Local semantic context analysis for automatic ontology matching

Giuseppe Fenza, Vincenzo Loia and Sabrina Senatore

Dipartimento di Matematica e Informatica,
Università degli Studi di Salerno,
via ponte don Melillo, 84084 - Fisciano (SA), Italy
E-mails: {gfenza, loia, ssenatore}@unisa.it.

Abstract—*This paper proposes an algorithm for concept matching, applied in the ontology mapping domain. Basic idea is to seek the effective semantics embedded in the concept name by analyzing the context in which it appears. Through simple interactions with the known lexicon WordNet, the right meaning associated to a concept is unequivocally elicited by exploring their local semantic context, viz. the surrounding concepts.*

This approach reveals interesting results for the word sense disambiguation, when polysemy problems requires a semantic interpretation.

Keywords— Semantic Web, concept matching, ontology mapping, WordNet, concept similarity.

1 Introduction

The Web comprises huge and disparate collections of information, whose size is estimated to overcome 11.5 billion web pages [1], though the search engines do not seem to cover the whole size (for instance, Google indexes approximately 9 billion pages). The diffusion of the Semantic Web has promised new models to support integrated access to web resources and services as well as intelligent applications for information processing on the Web.

Ontologies represents a conceivable solution for data representation as well as the knowledge sharing, aimed at the integration of the Web content in a unique and coherent view. Nevertheless, due to the decentralized nature of the Web, a plethora of ontologies has been defined and disseminated on the net; often they describe overlapped application domains; sometimes are specialized for specific domain.

It is evident the exigency to find some semantic correspondence among concepts which refer to different ontologies in order to get a "semantic reconciliation", aimed at establishing interoperability between semantic Web applications and a more homogeneous integration of information [22].

Ontology mapping has been proposed as an effective way of handling the semantic heterogeneity problem. It plays a central role in many application domains, such as e-commerce, semantic web services matchmaking [22], information integration, query mediation [23], etc..

It is the process that discovers a set of semantic correspondence between some entities of different ontologies. Many research studies have yield approaches and tools for (semi) automatic ontology matching [2], [3]; structural and linguistic matching has been taken into account; often combined approaches for the matching valuation provide efficient results too.

This paper presents a simple approach to concept matching, based on the linguistic similarity. The main idea is to discriminate the effective meaning of a word by analyzing the context in which it appears, in order to overcome the polysemy problems and the lexical ambiguity.

The approach achieves a primitive ontology mapping, by discovering semantic correspondences between concepts of two ontologies (implemented as graph-like structures). A graphical interface presents the discovered mapping.

The paper is organized as follows. Section 2 sketches a state of art in the ontology matching domain. Section 3 introduces our approach, describing the basic algorithm and then in Section 4, implementation details and some relevant case studies are presented in order to validate its applicability. Finally, conclusions close the paper.

2 Related works

In the last years, ontologies are increasingly being used to support the integration of information. Yet, their diffusion in many Web areas emphasizes impressive heterogeneity among information sources and in particular in the formal model exploited to encode the domain conceptualizations.

In literature, different types of heterogeneity have been identified, mainly split in the following classes.

- *Syntactic heterogeneity*: represents the heterogeneity due to differences at the language level; for instance, when two ontologies are defined by using different knowledge representation formalisms even though the meaning is the same [4].
- *Terminological heterogeneity*: occurs when different names are given to similar ontology entities [6].
- *Semantic heterogeneity*: is often called conceptual heterogeneity. It occurs when the same domain of interest is modelled in different way, for instance, exploiting different types of axioms for defining concepts or just giving diverse expressive values to them [5].

Many efforts have been done from the research communities to find a solution to the matching problem by developing a variety of tools and mostly providing a well-founded formalization [7, 8, 9]. Matching ontologies (or schemas) has been recognized as a critical operation in many

ISBN: 978-989-95079-6-8

application domains. It takes as input two ontologies, each one consisting of a set of entities and produces as output the "connections" (which may represent equivalences or just relations such as consequence, subsumption, or disjointness) holding between these entities.
Systems for ontology matching such as Cupid [11], OntoBuilder [10], Similarity Flooding [2], Clio [12], Glue [13], S-Match [3], OLA [14] and Prompt [15] aim at providing an alignment, namely a set of correspondences between semantically related entities of different ontologies. Finding the correspondences which are effective from the user's viewpoint is often not easy.
In [17] a tree-based classification of elementary matching approaches is given. At granularity level, matching techniques can be distinguished into the following classification criteria:
– Element-level vs. structure-level: Element-level matching techniques discover correspondences by analyzing entities or their relative instances, without considering their relations with other entities or their instances. Structure-level techniques instead compute correspondences by studying the structure in which the concepts or their instances appear [16].
– Syntactic vs. external vs. semantic: The syntactic techniques consider the input with regard to its sole structure, according to some well stated algorithm. On the other hand, external techniques exploit external resources (such as human support, thesaurus, etc.) and common knowledge in order to interpret the input [17]. Finally, semantic techniques use some formal semantics (i.e. model-theoretic approaches) to interpret the input and justify their results. Semantic matching [3] is based on the idea that labels at concept nodes, which are written in natural language, are translated into propositional formulas which codify the intended meaning of the labels themselves. This way the matching problem becomes a propositional unsatisfiability problem, which can then be efficiently implemented through propositional satisfiability (SAT) solvers [18].
Although many advances in the ontology matching research topic have been addressed, nevertheless, there are still open issues to investigate in order to get a more efficient and effective integration of ontology matching tools in the web applications domain.

3 Ontology Matching

The ontology matching problem can be seen as a problem of concept matching. Two concepts match, if they are similar. In this section some formal notions are given to delineate a common background, useful to describe our ontology matching approach. Then the ontology matching method is detailed.

3.1 Scope of a concept into an ontology

Concept matching requires the evaluation of the similarity between two concepts. This approach aims at discovering linguistic similarities between the involved entities. In general, linguistic similarities are based on morphology and semantics, which are associated to the words that describe the relative entities. Thus, in this approach the similarity between two entities of different ontologies is evaluated not only by investigating the semantics of the entities names, but also taking into account the local context, through which the effective meaning is described.
Often the same word placed in different textual contexts assumes completely different meanings. In addition, lexicons are not able to disambiguate situations in which homonyms occur.
In order to deal with lexical ambiguity, this approach introduces the notion of "scope" of a concept which represents the context where the concept is placed.

Definition 1: *Let O be ontology and c is a concept in O. The scope of c, with radius r,* ***scope(c, r)*** *is a set of all the concepts outgoing from c included in a path of length r, with centre c. More formally:*

$$scope(c,r) = \{c' |\, c' \in O\ , dist(c,c') < r\}$$

where $dist(c,c')$ *is the number of edges that are in the path from concept c to concept c'. Let us note that* $dist(c,c') = 0\ iff\ c = c'$.

Figure 1 sketches the idea. The scope defines a round area composed of all concepts that are connected directly or indirectly to the central node. This area represents the context. Increasing the radius means to enlarge the scope (i.e. this area) and, consequently, the set of neighbour concepts that intervene in the description of the context.
Fixed the radius of the scope of two concepts α and β, belonging to different ontologies, our goal is to find out a semantic relationship exist between them.

Definition 2. Let *α be a concept in the ontology O. The concept name of α,* ***a= label(α)*** *is the linguistic label associated to the concept α.*

Definition 3. Given two concepts *α , β belonging respectively to* ontologies *O and O', let* $\mathcal{L}$ *be a lexicon and a= label(α) and b= label(β). Then, let lex (a, b) ∈ [0, 1] be a lexical similarity associated to the pair of concept names (α, β), with α∈O , β∈O'. The set L is composed of all pairs, defined as follows:*

$$L = \{(\alpha', \beta') | \forall\ \alpha \in scope(\alpha', r)\ and\ \forall\ \beta \in scope(\beta', r)\ and\ \exists lex(a, b) \neq 0)\}$$

WordNet is the lexicon exploited in this approach. It is one of the most common lexical databases, used for research studies in computational linguistics, text analysis, etc. The function *lex(a, b)* is the similarity metric.
Many similarity functions have been implemented in the WordNet package. In particular, a measure specifically developed for WordNet is the similarity proposed by Wu and Palmer [19]. It is simple to compute and results an efficacy metric for ontology matching.

ISBN: 978-989-95079-6-8

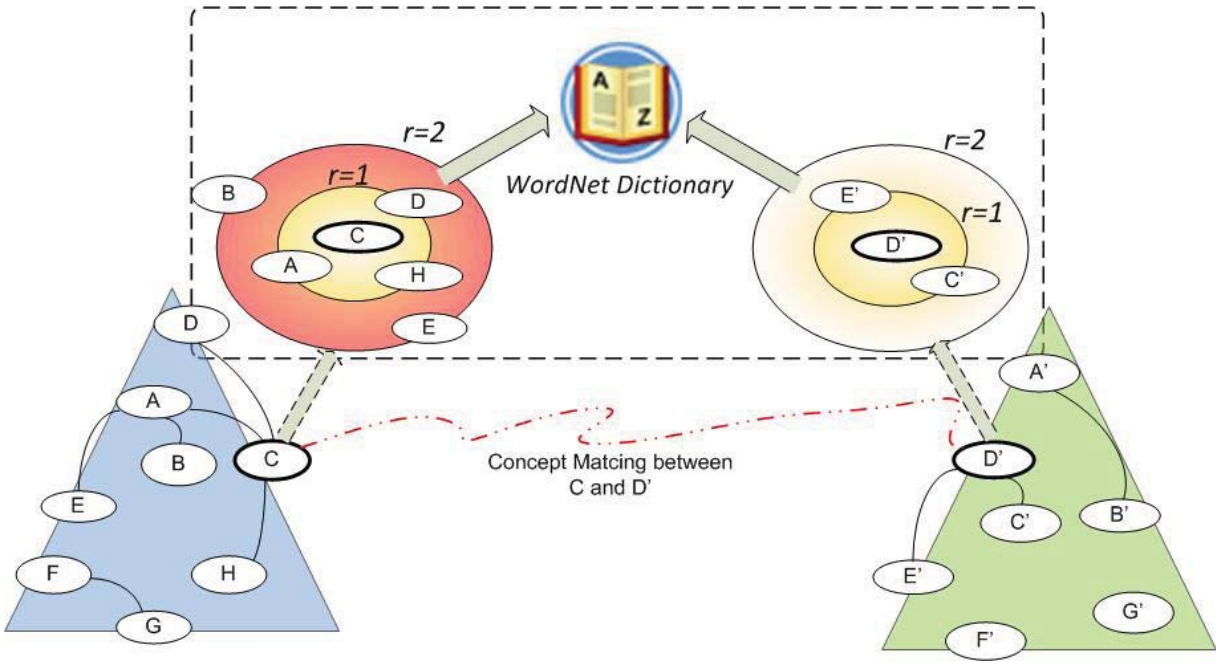

Figure 1: Intuitive idea of concept matching, exploiting the *scope* and the effective *sense* of the concept names

The Wu-Palmer similarity is based on the observation that although two concepts near the root of a hierarchy are close to each other in terms of edges, they can be very different semantically; conversely, two classes under one concept that are distant as number of edges, could be more similar conceptually. The idea is to find a common closest hyperonym in the path from them to the root.
The principle of similarity computation between two nodes is based on the distance which separates these two nodes from the root node and the distance which separates their closest common ancestor from the root too.
Let us assume that the arcs in ontology represent uniform distances (i.e. all the semantic edges have the same weight).

Definition 4. *Let c_1 and c_2 be two ontology elements in the ontology O. The distance between two nodes $\delta(c_1, c_2)$ is represented by the minimum number of edges that connect them.*

Definition 5. *Given an ontology O formed by a set of nodes and a root node R. Let c_1 and c_2 be two ontology concepts of which we will calculate the similarity. Then, g is the common ancestor of c and c'. The Wu–Palmer similarity is defined by the following expression[19]:*

$$sim(c_1, c_2) = \frac{2 \times \delta(g, R)}{\delta(c_1, R) + \delta(c_2, R)}$$

3.2 Concept Sense Discrimination

Through the given definitions, it is possible to individuate the meaning of a name associated to an ontology concept. Given a word, WordNet provides a list of all the synsets and word senses, related to that word. Just to give an example, submitting the word "table" to WordNet, the result is the following list.

```
The noun table has 6 sense (first 3 form tagged
texts)

1. (57) table, tabular array -- (a set of data
   arranged in rows and columns; "see table 1")
2. (25) table -- (a piece of furniture having a
   smooth flat top that is usually supported by
   one or more vertical legs; "it was a sturdy
   table")
3. (5) table -- (a piece of furniture with
   tableware for a meal laid out on it; "I
   reserved a table at my favorite restaurant")
4. mesa, table -- (flat tableland with steep
   edges; "the tribe was relatively safe on the
   mesa but they had to descend into the valley
   for water")
5. table -- (a company of people assembled at a
   table for a meal or game; "he entertained the
   whole table with his witty remarks")
6. board, table -- (food or meals in general;
   "she sets a fine table"; "room and board")
```

Let us note all the senses are ranked with respect to the frequency of the term (shown in the parenthesis for the first three senses) in the reference context (according to the spoken common sense).
The following pseudo-code details the algorithm for discriminate the actual sense of a word associated to a given concept. The algorithm takes as input an ontology O, a reference concept α in that ontology and a word *w*. The word *w* represents the name associated to the concept α (i.e. w=label(α)).
Let us note the semantic difference between the concept (or class) in an ontology and the label associated to that concept. The algorithm replies to question like "I would like to know the effective sense of the word w, placed in the context (or scope) of the concept α".

ISBN: 978-989-95079-6-8

Algorithm1

```
Input: Ontology O, concept α ∈ O, radius r and the
word w

Output: sense number of w

1: build an array N[|synset(w)|];
2: for each t1 in synset(w)
3:    initialize N[senseNumber(t1)] = 0;
4:    for i = 1 to r
5:       set M = ∅;
6:       for each c in scope(α,i) - scope(α,i-1)
7:          set S = ∅;  //set of similarity values
8:          for each t2 in synset(label(c))
9:             sim = similarity_WP(t1, t2)
10:            S = S U {sim};
11:         end for each
12:
13:         set max = maximum in S;
14:         max = max / |scope(α, i)|;
15:         M = M U {max};
16:      end for each
17:      set Sum = summation of all the values in M
18:      Sum = Sum /i;
19:      N[senseNumber(t1)] += Sum;
20:   end for
21: end for each
22: i1 = sense number with the highest
                              frequency in N.
23: return i1
```

First step (line 1) is to declare a vector structure whose size corresponds to the number of synsets (or senses) associated to the given word w. Goal is to maintain in each cell of the vector a pertinence value that represents how much the word w is semantically related to that sense (or belongs to that synset).
The algorithm selects all the concepts in the scope of α (belonging to the reference ontology O) by varying the radius (lines 4-6), in order to get different set of terms. Then compute the Wu-Palmer similarity between two terms coming from the concept name of α and the word *w* (line 9).
Just to give an idea about how the algorithm works, let us look at Figure 2. For simplicity, let us suppose *w* = *label(α)* = "Table". The algorithm evaluates the similarity between all the terms coming from both the WordNet synset of "Table" and the scope of concept "Table" in the ontology.
In particular, Figure 2 shows the flat ring shaped areas in different colours, by varying the radius.
For each concept *c* in the ring shaped area (computed as the difference of the areas between two successive radii, see line 6), the max similarity values between the name associated to *c* and a concept name in synset of α are maintained in M (line 15).
At the end of the two loops (lines 6-11) the variable *Sum* contains the sum of all the max similarity values computed for each couple of terms coming from the fixed term t1 of synsets of α and all the terms in the synset of *w*.
The *Sum* is "weighted" with respect to the current radius (line 18). The final value of *Sum* is added to the value stored in the cell associated to the sense of some term t1. This is repeated for each t1 in the synset of *w*. At the end, in each cell of the vector N there is a value, associated to each term in the synset of *w*.

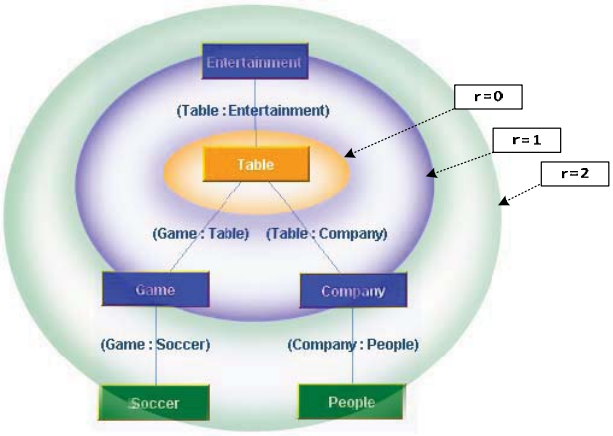

Figure 2. A sketched ontology and the scope of concept "Table" at different radius r.

The index of the vector cell, whose value is the maximum, represents the sense number to give as output.
Applying the algorithm on the ontology of Figure 2, the retuned index value is 5.
Abstractly speaking, the approach helps the user to individuate the right meaning of a word, given the context.

3.3 Ontology mapping

As said, the ontology mapping outlines correspondences or matches between concepts coming from two different ontology. In this approach, the concept matching is measured by computing a similarity between concepts at linguistic level.
The algorithm described in the previous section has been exploited to evaluate the concept matching. More formally, given two ontologies *O* e *O'* and two concepts *c* and *c'*, belonging respectively to these two ontologies, there is a match between *c* and *c'* if a similarity between them exists, computed as follows:

1. Algorithm1 is invocated twice: one time it takes as input the concept *c* and the ontology *O* and another time, by giving the concept *c'* and the ontology *O'*. Outputs of these two independent executions of Algorithm1 are two indexes, *i* for the synset of *c* and *i'* for the synset of c'. As said, they identify the sense number associated to each concept.

2. Once discovered the sense of involved concepts, the affinity between the concepts is computed by a similarity measure (for instance, the Wu-Palmer similarity defined in Section 3.1) between *c* and *c'*.

In order to obtain all the semantic correspondences among the concepts in the two ontologies, this procedure can be applied for each couple of concepts coming from two reference ontologies. Final result is an ontology mapping; a similarity value is assigned to each discovered correspondence between concepts in the two ontologies.

ISBN: 978-989-95079-6-8

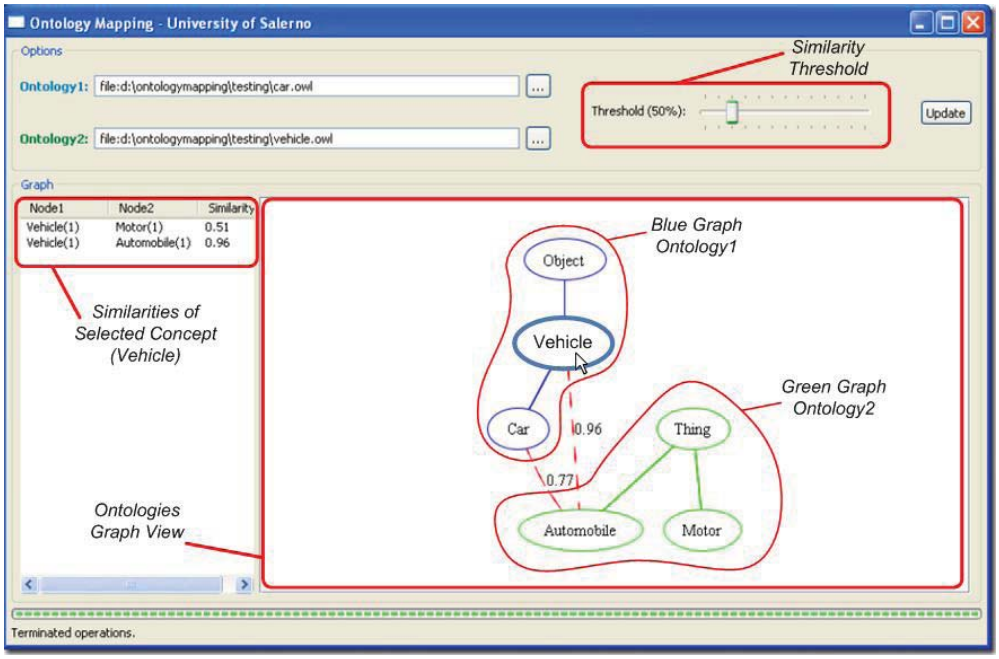

Figure 3. Ontology Mapping GUI

4 User interface and application details

The algorithm has been implemented in Java language. It exploits Java WordNet Similarity Library (JWSL) [20] for accessing the WordNet database. JSWL implements the most commons similarity and relatedness measures between words and get information about *synsets*.

A simple graphical interface has been designed for the user interaction (see Figure 3). It has been implemented by exploiting Graphviz library [21]. It is an open source graph visualization software which provides several main graph layout programs.

Let us note the ontologies have been represented (and implemented) as graphs, whose nodes represent the concepts whereas the edges connects two nodes when a relationship exists between them. In this version, a restriction on just hierarchies or taxonomies (i.e. only IS-A relationships) is considered.

The user can load the two ontologies and the final mapping is drafted in the *Ontology Graph View* panel (see Figure 3). The correspondences discovered by the mapping are sketched in Figure 3 as dotted lines between the concepts. A weight (value of similarity) is associated to each arc.

Figure 3 shows on the right hand of the interface, a sliding bar for setting the similarity threshold. Moving the cursor, the threshold is modified, returning in the graph view panel just the arcs whose similarity values are greater than this threshold.

Moreover, when the user clicks on a concept (node) in the graph view panel, all its connections with other concepts and the relative similarity values are shown on the left of the interface.

4.1 Some case studies

The algorithm has been applied on some sketched ontologies. The application acquires the ontologies in OWL format, although this prototypal version is limited to work on hierarchical structure.

In the interface of Figure 3, a simple example of ontology mapping is shown. The case study has been built ad-hoc; just the relevant portions of two ontologies are shown.

The Ontology1 (in blue color in Figure 3) is described by the concepts named “Object”, “Vehicle” and “Car” , whereas the Ontology2 (green color) is composed of the concept with names “Thing”, “Automobile” and “Motor”.

The application adds an arc between two concepts of different ontologies, when a similarity value (in the range [0, 1]) is found.

Let us note there exists an arc between the concept names “Vehicle” and “Automobile” with an high similarity (0.96), but the arc between “Vehicle” and “Motor” is not inserted in the graph view, because the similarity between these concept names assumes values lower than the selected threshold.

Similar considerations can be done on the ontology mapping shown in Figure 4 (although the GUI is not shown).

Two ontologies are drafted in different colors. Both of them describe a similar domain (the living beings).

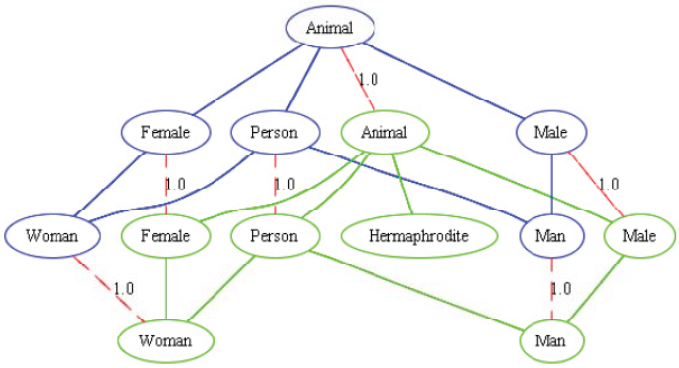

Figure 4: Example of Ontology Matching on the human beings domain.

ISBN: 978-989-95079-6-8

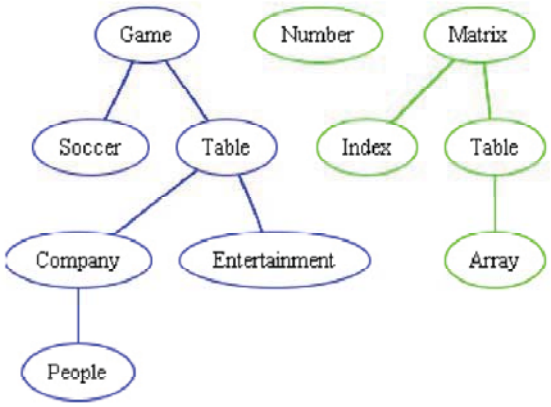

Figure 5: No concept matching occurs between these two ontologies, due to different meanings of the reference ontology domains.

Because the ontologies present concepts with similar names and meanings, the similarity values on the added arcs (in red) are equal to 1 (the maximum similarity).
Finally, Figure 5 shows a case of no concept match. The two ontologies share the same concept name "Table", but the reference context is completely different. In fact, Algorithm1 returns the sense number 5 for the ontology on the left (colored blue) and the sense number 1 for the ontology on the right, respectively (see Section 3.2 for details). Thus no arcs can be placed between them.
Let us note, in the example in Figure 5, the node with concept name "Number" is disconnected by the others, because the algorithm does not analyzes the OWL ontology relationships that are not hierarchical.

5 Conclusions

The paper describes a study for ontology mapping based on the discovering of linguistic similarity. A graphical interface allows the user to see the final mapping, presenting a similarity value for each discovered concept matching.
This algorithm represents an initial version of an ontology matching method and needs some extension for processing all the ontology relationships. Its efficacy would be visible if it was exploited in combination with structure based ontology matching. Further developments are addressed to reach these issues. Nevertheless, its applicability may not restricted to ontology matching problems. The basic algorithm could be used for Information Retrieval problems, for instance when the textual analysis requires the word sense disambiguation. The context of a word could be described by defining a "sliding window" on the text surrounding the analyzed word, in order to discover its appropriate sense. Furthermore, the algorithm could be exploited for discriminating common sense words, in specialized contexts where the word meaning depends on the application domain (or the reference ontology).

References

[1] Gulli, A. and Signorini, A. 2005. The indexable web is more than 11.5 billion pages. In *Special interest Tracks and Posters of the 14th international Conference on World Wide Web* (Chiba, Japan, May 10 - 14, 2005). WWW '05. ACM, New York, NY, 902-903.

[2] S. Melnik, H. G. Molina, E. Rahm, Similarity Flooding: A Versatile Graph Matching Algorithm and Its Application to Schema Matching, *In Proceedings of the 18th International Conference on Data Engineering*, San Jose, CA, 2002, pp.117-128.

[3] F. Giunchiglia, M. Yatskevich, E. Giuchilglia, Efficient Semantic Matching, *In Proceedings of ESWC, Heraklion*, Greece, 2005, pp.272-289.

[4] J. Euzenat and H. Stuckenschmidt. The 'family of languages' approach to semantic interoperability. In Borys Omelayenko and Michel Klein, editors, Knowledge transformation for the semantic web, pages 49–63. IOS press,Amsterdam (NL), 2003.

[5] M. Klein. Combining and relating ontologies: an analysis of problems and solutions. In Proc. IJCAI Workshop on Ontologies and Information Sharing, Seattle (WA US), 2001.

[6] N. Noy and M. Klein. Ontology evolution: Not the same as schema evolution. Knowledge and Information Systems, 6(4):428–440, 2004.

[7] P. Bernstein, A. Halevy, and R. Pottinger. A vision of management of complex models. *ACM SIGMOD Record*, 29(4):55–63, 2000

[8] P. Bouquet, M. Ehrig, J. Euzenat, E. Franconi, P. Hitzler, M. Krotzsch, L. Serafini, G. Stamou, Y. Sure, and S. Tessaris. Specification of a common framework for characterizing alignment. Deliverable D2.2.1, Knowledge web NoE, 2004

[9] A. Zimmermann, M. Krotzsch, J. Euzenat, and P. Hitzler. Formalizing ontology alignment and its operations with category theory. In Proc. 4th International Conference on Formal Ontology in Information Systems (FOIS),pages 277–288, Baltimore (MD US), 2006

[10] A. Gal, G. Modica, H. Jamil, and A. Eyal. Automatic ontology matching using application semantics. AI Magazine, 26(1):21{32, 2005.

[11] J. Madhavan, P. Bernstein, and E. Rahm. Generic schema matching with Cupid. In Proceedings of the 27th International Conference on Very Large Data Bases (VLDB), pages 48-58, Rome, Italy, 2001.

[12] R. Miller, M. Hernandez, L. Haas, L.-L. Yan, C. Ho, R. Fagin, and L. Popa. The Clio project: Managing heterogeneity. SIGMOD Record, 30(1):78-83,2001.

[13] A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Learning to map between ontologies on the semantic web. In Proceedings of the 11th International Conference on World Wide Web (WWW), pages 662{673, Honolulu, USA, 2002.

[14] J. Euzenat and P. Valtchev. Similarity-based ontology alignment in OWLlite. In Proceedings of the 15th European Conference on Artificial Intelligence (ECAI), pages 333-337, Valencia, Spain, 2004.

[15] N. Noy and M. Musen. The PROMPT suite: interactive tools for ontology merging and mapping. International Journal of Human-Computer Studies, 59(6):983-1024, 2003.

[16] J. Kang and J. Naughton. On schema matching with opaque column names and data values. In Proc. 22nd International Conference on Management *of Data (SIGMOD)*, pages 205–216, San Diego (CA US), 2003.

[17] P. Shvaiko and J. Euzenat. A survey of schema-based matching approaches. Journal on Data Semantics, IV:146–171, 2005.

[18] D. Le Berre JSAT: the java satisfiability library http://www.sat4j.org/

[19] Z. Wu and M. Palmer. Verb semantics and lexical selection. In Proc. 32nd Annual Meeting of the Association for Computational Linguistics (ACL), pages 133–138, Las Cruces (NM US), 1994.

[20] Java WordNet Similarity Library (JWSL) Available: http://grid.deis.unical.it/similarity/

[21] Graphviz - Graph Visualization. Available: http://www.graphviz.org/

[22] G. Fenza, V. Loia, S. Senatore, A hybrid approach to semantic web services matchmaking, International Journal of Approximate Reasoning, Volume 48, Issue 3, August 2008, pp. 808-828.

[23] G. Fenza, V. Loia, S. Senatore: *Improving Fuzzy Service Matchmaking through Concept Matching* Discovery. FUZZ-IEEE 2007: pp. 1-6

ISBN: 978-989-95079-6-8