UNSUPERVISED ALGORITHM FOR THE CONCEPT DISAMBIGUATION IN ONTOLOGIES
Semantic Rules and Voting System to Determine Suitable Senses

Isaac Lera, Carlos Juiz and Ramon Puigjaner
Departament Matemàtiques i Informàtica, Universitat de les Illes Balears, crt. Valldemossa 7.5km, Palma, Spain

Keywords: Word sense disambiguation, Ontology, Semantic web, Ontology matching.

Abstract: We present a new unsupervised algorithm which uses external resources and does not require any training to determine the sense more suitable of an ontology concept. We try to find out lexical coincidences among terms of both resources: ontology and WordNet. Through a voting system, we give weight each sense according to measurable parameters and logic rules, in function of semantic role of each correspondence element.

1 INTRODUCTION

Word Sense Disambiguation (WSD) consist on "determine which of the senses of an ambiguous word is invoked in a particular use of the word". These techniques are used across several applications: machine translation, data search, knowledge discover, data integration, etc., where one of their end goals is an effective processing of overwhelming amounts of data.

We have decided to discover the sense of ontology concepts which are represented by Ontology Web Language (OWL), standard of W3C. We focus on this representation for the following reasons: one of them is implicit in one goal of the Semantic Web (SW) to relate the meaning with data in an unequivocal way. OWL is a formal language with a logic structure which is highly related with other ontology elements: concepts, axioms, instances, properties, and individuals -it provides new mechanisms to relate elements-, and other reason is the numerous projects, applications and resources based on SW that arise every day. Ontology Matching (OM) is a vital subtask of Ontology Engineering, numerous tasks depend on it: edition, query processing, ontology and data repository managing, and visualization. OM are a set of techniques to establish relationships among ontology elements that they share some common meaning. For that, when we map two elements is necessary to find out their meaning applying WSD techniques.

2 RELATED WORK

We mention some works that serve us to introduce this work. Liu *et al.*'s approach (Liu et al., 2005) have some overlaps with our approach. They exploit from WordNet links as synonyms, hyponyms, hyperonyms and synonyms' definitions to determine the sense of main query words. Their system checks 4x4 coincidences across query words, instead our algorithm considers 8x8 matches with extra semantic rules and semantic mapping with ontologies. When they found coincidences in the apparition of words with same name then they assign the corresponding senses of these words. We have thought up a voting system where the elements and semantic of the correspondence determine the weight of each sense.

Banek *et al.* (Banek et al., 2008) present a "CSD" algorithm that increases the efficiency of Ontology Matching process. They calculate a probability function comparing the taxonomy of the ontology with a portion of taxonomy of WordNet, establishing a correspondence among the name of ontology class and WordNet noun. The neighbourhood of a class within each portion of taxonomy is based on next links: direct subclasses and superclasses, ranges of its own properties, and ranges where property concept is part of the range. In our case, we use most of ontology relationships regard to the concept, i.e. equivalent classes or transitive properties, individuals, all POS categories of WordNet, among others.

Khelif *et al.* (Khelif et al., 2008) developed a algorithm for CSD based on distance that they defined on ontology data. This distance is computed by weight-

Lera I., Juiz C. and Puigjaner R..
UNSUPERVISED ALGORITHM FOR THE CONCEPT DISAMBIGUATION IN ONTOLOGIES - Semantic Rules and Voting System to Determine Suitable Senses.
DOI: 10.5220/0003094403880391
In *Proceedings of the International Conference on Knowledge Engineering and Ontology Development* (KEOD-2010), pages 388-391
ISBN: 978-989-8425-29-4

ing the edges of the path between both classes. This path is a combination among hierarchical links, the domain and range of object properties. They tested their approach in the extraction of annotations from the Unified Medical Language System (UMLS) obtaining an highest average precision. In our case, we can apply our algorithm in any open discourse with generic ontologies. Castano *et al.* (Castano et al., 2003) presented an ontology matching algorithm where the semantic affinity between two concepts is evaluated in function of their relationships in the thesaurus and in their contexts.

Most of WSD approaches are evaluated applying a standard data set benchmark, called Senseval (Kilgarriff, 1998). The second version of Senseval contains a set of corpus, lemmas and instances where authors can test the robustness and precision of theirs algorithms. In our case, we can not apply this benchmark since our data source have to be represented with OWL language.

3 DISAMBIGUATION CONCEPT PROCESS

Our idea behind this algorithm is based on the capacity of finding correspondences between elements of both structures, and later the weighting each correspondence according a vote system that combines following information giving an assessment: lexical function among elements, semantic function of the correspondence, and the ambiguity of both elements. Thus, we have defined a category of elements that we can find in WordNet and the ontology and also, the voting system of each correspondence. Both tasks are complemented establishing the context that it reduces the search space and increases the accuracy and computational cost effectiveness.

3.1 Elements

In an OWL ontology, we manage only concepts, but the rest of elements (properties, axioms, and instances) have an influence on voting system. An OWL concept either can be simple like: *dog* and *hotdog* or can be composed of two terms like: *WarmTemperature* and *PizzaTopping*. For that reason, a concept has one o more possible equivalences in WordNet (*iff* it is right spelling). A tokenization process which is based on capitalisation letters (i.e. *PizzaTopping*, *hotAir* but not *hotDog*) and special characters (i.e. *hot_dog*, *hot#Temperature*) splits the words of a concept. If the concept has two words then it will be manage like two different words in the algorithm. All user definitions of ontology elements go through a tokenization (concepts, individuals and properties) and stemming process (properties), both tasks increase the flexibility degree of our algorithm.

Anyway, a simple or composed concept is transformed in a word with meaning in WordNet. Each word in WordNet has a set of senses. Each sense has a set of synonyms/antonyms, hyperonym/hyponyms, meronyms/holonyms, and one definition (gloss) which adds more terms. Glosses' terms are previously filtered and stemmed since we avoid unnecessary analysis (i.e. articles, prepositions) and loss by noun coincidences (i.e. plurals, verb forms, etc.), respectively.

3.2 Nomenclature

To ease the reading, a concept from ontology is defined by C, each concept in WordNet has a set of simple words: $C \equiv sW_i$. Each simple word has a set of senses: $wS_i : S_i$. Hereafter, i-indices are independent among elements. Each sense has a group of gloss' terms: $wS_i : S_i : t_i$, that is equivalent to $t_i : S_i : wS_i$. Also, each sense has a set of semantic WordNet constructors: $S_i : Hpon_i, Hper_i, Syn_i, Ant_i, Mer_i, Hol_i$. Thus, we could say that ":" means "has", and i-index means "element of set". Each, $Hpon_i, Hper_i$, etc. can be considerer like a sW_k, being a recursive representation.

3.3 Correspondences

Elements can generate a correspondence when its lexical coincide with other noun. We save this correspondence: both elements and their semantic function. The function of each term is provided by the semantic function in WordNet (syn., hyper., etc.) and in the ontology (superclass, subclass, equivalent and disjoint classes, and properties).

Only there will be a correspondence when we can obtain at least one sense from both concepts. For example, a matching between $simpleWords\ sWi = sWk$ is not useful for our case. Instead, in a term relation $sW_i : S_i : t_i{=}t_j : S_j : sW_j$ both senses are present. Also, this case $sW_i : S_i : Syn_i = sW_j$ is valid. Therefore, there are 8x8 possible combinations for each two simple words.

3.4 Voting System

Each correspondence is a vote and each vote has a different weight according to the semantic of elements involved and the ambiguity of their senses. For example, a concept ($C_i \equiv sW_i$) has a direct or indirect superclass ($C_i \subset C_j$) in the ontology. We can search this superclass concept(its equivalence in WordNet sW_j) trough a route among all senses and their hyperonyms across the whole tree of WordNet $sWi : \{S_i : Hper_i\}^{recursive\ search} = ?(Syn_j|sW_j) : C_j$. The ambiguity of simple words has an important role in this voting system, since an concept that has a relationship (i.e. equivalentClasses, disjointClasses) with another concept that it has one sense then this sense either has more weight or modifies all senses of that concept.

There are some type of cases that can simplify the determination of senses. We describe them only generically due to space limit:

- **Hierarchical Cases.** When a concept has a parent or a child, we can search the possible parents/children of this concept in WordNet tree. When we found this coincidence, we have identified the pair of senses that it complies this condition.
- **Equivalent & Disjoint Cases.** When a concept has an equivalent class, then we can determine one of them has only one sense and then put this one sense in the other word.
- ***Restrictions, Object & DataType Properties* Cases.** We difference two kind of data: property signature (property name, domain, and range) and semantic function. Property signature works like a frequency vote. In object properties, the semantic function permits to define specific rules when there are other correspondences among this concepts. Only transitive properties have been considered since that can relate other concepts with some information. Other restrictions or constructors are `unionOf` and `complemenfOf`. Both votes works like previous case of transitive property.
- ***Individuals* Cases.** Sometimes ontology designer introduces some words or entities like `individual` of a generic class. Our approach manages this type of cases where these correspondence receives more weight since individuals are some specific elements of something particular.
- **Normal Cases.** Finally, we consider the rest of correspondence like frequencies with different weights in function of each simple word's ambiguity. For example, a term correspondence $sW_i : S_i : t_i{=}t_j : S_j : sW_j$ increases the weights of both simple words with a constant value. If any simple word is not ambiguous then this weight is bigger than initial value. We consider that the relationships with non ambiguous words defines strong semantic links.Other cases of interest are: synonym correspondence $sW_i : S_i : Syn_i{=}Syn_j : S_j : sW_j$ both words share same sense, we assume that both concepts are equivalents; $sW_i : S_i : Ant_i{=}Syn_j : S_j : sW_j$ both concepts are disjoint and this correspondence will not be dealt.

3.5 Search Space

Some ontology elements are more related than other elements. These relationships make possible to discover quickly new correspondences and senses. Thus, we have opted to split the process in two parts: one part discovers the sense of the candidates concepts (concepts more related) and second part, the senses are discovered by the algorithm that it knows the sense of previous elements. This selection of concepts decreases the search space and reduces computer resources consumption. Our selection of predominant concepts is based on our previous work (Lera et al., 2008), where we have modified the idea that is presented in that paper applying clustering algorithms instead a simple formulas. The clustering parameters are the depth relative (the highest depth and its depth according with its children), the number of direct subclasses, the number of relationships with range on self and the number of individuals.

4 DEVELOPMENT

To avoid the consumption of resources, we have developed a procedure where all concept are analysed in function of its importance within context. Furthermore, during the analyse process we save data from future possible concepts, avoiding unnecessary computations. We have used a hash index that clusters with coincident elements. When a new element appears we add the element in this structure, whether proceed we create the correspondence or we modify the senses according rules avoiding unnecessary searches around other branches.

When we try to find coincidences in the multiples branches of hierarchies we apply a breadth first search (BFS) algorithm with a threshold of depth. Also, the algorithm of Stemming we have used the Lancaster Algorithm[1].

5 EVALUATION

To evaluate our algorithm, it is necessary the availability ontologies of reference, it is to say, that these ontologies are handmade specifically for disambiguation benchmarking. In the absence of that, we use

[1]www.comp.lancs.ac.uk/computing/research/stemming/

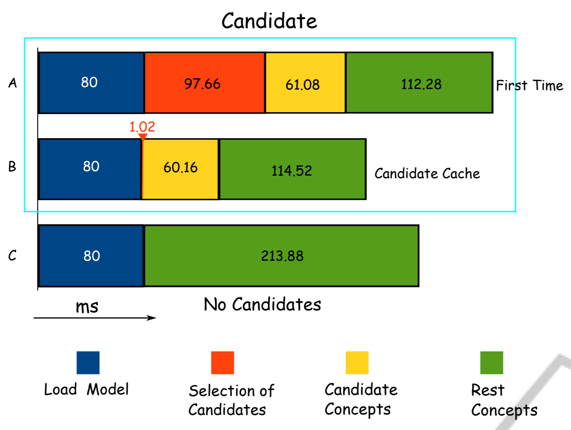

Figure 1: Algorithm's response time.

datasets from Ontology Alignment Evaluation Initiative (OAEI) such as: anatomy, conference, directory, etc. We have used the conference dataset[2], it is a collection of ontologies describing the domain of organising conferences. We have analysed all of them and we have taken time response measures each part of our algorithm. The accuracy is checked manually and practically results are identical in both situations.

Instead, the mean of response times change whether we compute the candidate concepts the first time (fig. 1: A-row), or whether we have already available this set of candidates (B-row). Also memory consumption is lower in this case than in the others. And finally, C-Row is a "normal" execution dealing all concepts in the same way.

6 CONCLUSIONS

We have presented an algorithm that is able to determine the sense of ontology concepts. To do that, all OWL constructors are considered and are mapped with WordNet structure to achieve a formal way to compare elements. All correspondences are dealing with the semantic function that have the elements involved. Some rules are triggered in special cases, for example, when we have equivalent concepts or transitive properties. In comparison with other approaches we have incorporated more resources with semantic knowledge to increase process efficiency in terms of success and computational cost. We do not be able to compare our results with another works since our input data are ontologies instead documents or other types of unstructured documents. We have noticed that if both structure WordNet-ontology and their names are similar, then the number of correspondences and rules triggered decreases considerably the algorithm execution time. Results encourage us to exploit some specific OWL constructors as equivalents complex axioms of Descriptive Logic and improving individual rules of each case.

[2]http://nb.vse.cz/ svabo/oaei2010/

ACKNOWLEDGEMENTS

This work is partially supported by the project *TIN2007-60440* from Spanish Ministry of Science and Innovation.

REFERENCES

Banek, M., Vrdoljak, B., and Tjoa, A. M. (2008). Word Sense Disambiguation as the Primary Step of Ontology Integration. In *Proceedings of the 19th international conference on Database and Expert Systems Applications*, pages 65–72. Springer-Verlag.

Castano, S., Ferrara, A., and Montanelli, S. (2003). H-match: an algorithm for dynamically matching ontologies in peer-based systems. In *In Proc. of the 1st Int. Workshop on Semantic Web and Databases (SWDB) at VLDB 2003*, pages 231–250.

Khelif, K., Gandon, F., Corby, O., and Dieng-Kuntz, R. (2008). Using the Intension of Classes and Properties Definition in Ontologies for Word Sense Disambiguation. In *Proceedings of the 16th international conference on Knowledge Engineering*, pages 188–197, Berlin, Heidelberg. Springer-Verlag.

Kilgarriff, A. (1998). SENSEVAL: An Exercise in Evaluating Word Sense Disambiguation Programs. In *Proceedings of the First International Conference on Language Resources and Evaluation*, pages 581–588.

Lera, I., Juiz, C., and Puigjaner, R. (2008). Quick Ontology Mapping Algorithm for distributed environments. In *Semantic Web and Web Services*, volume 1, pages 107–113. CSREA Press.

Liu, S., Yu, C., and Meng, W. (2005). Word Sense Disambiguation in Queries. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 525–532. ACM.