An Ontology-Based Similarity between Sets of Concepts

Valentina Cordì, Paolo Lombardi, Maurizio Martelli and Viviana Mascardi
Dipartimento di Informatica e Scienze dell'Informazione – DISI,
Università di Genova, Via Dodecaneso 35, 16146, Genova, Italy.
Email: cordi@disi.unige.it, 2001s003@educ.disi.unige.it, martelli@disi.unige.it, mascardi@disi.unige.it

***Abstract*— To help sharing knowledge in those contexts where documents and services are annotated with semantic information, such as the Semantic Web, defining and implementing the similarity between sets of concepts belonging to a common ontology may prove very useful. In fact, if both the required and the provided pieces of information (be they textual documents, services, images, or whatever) are annotated with sets of concepts taken from a reference ontology O, the evaluation of how good a piece of information P is, w.r.t. the required one R, may be based on the similarity between the two sets of concepts that describe P and R.**

One of the first applications of the agent technology, aimed at "reducing work and information overload", was that of retrieving and filtering information in an automatic way. Thus, the possibility to calculate the semantic distance between two sets of concepts finds a natural application in the agent field, in particular for improving those agents that act as "digital butlers" for their human owners, by exploring the Semantic Web and looking for useful documents and/or services.

Unfortunately, the metrics for calculating the semantic distance between two sets of concepts that can be found in the literature, are often very simple and do not meet some requirements that, up to us, make the metric closer to the common sense reasoning. For this reason, we have designed and implemented two new algorithms for computing the similarity between sets of concepts belonging to the same ontology.

I. Introduction

According to the Wikipedia encyclopedia (`http://en.wikipedia.org/wiki/`),

> *The Semantic Web is a project that intends to create a universal medium for information exchange by giving meaning (semantics), in a manner understandable by machines, to the content of documents on the Web.*

The intent of the Semantic Web is thus to enhance the usability and usefulness of the Web by means of common metadata vocabularies (ontologies [7]) and standard languages suitable for defining them, such as XML (`http://www.w3.org/TR/2004/REC-xml-20040204/`), XML Schema (`http://http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/`), RDF [6], RDF Schema [2], and OWL [17].

In order to share knowledge within the Semantic Web, two problems must be addressed:

1) The similarity between sets of concepts belonging to the same ontology O must be defined, in order to allow a user (be it a human or a software agent) interested in documents dealing with topics in the set S_1, to retrieve also documents dealing with topics in the set S_2, if S_1 and S_2 are "close enough" with respect to O.
2) Since it is not always possible to have a unique ontology O to use as a reference for comparing sets of concepts, documents are often tagged not only with a set of concepts, but also with the ontology from which the tagging concepts come from. In this case, maps between different ontologies must also be provided.

These two problems, although typical of the Semantic Web context, can be found in many other application scenarios, like Multiagent, Peer-to-Peer and Grid systems, where the comparison of set of concepts is required to provide more precise answers to the user' requests. For example, [8] describes a multiagent system for semantic-driven information retrieval, based on the peer-to-peer model, where routing of requests is performed by computing the similarity between the set of concepts advertised by each agent (the concepts that are dealt with by the agent's documents) and the set of concepts that characterise the documents looked for by the requesting peer. The similarity is evaluated by referring to a unique ontology that describes the system's domain, and from which concepts appearing both inside advertisements and requests are taken.

More in general, all the application scenarios where an "intelligent" information retrieval and filtering is required, may take advantage of programs and instruments that allow to compare two sets of concepts w.r.t. an ontology. In fact, all the "agents that reduce work and information overload" [15] usually need to compare the meta-information extracted from the retrieved document, with the meta-information that describes the user's interests. Very often, this meta-information consists either of a set of keywords whose similarity can be computed by using lexical ontologies such as WordNet, or of a set of concepts taken from a reference ontology.

In this paper, we address the problem of defining a similarity metric between sets of concepts belonging to the same ontology. In Section II we review some existing metrics defined in the literature to measure the similarity between two concepts (and, very rarely, between two sets of concepts) in either a taxonomy or an ontology. In Section III, we describe our algorithms and provide motivations for the choice we made in their design, and in Section IV we discuss our algorithms and we conclude the paper with the future directions of our work.

II. An overview of existing metrics

There are several techniques used to measure the similarity of two concepts belonging to the same taxonomy, and these techniques can be also applied to ontologies.

In particular the three main techniques are either 1) based on the distance between concepts, or 2) based on information content, or 3) based on a glossary. In this paper we take into account only the first one.

A. Bouquet, Kuper, Scoz and Zanobini's metric

In [1], Bouquet, Kuper, Scoz and Zanobini introduce two kinds of distances, one between simple concepts, and one between sets of simple concepts.

– *Ontological Distance between simple concepts.* The Ontological Distance between c and c', written $D_s(c, c')$, is the length of the minimal path between the nodes corresponding to c and c' in the ontology O, if such a path exists, and is 0 otherwise. The ontology is defined in the usual way, as a graph where nodes are labelled with concepts and arcs are labelled with relations between couples of concepts.

– *Ontological Distance between sets of simple concepts*. Let A and B be two sets of simple concepts. The ontological distance between the sets A and B, $D_c(A, B)$, is the sum of $D(c, c')$ for each c in A and c' in B.

Since this definition involves some redundancy, the notion of normalized set of simple concepts is introduced:

– *Normalized set of simple concepts*. Let K be the set of simple concepts occurring in a complex concept, namely a concept that is built from simple concepts defined in some ontology O and organized in a classification structure. A normalized set of simple concepts K' contained in K is defined as the set of all c belonging to K such that there is no path from c' to c in O for some c' belonging to K.

The ontological distance between complex concepts CA and CB is then defined as $D_c(A', B')$, where A' and B' are the normalized sets of simple concepts for A and B respectively, and A and B are the sets of simple concepts occurring in CA and CB, respectively.

B. Haase, Siebes, and van Harmelen's metric

In [9], the similarity between two concepts belonging to the same ontology, where a *SubTopic* relation is defined, is evaluated as

$$S(t_1, t_2) = \begin{cases} e^{-\alpha l} \cdot \frac{e^{\beta h} - e^{-\beta h}}{e^{\beta h} + e^{-\beta h}} & \text{if } t_1 \neq t_2 \\ 1 & \text{otherwise} \end{cases}$$

where l is the length of the shortest path between topic t_1 and t_2 according to the *SubTopic* relation, h is the level in the tree of the direct common subsumer from t_1 and t_2, and $\alpha >= 0$ and $\beta >= 0$ are parameters scaling the contribution of shortest path length l and depth h, respectively. The intuition behind using the depth of the direct common subsumer in the definition of the similarity is that topics at upper layers of hierarchical semantic nets are more general and are semantically less similar than topics at lower levels.

Given the function for calculating the similarity between two individual topics, it is possible to define the distance between two sets of concepts as

$$SF(s, e) = \frac{1}{|s|} \cdot \sum_{t_i \in s} \max_{t_j \in e} S(t_1, t_j)$$

C. Castano, Ferrara, Montanelli, and Racca's metric

In [5], a term affinity function $A(t, t')$ is defined to evaluate the affinity between two terms t and t' with respect to a thesaurus *Th* of terms and terminological relationships among them.

$A(t, t')$ is equal to the value of the highest-strength path of terminological relationships between t and t' in Th if at least one path exists, and is 0 otherwise. A path strength is computed by multiplying the weights associated with each terminological relationship involved in the path, that is:

$$A(t, t') = \begin{cases} \max_{i=1..k} \{W_{t \rightarrow_i^n t'}\} & \text{if } k > 1 \\ 0 & \text{otherwise} \end{cases}$$

where: k is the number of paths between t and t' in Th; $t \rightarrow_i^n t'$ denotes the *i*th path of length $n \geq 1$; $W_{t \rightarrow_i^n t'} = W_{1_{tr}} \cdot W_{2_{tr}} \cdot \cdots \cdot W_{n_{tr}}$ is the weight associated with the *i*th path, where $W_{j_{tr}}$ such that $j = 1, 2, ..., n$ denotes the weight associated with the *j*th terminological relationship in the path.

In [3], Bulskov, Knappe, and Andreasen use basically the same measure, namely the maximal multiplicative weighted path length, on ontologies expressed in ONTOLOG [16].

D. Rada, Mili, Bicknell, and Blettner's metric

In [18] the conceptual distance between any two concepts is defined as the shortest path through a semantic network. The semantic network taken into account is MeSH, a hierarchical network of biomedical concepts that (at the time the paper was published, namely in 1989) consisted of about 15,000 terms organized into a nine-level hierarchy, with concepts related by a *broader-than* relationships, which includes both *is-a* and *part-of* relationships.

E. Leacock and Chodorow's metric

The measure presented in [13] is similar to that defined by Rada, Mili, Bicknell, and Blettner, since it is based on the length of the shortest paths between noun concepts in a *is-a* hierarchy. Leacock and Chodorow's measure of similarity is thus defined as follows:

$$sim_{LeaCho}(c1, c2) = \max \left[-\log \left(\frac{length(c1, c2)}{(2D)} \right) \right]$$

where $length(c1, c2)$ is the shortest path length between the two concepts and D is the maximum depth of the taxonomy.

As we can see, the value of the shortest path length is scaled by the depth D of the hierarchy, where depth is defined as the length of the longest path from a leaf node to the root node of the hierarchy.

F. Wu and Palmer's metric

Wu and Palmer [20] define a measure of similarity that is also based on path lengths, however, they focus on the distance between a concept to the root node.

Resnik [19] reformulates their measure slightly. This measure finds the distance to the root of the most specific node that intersects the path of the two concepts in the *is-a* hierarchy. This intersecting concept is the most specific concept that the two concepts have in common, and is known as the "lowest common subsumer" (*lcs*). The distance of the *lcs* is then scaled by the sum of the distances of the individual concepts to the node.

The measure is formulated as follows:

$$sim_{WuPal}(c1, c2) = \frac{2 \cdot depth(lcs(c1, c2))}{depth(c1) + depth(c2)}$$

where $depth$ is the distance from the concept node to the root of the hierarchy.

G. Hirst and St.Onge's metric

Hirst and St.Onge [10] introduce a measure of relatedness that considers many other relations beyond the *is-a* one, and that is used for lexical ontologies. This measure classifies relations as: horizontal, upward, or downward.

- *Upward relations* connect more specific concepts to more general ones (i.e., *is-a*)
- *Downward relations* connect more general concepts to more specific ones (i.e., *is-a-kind-of*)
- *Horizontal relations* maintain the same level of specificity.

The measure defined by Hirst and St.Onge has three levels of relatedness: extra strong, strong and medium strong. An extra strong relation is based on the syntactic form of the words, while two words representing the same concept (i.e., synonyms) have a strong relation between them. The medium-strong relation is determined by a set of allowable paths between concepts. If a path that is neither too long nor too winding exists, then there is a medium-strong relation between the concepts. The score given to a medium-strong relation considers the path length between the concepts and the number of changes in direction of the path:

$path\ weight =$
$C - path\ length - (k \times \#changes\ in\ direction)$

III. Computing the similarity of two sets of concepts: A new proposal

The two algorithms we have designed and implemented to compute the similarity of two sets of concepts, work on ontologies represented in OWL, RDF and DAML+OIL [11]. Both algorithms are based on the definition of the "sim_c" function for calculating the similarity of a concept w.r.t. a set of concepts. Thus, we first introduce "sim_c" in Section III-A, and then we introduce the two algorithms in Section III-B.

A. Computing the similarity between a concept and a set of concepts

Our algorithm for computing the similarity between a concept and a set of concepts is an extension of Dijkstra's algorithm (shown in Algorithm 1), where there may be more than one destination node.

Algorithm 1: $Dijkstra(G, w, s)$

```
foreach vertex v ∈ V[G] do
    d[v] := ∞;
    previous[v] := nil;
end
d[s] := 0;
S := ∅;
Q := V;
while Q ≠ ∅ do
    u := extract_min(Q);
    S := S ∪ {u};
    foreach edge (u, v) outgoing from u do
        if d[v] > d[u] + w(u, v)  /* Relax(u,v)  */
        then
            d[v] := d[u] + w(u, v);
            previous[v] := u;
            Q := update(Q);
        end
    end
end
```

Initialisation: The difference in the initialisation phase w.r.t. Dijkstra's algorithm is that in ours, the value of a path is evaluated as the product of the weights of the path's edges (kept in a $w[\_,\_]$ matrix), and we prefer paths with a higher value[1]. Thus, we must initialise the similarity (kept in a $d[\_]$ array) of the source concept s from itself as if it were the best possible similarity (1), and the other similarities as if they were the worst possible ones (0).

In our algorithm for calculating the "sim_c" function we use the $pi[\_]$ array that, for each concept in the ontology, keeps track of its predecessor (if any) in the current best estimated path towards the destination node(s). The pi data structure allows us to store a spanning tree of the ontology, characterised by the nodes $pi[j]$ for $1 \leq j \leq |O|$ (where $|O|$ is the number of concepts belonging to the ontology) and by the edges $(pi[j], j)$, for $1 \leq j \leq |O|$.

[1]The weights between couples of concepts kept in the $w[\_,\_]$ matrix, representing the similarity of pairs of adjacent concepts, are decided at design time by the ontology developer who is supposed to be an expert of the ontology domain.

Algorithm 2: *initialise_single_source*(*Ontology* o, *Concept* s)

```
foreach concept c in Concepts(o) do
    d[c] := 0 ;
    pi[c] := nil ;
end
d[s] := 1;
```

Relaxation: The relaxation procedure checks whether the current best estimate of the similarity between s and v ($d[v]$) can be improved by going through u (i.e. by making u the predecessor of v). With respect to Dijkstra's algorithm, here we changed a $>$ with a $<$ in the condition of the **if** statement, and a $+$ with a $*$ in the evaluation of the path's total weight.

Algorithm 3: *relax*(*Concept* u, *Concept* v, *double* $w[][]$)

```
if d[v] < d[u] * w[u,v] then
    d[v] := d[u] * w[u,v] ;
    pi[v] := u ;
end
```

Implementation of the sim_c function: The *sim_c* algorithm, whose pseudo-code is shown in Algorithm 4, evaluates the similarity between a concept s and a set of concepts $target$.

The algorithm returns a value in $[0, 1]$. If s belongs to $target$, 1 is returned and the algorithm stops, otherwise the algorithm starts by exploring the paths from s to the nodes in $target$. The *lower_bound* parameter is the value under which results are considered no longer relevant, and may range in $[0, 1]$.

Post processing phase: All the paths with a better value than *lower_bound* have been explored; the *current_highest_similarity* array contains 0 for those nodes in the local ontology whose similarity with s is not relevant; they contain a value different from 0 (and surely $>$ *lower_bound*) for the other ones. Now, we only need to combine these values in order to obtain a final value in $[0, 1]$. The mathematical function used to combine these values is the following:

$$combine\_final\_value(x_1 \dots x_n) = f(x_1)$$

where the sequence $(x_1 \dots x_n)$ is ordered and

$$f(x_i) = \begin{cases} x_i + (1 - x_i) * f(x_1) & \text{if } i < n \\ x_i & \text{if } i = n \end{cases}$$

This function, when applied to the ordered sequence of values is in the $[0, 1]$ range representing the similarity of the source concept with all the other concepts of the matrix (when this similarity is different from 0), allows us to obtain a value which is in the $[0, 1]$ range, and that gives more weight to the higher values, but still allowing the lower values to contribute to the final outcome.

Algorithm 4: *double sim_c*(*Ontology* o, *Concept* s, *set*(*Concept*) *target*, [0..1] *lower_bound*)

```
if s ∈ target then
    return 1
else
    initialise_single_source(o, s);
    foreach concept c ∈ target do
    current_highest_similarity[c] := 0;
    go_on := true;
    /* Make S equal to the source
       concept                                  */
    S := s;
    /* the current concept is the source
       concept                                  */
    u := s;
    C := Concepts(o);
    /* while there are still concepts to
       explore, and the paths through
       these concepts may prove to be
       better than the current path to
       one of the concepts in target            */
    while C ≠ S and go_on do
        foreach concept c ∈ Adjacent(u) do
            /* the path's weights going
               through "v" are evaluated
               and updated                      */
            relax(u, c, w);
        end
        /* the most promising concept to
           reach one of the concepts in
           Loc namely the one such that
           d[u] is higher is found              */
        u := extract_most_similar(C \ S);
        if d[u] > lower_bound then
            S := S ∪ {u};
            /* if the path leads to the
               destination, we wonder if
               this path is better than the
               previous one found, leading
               to the destination (if any),
               and we eventually update the
               value of the current best
               path                             */
            if u ∈ target then
                if d[u] > current_highest_similarity[u]
                then
                    current_highest_similarity[u] :=
                    d[u];
                end
            end
        else
            /* if all the paths that still
               remain to be considered, are
               worst than the lower_bound,
               the algorithm stops              */
            go_on := false;
        end
    end
    odered_val := order(current_highest_similarity);
    return combine_final_value(ordered_val);
end
```

B. Computing the similarity between two sets of concepts

In the following we present two different algorithms for computing the similarity between two sets of concepts. The first is based on a mathematical formula while the second is based on a recursive approach. There are some conditions that, in our opinion, an algorithm for computing the similarity between two sets of concepts should meet :

1) It is not sufficient that a concept in the source set matches a concept in the target set to obtain 1 as result.
2) To obtain 1 as result, all the concepts in the source set should have a related concept in the target set.
3) If there are many concepts with high similarity in the two sets, we would like to "prize" them by having a similarity function that respects the inequality:
$$similarity(o, target, source, lower\_bound) > \frac{\sum_{s \in source} sim\_c(o, target, source, lower\_bound)}{|source|}$$
On the other hand, if there are many concepts with low similarity in the two sets of concepts, we would like to "punish" them by having a similarity function that respects the inequality:
$$similarity(o, target, source, lower\_bound) < \frac{\sum_{s \in source} sim\_c(o, target, source, lower\_bound)}{|source|}$$

1) Algorithm $similarity\_by\_m\_th\_root$.: The first algorithm is based on the following formula:

$$\sqrt[m]{\frac{\sum_{k=0}^{n} (a_k)^m}{n}} \quad where\ a_i \in \mathbb{Z}$$

The algorithm that uses this formula is described below in pseudo-code; it takes as input five parameters: the ontology, the target set, the source set, a coefficient representing the value of m in the previous formula and the $lower\_bound$ under which results are considered no longer relevant.

Algorithm 5: $double\ similarity\_by\_m\_th\_root(Ontology\ o, set(Concept)\ target, set(Concept)\ source, int\ m, [0..1]\ lower\_bound)$

```
result := 0;
foreach concept c ∈ source do
    result := result + sim_c(o, c, target, lower_bound)^m;
end
return ∛result / |source|;
```

2) Algorithm $similarity\_by\_recursive\_eval$.: The second algorithm is also based on the sim_c algorithm for computing the similarity between a concept and a set of concepts illustrated previously, but the different idea is to evaluate the similarity between the target set and the elements of the source set, and to use a recursive function to calculate the similarity. The entire pseudo-code is illustrated below.

Algorithm 6: $double\ similarity\_by\_recursive\_eval(Ontology\ o, set(Concept)\ target, set(Concept)\ source, [0..1]\ lower\_bound)$

```
similarity := nil;
foreach concept c ∈ source do
    similarity.add(sim_c(o, c, target, lower_bound));
end
return calculate(similarity);
```

Algorithm 7: $double\ calculate(set(Double)\ similarity)$

```
if similarity.isEmpty() then
    return 0
else
    value = max(similarity);
    similarity.remove(value);
    return (value + (1 − value) ∗ calculate(similarity));
end
```

IV. Discussion and future work

The $similarity\_by\_m\_th\_root$ algorithm meets the requirements specified in Section III-B. It defines a means where high values have an higher impact in computing the final result than low values. The m parameter allows us to specify how much the higher values should be "prized" with respect to lower ones.

The $similarity\_by\_recursive\_eval$, instead, does not satisfy the third requirement: in fact, if one (and only one) concept belongs both to the $source$ set and to the $target$ set, the result of the algorithm is 1, and thus the first condition specified in Section III-B is not met. However our experiments, discussed in [14], demonstrated that $similarity\_by\_recursive\_eval$ has a better performance than $similarity\_by\_m\_th\_root$.

Although many techniques for computing the similarity between concepts belonging to the same ontology or taxonomy exist, the use of that based on the path length is very common in the scientific community. Our definition of the similarity between concepts is also very common, and thus it is not a novelty w.r.t. the existing literature. However, we have also defined the similarity between a single concept and a set of concepts, and between two sets of concepts, establishing some criteria that our definitions should meet. Up to our knowledge, the only metrics for defining the similarity between two sets of concepts are those defined in [1] and [9], but none of them meets our requirements, that, instead, are met by our $similarity\_by\_m\_th\_root$ definition.

Our algorithms have been implemented using the Jena framework (`http://jena.sourceforge.net`), and can be downloaded from `http://www.disi.unige.it/person/MascardiV/Software/SoftwarePaoloLombardi.html`. The main direction of our work consists of integrating our implemented metrics into the P2P system described in [8], that is being developed using JXTA (`http://www.jxta.org`). Since both Jena and JXTA are based on Java, the integration should be easy to implement.

Although this system was born as a pure P2P system, we can easily see it as a MAS. In fact, peers are autonomous (they actively push their expertise to the other peers in the system, as discussed below), reactive (they react to incoming requests), proactive (they have a long term goal of retrieving as many relevant documents as possible), and social (communication is asynchronous and uses a structured, XML-based communication language). Finally, peers are aware of the features, the topology and the inhabitants of the P2P system where they live, so they are in some sense situated in they software environment. Since peers respect the well-known definition of agent given in [12], in the following we will use the term agent instead of the term peer.

In the system under consideration, agents may register to one or more thematic groups. Relevant information retrieval is achieved through the use of a thematic global ontology (TGO) for each theme dealt with by the system; the TGO associates a semantics with the resources to be shared within the thematic group. All the agents that register to a thematic group share the TGO of the group. Each arc between two concepts belonging to the TGO is weighted with a value in $[0, 1]$ that represents the similarity between the two concepts. Each agent A is characterised by a set of concepts of interest CoI_A such that $CoI_A \subseteq V$, where V are the concepts in the TGO. Agents actively and autonomously push their expertises by sending *advertisements*, containing the concepts of the TGO that better describe the resources they share, so each agent A is also characterised by the sets of advertised concepts that it pushes towards the system, Adv_{A_i}, for $i \in 1, .., n$ such that $Adv_{A_i} \subseteq V$.

In order to allow an agent A to understand if an agent B sending an advertisement Adv_{B_j}, shares resources that may be of interest for A, the similarity between Adv_{B_j} and CoI_A w.r.t. the TGO should be evaluated. By integrating our $similarity\_by\_nth\_root$ algorithm in the system, we would allow agent A to evaluate $similarity\_by\_nth\_root(TGO, CoI_A, Adv_{B_j}, lower\_bound)$ thus obtaining the required similarity.

REFERENCES

[1] P. Bouquet, G. Kuper, M. Scoz and S. Zanobini. Asking and answering semantic queries. In *Proc. of Meaning Coordination and Negotiation Workshop (MCNW-04) in conjunction with International Semantic Web Conference (ISWC-04)*, 2004

[2] J. Broekstra,M. Klein, S. Decker, D. Fensel, F. van Harmelen and I. Horrocks. Enabling knowledge representation on the Web by extending RDF schema, In *Proc. of the 10th international conference on World Wide Web*, pp. 467–478, 2001

[3] H. Bulskov, R. Knappe and T. Andreasen. On Measuring Similarity for Conceptual Querying. In *Proc. of the 5th International Conference on Flexible Query Answering Systems*, Springer-Verlag publisher, pp. 100–111, 2002.

[4] S. Castano, A. Ferrara, S. Montanelli, E. Pagani, and G. P. Rossi. Ontology-addressable contents in P2P networks. In *Proc. of the 1st SemPGRID Workshop*, 2003.

[5] S. Castano, A. Ferrara, S. Montanelli, G. Racca. Semantic Information Interoperability in Open Networked Systems. In *Proc. of the Int. Conference on Semantics of a Networked World (ICSNW), in cooperation with ACM SIGMOD 2004*, pp. 215–230, 2004.

[6] S. Decker, S. Melnik, F. van Harmelen, D. Fensel, M. C. A. Klein, J. Broekstra, M. Erdmann and I. Horrocks. The Semantic Web: The Roles of XML and RDF, IEEE Internet Computing, 4(5), pp. 63–74, 2000

[7] T. R. Gruber. A translation approach to portable ontology specifications. Knowledge Acquisition, 5:199–220, 1993.

[8] G. Guerrini, V. Mascardi, M. Mesiti. A Semantic Information Retrieval Advertisement and Policy Based System for a P2P Network. In *Proc. of the DBISP2P Conference*, 2005.

[9] P. Haase, R. Siebes, F. van Harmelen. Peer Selection in Peer-to-Peer Networks with Semantic Topologies. In *Proc. of International Conference on Semantics of a Networked World: Semantics for Grid Databases*, 2004.

[10] G. Hirst, D. St. Onge. Lexical chains as representations of context for the detection and correction of malapropisms, In Fellbaum MIT Press, pp. 305–332, 1998.

[11] I. Horrocks, P. F. Patel-Schneider, and F. van Harmelen. Reviewing the design of DAML+oil: an ontology language for the semantic web. In *Proc. of the 18th National Conference on Artificial Intelligence*, pp. 427–428, 2002.

[12] N. R. Jennings and K. Sycara and M. Wooldridge. A Roadmap of Agent Research and Development. Autonomous Agents and Multi-Agent Systems, 1:7–38, 1998.

[13] C. Leacock, M. Chodorow. Combining local context and WordNet similarity for word sense identification, In Fellbaum MIT Press, pp. 265–283, 1998.

[14] P. Lombardi. Progettazione ed implementazione di una nuova metrica sulle ontologie. Master Thesis, DISI, Università degli Studi di Genova, 2005. In Italian. `http://www.disi.unige.it/person/MascardiV/Download/Lombardi.zip`.

[15] P. Maes. Agents that Reduce Work and Information Overload. Communications of the ACM, 37(7), 1994.

[16] J. F. Nilsson. A Logico-algebraic Framework for Ontologies ONTOLOG. In *Proc. of the First International OntoQueryWorkshop Ontology-based interpretation of NP s*, 2001.

[17] P. F. Patel-Schneider, P. Hayes, and I. Horrocks. Web Ontology Language (OWL) Abstract Syntax and Semantics. Technical report, W3C.

[18] R. Rada, H. Mili, E. Bicknell, M. Blettner. Development and application of a metricon semantic nets. IEEE Transaction on Systems, Man and Cybernetics 19(1), pp. 17–30, 1989.

[19] P. Resnik. Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural languag, Journal of Artificial Intelligence Research 11, pp. 95–130, 1998.

[20] Z. Wu, M. Palmer. Verb semantics and lexical selection, In 32nd Annual Meeting of the Association for Computational Linguistics, Las Cruces, New Mexico, pp. 133–138, 1994.