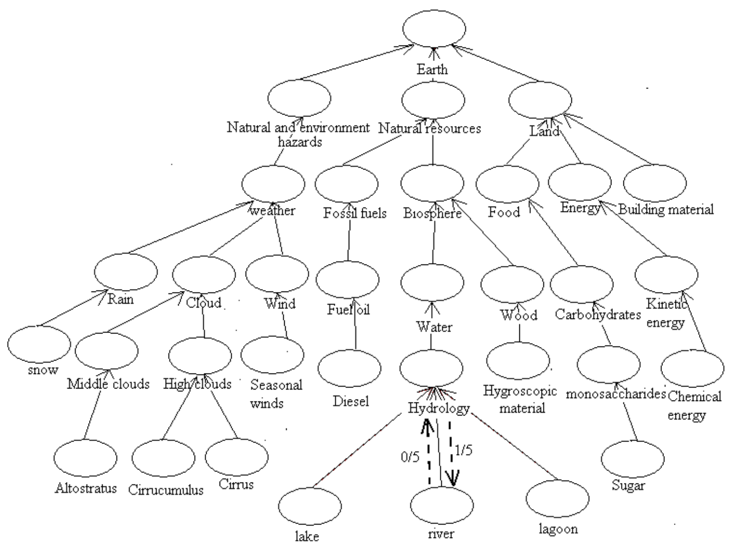


Figure 1. A hierarchy. conf(river, Hydrology)=0 but conf(Hydrology, river) = 0.2

2.2 Confusion [3]

What is the capital of Germany? *Berlin* is the correct answer; *Frankfurt* is a close miss, *Madrid* a fair error, and *sausage* a gross error. What is closer to a *cat,* a *dog* or an *orange*? Can we measure these errors and similarities? Can we systematize or organize these values? Hierarchies (§2.1) of symbolic values allow measuring the similarity between these values, and the error when one is used instead of another. This measurement is accomplished by the theory of confusion [3] and the function conf.

Confusion, contradiction or inconsistency arises when a concept in hierarchy A has a relation that is incompatible, contradicts or negates other relation of the same concept in B. For instance, Earth in A has relation *shape* flat; and in B Earth has the relation *shape* round. Contradiction arises from two relations: in our example, the shapes are not the same, are inconsistent and *shape* can only have a single value.
Because OM must copy concepts keeping the semantics of the sources in the result, and both semantics are incompatible, a contradiction is detected. It is not possible to keep both meanings because they are inconsistent.[1] OM uses confusion to solve this.

Function CONF(r, s), called the *absolute confusion,* computes the confusion that occurs when object r is used instead of object s, as follows:

CONF(r, r) = CONF(r, s) = 0, when s is some ascendant of r;

CONF(r, s) = 1 +CONF(descendant_of (r), s) in other cases.

[1] OM assumes A and B to be well-formed ontologies (each without contradictions). Even then, an inconsistency can arise when considering their joint knowledge.

CONF is the number of *descending* links when one travels from *r* (the used value) to *s* (the intended value), in the hierarchy to which *r* and *s* belong.

Absolute confusion CONF returns a number between 0 and *h*, where *h* is the height of the hierarchy.

Definition.

conf(*r*, *s*), the confusion when using *r* instead of *s*, is:

conf(*r*, *s*) = CONF(*r*, *s*) / *h*

conf returns a number between 0 and 1. *Example:* In Figure 1, conf(Hydrology, river) = 0.5. OM uses conf, whereas [3] describes CONF. Confusion is not a symmetric function: conf (*a, b*) is not the same as conf(*b, a*).

2.3 Ontology [1]

Formally, an ontology is a tuple (C, R) where C is a set of *concepts,* some of which are *relations*; and R is a set of *restrictions* of the form ($r\ c_1\ c_2 \ldots c_k$)) among relation *r* and concepts c_1 through c_k. It is said that the *arity* of *r* is *k*.
Computationally, an ontology is a data structure where information is stored as nodes (representing concepts such as *house*, *computer*, *desk*) and relations (representing restrictions among nodes, such as *cuts*, *transcribes* or *hair color*, as in (*cuts* hammer wood), (*transcribes* printer document) (Figure 2). Usually, the information stored in an ontology is “high level” and it is known as *knowledge.* Notice that relations are also concepts.

2.4 COM [2]

Given two ontologies B and C, COM [2] is an important algorithm that, given a concept $C_C \in C$, finds *cms* = COM(C_C, B), the most similar concept (in B) to C_C.

2.5 OM Notation [1]

OM Notation represents ontologies through a notation like XML. The labels describe the concepts and their restrictions. In OM Notation, • relations are concepts; • relations are n-ary relations; • a particular case of a relation is a *partition.*

2.6 Computer-aided Ontology merging

Initially, merging was accomplished with the help of a user.

Previous solutions to §1.1. Prompt [17], Chimaera [7], OntoMerge [6] and ISI [15] rely on the user to solve the most important problems found in the process, and are considered non automatic mergers. FCA-Merge [10], IF-Map [24] require consistent ontologies that are expressed in a formal notation employed in Formal Concept Analysis [4], which limits their use. CUPID [16], CTX-Match [14] and [13] are notable advancements towards automatic merging, but each one of them focuses on a single aspect of the merging process. [24] uses WordNet and user intervention.

Our solution to §1.1 is the *OM algorithm* (§3), which performs the fusion in a

- ❖ robust (OM forges ahead and does not fall into loops),
- ❖ consistent (without contradictions),
- ❖ complete (the result contains *all* available knowledge from the sources, but it avoids redundancies and detects synonyms, among other tasks) and
- ❖ automatic manner (without user intervention).

2.7 Knowledge support for OM

OM uses some built-in knowledge bases and knowledge resources, which help to detect contradictions, find synonyms, and the like. These are:

1. In the coding, stop words (in, the, for, this, those, it, and, or…) are expunged (ignored) form word phrases;
2. Words that change the meaning of a relation (without, except…) are considered;
3. A number of hierarchies are built-in into OM, in order to ease the computation of confusion;

In the near future,

4. OM can rely on external language sources (WordNet, dictionaries, thesaurus..);
5. OM will use as base knowledge the results of previous merges!

3. OM Algorithm for automatic merging of ontologies [1]

This algorithm fuses two ontologies A and B into a third ontology $C = A \cup B$[2] containing the information in A, plus the information in B not contained in A, without repetitions (redundancies) nor contradictions.

OM proceeds as follows:

1. $C \leftarrow A$. Ontology A is copied into C. Thus, initially, C contains A.
2. Add to each concept $C_C \in C$ additional concepts from B, one layer at a time, contained in or belonging to the restrictions (relations) that C_C has already in C. At the beginning, concept C_C is the root of ontology C. Then, C_C will be each of

[2] Symbol $\cup$ when it referes to ontology merging, it means not only set union, but "careful" merging of concepts, using their semantics.

the descendants of C_C, in turn, so that each node in C will become C_C.[3] For each $C_C \in C$, COM (§2.4) looks in B for the concept that best resembles C_C, such concept is called the *most similar concept* in B to C_C, or *cms.* Two cases exist:

A. If C_C has a most similar concept *cms* $\in$ B, then:
   - i. Relations that are synonyms (§3.1, example 2) are enriched. To enrich a concept in C is to add to its definition the new words that are in the definition of another concept in B, when both are found to be synonyms.[4]
   - ii. New relations (including partitions) that *cms* has in B, are added to C_C. For each added relation, concepts related by that relation and not present in C are copied to C. Example: if (*cms* color `Red`) $\in$ B and concept `Red` is not in C, this concept `Red` is copied to C, together with its ascendants, if they are not in C.
   - iii. Inconsistencies (§2.2) between the relations of C_C and those of *cms* are detected.
     1. If it is possible, by using confusion, to resolve the inconsistency, the correct concepts are added to C. Example: Let (AcmeCorp *incorporated_in* Maryland) $\in$ B and (AcmeCorp *incorporated_in* USA) $\in$ C. Since *incorporated in* has arity 1 (is single-valued), a contradiction is detected and solved because conf(Maryland, USA)=0, thus (AcmeCorp *incorporated_in* USA) changes to (AcmeCorp *incorporated_in* Maryland) in C_C.
     2. When the inconsistency can not be solved, OM rejects the contradicting information in B, and C_C keeps the original relation from A.
   - iv. Concepts that are descendant of *cms* not present in C are copied to C, in a superficial manner.**¡Error! Marcador no definido.**

B. *cms* $\notin$ B. That is, B contains no object *cms* resembling C_C.
   - i. $C_C \leftarrow$ next descendant of C_C (Take the next descendant of C_C).
   - ii. Go back to step 2 until all the nodes of C are visited (including the new nodes that are being added by OM as it works).

3.1 Using OM. Examples

In this section, figures show only relevant parts of ontologies A, B and the result C, because they are too large to fit (for instance, complete A of Figure 3 has 234 nodes).

[3] The ontology C is searched *depth-first*: first, C_C is the root. Then, C_C is the first child of the root, then C_C is the first child of *this* child (a grand son of the root)… Thus, a branch of the tree is traveled until the deepest descendant is reached, before we consider another branch. Since the OM notation uses trees to represent ontologies, OM finds easy to do these travels.

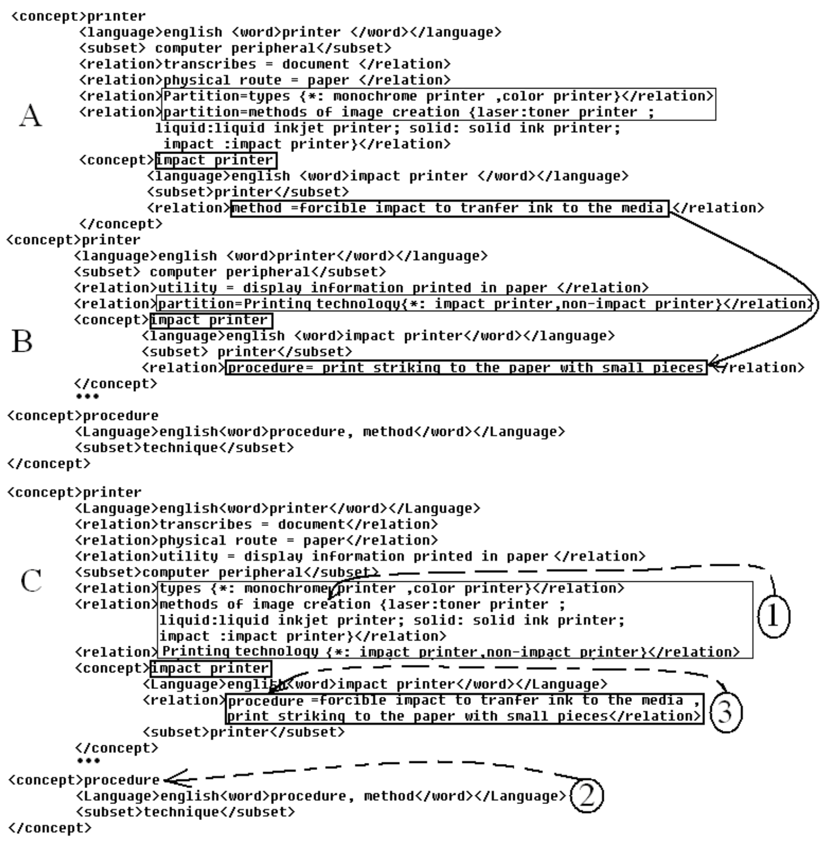
[4] For instance, in A we find `impact printer` *method* (Figure 2) and in B we find `impact printer` *procedure* and C_C = `impact printer` in A has a most similar concept *cms* = `impact printer` in B, and *method* and *procedure* are found by OM to be synonyms, then all the words in the definition of *procedure* in B are copied into the definition of *method* in C.

Example 1. Merging ontologies with inconsistent knowledge. Differences between A and B could be due to: different subjects, names of concepts or relations; repetitions; reference to the same facts but with different words; different level of details (precision, depth of description); different perspectives (people are partitioned in A into male and female, whereas in B they are young or old); and contradictions. Example: Let A contain: veteran *John Nash Sr. Was born in Bluefield,* while B contains: mathematician *John Forbes Nash was born in West Virginia*. Both ontologies duplicate some information (about Nash's place of birth), different expressions (veteran versus mathematician), different level of details (Bluefield versus West Virginia), and contradictions (*John Nash Sr.* vs. *John Forbes Nash*). A person will have in her mind a consistent combination of information: *John Nash Sr.* and *John Forbes Nash* are not the same person, or perhaps they *are* the same, they are synonyms. If she knows them she may deduce that one is the son and the other is the father. We solve these problems everyday, using previously acquired knowledge (§2.7) and *common sense knowledge* [8], which computers lack. Also, they did not have a way to gradually and automatically grow their ontology. OM "measures" the inconsistency (of two apparently contradicting facts) by asking conf to determine the size of the confusion in using Bluefield in place of West Virginia and vice versa, or the confusion of using *John Nash Sr.* instead of *John Forbes Nash.* Small inconsistencies cause C to retain the most specific value, while if it is large, OM keeps C unchanged (ignoring the contradicting fact from B). In case of inconsistency, A prevails.[5]

Example 2. Joining partitions, synonym identification. Figure 3 has the same numbers as the cases below, for easy identification.

1. Copying new partitions. A has two partitions: types and methods of image creation. B has a different partition: printing technology. Thus, printing technology is added to C (result not shown in C of Figure 3 for lack of space).
2. Copying concepts. Concept *procedure* in B is copied to C, since is not in A. Its ascendant (not shown in Figure 4) is also copied to C, if not already there.
3. Changing a simple concept into a full concept. Synonym identification. Adding more semantics to a relation. Relation *method* in A is copied to C; then, *procedure* in B is identified as a synonym of *method,* so the name *method* in C changes to *procedure.* In addition, *procedure* is a concept in B (it was just a phrase, a simple concept in A), so it becomes a (full) concept in C. Then, new semantics is added to *procedure* in C by adding to its definition "print striking to the paper with small pieces" which came from B.

[5] We can consider that an agent's previous knowledge is A, and that such agent is trying to learn ontology B. In case of inconsistency, it is natural for the agent to trust more its previous knowledge, and to disregard inconsistent knowledge in B as "not trustworthy" and therefore not acquired – the agent refuses to learn knowledge that it finds inconsistent, if the inconsistency (measured by conf) is too large.



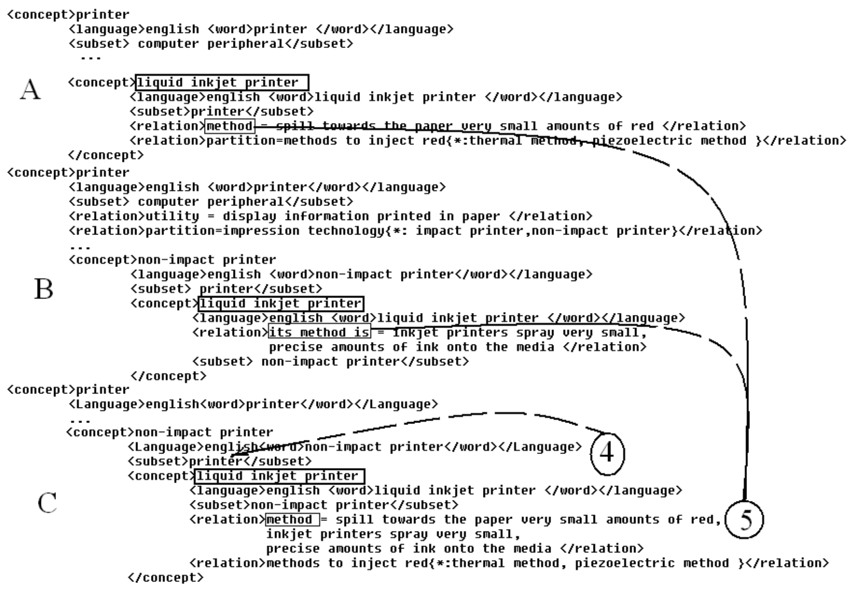
```
<concept>printer
        <language>english <word>printer </word></language>
        <subset> computer peripheral</subset>
        <relation>transcribes = document </relation>
        <relation>physical route = paper </relation>
        <relation>Partition=types {*: monochrome printer ,color printer}</relation>
        <relation>partition=methods of image creation {laser:toner printer ;
                 liquid:liquid inkjet printer; solid: solid ink printer;
                  impact :impact printer}</relation>
        <concept>impact printer
                <language>english <word>impact printer </word></language>
                <subset>printer</subset>
                <relation>method =forcible impact to tranfer ink to the media </relation>
        </concept>
<concept>printer
        <language>english <word>printer</word></language>
        <subset> computer peripheral</subset>
        <relation>utility = display information printed in paper </relation>
        <relation>partition=Printing technology{*: impact printer,non-impact printer}</relation>
        <concept>impact printer
                <language>english <word>impact printer</word></language>
                <subset> printer</subset>
                <relation>procedure= print striking to the paper with small pieces </relation>
        </concept>
        •••
<concept>procedure
        <Language>english<word>procedure, method</word></Language>
        <subset>technique</subset>
</concept>

<concept>printer
        <Language>english<word>printer</word></Language>
        <relation>transcribes = document</relation>
        <relation>physical route = paper</relation>
        <relation>utility = display information printed in paper </relation>
        <subset>computer peripheral</subset>
        <relation>types {*: monochrome printer ,color printer}</relation>
        <relation>methods of image creation {laser:toner printer ;
                  liquid:liquid inkjet printer; solid: solid ink printer;
                  impact :impact printer}</relation>
        <relation> Printing technology {*: impact printer,non-impact printer}</relation>
        <concept>impact printer
                <Language>english<word>impact printer</word></Language>
                <relation>procedure =forcible impact to tranfer ink to the media ,
                          print striking to the paper with small pieces</relation>
                <subset>printer</subset>
        </concept>
        •••
<concept>procedure
        <Language>english<word>procedure, method</word></Language>
        <subset>technique</subset>
</concept>
```

Figure 2. Relations *method* in A and *procedure* in B are synonyms, thus both definitions (of *method* and of *procedure*) are added to C

Example 3. Removing redundant relations and comparing relations. Figure 3 has the same numbers as the cases below, for easy identification.

4. Removing redundant relations. liquid inkjet printer in A is the descendant of printer, whereas in B it is the descendant of non-impact printer, which is descendant of printer. Adding both relations (subset) to liquid inkjet printer in C provokes copying non-impact printer to C (which was not in C). As a result, liquid inkjet printer would have two ascendants: printer and non-impact printer. Since non-impact printer is a descendant of printer, the relation liquid inkjet printer subset of printer is redundant. OM expunges this to keep only liquid inkjet printer subset of non-impact printer, and non-impact printer subset of printer. Final result is C in Figure 3.
5. Comparing relations. The relations *method* in B and *its method is* in A are considered to be the same, because (§2.7) connectors *and*, *or*, *its*… are ignored.



```
<concept>printer
        <language>english <word>printer </word></language>
        <subset> computer peripheral</subset>
          ...
        <concept>liquid inkjet printer
                <language>english <word>liquid inkjet printer </word></language>
                <subset>printer</subset>
                <relation>method = spill towards the paper very small amounts of red </relation>
                <relation>partition=methods to inject red{*:thermal method, piezoelectric method }</relation>
        </concept>
<concept>printer
        <language>english <word>printer</word></language>
        <subset> computer peripheral</subset>
        <relation>utility = display information printed in paper </relation>
        <relation>partition=impression technology{*: impact printer,non-impact printer}</relation>
        ...
        <concept>non-impact printer
                <language>english <word>non-impact printer</word></language>
                <subset> printer</subset>
                <concept>liquid inkjet printer
                        <language>english <word>liquid inkjet printer </word></language>
                        <relation>its method is = inkjet printers spray very small,
                                  precise amounts of ink onto the media </relation>
                        <subset> non-impact printer</subset>
                </concept>
<concept>printer
        <Language>english<word>printer</word></Language>
        ...
        <concept>non-impact printer
                <Language>english<word>non-impact printer</word></Language>
                <subset>printer</subset>
                <concept>liquid inkjet printer
                        <language>english <word>liquid inkjet printer </word></language>
                        <subset>non-impact printer</subset>
                        <relation>method = spill towards the paper very small amounts of red,
                                  inkjet printers spray very small,
                                  precise amounts of ink onto the media </relation>
                        <relation>methods to inject red{*:thermal method, piezoelectric method }</relation>
                </concept>
```

Figure 3. Relations *method* in A and *its method is* in B are the same, so they are merged in a single relation *method* in C (label 5)

3.2 Additional examples for real cases

OM has merged ontologies derived from real documents. The ontologies were obtained manually from several documents [18-22] describing, say, the same animal. The obtained ontologies were merged (automatically) by OM. Validation of results has been made manually, obtaining good results (table 1).

Table 1. Performance of OM in some real examples: C = A ∪ B. First column gives the number of nodes (112 in Turtles) and relations (12) in A and in B (72 nodes, 21 relations)

Ontologies A y B	**Relations in C**	**Nodes in C**	**error**
Turtles [22]. 112, 12, 72, 21.	All relations were copied	All nodes were copied	0
Hammer [21]. 112, 12, 72, 21.			0
Poppy [20]. 112, 12, 72, 21.			0
100 Years of Loneliness [18]. 112, 12, 72, 21.	12 out of 467 were not copied	8 out of 141 were not copied	0.056
Oaxaca [19]. 112, 12, 72, 21.	All relations were copied	1 out of 309 was not copied	0.003

3.3 Conclusions

The paper presents an automatic, robust algorithm that fuses two ontologies into a third one, while keeping the knowledge in the sources. It solves some inconsistencies and avoids adding redundancies to the result.
The examples shown, as well as others in [1], provide evidence that OM does a good job, in spite of joining very general or very specific ontologies. This is because the algorithm takes into account not only the words in the definition of each concept, but its semantics [context, synonyms, resemblance (through conf) to other concepts…] too. In addition, its base knowledge (§2.7) helps.

References

1. Cuevas-Rasgado, A. Merging of ontologies using semantic properties Ph. D. thesis in progress. CIC-IPN, Mexico.
2. Guzman, A., and Olivares, J. Finding the Most Similar Concepts in two Different Ontologies. *Lecture Notes in Artificial Intelligence* 2972, Springer-Verlag. 129-138. 2004
3. Levachkine, S., and Guzman, A. Hierarchy as a new data type for qualitative values. *Journal Expert Systems with Applications* **32**(3), June 2007.
4. Bemhard Ganter, Gerd Stumme, Rudolf Wille. *Formal concept analysis: Foundations and applications.* 1st ed. A Ed. New York, NY: Springer.
5. Connolly, D., van Harmelen, F., Horrocks, I, *et al*. DAML+OIL Reference description. March 2001. W3C Note 18. 2001. http://www.w3.org/TR/2001/NOTE-daml+oil-reference-20011218
6. Dou D., McDermott, D., and Qi. P. Ontology Translation by Ontology Merging and Automated Reasoning. *Proc. EKAW Workshop on Ontologies for Multi-Agent Systems*. 2002
7. Deborah L. McGuinness, Fikes, R., Rice, J., and Wilder, S. The Chimaera Ontology Environment Knowledge. *Proceedings of the Eighth International Conference on Conceptual Structures Logical, Linguistic, and Computational Issues*. Darmstadt, Germany. 2000.
8. Lenat, D., and Guha, V. *Building Large Knowledge-Based Systems*. (Addison-Wesley 1989).
9. F. Manola, E. Miller. RDF Primer. W3C Recommendation. 10 February 2004. http://www.w3.org/TR/2004/REC-rdf-primer-20040210/
10. Gerd Stumme, Alexander Maedche. Ontology Merging for Federated ontologies on the semantic web. In: E. Franconi, K. Barker, D. Calvanese (Eds.): *Proc. Intl. Workshop on Foundations of Models for Information Integration*, Viterbo, Italy, 2001. LNAI, Springer 2002 (in press).

12. Knowledge Interchange Format. Draft proposed. American National Standard [dp ANS] NCITS.T2/98-004. http://logic.stanford.edu/kif/dpans.html
13 Kotis K., Vouros G. and Stergiou, K. Towards Automatic of Domain Ontologies: The HCONE-merge approach. *Journal of Web Semantics* (JWS), Elsevier, vol. **4**:1, pp 60-79, 2006. In ScienceDirect: http://authors.elsevier.com/sd/article/S1570826805000259.
14. L. Serafini, P. Bouquet, B. Magnini, and S. Zanobini. An Algorithm for Matching Contextualized Schemas vio SAT. In Proceedings of CONTEXT 03. 2003
15. Large Resources. Ontologies (SENSUS) and Lexicons: http://www.isi.edu/natural-language/projects/ONTOLOGIES.html
16. Madhavan, J., Bernstein, P., Rahm, E., 2001. Generic schema matching using Cupid. In: 27th International Conf. On Very Large Data Bases (VLDB'01). Rome, Italy.

17. Noy, N., and Musen, M. PROMPT: Algoritm and Tool for Automated Ontology Merging and Alignment. Stanford Medical Informatics, Stanford University, CA 94305-5479, {noy, musen}@smi.stanford.edu In Proc. Of the National Conference on Artificial Intelligence, 2000.
18.Ontologies about Cien Años de Soledad: http://html.rincondelvago.com/cien-anos-de-soledad_gabriel-garcia-marquez_22.html and http://www.monografias.com/trabajos10/ ci-so/ciso.shtml
19.Ontologies about Oaxaca: http: www.oaxaca-mio.com/atrac_turisticos/infooaxaca.htm and http: www.elbalero.gob.mx/explora/html/oaxaca/geografia.html
20.Ontologies about poppy: http://es.wikipedia.org/wiki/Amapola and http://www.buscajalisco.com/bj/salud/herbolaria.php?id=1
21.Ontologies about tools and products: http://sumesa.com/
22.Ontologies about turtles: www.damisela.com/zoo/rep/tortugas/index.htm and http://www.foyel.com/cartillas/37/tortugas_-_accesorios_para_acuarios_i.html
24. Kalfoglou, Y., and Schorlemmer, M. Information-Flow-based Ontology Mapping. Proceedings of the 1st International Conference on Ontologies, Databases and Applicatio of Semantics (ODBASE'02), Irvine, CA, USA. 2002.