Versatile Structural Disambiguation for Semantic-aware Applications *

Federica Mandreoli
Dip. Ing. dell'Informazione
Univ. di Modena e
Reggio Emilia
I-41100 Modena, Italy
fmandreoli@unimo.it

Riccardo Martoglia
Dip. Ing. dell'Informazione
Univ. di Modena e
Reggio Emilia
I-41100 Modena, Italy
rmartoglia@unimo.it

Enrico Ronchetti
Dip. Ing. dell'Informazione
Univ. di Modena e
Reggio Emilia
I-41100 Modena, Italy
eronchetti@unimo.it

ABSTRACT

In this paper, we propose a versatile disambiguation approach which can be used to make explicit the meaning of structure based information such as XML schemas, XML document structures, web directories, and ontologies. It can be of support to the semantic-awareness of a wide range of applications, from schema matching and query rewriting to peer data management systems, from XML data clustering to ontology-based automatic annotation of web pages and query expansion. The effectiveness of the achieved results has been experimentally proved and is founded both on a flexible exploitation of the structure context, whose extraction can be tailored on the specific application needs, and of the information provided by commonly available thesauri such as WordNet.

Categories and Subject Descriptors:
H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing — *linguistic processing*

General Terms: Algorithms.

Keywords: word sense disambiguation, semantic web, structure based information.

1. INTRODUCTION

In recent years, knowledge based approaches, i.e. approaches which exploit the semantics of the information they access, are rapidly acquiring more and more importance in a wide range of application contexts. We refer to "hot" research topics, like schema matching and query rewriting [9, 15], also in peer data management systems (PDMS) [13], XML data clustering and classification [19, 20] and ontology-based annotation of web pages and query expansion [8, 10], all going in the direction of the Semantic Web "... an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation" [4]. In these contexts, most of the proposed approaches share a common basis: They focus on the structural properties of the accessed information, which are represented adopting XML or ontology based data models, and their effectiveness is heavily dependent on knowing the right meaning of the employed terminology. For instance, the graph matching algorithm proposed in [16] converts the schemas to be matched into directed labelled graphs, assumes that a similarity measure between nodes has been defined and the matching computation essentially relies on a mechanism of similarity spreading. Obviously, the more the similarity measure is able to quantify the semantic closeness of the node's labels, the more the obtained mappings are effective. In the same way, the approach for automatic classification of XML data proposed in [20] views tags as high-quality features of XML documents and exploits their meaning in the classification process. Generally speaking, due to the ambiguity of natural languages, terms describing information usually have several meanings and making explicit the semantics of information goes through the tricky task of deriving from the context the most appropriate meanings. For example, Fig. 1 shows the hierarchical representation

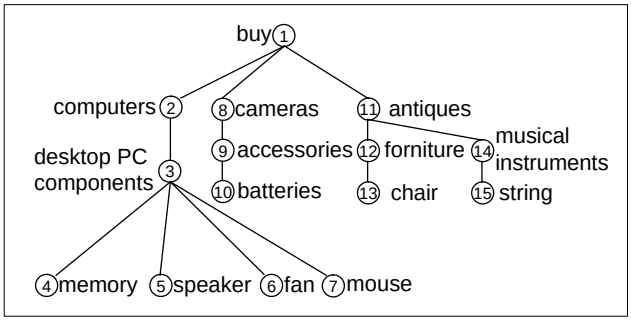


Figure 1: A portion of the eBay® categories.

of a portion of the categories offered by eBay®, one of the most famous world's online marketplaces (nodes are univocally identified by their pre-order values). It contains many polysemous words, from `string` to `batteries` and `memory`, to which commonly available vocabularies associates several meanings. The information given by the surrounding nodes allows us to state, for instance, that `string` are "stringed

*This work is partially supported by the Italian Council co-funded project WISDOM.

CIKM'05, October 31–November 5, 2005, Bremen, Germany.

instruments that are played with a bow" and `batteries` are electronic devices and not a group of guns or whatever else.

For most of the knowledge based approaches present in the literature, the problem of making explicit the meanings of words is usually demanded to human intervention and till now machine-based solutions have only been marginally addressed in the context of structure based information. On the other hand, in the most cutting edge semantic-aware application contexts the role of humans is limited to that of user while, when some kind of human intervention is provided for, unassisted semantic annotation is a quite tedious task.

In this paper, we propose a generic approach for the disambiguation of graph-like structured information, mainly focusing on trees. It can be used to make explicit the meaning of a wide range of structure based information, including XML schemas, the structures of XML documents, web directories, and ontologies. Starting from the lesson learnt in the word sense disambiguation (wsd) field [11], where several solutions have been proposed for free text, we have conceived a versatile approach which tries to disambiguate the terms occurring in the nodes' labels by analysing their context and by using an external knowledge source. More precisely, starting from a given node, we support several ways of navigating the graph in order to extract the context which can thus be tailored on the specific application needs. Moreover, the disambiguation method does not depend on training data or extensions, which are not always available. For instance, in a PDMS, peers not necessarily store actual data. We follow instead a different approach: The exploitation of the information provided by commonly available thesauri such as WordNet [17]. In particular, disambiguation is founded on the hypernymy/hyponymy hierarchy, as suggested by most of the classic wsd studies, and the sense contexts, extracted from the thesaurus, can be compared against the graph context to refine the results. The outcome of the overall process is a ranking of the plausible senses for each term. In this way, we are able to support both the assisted annotation and the completely automatic one whenever the top sense is selected. The disambiguation approach has been implemented and extensively evaluated through tests performed on three groups of trees differing in the level of specificity and polysemy. Experimental results show the good effectiveness of the proposed approach, also in particularly involved settings.

The rest of the paper is organized as follows: Section 2 presents an overview of our disambiguation approach, while the proper disambiguation algorithm is presented in Section 3. Experimental evaluation is provided in Section 4 and related works in Section 5. Finally, Section 6 concludes the paper.

2. OVERVIEW OF THE APPROACH

In this section we present the functional architecture of a generic tree disambiguation service (see Fig. 2) and introduce relevant terminology. Being trees particular kinds of graphs, without loss of soundness, in the following we will use indifferently the terms tree and graph. Indeed, at the end of the present section, we will show that the service can be straightforwardly extended to graphs. We emphasize that no extension or training data is required for our disambiguation purposes as they are not always available. The only external source is a thesaurus associating each word with the concepts or senses it is able to express.

The service is able to disambiguate XML schemas, the structures of XML documents, web directories, and, in general, such information descriptions which can be represented as trees. As a particular case, XML schemas are represented as trees which make explicit the structural relationships between the involved elements, thus capturing the element context, and abstract from the complexity of the language syntax. The tree contains a set of nodes whose labels must be disambiguated and a set of arcs which connect pairs of nodes and which may as well be labelled (e.g. `type`, `property`). The individuation of the correct sense for each label can be possible by analysing the context of the involved terms and by using an external knowledge source. Arcs are particularly important as they connect each label with its context. Each arc label is associated with two weights between 0 and 1 (default value 1), one for each crossing direction (direct and inverse). Weights will be used to compute the distance between two nodes in the graph and the lower the weight of an arc is the closer two nodes connected by such arc are.

The "terms/senses selection" component in Fig. 2 takes the label of each node N of the tree, extracts the contained terms (which can also be more than one as for instance `desktop PC components` in Fig. 1) and associates each of these terms (t, N)[1] with a list of senses $Senses(t, N) = [s_1, s_2, \ldots, s_k]$. In principle, such list is the complete list of senses provided by the thesaurus but it can also be a shrunk version suggested either by human or machine experts or as feedback of a previous disambiguation process.

Each polysemous term (t, N) is then associated with its context. The context is first extracted from the tree but it does not necessarily coincide with the entire tree. Indeed, different applications require different contexts. For instance, while disambiguating the term `string` in the `musical instruments` category of eBay®, using categories such as `women's clothing` would be quite misleading. Thus we support different contexts by means of different crossing settings. By default, the nodes reachable by the term's node N through any arc belong to the term's context. The set of crossable arc labels and the corresponding crossing directions is shrinkable, that is it is possible to specify which kinds of arcs are crossable, in which direction and the maximum number of crossings (distance from the term's node). Moreover, as we deal with trees, we also provide the possibility of including the siblings of the term's node in the context. The above options can be freely combined. As a special case, let us consider trees having no label on the arcs. It actually represents the conceptual structure of the most common application contexts such as web directories, XML documents, and XML schemas. When the only crossing direction is the direct one, the context is defined by the descendants or subtree of the term's node. Conversely, it is represented by the ancestors. For instance, for the eBay example, one of the best crossing settings is to include ancestors, descendants, and siblings whereas the whole structure would be useful for structures dealing with more "contextualized" topics such as book descriptions.

Given a crossing setting, the "graph context extraction" component in Fig. 2 contextualizes each polysemous term (t, N) by extracting its graph context $Gcontext(t, N)$ from

[1]Notice that the same term could be included more than once and that the disambiguation is strictly dependent on the node each instance belongs to.

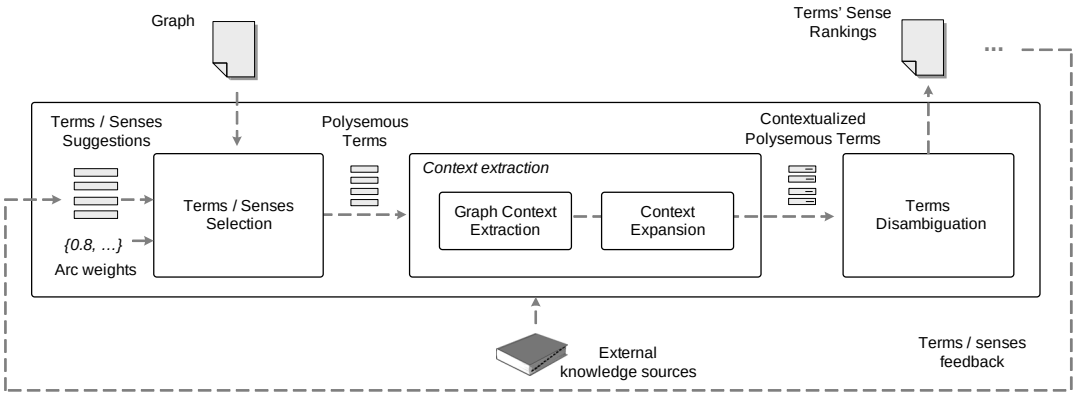


Figure 2: A generic graph disambiguation service.

the set of terms belonging to the reachable nodes. Not all nodes contribute with the same weight to the disambiguation of a term. In principle, the more one node is close to the term's node and is connected by arcs with low weights the more it influences the term disambiguation. For this reason, we associate each reachable node N_c in the context with a weight $weight(N_c)$ computed as follows. Given the path from the node N_c to the term's node N, we count the number of instances corresponding to each pattern specified in the crossing setting (i.e. arc label and arc crossing direction) and we define the distance d between N and N_c as the sum of the product of the weights associated to each pattern and the corresponding number of instances. Then, $weight(N_c)$ is computed by applying a gaussian distance decay function defined on d:

$$weight(N_c) = 2 \cdot \frac{e^{-\frac{d^2}{8}}}{\sqrt{2\pi}} + 1 - \frac{2}{\sqrt{2\pi}}$$

Thus each element of the graph context is a triple $((t_c, N_c), Senses(t_c, N_c), weight(N_c))$ defined from each term t_c belonging to each reachable node N_c. For instance, assume that in the eBay example the context is made up of the siblings and ancestors, that the weight of the parent/child arcs is 1 in the direct direction and 0.5 in the opposite one, and that the maximum number of crossings is 2. The graph context of the term $(\texttt{mouse}, 7)$ is made up of the terms $(\texttt{computers}, 2)$, $(\texttt{desktop}, 3)$, $(\texttt{PC}, 3)$, $(\texttt{components}, 3)$, $(\texttt{memory}, 4)$, $(\texttt{speaker}, 5)$, and $(\texttt{fan}, 6)$. The distance between node 7 and nodes 2, 3, 4 (5 and 6) are 1 (i.e. 2 arcs crossed in the opposite direction with weight 0.5), 0.5 (i.e. 1 arc crossed in the opposite direction with weight 0.5), and 1.5 (i.e. 1 arc crossed in the opposite direction with weight 0.5 and 1 arc crossed in the direct direction with weight 1), respectively. Then, $weight(2) = 0.91$, $weight(3) = 0.95$, and $weight(4) = weight(5) = weight(6) = 0.8$.

The context of each term (t, N) can be expanded by the contexts $Scontext(s)$ of each sense s in $Senses(t, N)$. It is particularly useful when the graph context provides too little information. In particular for each sense we consider the definitions, the examples and any other explanation of the sense provided by the thesaurus. As most of the semantics is carried by noun words [11], the "context expansion" module in Fig. 2 defines $Scontext(s)$ as the set of nouns contained in the sense explanation.

Finally, each term (t, N) with its senses $Senses(t, N)$ is disambiguated by using the previously extracted context. The proper disambiguation process is the subject of the following Section. The result is a ranked version of $Senses(t, N)$ where each sense $s \in Senses(t, N)$ is associated with a confidence $\phi(s)$ in choosing s as a sense of (t, N).

The overall approach is quite versatile. It supports several disambiguation needs by means of parameters which can be freely combined, from the weights to the graph context. Moreover, the ranking approach has been conceived in order to support two types of graph disambiguation services: The assisted and the completely automatic one. In the former case, the disambiguation task is committed to a human expert and the disambiguation service assists him/her by providing useful suggestions. In the latter case, there is no human intervention and the selected sense can be the top one. Moreover, the above approach can be straightforwardly applied also to graphs. Indeed, only the context extraction phase accesses the submitted structure whereas the actual disambiguation algorithm is completely independent from it. The only problem is in the weight computation where more than one path can connect a pair of nodes. In this case, the one with the lower distance could be selected. In this way, we would be able to disambiguate ontologies written in different languages such as OWL and RDF where arc labels are quite frequent (e.g. in RDF arcs can be of `subClassOf` type or `range` type or many other types). However, at present, trees are our main focus of interest and, in particular, trees having no label on the arcs which have been subject of our tests, while we plan to deal with general graphs in the future.

3. THE DISAMBIGUATION ALGORITHM

The algorithm for disambiguation we devised follows a relational information and knowledge-driven approach. Indeed, the context is not merely considered as a bag of words but other information such as their distance from the polysemous term to be disambiguated and semantic relations are also extracted. Moreover we use additional information provided by thesauri: The hypernymy/hyponymy relationships among senses and the sense explanations and frequencies of use. In particular, for the disambiguation algorithm we used WordNet [17] which is a fairly comprehensive common-sense thesaurus.

The algorithm is shown in Fig. 3. It takes in input a term (t, N) to be disambiguated and produces a vector ϕ of confidences in choosing each of the senses in $Senses(t, N)$.

```
algorithm Disambiguate(t, N)
//graph context contribution
(01) φ_G = [0, ..., 0]   (# senses in Senses(t_c, N))
(02) norm = 0
(03) for each (t_c, N_c) in Gcontext(t, N)
(04)     φ_G = φ_G + weight(N_c) * TermCorr(t, t_c, norm)
(05)     norm = norm * weight(N_c)
(06) φ_G = φ_G / norm
//expanded context contribution
(07) for i from 1 to the number of senses in Senses(t, N)
(08)     if expanded context
(09)         φ_E[i] = ContextCorr(Gcontext(t, N), Scontext(s_i))
(10)     φ_U[i] = decay(s_i)
(11) φ = α(γ * φ_G + ϵ * φ_E) + β * φ_U
```

Figure 3: The disambiguation algorithm

In particular, given $Senses(t, N) = [s_1, s_2, \ldots, s_k]$, ϕ is a vector of k values and $\phi[i]$ is the confidence in choosing s_i as sense of (t, N). The obtained confidence vector tunes two contributions (line 11): That of the context, whose weight is expressed by the constant α and which is subdivided in the graph context (confidence vector ϕ_G, weight γ) and the expanded context (confidence vector ϕ_E, weight ϵ), $\gamma+\epsilon = 1$, and that of the frequency of sense use in English language (confidence vector ϕ_U) with weight β, $\alpha + \beta = 1$.[2]

The terms surrounding a given one provide a good informational context and good hints about what sense to choose for it. The contribution of the graph context is computed from step 1 to step 6. In particular, ϕ_G is the sum of the values measuring the level of semantic correlation between the polysemous term t and the ones in the graph context $Gcontext(t, N)$ (step 4). The contribution of each context term (t_c, N_c) is weighted by the relative position in the graph of the t_c's node, N_c, w.r.t. N (i.e. $weight(N_c)$). Finally in step 6 the whole vector ϕ_G is divided by the $norm$ value in order to obtain normalized confidences.

```
function TermCorr(t, t_c, norm)
(1) c(t, t_c) is the minimum common hypernymy of t and t_c
(2) φ_C = [0, ..., 0]
(3) for i from 1 to the number of senses in Senses(t, N)
(4)     if c(t, t_c) is ancestor of s_i
(5)         φ_C[i] = sim(t, t_c)
(6) norm = norm + sim(t, t_c)
(7) return φ_C
```

Figure 4: The $TermCorr()$ function

The basis of function $TermCorr()$ (see Fig. 4) derives from the one in [18]. As in [18], the confidence in choosing one of the senses associated with each term is directly proportional to the semantic similarities between that term and each term in the context; the intuition behind the similarity is that the more similar two terms are, the more informative will be the most specific concept that subsumes them both. However, our approach differs in the semantic similarity measure $sim(t, t_c)$ as it does not rely on a training phase on large pre-classified corpora but exploits the hypernymy hierarchy of the thesaurus. In this context, one of the most promising measures is the Leacock-Chodorow [12], which has been reviewed in the following way:

$$sim(t, t_c) = \begin{cases} -ln\frac{len(t,t_c)}{2 \cdot H} & \text{if } \exists \text{ a common hypernymy} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where $len(t, t_c)$ is the minimum among the number of links connecting each sense in $Senses(t, N)$ and each sense in $Senses(t_c, N_c)$ and H is the height of the hypernymy hierarchy (in WordNet it is 16). Moreover, we define the minimum common hypernym $c(t, t_c)$ of t and t_c as the sense which is the most specific (lowest in the hierarchy) of the hypernyms common to the two senses (i.e. that crossed in the computation of $len(t, t_c)$). For instance, in WordNet the minimum path length between the terms "cat" and "mouse" is 5, since the senses of such nouns that join most rapidly are "cat (animal)" and "mouse (animal)" and the minimum common hypernym is "placental mammal". Obviously these two values are not computed within the function but once for each pair of the involved terms. Eq. 1 is decreasing as one moves higher in the taxonomy thus guaranteeing that "more abstract" is synonymous of "less informative". Therefore, function $TermCorr()$ increases the confidence of such senses in $Senses(t, N)$ which are descendants of the minimum common hypernym (lines 3-4) and the increment is proportional to how informative the minimum common hypernym is (line 5). At the end of the process (Fig. 3, line 6), the value assigned in ϕ_G to each sense is then the proportion of support it receives, out of the support possible which is kept updated by function $TermCorr()$ (line 6) and in the main algorithm (Fig. 3, line 5).

```
function ContextCorr([t_1, ..., t_n], [t_1^s, ..., t_m^s])
(1) φ_C = [0, ..., 0]
(2) for i from 1 to n
(3)     φ_T = [0, ..., 0]
(4)     norm = 0
(5)     for j from 1 to m
(6)         φ_T = φ_T + TermCorr(t_i, t_j^s, norm)
(7)     φ_C[i] = max(φ_T / norm)
(8) return mean(φ_C)
```

Figure 5: The $ContextCorr()$ function

Beside the contribution of the graph context, also the expanded context can be exploited in the disambiguation process (Fig. 3, lines 7-9). In this case, the main objective is to quantify the semantic correlation between the context $Gcontext(t, N)$ of the polysemous term (t, N) and the explanation of each sense s in $Senses(t, N)$ represented by $Scontext(s)$. In particular, the confidence in choosing s is proportional to the computed similarity value (Fig. 3, line 9). The pseudocode of function $ContextCorr()$ is shown in Fig. 5. It essentially computes the semantic similarity between each term t_i in the graph context and the terms in the sense context $Scontext(s)$ (lines 3-7) by calling the $TermCorr()$ function for each term t_j^s in $Scontext(s)$ (line 6) and then by computing the maximum of the obtained confidence vector ϕ_T. The returned value (line 8) is the mean of the similarity values computed for the terms in $Gcontext(t, N)$.

The last contribution is that of function $decay()$, exploiting the frequency of use of the senses in English language (Fig. 3, line 9). In particular, WordNet orders its list of senses $WNSenses(t)$ of each term t on the basis of the fre-

[2] All operations on the vectors are the usually defined ones.

quency of use (i.e. the first is the most common sense, etc.). We increment the confidence in choosing each sense s in $Senses(t, N)$ in a way which is inversely proportional to its position, $pos(s)$, in such ordered list:

$$decay(s_i) = 1 - \rho \frac{pos(s_i) - 1}{|WNSenses(t)|}$$

where $0 < \rho < 1$ is a parameter we usually set at 0.8 and $|WNSenses(t)|$ is the cardinality of $WNSenses(t)$. In this way, we quantify the frequency of the senses where the first sense has no decay and the last sense has a decay of 1:5. Such an adjustment attempts to emulate the common sense of a human in choosing the right meaning of a noun when the context gives little help.

As a final remark, notice that for the sake of simplicity of presentation, algorithm `Disambiguate()` takes one term at a time. However, for efficiency reasons, in the actual implementation the $sim()$ computation is performed only once for a given pair of terms (also swapped as $sim()$ is a symmetric measure).

4. EXPERIMENTAL EVALUATION

In this section we present the results of an actual implementation of our disambiguation approach.

4.1 Experimental setting

Tests were conceived in order to show the behavior of our disambiguation approach in different scenarios. In particular we tested 3 groups of trees characterized by 2 dimensions of interest. The first dimension, *specificity*, indicates how much a tree is contextualized in a particular scope; trees with low specificity can be used to describe heterogeneous concepts, such as a web directory, whereas trees with high specificity are used to represent specialized fields such as data about movies and their features and staff. The second dimension, *polysemy*, indicates how much the terms are ambiguous. Trees with high polysemy contain terms with very different meanings: For instance, `rock` and `track` whose meanings radically change in different contexts. On the other hand, trees with low polysemy contain mostly terms whose senses are characterized by subtle shades of meaning, such as `title`. For each feasible combination of these properties we formed a group by selecting the three most representative trees. Group1 is characterized by a low specificity and a polysemy which increases along with the level of the tree; it is the case of web directories in which we usually find very different categories under the same root and a low polysemy at low levels and high polysemy at the leaf level. The trees we selected for Group1 are a small portion of Google™'s and Yahoo®'s web directories and of eBay®'s catalog. Group2 is characterized by a high specificity and a high polysemy; we chose structures extracted from XML documents of Shakespeare's plays, Internet Movie Database (IMDb®, `www.imdb.org`) and a possible On Line Music Shop (OLMS). Finally, Group3 is characterized by a high specificity and a low polysemy and contains representative XML schemas from the DBLP and SIGMOD Record scientific digital libraries and the Dublin Core Metadata Initiative (DCMI®, `dublincore.org`) specifications. Low specificity and high polysemy are hardly compatible, therefore we will not consider this one as a feasible combination.

Tab. 1 shows the features of each tree involved in our experimental evaluation. From left to right: The number of terms, the mean and maximum number of terms' senses, the percentage of correct senses between all the possible senses and the average similarities among the senses of each given term in the tree (computed by using a variant of Eg. 1). Notice that our trees are composed by 15-40 terms. Even though they are not particularly big, their composition allows us to generate a wide and significant variety of graph contexts. The other features are instead important in order to understand the difficulty of the disambiguation task: For instance, higher is the number of senses of the involved terms more difficult will be their disambiguation. The mean number of senses of Group2 and Group3 is almost double than that of Group1, thus we expect their disambiguation to be harder. This is confirmed by the percentage of correct senses between all the possible senses, which can be considered an even more significant "ease factor" and is higher in Group1. The last feature partially expresses how the trees are positioned w.r.t. the polysemy dimension: the higher is the average of the sense similarity the lower is the polysemy and the different senses have a closer meaning. This is true in particular for Group1 and Group3 trees, confirming the initial hypothesis.

Table 1: Features of the tested trees

	# terms	# senses		Perc.	Sense
		mean	max	correct	simil.
eBay	16	3.062	8	0.327	3.321
Google	23	3.522	11	0.296	3.201
Yahoo	15	2.733	6	0.366	3.372
Group1	18.000	3.106	8.333	0.330	3.298
IMDb	41	3.854	10	0.278	2.991
OLMS	21	6.286	17	0.159	2.31
Shakes.	15	8	29	0.133	2.152
Group2	25.667	6.047	18.667	0.190	2.484
DBLP	14	5.429	11	0.224	2.6
DCMI	17	5	10	0.235	2.983
Sigmod	18	6.444	13	0.198	2.901
Group3	16.333	5.624	11.333	0.219	2.828

4.2 Effectiveness evaluation

In our experiments we evaluated the performances of our disambiguation algorithm mainly in terms of effectiveness. Efficiency evaluation is not crucial for a disambiguation approach and is beyond the goal of this article so it will not be deepened (in any case, the disambiguation process for the analysed trees required at most few seconds). Traditionally, wsd algorithms are evaluated in terms of precision and recall figures [11]. In order to produce a deeper analysis not only of the quality of the results but also of its possible motivations w.r.t. the different tree scenarios, we considered the precision figure along with a number of newly introduced indicators. Recall parameter is not considered because its computation is usually based on frequent repetitions of the same terms in different documents, and we are not interested in evaluating the wsd quality from a single term perspective.

The disambiguation algorithm has first been tested on the entire collection of trees using the default graph context: all the terms in the tree. Fig. 6 shows the precision results for the disambiguation of the three groups. Three contributions are presented: The graph context one (Graph), the expanded context one (Exp) and the combined one (Comb). In general, precision P is the mean of the number of terms correctly disambiguated divided by the number of terms in

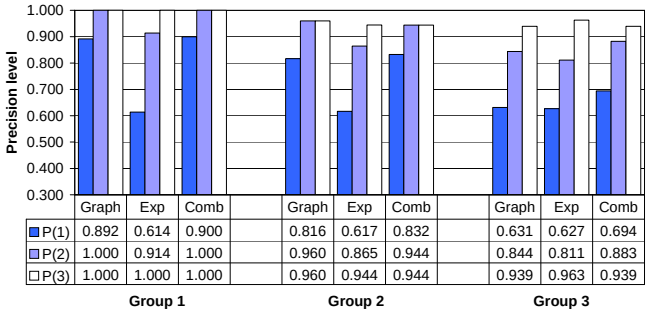


Figure 6: Mean precision levels for the three groups

Table 2: Delta values of the selected senses

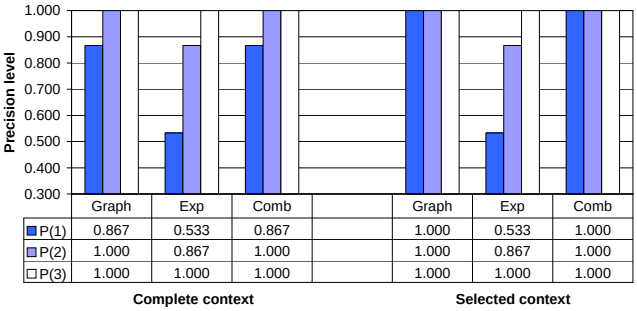
	Delta to last (rank)			Delta (conf)	
	rank1	rank2	rank3	to foll.	from top
eBay	2.154	0.667	-	0.307	-0.043
Google	2.5	2	-	0.244	-0.003
Yahoo	1.733	-	-	0.184	0
Group1	2.129	1.333	-	0.245	-0.015
IMDb	2.444	2.667	-	0.14	-0.017
OLMS	4.118	10.5	-	0.171	-0.02
Shakes.	9.125	2.2	-	0.171	-0.042
Group2	5.229	5.122	-	0.161	-0.026
DBLP	3.4	6.667	-	0.142	-0.039
DCMI	3.273	4	5	0.125	-0.039
Sigmod	5.5	4.375	-	0.168	-0.035
Group3	4.058	5.014	5	0.145	-0.038

the trees of each group. Since we have at our disposal complete ranking results, we compute precision P(M) at different levels of quality, by considering the results up to the first M ranks: For instance, P(1) will be the percentage of terms in which the correct senses are at the top position of the ranking. Combination of graph context and expanded context contributions produces good P(1) precision levels of 90% and of over 83% for groups Group1 and Group2, respectively. Precision results for Group3 are lower (nearly 70%), but we have to consider the large number and higher similarity between the senses of the involved terms; even in this difficult settings, the results are quite encouraging, particularly if we notice that P(2) is above 88%. As to the effectiveness of the context expansion, notice that its contribution alone (Exp) is generally very near to the graph context one, particularly in the complex Group3 setting, meaning a good efficacy of this approach too; further, in all the three cases the combination of the two contributions (Comb) produces better results than each of the contributions alone. This is achieved by using optimal values for the α, γ (0.7) β, and ϵ (0.3) weights, as obtained from a series of exploratory tests.

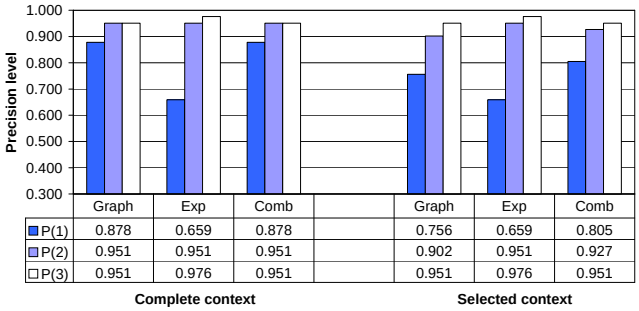
The next step was to evaluate the different behaviors of the trees disambiguation by varying the composition of their terms' context. We tested an extensive range of combinations for all the available trees, with "selected" contexts including only ancestor, descendant and/or sibling terms, and discovered two main behaviors: Group1 trees respond well to a more careful context selection, while Group2 and Group3 show an opposite trend. Fig. 7 shows two illustrative comparisons between complete and selected contexts for Yahoo tree (Group1, Fig. 7-a) and IMDb tree (Group2, Fig. 7-b). Notice that, in the first case, the combined precision P(1) raises from 86% to a perfect 100% for a selected setting involving only ancestors, descendants and siblings. This is due to the fact that Group1 concepts are very heterogeneous and including in the context only directly related terms reduces the disambiguation "noise" produced by completely uncorrelated ones. For instance, when the complete Yahoo tree is used to disambiguate the term `hygiene` in the `health` category, the top sense is that related to the health science as the process is wrongly influenced by terms like `neurology` and `cardiology` contained in the `medicine` category. Instead, when the tree terms are specific and more contextualized, such as in the other two groups, the result is the opposite: Notice the IMDb combined precision dropping from nearly 88% to 80% when only ancestors and descendant terms are kept (Fig. 7-b).

Precision figures are the fundamental way to evaluate a wsd approach, however, in our case, we wanted to analyze the results more in depth and from different perspectives. For instance, precision P(1) might be high thanks to the effectiveness of the approach but also for the possibly small number of senses of the involved terms (think of terms with just one sense). In order to deepen our analysis, we computed additional "delta" parameters (see Tab. 2): The left part of the table shows delta values between rank positions, while the right part shows delta values between confidences. Delta rank values express the mean difference between the position in the ranking of the correct sense and that of the last one; we computed them when the right senses appear in the first (rank1 in table), second (rank2) and third (rank3) position. For a given rank, we indicate by a '-' the situation where there are no correct senses with that rank. In general, the higher the "delta to last" rank value is, the harder the disambiguation task should be. At a first glance, Group2 and Group3 confirm their inherent complexity w.r.t. Group1, where rank1 delta values are nearly double. Also notice the very high rank1 delta of some trees, such as the Shakespeare one, meaning that our approach correctly disambiguates also terms with very high number of senses. Further, we wanted to analyze the actual confidence values and, in particular: How much the right senses' confidences are far from the incorrect ones, i.e. how much the algorithm is confident in its choices (delta confidence to the followings, first column of the right part of the table), and, when the choice is not right, how much the correct sense confidence is far from the chosen one (delta confidence from the top). We see that the "to the followings" values are sufficiently high (from 14% of Group3 to over 24% of Group1), while the "from the top" ones are nearly null, meaning very few and small mistakes. Notice that the wsd choices performed on Group1, which gave the best results in terms of precision, are also the most "reliable" ones, as we expected.

In Tab. 2 we showed aggregate delta values for each group, however we also found interesting to investigate the visual trend of the delta confidences of the terms of a tree. Fig. 8 shows the double histogram composed by the delta to the followings (top part) and the delta from the top (bottom part) values, where the horizontal axis represents the 21 terms of the On Line Music Shop tree. Notice that for two terms no contributions are present: This is due to the fact that these terms have only one available sense and, thus, their disambiguation is not relevant. Further, the graph shows that,



(a) Typical Group1 behavior (Yahoo tree)



(b) Typical Group2 behavior (IMDb tree)

Figure 7: Mean precision levels comparison between complete context (whole tree) and a selected context

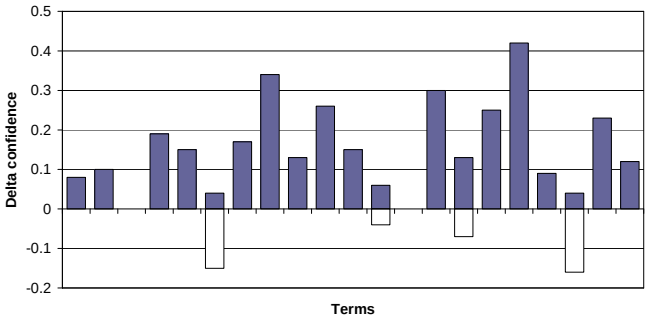


Figure 8: Confidence delta values for OLMS

when the upper bars are not particularly high (low confidence), the bottom bars are not null (wrong disambiguation choices), but only in a very limited number of cases. In most cases, the upper bars are evidence of good disambiguation confidence and reliability, with peaks of over 40%.

Up to now we have not considered the contribution of the terms/senses' feedback to the overall effectiveness of the results, in particular in the disambiguation of the most ambiguous terms in the tree. For illustration, suggesting the correct meaning of the term `volume` in the DBLP tree as a book helps the algorithm in choosing the right meaning for `number` as a periodic publication. Moreover, suggesting the correct meaning of the term `line` (part of character's speech) in the Shakespeare tree produces better disambiguation results, for instance for the `speaker` term, where the position of the right sense passes from second to first in the ranking. Notice that, in this case, and in many others, the feedback on the term merely confirms the top sense in the ranking (i.e. our algorithm is able to correctly disambiguate it); nonetheless, this has a positive effect on the disambiguation of the near terms since the "noise" produced by the wrong senses is eliminated. The flexibility of our approach allows also to benefit from a completely automatic feedback, where the results of a given run are refined by automatically disabling the contributions of all but the top X senses in the following runs. We can generally choose a very low X value, such as 2 or 3, since the right sense is typically occupying the very top positions in the ranking. For instance, by choosing $X = 2$ in the SIGMOD tree, the results of the second run show a precision increment of almost 17%, and similar results are generally obtainable on all the considered trees.

5. RELATED WORK

Before discussing the few approaches proposed for the "structural" disambiguation problem, we will first briefly review our disambiguation approach in the more "classic" and well studied field of wsd for free text. The necessity of looking at the context of a word in order to correctly disambiguate it is universally accepted, nonetheless two different approaches exist: The bag of words approach, where the context is merely a set of words next to the term to disambiguate, and the relational information approach, which extends the former with other information such as their distance or relation with the involved word. The disambiguation algorithm developed in this paper adopts the relational information approach which is more complex but generally performs much better. In the literature, a further distinction is based on the kind of information source used to assign a sense to each word occurrence [11]. Our disambiguation method is a knowledge-driven method as it combines the context of the word to be disambiguated with additional information extracted from an external knowledge source, such as electronically oriented dictionaries and thesauri. Such approach often benefits from a general applicability and is able to achieve good effectiveness even when it is not restricted to specific domains. WordNet is, with no doubt, the most used external knowledge source [6] and its hypernym hierarchies constitute a very solid foundation on which to build effective relatedness and similarity measures, the most common of which are the path based ones [12]. Further, the descriptions and glosses provided for each term can deliver additional ways to perform or refine the disambiguation: The gloss overlap approach [3] is one of them. Among the alternative approaches, the most common one is the corpus-based or statistic approach where the context of a word is combined with previously disambiguated instances of such word, extracted from annotated corpora [1, 2]. Recently, new methods relying on the entire web textual data, and in particular on the page count statistics gathered by search engines like Google [7, 8] have also been proposed. However, generally speaking, the problem of such approaches is that they are extremely data hungry and require extensive training, huge textual corpora, which are not always available, and/or a very large quantity of manual work to produce the sense-annotated corpora they rely on.

This problem prevents their use in the application contexts we refer to, as even "raw" data are not always available (e.g. in a PDMS, peers not necessarily store actual data).

Structural disambiguation is acknowledged as a very real and frequent problem for many semantic-aware applications. However, to our best knowledge, up to now it has only been partially considered in two contexts, schema matching and the XML data clustering, and few actual structural disambiguation approaches have recently been presented. In many schema matching approaches, the semantic closeness between nodes relies on syntactic approaches, such as simple string matching possibly considering its synonyms (e.g. [16, 14]). Also, a good number of statistical wsd approaches have been proposed in the matching context (e.g. [13]). However, as we already outlined, they rely on additional data which may not always be available. As to the proper structural disambiguation approaches, in [20] the authors propose a technique for XML data clustering, where disambiguation is performed on the documents' tag names. The local context of a tag is captured as a bag of words containing the tag name itself, the textual content of the element and the text of the subordinate elements and then it is enlarged by including related words retrieved with WordNet. This context is then compared to the ones associated to the different WordNet senses of the term to be disambiguated by means of standard vector model techniques. In a similar scenario, the method proposed in [19] performs disambiguation by applying a shortest path algorithm on a weighted graph constructed on the terms in the path from each node to the root and on their related WordNet terms. For the graph construction, WordNet relations are navigated just one level. In a schema matching application, [5] presents a node disambiguation technique exploiting the hierarchical structure of a schema tree together with WordNet hierarchies. In order for this approach to be fully effective, the schema relations have to coincide, at least partially, with the WordNet ones, and this appears as a quite strong requirement.

Generalizing, our approach differs from the existing structural disambiguation approaches as it has not been conceived in a particular scenario but it is versatile enough to be applicable to different semantic-aware application contexts. It fully exploits the potentialities of the context of a node in a graph structure and its extraction is flexible enough to include relational information between the nodes and different kinds of relationships, such as ancestors, descendants or siblings. Further, we fully exploit WordNet hierarchies, and in particular the hypernym ones which are the most used for building effective relatedness measures between terms in free text wsd.

6. CONCLUSIONS

Structural disambiguation is acknowledged as a very real and frequent problem for many semantic-aware applications. As to our knowledge, this is the first work which proposes a versatile approach for the disambiguation of graph-like structured information. Our main aim was to provide a significant improvement to the semantic-awareness of a wide range of knowledge based applications. The experimental results, showing the very good effectiveness of the approach, are quite encouraging and induce us to continue in this direction. In our future work, we will deeply analyse the ontology disambiguation problem, also by evaluating the performance on generic graphs, and the feedback process.

7. REFERENCES

[1] E. Amitay, R. Nelken, W. Niblack, R. Sivan, and A. Soffer. Multi-resolution disambiguation of term occurrences. In *Proc. of CIKM*, 2003.

[2] J. Artiles, A. Penas, and F. Verdejo. Word Sense Disambiguation based on term to term similarity in a context space. In *Proc. of Senseval-3*, 2004.

[3] S. Banerjee and T. Pedersen. Extended gloss overlaps as a measure of semantic relatedness. In *Proc. of IJCAI*, 2003.

[4] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, 284(5), 2001.

[5] P. Bouquet, L. Serafini, and S. Zanobini. Semantic coordination: a new approach and an application. In *Proc. of ISWC*, 2003.

[6] E. Budanitsky and G. Hirst. Semantic distance in wordnet. In *Proc. of NAACL Workshop on WordNet and Other Lexical Resources*, 2001.

[7] R. Cilibrasi and P. Vitanyi. Automatic meaning discovery using Google. Technical report, University of Amsterdam, 2004.

[8] P. Cimiano, S. Handschuh, and S. Staab. Towards the self-annotating web. In *Proc. of WWW Conf.*, 2004.

[9] H. Do, S. Melnik, and E. Rahm. Comparison of schema matching evaluations. In *Proc. of WebDB Workshop*, 2002.

[10] M. Ehrig and A. Maedche. Ontology-focused crawling of web documents. In *Proc. of SAC*, 2003.

[11] N. Ide and J. Veronis. Introduction to the Special Issue on Word Sense Disambiguation: The State of the Art. *Computational Linguistics*, 24(1), 1998.

[12] C. Leacock and M. Chodorow. Combining local context and WordNet similarity for word sense identification. In C. Fellbaum, editor, *WordNet: An electronic lexical database*. MIT Press, 1998.

[13] J. Madhavan, P. A. Bernstein, A. Doan, and A. Y. Halevy. Corpus-based schema matching. In *Proc. of ICDE*, 2005.

[14] J. Madhavan, P. A. Bernstein, and E. Rahm. Generic Schema Matching with Cupid. In *Proc. of VLDB Conf.*, 2001.

[15] F. Mandreoli, R. Martoglia, and P. Tiberio. Approximate Query Answering for a Heterogeneous XML Document Base. In *Proc. of WISE Conf.*, 2004.

[16] S. Melnik, H. Garcia-Molina, and E. Rahm. Similarity Flooding: A Versatile Graph Matching Algorithm and ist Application to Schema Matching. In *Proc. of ICDE*, 2002.

[17] G. A. Miller. WordNet: A Lexical Database for English. *CACM*, 38(11), 1995.

[18] P. Resnik. Disambiguating Noun Groupings with Respect to WordNet Senses. In *Proc. of Workshop on Very Large Corpora*, 1995.

[19] A. Tagarelli and S. Greco. Clustering Transactional XML Data with Semantically-Enriched Content and Structural Features. In *Proc. of WISE Conf.*, 2004.

[20] M. Theobald, R. Schenkel, and G. Weikum. Exploiting Structure, Annotation, and Ontological Knowledge for Automatic Classification of XML Data. In *Proc. of WebDB Workshop*, 2003.