An Approach to Knowledge-Based Word Sense Disambiguation Using Semantic Trees Built on a WordNet Lexicon Network

Andrei MINCĂ*, Ştefan DIACONESCU*
* Department of Research and Development, SOFTWIN
Bucharest, Romania
aminca@softwin.ro, sdiaconescu@softwin.ro

***Abstract*—This paper proposes a knowledge-based WSD (Word Sense Disambiguation) method derived from the Lesk algorithm. The proposed method considers an extension of the definition domain of the Lesk algorithm by creating a lexicon network from tagged lexicon glosses. We present several methods that adjust the lexicon network in order to better describe the natural language. Further, on the pre-processed lexicon network we build competence and definition semantic trees for each sense that will be used to compute costs of semantic similarity between words senses. For this purpose we use a WSD window limited to a phrase, and a similar reasoning for larger contexts. For testing we apply our methods to the recently WordNet tagged glosses.**

Keywords-Lesk algorithm, knowledge-based Word Sense Disambiguation, WordNet tagged glosses, lexicon network, competence and definition semantic trees

I. INTRODUCTION

Most languages, either formal or natural languages, have several ambiguous constituents: words, symbols, tokens.

Word sense disambiguation (WSD) is the process of establishing the concept which a given use of a word represents. WSD is done usually in natural language processing systems by examining semantic domains for each word with restriction mechanisms. The flexibility of natural languages increases the complexity of this process. Doing this automatically makes it even harder.

However, noticeable progress has been done in this matter. Most important is the development and formalization of knowledge-based lexicons. Machine-readable dictionaries are available for the majority of natural languages. More complex lexicons store a variety of semantic relations between lexicon entries from one or more languages. WordNet is the most popular for English. Moreover, EuroWordNet is a joint effort for European languages. Knowledge-based lexicons combined with several methods and algorithms, that harness the machine's computing power, transform this problem into a more appealing one.

In this paper we describe a method for knowledge-based WSD that makes use of machine-readable dictionaries and a certain type of semantic relation. Our method is also an "All-words WSD method". We propose the use of semantically tagged glosses to create a lexicon network. Once available, the lexicon network provides means to better describe the relations between word senses. We consider building semantic trees for each word sense using the lexicon network. These semantic trees will be used to establish similarity scores, between senses, which will represent the main source for WSD.

II. PREVIOUS WORK

In [1] we find the most general and well-known attempt to utilize information in machine-readable dictionaries for WSD, that of Lesk (1986), which computes a degree of overlap--that is, number of shared words--in definition texts of words that appear in a ten-word window of context. The sense of a word with the greatest number of overlaps with senses of other words in the window is chosen as the correct one. The Lesk algorithm is also a knowledge-based WSD method.

Many variations of the Lesk algorithm have been proposed, including: i) versions of the algorithm that attempt to solve the combinatorial explosion of possible word sense combinations when more than two words are considered, ii) algorithm variations where each word in a given context is disambiguated individually, by measuring the overlap between its corresponding dictionary definitions and the current sentential context, and iii) alternatives where the semantic space of a word meaning is augmented with definitions of semantically related words. All versions reported encouraging results.

For example, in [2] the Lesk algorithm is adapted to the online lexical database called WordNet [3], freely distributed by Princeton University. Besides storing words and their meanings, WordNet also defines a rich set of relationships between words. For example it says that a dog *is a* canine which *is a* carnivore, etc. It tells us that to bark *is one way* to interact, and that one sense of bark *is a kind of* a covering. It says that pine and cone are two *kinds of* coniferous trees while sandwiches form *a part of* breakfast. The use of these relations significantly improves the disambiguation process.

Further on, in [4] and [5] a complex language for abstracting grammar, named GRAALAN, is proposed which can store even richer sets of word or word sense relations. For example, besides the usual relations already present in WordNet-like projects, there are also relation types like

e*xtension from, extension in, reduction from, reduction in, figurative of, literal for, abstract for* and others.

In [6] is presented a method for the automated disambiguation of glosses in lexical knowledge resources with a novel graph-based algorithm. This method is based on the identification of edge sequences which constitute cycles in the dictionary graph (possibly with one edge reversed) and relate a source to a target word sense. Experiments performed on WordNet and several machine-readable dictionaries showed noticeable results.

We acknowledge all this rich information and we concentrate our methods on the semantically tagged glosses relations form and a lexicon network built on such relations. These relations are freely available in the WordNet project [7].

III. SEMANTICALLY TAGGED GLOSSES

We want to improve WSD using a method similar to that of Lesk algorithm. Our method extends the definition domain of the original Lesk algorithm by making use of a specific relation type: semantically tagged glosses. In this type of relation word-forms from the definitions ("glosses") in a lexicon are, manually or automatically, linked to the context-appropriate sense in the lexicon. Thus, the glosses are a sense-disambiguated corpus against the source lexicon.

Having a sense-disambiguated lexicon we can further extend the reasoning behind the Lesk Algorithm and look for matching between senses, instead of words, used in words' definition. Thus the matching scores are refined.

We also consider looking further for sense matching by recursively expanding every sense with its definitions. The following figure contains an example:

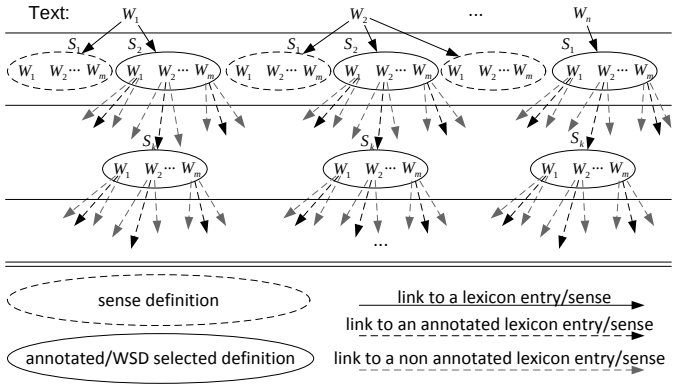

Figure 1. Lesk algorithm reasoning extended. Every annotated sense is extended with its definition that also has words with disambiguated senses and so on.

Creating this type of relations is often a time consuming task considering that in a common natural language lexicon are more or less 1.500.000 *useful* "semantically tagged glosses" relations. The *potential* number of "semantically tagged glosses" relations is in fact the total number of words found in glosses. Not all words in a gloss are considered relevant and therefore not disambiguated in a sense's gloss, resulting in a smaller number of real (useful) relations than the potential number of such relations. We also consider giving a cost to these relations according to their contribution in the context of the sense's definition, named *relevance* cost.

A. WordNet semantically tagged glosses

WordNet, besides storing words and their meanings and defining a rich set of relationships between words, has also semantically tagged glosses with sense disambiguated information. The following table provides some statistics:

TABLE I. WORDNET STATISTICS ON SEMANTICALLY TAGGED GLOSSES

Tokenized text	Multi-word forms	Taggable lemmas
Types 47334 Tokens 1621129	man 7168 auto 45967 all 53135	Types 55561 Tokens 1504077

Because WordNet is organized in synsets [2] we modified the original structure in order to obtain a lexicon like form. This was done by multiplying a synset gloss to a number equal to the synset's elements. In conclusion every synset element is now a lexicon entry with a gloss associated. Using this method, with the new glosses tagged with sense disambiguated information we identified 147.300 lexicon entries, 206.941 senses with 3.116.604 *potential* "semantically tagged glosses" relations (number of total words in glosses) between WordNet-like lexicon entries' senses from which just 936796 are *useful* relations (number of total words disambiguated, present in the new set of glosses obtained after multiplication).

IV. LEXICON NETWORK

The semantically tagged glosses as relations form a large network as an oriented graph of relations between word senses. This large lexicon network can be pre-processed further in order to offer a consistent and compact view for the word senses relations in a natural language.

Types of transformations on the lexicon network:

i) modifying the lexicon network so that it will have lesser strong connected components;

ii) reconstruct the network in order to obtain a leveled lexicon network, using different criteria; this implies removing all the strong connected components.

Both approaches can use different constraints like: sorting the senses by their relations sets, upward or downward. Each sense has two relations sets. The first relations set is formed from relations with other senses that define the current sense, named *definition relations*. The second relations set uses relations with other senses that the current sense is defining, named *competence relations*.

The second approach produces two sets of entities with special properties: *universals* - word senses situated on the top level (are not referred by any other sense), *primitives* - word senses situated on the bottom level (are not referring no other sense).

The second approach can also use other constraints for example: a minimum number of levels, a maximum number of levels, a minimum number of primitives, a minimum numbers of universals or a minimum number of deleted relations. A

combination of these can also be applied. These lexicon network transformations will be applied only once, and the resulting network will be used further in the disambiguation process. An unmodified lexicon network can also be used.

An example of a minimal lexicon network in the original form is given in figure 2.

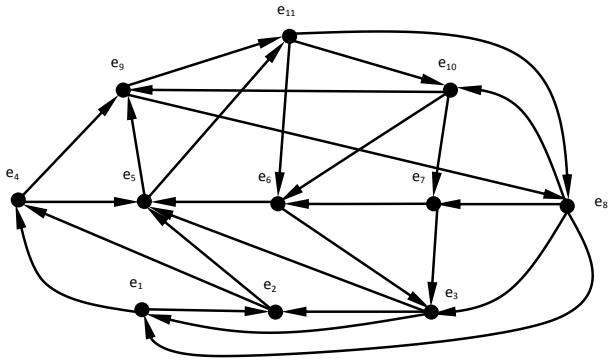

Figure 2. A minimal lexicon network in the original form

The same lexicon network, after a leveling transformation is shown below. This example has only one *universal* and one *primitive*.

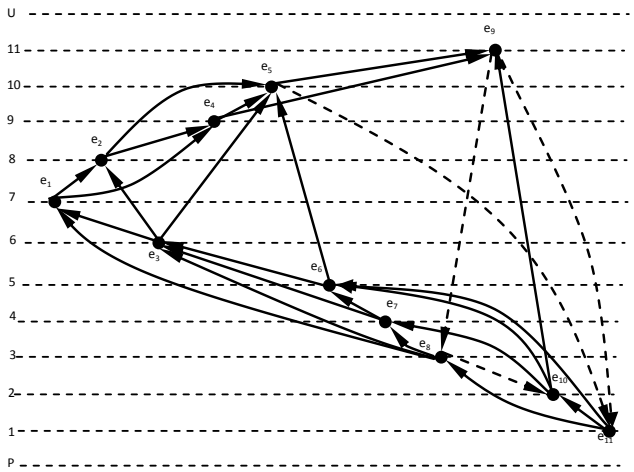

Figure 3. A leveled minimal lexicon network

V. SEMANTIC TREES

In order to obtain a larger definition domain we will use the lexicon network to build semantic trees for each word sense. A semantic tree is a perspective of sense S of the lexicon network LN. For each sense S we can build two different semantic trees: a *competence* tree and a *definition* tree.

A *competence* tree for sense S is a semantic tree build recursively from relations to senses that have sense S in their *definition* domain. In other words, it is a tree formed with senses that are directly or indirectly "*related to"* sense S.

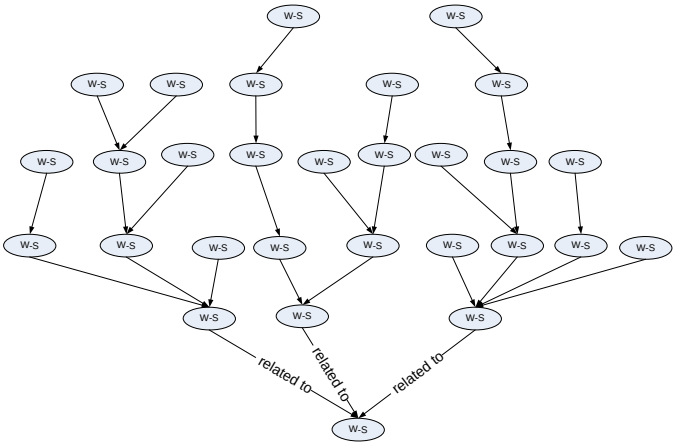

Figure 4. A word sense with the competence tree

A *definition* tree for sense S is a semantic tree build recursively from relations to senses that have sense S in their *competence* domain. In other words, it is a tree formed with senses which with sense S has a direct or indirect *"related to"* relation.

These trees can be built using several size constraints: number of nodes limit, number of levels - depth limit, or width limit.

Formal examples of such trees are given in figures 4 and 5.

These semantic trees provide an extended resource to compute similarity costs between senses.

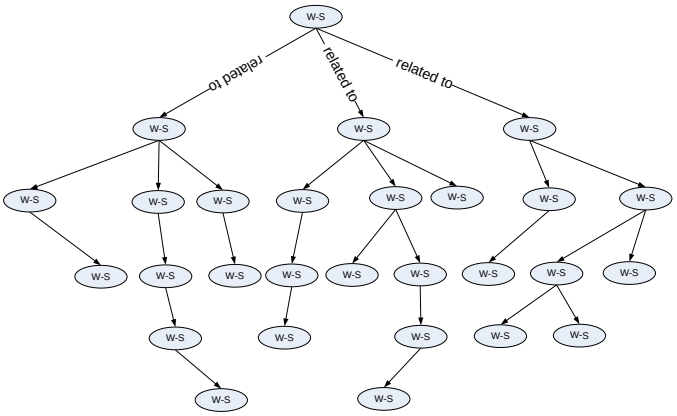

Figure 5. A word sense with definition tree

VI. COMPUTING BEST VARIANT

The final purpose of WSD is to decide the use of a certain sense for a word in a given text. To perform WSD for a given word, a subset of a certain size from the text is often used, called a window. This window is actually the context on which the WSD will be done for the targeted word. In this paper, we propose a window size equal to the phrase in which the word is found.

At first, all the senses for all the words in the phrase are identified. Pairs of senses belonging to different words are

considered. For each such pair a similarity cost is computed using the following algorithm.

The similarity cost of a pair of senses, computed as below, is a percentage value:

$$GPP_{ij} = \frac{GPT_U + GPT_D}{2} \quad (1)$$

where:

- GPT_U - represents the similarity cost/grade between the competence tree of sense *i* and the competence tree of sense *j*
- GPT_D - represents the similarity cost/grade between the definition tree of sense *i* and the definition tree of sense *j*

and GPT, is computed also as a percentage value, using the formula:

$$GPT = \sum_{l=1}^{L} GPN_l \cdot \frac{c^{L-l}}{c^L - 1} \quad (2)$$

where:

- L - number of tree levels
- l - current tree level
- c - a coefficient of relative relevance for tree levels
- GPN_l - similarity cost/grade for the current level in tree

and GPN, is computed also as a percentage value, using the formula:

$$GPN = \frac{\sum_{n=1}^{N} GR_n \cdot p}{N} \quad (3)$$

where:

- N - number of tree level nodes
- n - current node on the current tree level
- GR_n - *relevance* cost/grade (defined earlier) for the current node inside the current level
- p - has a bit value, 0 if the current node is not present in the similar sense tree and thus it has no influence, 1 if the current node was found

and GR, also a percentage value, it is the cost of the relation between the current node and the parent node.

These similarity costs for pair senses are stored in a matrix with NS * NS size, where NS is the total number of senses in a phrase.

TABLE II. SIMILARITY COSTS MATRIX FOR PAIRS OF SENSES

	W_1	W_2	W_3	...	W_N
W_1		$M_{ns_1ns_2}$	$M_{ns_1ns_3}$	...	$M_{ns_1ns_N}$
W_2	$M_{ns_2ns_1}$		$M_{ns_2ns_3}$	...	$M_{ns_2ns_N}$
W_3	$M_{ns_3ns_1}$	$M_{ns_3ns_2}$		...	$M_{ns_3ns_N}$
...	...	...	...		...
W_N	$M_{ns_Nns_1}$	$M_{ns_Nns_2}$	$M_{ns_Nns_3}$	...	

Every element in the above matrix is also a matrix with an NS_i * NS_j dimension that stores the similarity costs between senses of word *i* and senses of word *j*. This matrix also has empty elements because we do not need to compute similarities between senses of the same word. Elements from the matrix are used further on to compute the best variant of combination of senses in the current phrase. The problem can be reduced to a task of computing a complete subgraph of a maximal cost in an oriented graph, this graph is constructed from all edges between the word senses.

Therefore, we have:

- G = (V, E), G is a complete N-partite graph, N is the number of words in the current phrase.
- $V = V_1 \cup V_2 \cup \ldots \cup V_N$, where V_i is a set of nodes for partition *i*, in other words the senses of word *i* in the phrase.
- $E = \{(s_{ik}, s_{jl}) \mid s_{ik} \in V_i, s_{jl} \in V_j, i \neq j\}$, where (s_{ik}, s_{jl}) is the edge from sense *k* of word *i* and sense *l* of word *j*

Cost of the edge (s_{ik}, s_{jl}) can be computed from:

$$Cost_{(s_{ik}, s_{jl})} = MGP_{s_{ik}, s_{jl}} + MGP_{s_{jl}, s_{ik}} \quad (4)$$

or

$$Cost_{(s_{ik}, s_{jl})} = \frac{MGP_{s_{ik}, s_{jl}} * MGP_{s_{jl}, s_{ik}}}{MGP_{s_{ik}, s_{jl}} + MGP_{s_{jl}, s_{ik}}} \quad (5)$$

where $MGP_{s_{ik}, s_{jl}}$ is the matrix value at (s_{ik} , s_{jl})

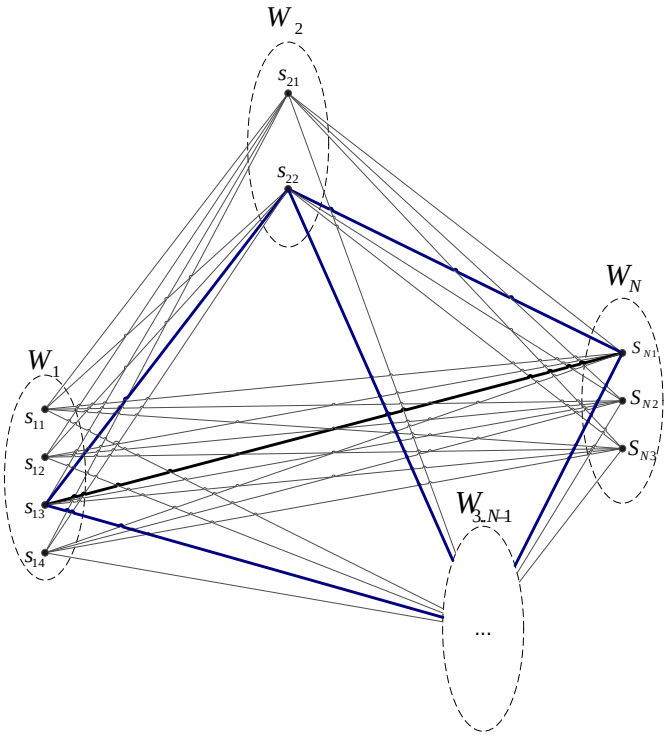

Figure 6. A graph for an N-word phrase

Figure 6 shows a N-partite graph. We highlighted an example of a best variant. There can be more than one best variant.

We define $V_S \subset V$ as a set of senses which can be a solution with $|V_S| = N$, $\forall i = \overline{0,N}$ $\exists k$ with $s_k \in V_S$ and $s_k \in V_i$. In other words, such a solution is a set of senses, one for each word. We look for that set V_S for which $\sum_{i \neq j} (s_i, s_j)$ has the highest value, where $s_i \in V_S$, $s_j \in V_S$ and $(s_i, s_j) \in E$.

We mention that N*(N-1)/2 represents the number of total edges in a WSD variant V_S . Computing all combinations is a time consuming procedure if considering a phrase with numerous words. The number of the total variants id: $\prod_{i=1}^{N} |V_i|$. For example, a phrase with 10 words, each with 5 senses in average, we will have 9765625 possible combinations. Thus, for computing the best variant we consider a backtracking similar algorithm with strong constraints. We assume that similar word senses have very high similarity costs and distinct word senses have very low similarity costs.

The algorithm starts from the following premises:

- all edges are sorted in a downward order which guarantees constructing a variant with maximal cost from the start.

It considers the following constraint:

- looking for other combination continues only if a better one can ever be found. In other words:

$$MaxVV <= \sum_{i=0}^{k-1} VE_i + VE_k * (\frac{N(N-1)}{2} - k) \quad (6)$$

where MaxVV is the current determined maximal value and VE is an edge value.

Backtracking with constraints:

step 1: order downwards the edges.

step 2: counter k (position in V_S) is set to 0 and the current combination values sum VV also set to 0.

step 3: select the next edge not yet selected with maximal cost that fits in the current variant with the k-1 edges already selected.

step 4: if there is no such edge we decrement counter k and we go to step 3.

step 5: if there is such edge we add the edge value to the variant cost.

step 6: if k#N we increment counter k and go to step 3.

step 7: if k=N check if a better solution exists than the one already found by (6), if true then k and the current variant cost are decremented until at least one word has no senses in the current variant, go to step 3; else the process stops.

VII. RESULTS

We present here our results obtained with different combination of constraints for reconstructing the lexicon network and build the semantic trees. We used WordNet semantically tagged glosses as relations and applied the methods to the same WordNet glosses.

We are aware that the results on other texts will not have a disambiguation accuracy greater than the ones on the texts in WordNet. This is so, firstly, because we used as a target test the source for our relations and secondly, because we transformed a synsets' "semantically tagged glosses" relations network in a lexicon entries senses' "semantically tagged glosses" relations network.

Our modified version of WordNet has 202144 glosses manually disambiguated. Its lexicon network has 936796 valid relations.

The usual formulas for WSD are:

- the *precision P* as the ratio between the number of target words disambiguated correctly and the total target words for which the disambiguation succeeded
- the *recall R* as the ratio between the number of target words disambiguated correctly and the total target word
- *F-measure* as a harmonic average between P and R.

TABLE III. WSD RESULTS ON WORDNET

LN Alg.	Relations Count	Global P	Global R	Global F-measure
1	936796	0.712066256018748	0.712057433924927	0.712061192955979
2	203812	0.535384423735973	0.535379421801158	0.535381568981607
3	832886	0.701633864263190	0.701624272640936	0.701628389873535
4	832886	0.701577270070855	0.701567678448601	0.701571795681200
5	832886	0.701663664359697	0.701654072737443	0.701658189970042

The above table shows our results on the WordNet using different transformations on the initial lexicon network.

Lexicon Network Algorithm 1: no operations on relations' graph-lexicon network are done.

LN Alg. 2: first we sort downward senses by their definition and competence sets and then we build a truncated lexicon network from relations by expanding every sense by one level.

LN Alg. 3: first we sort downward senses by their definition and competence sets and then we build a truncated lexicon network from relations by expanding every sense by one level in every SCC (strong connected component).

LN Alg. 4: first we sort downward senses by their competence and definition sets and then we build a truncated lexicon network from relations by expanding every sense by one level in every SCC.

LN Alg. 5: first we sort upward senses by their definition and competence sets and then we build a truncated lexicon network from relations by expanding every sense by one level in every SCC

For each different algorithm that transforms the lexicon network we may get different numbers for the relations count in the lexicon network.

VIII. CONCLUSIONS

We proposed a method of expanding the Lesk algorithm reasoning by building a lexicon network from semantically tagged glosses. Also, we presented different methods to transform the resulting lexicon network to better describe the natural language. Further, we described a method to build semantic trees for each sense using such a lexicon network and also a method to reduce the computational volume for the best variant detection.

Finally, testing our methods on WordNet we showed good results for WSD using only semantic trees similarity costs.

FUTURE WORK

Even if the computation volume was reduced by considering strong constraints on a backtracking type like algorithm we acknowledge that combinations of a large number of strong related words can be imagined.

We consider different WSD window sizes and intelligent methods of combining words thus creating context groups for WSD that will dramatically reduce the computation volume.

REFERENCES

[1] M. Lesk, "Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone," Proceedings of SIGDOC '86, 1986.

[2] S. Banerjee and T. Pedersen, "An adapted Lesk algorithm for word sense disambiguation using WordNet," Proceedings of the Third International Conference on Intelligent Text Processing and Computational Linguistics, Mexico City, 2002.

[3] WordNet at Princeton University, http://wordnet.princeton.edu

[4] Șt. Diaconescu and I. Dumitrașcu, "Complex Natural Language Processing System Architecture," Advances in Spoken Language Technology, The Publishing House of the Romanian Academy, Bucharest, 2007, pp. 228-240.

[5] Șt. Diaconescu,"Natural Language Syntax Description using Generative Dependency Grammar," POLIBITS 38, 5-18, July-December 2008, ISSN: 1870-9044.

[6] Semantically Tagged glosses for WordNet, http://wordnet.princeton.edu/glosstag.shtml.

[7] R. Navigli, "Using Cycles and Quasi-Cycles to Disambiguate Dictionary Glosses", EACL '09 Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics, Stroudsburg, PA, USA, March 2009.

[8] R. Navigli, "Word sense disambiguation," A survey. ACM Comput. Surv. 41, 2, Article 10, February 2009.

[9] S. P. Ponzetto, R. Navigli, "Knowledge-rich Word Sense Disambiguation rivaling supervised systems," Proc. of the 48th Annual Meeting of the Association for Computational Linguistics (ACL), pp. 1522-1531, 2010.