Can Syntactic and Logical Graphs help Word Sense Disambiguation?

Amal Zouaq, Michel Gagnon, Benoit Ozell
Ecole Polytechnique de Montréal, C.P. 6079, succ. Centre-ville, Montréal (Québec) H3C 3A7
E-mail: {amal.zouaq, michel.gagnon, benoit.ozell}@polymtl.ca

Abstract

This paper presents a word sense disambiguation (WSD) approach based on syntactic and logical representations. The objective here is to run a number of experiments to compare standard contexts (word windows, sentence windows) with contexts provided by a dependency parser (syntactic context) and a logical analyzer (logico-semantic context). The approach presented here relies on a dependency grammar for the syntactic representations. We also use a pattern knowledge base over the syntactic dependencies to extract flat predicative logical representations. These representations (syntactic and logical) are then used to build context vectors that are exploited in the WSD process. Various state-of-the-art algorithms including Simplified Lesk, Banerjee and Pedersen and frequency of co-occurrences are tested with these syntactic and logical contexts. Preliminary results show that defining context vectors based on these features may improve WSD by comparison with classical word and sentence context windows. However, future experiments are needed to provide more evidence over these issues.

1. Introduction

The task of word sense disambiguation (WSD) can be regarded as one of the most important tasks for natural language processing applications including semantic interpretation of texts, semantic web applications, paraphrasing and summarization. One issue of current word sense disambiguation methods is that the most successful techniques are supervised, which means that annotated corpora should be available to train the systems. However, this kind of data is heavy to produce and cannot be created for each new domain to be disambiguated. This indicates that more efforts should be put on unsupervised word sense disambiguation techniques. Furthermore, one vital issue that should generally be solved for this kind of systems is the choice of an adequate context. Usually, this context is defined as a window of words or sentences around the word to be disambiguated. The question raised by this paper is whether defining this context using syntactic and logical features can be beneficial to WSD. This paper briefly presents a natural language processing pipeline that outputs logical representations from texts and disambiguates the logical representations using various WSD algorithms. The paper also presents different context definitions that are used for WSD. Preliminary results show that logical and syntactic features can be of interest to WSD. The main contribution of this paper is the use of syntactic and semantic information for WSD in an unsupervised manner.

The paper is organized as follows: First, section 2 explains the pipeline that creates logical representations and presents the various WSD algorithms and the contexts used in this study. Section 3 presents experiments that are conducted over a small corpus and shows preliminary results. It also describes the results of our system on the Senseval English lexical Sample Task before drawing a conclusion.

2. State of the Art

There have been previous works in the WSD and Semantic Role Labeling (SRL) communities (Tanaka et al., 2007) (Merlo and Musillo, 2008) that try to incorporate syntactic and lexical information for WSD and SRL. For e.g. (Tanaka et al., 2007) show that exploiting rich semantic information improves the precision of the results by 2- 3%. Their approach uses a machine learning algorithm which is trained over a TreeBank, and an HPSG lexicon which describes syntactic and semantic features for each lexeme. Such a supervised approach may provide interesting results, but it requires resources that are costly to acquire. Our aim is to define contexts based on syntactic and logical features, but without necessarily resorting to a supervised WSD or to linguistic resources.

3. Prototype Implementation

The system is built using a modular design and is intended to be as generic and reusable as possible. It is composed of a syntactic analyzer, the dependency module of the Stanford Parser (De Marneffe et al., 2006), a logical analyzer based on dependency grammars and finally a WSD module.

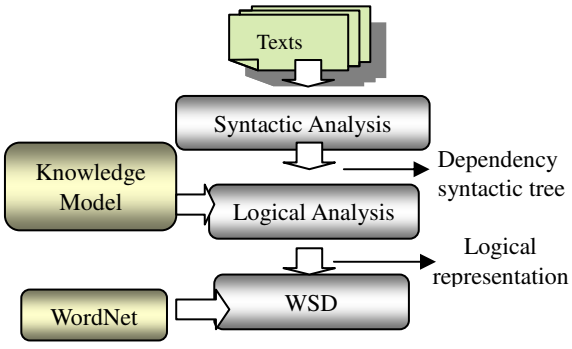

Figure 1. The prototype pipeline

Using the dependencies (the grammatical structure) created by the Stanford parser (De Marneffe et al., 2006),

and based on the work of (Zouaq, 2008), we developed a Prolog logical analyzer that searches specific patterns in the grammatical structure and that transforms them into a flat predicative logical representation. This representation relies on general and universal categories that are found in all the semantic role labelling systems: event, statement, entity, named_entity, attribute, time, measure, circumstance, etc. With these categories, it is easy to express various information contexts and to remain independent from a particular domain. An example of a logical representation based on these categories is shown below (figure 2). The figure also shows a simple pattern: nsubj(Verb, Subject).

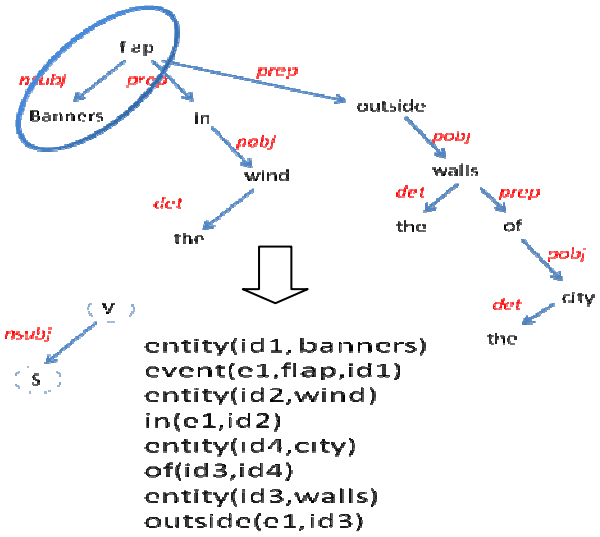

Figure 2. A dependency syntactic representation and the resulting logical representation

Although it is possible to create logical representations using only lexical items, we believe that these universal categories can help WSD and represent a step towards the sentence semantics. Starting from these logical expressions (Figure 2), the system should then be able to create a semantic representation using a specific set of roles (for WSD). Since we propose a modular framework, independent from a particular role terminology, we chose the well-known lexical dictionary WordNet (Fellbaum, 1998). However, all the resources used here are interchangeable and it is possible to use ontologies instead of WordNet for example (Zouaq et al., 2010). Moreover, it is also possible to use another dependency parser with minimal changes in the logical analyzer.

3.1. Logic-based Word Sense Disambiguation

Using the kind of pipeline proposed in figure 1, we can define the task of WSD as a three-phase process, requiring the syntactic parsing, the identification of the sentence logical structure (predicates, arguments) and finally the categorization of the obtained structure, which represents the actual WSD of the sentence constituents.

An example of a WordNet-based disambiguation for the logical representation presented above is shown below.

outside(e1, id3), of(id3, id4), entity(id4, WN: city%1:15:00::), resolve(id4), entity(id3, WN: wall%1:06:01::), resolve(id3), in(e1, id2), entity(id2, WN: wind%1:04:01::), resolve(id2), event(e1, WN: flap%2:38:00::, id1), entity(id1, WN: banner%1:06:00::).

This represents the core of the paper. How can we obtain the most reliable logical representations using WSD?

3.2. Context Definitions

At this point, context definition represents an essential step for effective results. Basically, a number of similarity-based methods use word windows and sentence windows (Ide and Veronis, 1998) to provide a vector space that defines the context of a word to be disambiguated. Here, we propose the use of the syntactic and the logical representations in order to define the context vector of the word (called syntactic context in the former and logical-semantic context in the latter) and we judge the performance of various WSD using these contexts by comparison with traditional window-based context and sentence-based contexts. The following "traditional" contexts were used:

- All previous sentences: starting from the current sentence, all the previous sentences are taken into account to build the context vector.
- 0 to 6 sentences: the context may be composed from 0 to 6 sentences around the sentence containing the word to disambiguate. 0 sentence means that you take into account only the current sentence. 1 sentence context means that you take the previous and the next sentence around the sentence to disambiguate, and so on.
- 0 to 6 words around the word to disambiguate means that you take only the current word (context with 0 word), 1 word before and 1 word after the word to disambiguate (context with 1 word), and so on.

We also defined graph-based contexts, which can be divided into **local contexts** and **global contexts**. A local context comprises only the direct relationships of the ambiguous word with the other words in the current sentence whereas a global context is built incrementally from all the previously defined contexts and updated with the disambiguated entities and events labels. Graph-based contexts are built using the results of dependency parsing as well as logical representations:

- A **local syntactic graph context** around a given word to disambiguate (here an entity or an event) is composed of all the direct dependency relationships from or to the word. A syntactic context for the word "*flap*" in the previous example is composed of the following words: [banner, in, outside] (see figure 2).
- A **local logical-semantic graph context** of an entity or an event is also composed of the elements which are in direct predicative relationships with the word to disambiguate. For instance, a logical-semantic context for the word *flap* is defined as follows: [banner, wind, walls] (we can see in Figure 2 that they denote entities that are related to the same event *e1*).

A global context for a sentence i contains all the previous

contexts from sentence 1 to i. As soon as some entity or event is disambiguated, its sense is added to the context.
For instance, given the current sentences:
Banners flap in the wind outside the walls of the city.
The gates seem getting closer.
The first global syntactic context for the word "gates" in the second sentence will contain all the words of the current sentence together with all the words of the previous sentence and the disambiguated entities and events:
[*Banners, flap, in, the, wind, outside, the, walls, of, the, city,* ***streamer****, gates, seem, getting, closer*].
Note that the disambiguated entities are in bold, and that the entities whose label is the same as the original word are not repeated. For example, wind is tagged with the sense Wind but is not repeated in the context vector.
The global logical graph for the word "gates" will contain only the identified entities and events plus the disambiguated entities and events: [banners, flap, wind, walls, city, ***streamer****, gates, seem, getting, closer*].
Below is an example that shows clearly the differences between syntactic and logical graphs. Given the sentence: "The peasant came, holding the rabbit by the ears", the following syntactic parse is generated:

```
root came/v
  nsubj peasant/n
    det the/d
  advmod back/rb
  xcomp holding/v
    dobj rabbit/n
      det the/d
    prep by/prep
      pobj ears/n
        det the/d
```

The corresponding logical form is:
entity(id1, peasant), entity(id2, rabbit), entity(id3, ears), by(e1, id3), event(e1, holding, id1, id2), event(e2, came, id1).
From this logical form, it is clear for example that there is a direct link between the event "holding" and the entity "peasant", which will be used in the disambiguation of the event. However, the syntactic context does not contain such a direct link.
Finally, another important measure for WSD is the **similarity metric**. The metric used in this work is the number of overlapping terms between the context of the word and the context of the word senses.

3.3. WSD Algorithms

Knowledge-based algorithms were implemented and were used to test various combinations between the algorithms and the contexts described above (word-based, sentence-based and graph-based). These algorithms are mainly similarity-based algorithms and include the ***Simplified Lesk algorithm*** (Kilgarrif and Rosenzweig, 2000) which rely on word sense glosses and ***Banerjee and Pedersen algorithm*** (Banerjee and Pedersen, 2003) which takes into account the sense gloss, its related words as well as all its direct lexical and semantic relationships (hypernyms, hyponyms, etc.). We also implemented ***the most frequent sense algorithm*** as a baseline. These algorithms always back-off to the most frequent sense in case they fail to disambiguate the word. We also developed another type of algorithm (minimally supervised) relying on co-occurrence frequency vectors extracted from an annotated corpus (SEMCOR and Senseval English lexical training data) hereafter designated as ***Frequency of co-occurrences*** vectors. These vectors contain the most frequent co-occurring words for a given term and help to determine the number of overlapping terms between these co-occurring terms and the context of the term to disambiguate. We also enriched the context definition of Banerjee and Pedersen with these co-occurrence frequencies (the algorithm is designated as ***Banerjee/Pedersen + Frequency of co-occurrences***), thus extending the vector space and increasing the chance of finding overlaps between the context of the word and the instance to be disambiguated.

4. Experiments

The experiments were conducted on a small corpus composed of children stories such as *Alice in Wonderland* comprising 185 sentences. This corpus (Corpus 1), composed of Text A, Text B and Text C, was manually annotated in order to build a gold standard. Corpus 1 was characterized by simple to complex syntactic grammatical constructs ranging from a simple "*Birds are flying*" to a complex sentence such as "*Alice had read several nice little histories about children who had got eaten up by wild beasts and other unpleasant things, all because they WOULD not remember the simple rules their friends had taught them: such as, that a red-hot poker will burn you if you hold it too long.*".

4.1. Various WSD Algorithms and Contexts

The results of the experiments on corpus 1 are presented in the various following tables. Table 1, 2 and 3 show the best results among all WSD algorithms and contexts. Table 2 and Table 3 list the algorithms and contexts used to obtain the results in Table 1, respectively. Since the disambiguation in Corpus 1 was performed on entities and events, the results are displayed in terms of precision and recall over these elements. The first column displays the precision of entity disambiguation, column 2 the recall of the entity disambiguation, and so on.
Precision and recall are calculated based on the following formulas:

Precision = items the system got correct / total number of items the system generated

Recall = items the system got correct / total number of relevant items (which the system should have produced)

%	Entity Precision	Entity Recall	Event Precision	Event Recall
Text A	91.96	67.32	86.36	68.67
Text B	93.83	93.83	87.30	87.30
Text C	77.99	64.68	57	50

Mean	87.93	75.28	76.89	68.66

Table 1. Best results obtained during WSD by taking all algorithms and contexts into account

Algorithms	Entity	Event
Text A	Banerjee/Pedersen + Frequency of co-occurrences	Frequency of co-occurrences
Text B	Banerjee and Pedersen	Simplified Lesk
Text C	Frequency of co-occurrences **or** Banerjee and Pedersen	Frequency of co-occurrences

Table 2. Algorithms used to obtain the results in Table 1.

Algorithms	Entity	Event
Text A	Previous Sentences Context	Global syntactic Graph Context
Text B	Previous Sentences Context	Local logical Graph Context
Text C	Word window 6 **or** Previous Sentences Context	Previous Sentences Context

Table 3. Contexts used to obtain the results in Table 1.

Overall, we can observe that the various WSD algorithms perform better (about 2-3% better) than the most frequent sense (Table 4) on these three texts.

%	Entity Precision	Entity Recall	Event Precision	Event Recall
Text A	89.28	65.36	86.36	68.67
Text B	91.36	91.36	84.13	84.13
Text C	76.08	63.09	49	42.98
Mean	85.57	73.27	73.16	65.26

Table 4. Most frequent sense results.

Regarding entities, we can observe, in Table 2, that Text B is better disambiguated using unsupervised WSD (Banerjee and Pedersen, 2003) while C obtains the same results either with the unsupervised WSD (Banerjee and Pedersen, 2003) or with the frequency of co-occurrences built based on SEMCOR. The other disambiguation tasks, especially for events, are better handled by taking into account frequencies of co-occurrences (2 cases) and Simplified Lesk (1 case).

It is obvious that the WSD approach has an impact on the results but context definition is also a major aspect. Table 5 presents the best results of our experiments using the syntactic and logical contexts with various WSD algorithms, and Table 6 and 7 show the contexts and the algorithms that enabled these results.

%	Entity Precision	Entity Recall	Event Precision	Event Recall
Text A	91.07	66.67	86.36	68.67
Text B	93.17	92.59	87.30	87.30
Text C	76.55	63.49	52.43	47.37
Mean	86.93	74.25	75.36	67.78

Table 5. Best results obtained during WSD (various algorithms) with logical and syntactic graph contexts.

Context	Entity	Event
Text A	Global logical graph context	Global syntactic graph context
Text B	Global syntactic graph context	Local Logical graph context
Text C	Global logical graph context	Global syntactic graph context

Table 6. Contexts used to obtain the results in Table 5.

Algorithm	Entity	Event
Text A	Banerjee and Pedersen + Frequency of co-occurrences	Frequency of co-occurrences
Text B	Banerjee and Pedersen	Simplified Lesk
Text C	Banerjee and Pedersen + Frequency of co-occurrences	Frequency of co-occurrences

Table 7. Algorithms used to obtain the results in Table 5

Two remarks can be made based on the results in tables 5, 6 and 7. First, the overall performance of the WSD is not very far from the performance of the best results presented in Table 1. This may indicate that using syntactic and logical graph contexts may be of interest to WSD. Global contexts (5 cases out of 6) seem to perform better than local ones (1 case out of 6), at least when coupled with the frequency of co-occurrences algorithm. Syntactic contexts (3 cases out of 6) appear the same number of times as logical ones. It is then difficult to draw a sound conclusion based solely on these experiments.

4.2. Experiments with Banerjee and Pedersen Algorithm

In order to provide better evidence on the interest of syntactic and logical graph contexts for WSD, we decided to run (Banerjee and Pedersen, 2003) algorithm on Corpus 1 and to identify the contexts used to obtain the best disambiguated results. Table 7 and 8 show these experiments.

%	Entity Precision	Entity Recall	Event Precision	Event Recall
Text A	90.18	66.01	86.36	68.67
Text B	93.83	93.83	79.36	79.36
Text C	77.99	64.68	44	38.6
Mean	87.33	74.84	69.91	62.21

Table 7. Best results obtained using (Banerjee and Pedersen, 2003)

%	Entity Precision	Event Precision
Text A	Local Logical Graph Context	Local Logical Graph Context
Text B	Previous Sentence Context	Word Window 6
Text C	Previous Sentence Context	Local Logical Graph Context

Table 8. Contexts used to obtain the results in Table 7.

As can be seen in Table 7, the results of the Banerjee and Pedersen algorithm on event disambiguation are much lower than the ones obtained in Table 1. In Table 8, we can notice that global graph and syntactic graph contexts have disappeared and that local graph contexts appear 3 times out of six.

A third experiment was to run (Banerjee and Pedersen, 2003) using only local logical graphs and local syntactic graphs and to compare the results with the best performance displayed in Table 7. The objective was to provide a comparison between syntactic and logical graphs as well as between graph-based contexts and more traditional contexts. The following tables (Table 9, 10 and 11) provide these results.

%	Entity Precision	Entity Recall	Event Precision	Event Recall
Text A	90.17	66.01	81.82	65.06
Text B	92.59	92.59	77.78	77.78
Text C	75.6	62.7	44	38.6
Mean	86.12	73.77	**67.87**	**60.48**

Table 9. Best results obtained using local logical graph contexts and (Banerjee and Perdersen, 2003) algorithm

%	Entity Precision	Entity Recall	Event Precision	Event Recall
Text A	87.5	64.05	74.24	59.03
Text B	87.04	87.04	76.19	76.19
Text C	75.6	62.7	44	38.6
Mean	83.38	71.26	64.81	57.94

Table 10. Best results obtained using local syntactic graph contexts and (Banerjee and Perdersen, 2003) algorithm

As shown in Tables 9 and 10, with (Banerjee and Pedersen, 2003) algorithm, entities and events are best disambiguated using local logical graphs. Other contexts (a 6 word window, current sentence and global syntactic and logical contexts) do not perform as well as local logical context (Table 11) except for the global syntactic graph which outperforms slightly the local logical context for entities.

%	Entity Precision	Entity Recall	Event Precision	Event Recall
SCW0	84.73	72.69	63.87	57.24
WW6	83.83	72.04	67.06	59.84
Global Logical Graph	84.41	72.4	58.77	52.97
Global Syntactic Graph	**86.51**	**74.15**	59.04	53.08

Table 11. Mean results obtained using various contexts and (Banerjee and Perdersen, 2003) algorithm.

Although we made a number of experiments taking into account various algorithms and contexts, in reality, only one algorithm and context must be chosen. It is then important to synthesize the results of these experiments. In general, based on a comparison with contexts containing all previous sentences, 0 to 6 sentences and 0 to 6 words around the word to disambiguate, and by taking into account all the WSD algorithms, it is possible to make the following comments:

- Using global graphs does not improve the results of event WSD with (Banerjee and Pedersen, 2003) algorithm;
- Global syntactic graphs give the best result for entities, close to the best possible results using (Banerjee and Pedersen, 2003) and outperform only slightly the local logical graphs. In general, these local logical graphs comprise attribute labels and related event labels;
- Local logical graphs seem to be the best combination for event disambiguation using (Banerjee and Pedersen, 2003) (Table 9). This may indicate that using just direct logical relationships may be enough to disambiguate a particular event, and that using more words in context may not be required for event disambiguation. Disambiguation results using a 6 word window come just after the local logical graphs, outperforming syntactic contexts.

Despite the small size of the corpus, graph-based contexts seem to be a possible way to obtain interesting results. This is at least the case with local logical graph and global syntactic graph contexts coupled with (Banerjee and Pedersen, 2003) for the disambiguation of entities and local logical graphs for the disambiguation of events.

We believe that these results may be further improved

given that our logical analyzer in its current development state may not cover enough English syntactic patterns to be competitive with the syntactic graphs, which are produced by a state-of-the-art dependency parser (De Marneffe et al., 2006). Based on this hypothesis, our assumption is that a more complete logical analyzer should exceed even better the results of a syntactic graph.

We run another experiment on the Senseval English Lexical Sample Data (Mihalcea et al., 2004) and we adapted our algorithms to tackle nouns, verbs and adjectives as this is required in Senseval. Senseval provides a set of texts with particular words to be disambiguated. In the following paragraph, the word tagged "head" should be disambiguated. You can notice, various problems in the punctuation as well as in some non-connected sentences (Sealing...).

<instance id="activate.v.bnc.00061340" docsrc="BNC">
<context>
So , to provide ample warning , fit smoke alarms . Ideally , site one in the hall and another on the landing . Avoid fitting one in the kitchen , as fumes from cooking are often enough to ***<head>activate</head>*** *the alarm . The one illustrated is by First Alert : like most types it can be simply screwed into a ceiling . SEALING GLAZING BARS*
</context>
</instance>

We ran a number of tests using the WSD algorithms and the traditional contexts described above (sentence windows, word windows, and all previous sentences). The best results (64.1% (fine-grained) and 69% (coarse-grained)) were obtained using Banerjee and Perdersen algorithm coupled with the frequencies of co-occurrence (extracted from Senseval Training Data). The best context with this WSD algorithm used a 2-sentence window and a cosine similarity to measure the similarity between the context vector and each word sense vector. We also run experiments using various WSD algorithms and the local graph contexts (syntactic and logical). Global contexts were not applicable to Senseval data as the disambiguation task was targeted to various unrelated paragraphs and not entire texts. Unfortunately, using graph-based contexts worsened the results of the best performing algorithm Banerjee and Pedersen coupled with frequencies of co-occurrences (Best value = 64.1 %, Local syntactic graph = 43.2% and local logical graph= 48%). However, the results were improved for the algorithm (Banerjee and Pedersen, 2003) alone (Table 12).

	2-sentence window	Local syntactic graph	Local logical graph
Fine-grained results	46.5 %	52.4%	50.2%
Coarse-grained results	54.9%	58.5%	57.4%

Table 12. Results obtained using (Banerjee and Pedersen, 2003) on Senseval Data

Local syntactic graphs perform better than logical graphs in Table 12. However, when we run other experiments using Frequency of co-occurrences, local logical graphs gave superior performance in comparison with other contexts (Sentence context window, syntactic graphs).
We noted two issues in the Senseval experiment. On one side, the syntactic parsing was very noisy, due to improper punctuation (see the example above) and syntactic errors in the dependency parsing. The syntactic links between the words had many erroneous or high-level underspecified dependencies. On the other side, we obtained very incomplete logical representations using our logical analyzer on this set of data. As previously said, this might indicate that the analyzer may not cover enough syntactic structures, but we also noticed that noisy syntactic relationships had a big impact over our analyzer, which looks for specific well-defined syntactic structures. Despite these two issues, local syntactic and logical graphs contexts raised the performance of some WSD algorithms. Improving the syntactic and logical analysis might have an impact over the accuracy of WSD.
This also raises the question of the kind of corpora that should be made available to the research community which is committed to full parsing for WSD. Obtaining better quality texts is probably one of the requirements of such approaches. Further work will tackle the enhancement of the logical analyzer, but also the manual definition of logical contexts in order to avoid any impact of bad syntactic and logical analyses on the WSD and to test our logical contexts on clean and non-noisy data.

5. Conclusion

This work presented a WSD approach based on syntactic dependencies and predicative logical representations. One interest of these logical representations is their natural identification of the role arguments. This paper presented an experiment on a small corpus and on Senseval data that shows that interesting preliminary results might be obtained using dependency and logical features. Based on these preliminary results and based on previous research works (Tanaka et al., 2007), our assumption is that logical contexts should help WSD, at least with some particular algorithms such as the ones presented in the experiments. We also believe that a semantic analysis involving anaphora and co-reference resolution might also help WSD, by providing links to previous information in the context definition. Further work should provide a more thorough evaluation of WSD using syntactic and logical features. The impact of the accuracy of logical form extraction on WSD should also be measured.

6. Acknowledgements

The authors would like to thank Prompt Quebec and

UnimaSoft Inc. for their financial support.

7. References

Banerjee, S. and Pedersen, T.: Extended gloss overlaps as a measure of semantic relatedness. In *Proc. of the Eighteenth International Joint Conference on Artificial Intelligence*, Acapulco, Mexico, pp. 805-810 (2003)

De Marneffe, M-C, MacCartney, B. and Manning. C.D.: Generating Typed Dependency Parses from Phrase Structure Parses. In *Proc. of LREC*, pp. 449-454 (2006)

Fellbaum, C.: WordNet: An Electronic Lexical Database. MIT Press (1998)

Ide, N. and Véronis, J.: Introduction to the special issue on word sense disambiguation: the state of the art. *Comput. Linguist.* 24(1) :2-40 (1998).

Kilgarriff, A. and Rosenzweig, R.: Framework and results for English SENSEVAL. *Computers and the Humanities* 34:15–48, (2000).

Merlo, P. and Musillo, G. : Semantic parsing for high-precision semantic role labelling. *In Proceedings of the Twelfth Conference on Computational Natural Language Learning,* pp. 1-8, ACL. (2008).

Mihalcea, R., Chklovski, T. And Kilgarriff, A.: The Senseval-3 English Lexical Sample Task Export, in *Proc. of Senseval-3*, pp. 25--28, Spain, (2004).

Tanaka, T., Bond, F., Baldwin, T., Fujita, S. and Hashimoto, C. : Word Sense Disambiguation Incorporating Lexical and Structural Semantic Information. *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pp. 477-485, ACL (2007).

Zouaq, A., Gagnon, M. and Ozell, B. (2010): Semantic Analysis using Dependency-based Grammars and Upper-Level Ontologies, In *Proceedings of the 11th International Conference on Intelligent Text Processing and Computational Linguistics*, (2010) (To appear).

Zouaq, A.: Une approche d'ingénierie ontologique pour l'acquisition et l'exploitation des connaissances à partir de documents textuels, Ph.D. Dissertation, University of Montreal (2008).