IJCLA VOL. 1, NO. 1-2, JAN-DEC 2010, PP. 85-101
RECEIVED 28/10/09 ACCEPTED 16/01/10 FINAL 09/03/10

Semantic Analysis using Dependency-based Grammars and Upper-Level Ontologies

AMAL ZOUAQ, MICHEL GAGNON AND BENOÎT OZELL

Ecole Polytechnique de Montréal, Montréal (Québec)

ABSTRACT.

Semantic analysis of texts is a key issue for the natural language processing community. However, this analysis is generally based on a deeply-intertwined syntactic and semantic process, which makes it not easily adaptable and reusable from a practical point of view. This represents an obstacle to the wide development, use and update of semantic analyzers. This paper presents a modular semantic analysis pipeline that aims at extracting logical representations from free text based on dependency grammars and assigning semantic roles to the logical representation elements using an upper-level ontology. An evaluation is conducted, where a comparison of our system with a baseline system shows preliminary results.

Keywords: Semantic Analysis, Logical Analysis, Dependency Grammars, Upper-level Ontologies, Word sense disambiguation.

1 INTRODUCTION

Semantic Analysis is the process of assigning a given sense to the different constituents of a sentence or a text. In the NLP community, most of the approaches, such as HPSG [14] and categorical grammars [10, 15], require the use of a semantic lexicon, i.e. is a dictionary that links words to semantic classes and roles and involves sub-categorization. In fact, this lexicon is the most important component of these grammars, since it is encoded as a set of lexical entries with syntactic and semantic information (feature structures or type-logical lambda-expressions). In the text mining community, template filling, which also involves knowledge about semantic arguments, is mainly used as a way to assign or extract meaning. Some problems arise from this kind of approach. Firstly, it is generally domain-dependent, especially in the Text Mining Community. This involves repeating the process for each new domain since the identified roles must suit the application domain in order to be accurate. Secondly, the construction of the lexicon implies a huge effort. It is generally language dependent. Thirdly, semantic analysis (involving sub-categorization) can be considered as an intertwined process of syntactic and semantic processing, which make it not easily modularised and updatable.

Based on these issues, we believe that there is a need of a looser coupling between the syntactic and semantic information. This paper presents a reflection on what should be a semantic analysis with the current technologies and formalisms available. It presents a pipeline that separates the process of extracting logical representations (the logical analysis) from the process of assigning semantic roles to the logical representation elements (the semantic annotation). These roles are defined in an upper-level ontology, SUMO [12]. The interest of the pipeline as proposed here is first the modular nature of the syntactic, logical and semantic analyzers, which enables easier updates and focused experimentations that identify the weaknesses of each component of the pipeline, and second, the definition of semantic roles in an ontology, which make the approach more easily interoperable. In fact, one of the major problems of SRL systems is the diversity of semantic roles and their various terminologies and formalisms [8], which hinder their comprehension from one SRL system to another.

The paper is organized as follows. Section 2 presents briefly the state of the art in computational semantics. Section 3 describes the system, including the knowledge model, the steps involved, as well as the required knowledge structures (SUMO and SUMO-WordNet). It also lists the syntactic patterns used in the logical analyzer and presents some of the WSD methods used to assign SUMO senses to the logical elements. Section 4 presents an experiment, shows the results in terms of precision and recall and compares our approach with baseline systems. Finally, section 5 summarizes the paper and outlines implications for NLP research.

2 RELATED WORK

These last years have shown interesting progress in the computational semantics research. While the majority of recent text-based extraction works relies on statistical-based shallow techniques [1], there is still a non negligible amount of research devoted to the implementation of hand-built grammars such as categorical grammars [15], HPSG [14], MRS [4] or TRIPS grammar [1]. These grammars are usually sets of syntactic rules coupled with semantic components, which indicate the role of the rule's arguments in terms of semantics. One drawback of this approach is that the lexicon is not easily obtainable and requires a lot of manual work from computational linguists. This makes the approach not easily scalable and not easily adaptable to new semantic analysis and new models. Moreover, rich grammars such as categorical grammars are not so easily obtainable or reusable.

Other works such as [13] have addressed the extraction of logical forms for semantic analysis as we do but they did not tackle, to our knowledge, the assignment of semantic roles to the logical forms. Finally, very recent works [3] show a growing interest in producing deep semantic representations by taking as input the result of a syntactic parser. This paper is in the same line of research. However our work aims at a looser coupling of the syntactic and semantic features and leaves the deep semantic aspects to a subsequent step of WSD.

3 A MODULAR PIPELINE FOR SEMANTIC ANALYSIS

The semantic analysis adopted in this paper is a modular pipeline that involves three steps (Fig.1):

1. Syntactic parsing of texts;
2. Logical analysis using a dependency-based grammar;
3. Semantic annotation based on the upper-level ontology SUMO involving word-sense disambiguation.

This modular process is a solution to the above mentioned issues related to current semantic analysis including creating a modular design clearly separating syntactic, logical and semantic annotation or extraction steps, providing a dependency-based grammar that could be comprehensible and reusable by the text mining and NLP community, making this grammar domain-independent and lexicon-independent, and finally using an ontology as a way to formally define semantic roles and make them understandable from a SRL system to another.

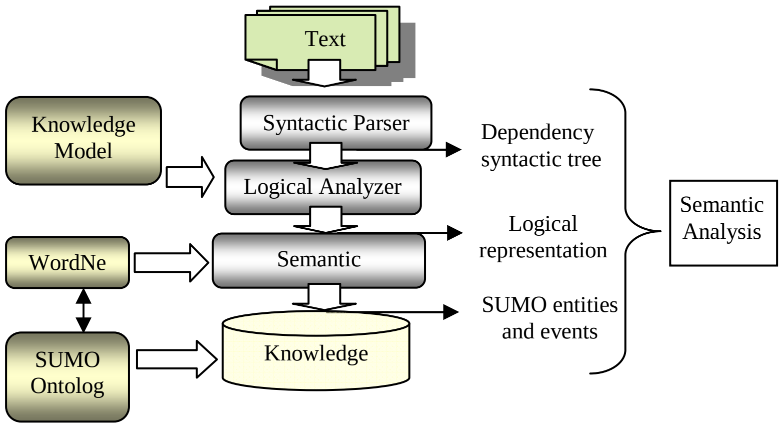

Fig. 1. The Semantic Analysis Pipeline

The following sections explain these steps as well as the linguistic resources needed at each step.

3.1 The Syntactic Analysis

The syntactic analysis is aimed at facilitating the subsequent steps of logical representation and semantic annotation. We believe that this analysis should be based on deep linguistic analysis and should not be limited to simple tagging or surface syntactic parsing. Our goal is to propose a method "easily" reproducible, reusable and able to extract domain-dependent and domain-independent patterns. This should be perfectly handled by dependency parsing.

Dependency parsing outputs grammatical relationships between each pair of words in a sentence. This formalism has proved its efficiency in text mining and we believe that it has the required characteristics of a good grammatical formalism, as it is intuitive, easily understandable, and it enables transparent encoding of predicate-argument structure. Moreover, current state-of-the-art dependency analyzers seem to be sufficiently robust to be considered as reliable tools for knowledge extraction and this is particularly true for the Stanford parser, according to current surveys [16].

Our system uses the basic dependencies format option in the Stanford Natural Language Processing Parser and its dependency component [6] and transforms the output grammatical relations into a

tree, represented as a Prolog term. It also enriches the typed dependencies with the word grammatical categories (such as verb (v), noun (n), …), which are obtained using the Stanford's parts-of-speech output. This operation is important because it enables distinguishing some patterns using their parts-of-speech. For instance, the parser output for the sentence *Banners flap in the wind outside the walls of the city* is presented below. This sentence will be used as an example throughout the whole process of semantic analysis.
nsubj(flap-2, Banners-1);prep(flap-2, in-3);det(wind-5, the-4);pobj(in-3, wind-5);prep(flap-2, outside-6);det(walls-8, the-7);pobj(outside-6, walls-8);prep(walls-8, of-9);det(city-11, the-10);pobj(of-9, city-11)
Banners/NNS flap/VBD in/IN the/DT wind/NN outside/IN the/DT walls/NNS of/IN the/DT city/NN./.

This is transformed into a tree, which is given as input to the logical analyzer:

```
root/tree(token(flap,2)/v,
    [nsubj/tree(token(banners,1)/n,[]),
     prep/tree(token(in,3)/prep,
        [pobj/tree(token(wind,5)/n,
           [det/tree(token(the,4)/d,[])])]),
     prep/tree(token(outside,6)/prep,
        [pobj/tree(token(walls,8)/n,
           [det/tree(token(the,7)/d,[]),
            prep/tree(token(of,9)/prep,
              [pobj/tree(token(city,11)/n,
                 [det/tree(token(the,10)/d,
        [])])])])])])]).
```

3.2 The Logical Analysis

The logical analyzer presented in this paper is based on the dependency syntactic tree produced in the syntactic analysis step and is strongly related to the approach presented in [17]. In fact, both systems rely on dependency syntactic patterns to extract logical representations. While these representations were used in [18] to generate domain ontologies using measures from graph theory, they are exploited here as an intermediate step towards an efficient modular semantic analysis. Moreover, one of the new central points of our approach is the use of an upper level ontology as the semantic lexicon for semantic analysis and the use of WSD algorithms in order to assign roles. The key points that should be outlined here is that, first, the logical analysis does not depend on a particular lexicon, a particular vocabulary or a particular domain. Second, this analysis is based on the compositionality principle, which states that a sentence semantic representation can be obtained by the semantic representation of its parts. Here we consider that a sentence logical representation requires the logical representation of its parts. To our knowledge, there is no previous proposal to create a compositional logical analyzer based on dependency grammars as we propose here.

3.2.1 The Knowledge Model

Although the logical analysis does not require the use of a semantic lexicon, it still needs a conceptual structure made up of a minimal set of categories. In this project, the categories include entities, named entities, events, statements, circumstances, time, number, purpose, measure and attributes. With these categories, chosen to be as general as possible, it is easy to express various information contexts and to remain independent from a particular domain. Although it would be possible to create logical representations using only lexical items, we believe that using these categories can help the semantic analysis.

There is a straightforward map between our knowledge model categories and grammatical relationships. The following table summarizes the mapping involved between the syntactic categories and the knowledge model. Sometimes, the knowledge model element is detected through a part-of-speech (POS) (e.g. verb, noun), but it may also be detected through a number of grammatical relationships (syntactic patterns) necessary to find the relevant element. In the example column, the words in bold indicate the syntactic category related to the knowledge model element. This knowledge model is subject to further enhancements in the future.

Table 1. Mapping knowledge model elements with syntactic categories

Knowledge Model Element	Syntactic Category	Example
Entity	Noun (n)	The **cat** eats.
Event	Verb (v)	The cat **eats**.
Statement	Any pattern involving a clausal complement with external subject (xcomp)	I **like** {to eat in the garden}$_{\text{xcomp}}$
Circumstance	Adverbial clause (advcl)	The accident happened {as the night was **falling**}$_{\text{advcl}}$
Time	Temporal modifier (tmod)	He swam in the pool {last **night**}$_{\text{tmod}}$
Number	Numeric Modifier (num)	**200** people came to the

		party
Attributes	1. Nominal subj. and copula 2. Adjectival complement 3. Adjectival modifier	1. The cat is **big** 2. He looks **tired** 3. He is a **happy** man
Measure	Measure	The director is 55 **years** old

Note that these mappings are not performed in isolation. In fact, relating a knowledge model element to a syntactic category occurs only in the context of detecting specific syntactic patterns. This prevents the system from incorrectly assigning a knowledge model element to a given lexical item. For example, many nominalizations should refer to events instead of entities. Assigning them in the context of a pattern enables us to avoid this confusion. These patterns are explained in the next section.

3.2.2 The dependency-based Grammar: a Pattern Knowledge Base

Besides the link between syntactic categories and knowledge model elements, the dependency-based grammar is composed of a set of patterns coupled with transformational rules. These rules exploit the dependency representation and create logical representations using the general categories introduced in the knowledge model. The grammar is divided into **core** and **modifiers** patterns and is composed, up to now, of 61 rules. Core syntactic patterns, such as the well-known *subject-direct object* pattern, represent main grammatical structures that are necessary for parsing the texts. Modifiers patterns represent modifiers such as prepositions, participial, purpose-clause, temporal modifiers and so on. The patterns are organized into a hierarchy where richer patterns containing the maximum number of relationships are at the top level. In our Prolog implementation, the hierarchy is simply organized as a set of rules where "richer" rules are fired first. It is worth noting that many patterns can be instantiated in a same sentence, including core patterns and modifiers patterns. Also, some patterns extract implicit knowledge. For instance, in the phrase "*the rabbit's head*", the logical analyzer will produce a predicated term *has-attr (rabbit, head*) from the grammatical relationship *poss (possessive)*. At the subsequent step of WSD, the "real" meaning of the relation (*part-of, possess*, etc.) will be assigned.

The following tables show some of these patterns and provide examples, some of them taken from the Stanford dependencies manual [5]. A grammatical relationship between brackets indicates that it is a child of the preceding relationship. For example, in *nsubj-xcomp[-dobj]*, a *dobj* relationship is a child of the *xcomp* relationship. In the examples column, the words in bold and italics represent the heads (root nodes) of the patterns. Head's syntactic category is indicated in italics in the beginning of each pattern. The reader is referred to [6] to understand the grammatical hierarchy and the corresponding grammatical links.

Table 2. Main syntactic patterns

Patterns	Examples
Verb-nsubj-dobj-iobj	{Mary}$_{\textbf{nsubj}}$ ***gave*** {Bill}$_{\textbf{iobj}}$ a {raise}$_{\textbf{dobj}}$
Verb-nsubj-dobj-xcomp	{The peasant}$_{\textbf{nsubj}}$ ***carries*** {the rabbit}$_{\textbf{dobj}}$, {holding}$_{\textbf{xcomp}}$ it by its ears
Verb-nsubj-dobj	{The cat}$_{\textbf{nsubj}}$ ***eats*** {a mouse}$_{\textbf{dobj}}$
Verb-nsubj-xcomp[-dobj]	{Michel}$_{\textbf{nsubj}}$ ***likes*** to {eat}$_{\textbf{comp}}$ {fish}$_{\textbf{dobj}}$
Adjective-nsubj-xcomp	{Benoit}$_{\textbf{nsubj}}$ is ***ready*** to {leave}$_{\textbf{xcomp}}$
Verb-csubj-dobj	What Amal {said}$_{\textbf{csubj}}$ ***makes*** {sense}$_{\textbf{dobj}}$
Verb-nsubj-expl	{There}$_{\textbf{expl}}$ ***is*** a small {bush}$_{\textbf{nsubj}}$
Adjective-nsubj-cop	{Benoit}$_{\textbf{nsubj}}$ {is}$_{\textbf{cop}}$ ***happy***
Noun-nsubj-cop	{Michel}$_{\textbf{nsubj}}$ {is}$_{\textbf{cop}}$ a ***man***
Verb-nsubj-acomp	{Amal}$_{\textbf{nsubj}}$ ***looks*** {tired}$_{\textbf{acomp}}$
Verb-xcomp-ccomp	Michel ***says*** that Benoit {likes}$_{\textbf{ccomp}}$ to {swim}$_{\textbf{xcomp}}$
Verb-nsubj	{The cat}$_{\textbf{nsubj}}$ ***eats***
Verb-dobj	Benoit talked to Michel in order to ***secure*** {the account}$_{\textbf{dobj}}$
Verb-nsubjpass-prep by	{The man}$_{\textbf{nsubjpass}}$ has been ***killed*** {by}$_{\textbf{prep}}$ the police
Verb-csubjpass-prep by	That he {lied}$_{\textbf{csubjpass}}$ was ***suspected*** {by}$_{\textbf{prep}}$ everyone
Verb-nsubjpass	{Bills}$_{\textbf{nsubjpass}}$ were ***submitted***

Table 3. Modifiers patterns

Modifiers Patterns (Modifiers)	Examples
Partmod[prep]	There is garden **surrounded by** houses.
Prep[pcomp]	They heard **about** Mia missing classes.
Prep (after a noun)	Vincent discovered the man **with** a telescope.
Prep (after a verb)	Bills were submitted **to** the man.
Amod	The **white** cat eats
Tmod	Vincent swam in the pool last **night**
Advcl	The accident happened as the night was **falling**.

Ccomp	Michel says that Benoit **likes** to swim.
Purpcl	Benoit talked to Michel in order to **secure** the account.
Infmod	The following points are to **establish**.
Measure	The director is 55 **years** old.
Num	The director is **55** years old.
Poss	The peasant taps the **rabbit**'s head with his fingers.
Quantmod	**About** 200 people came to the party.
Advmod	**Genetically** modified food is dangerous.
Rcmod	Michel loves a cat which Benoit **adores**.

At present, the grammar does not handle anaphora resolution automatically and conjunctions are computed based on a distributive interpretation only, which may not lead to a correct interpretation in some cases. Future work should tackle these issues.

3.2.3 The Transformational Approach

Each pattern is a Prolog rule that builds a logical representation according to the fired pattern. Since we use a compositional approach, each fired rule builds a part of the sentence analysis. The resulting representation is a predicative flat formula composed of predicates (the knowledge model elements) applied to lexical elements, as well as predicates resulting from prepositional relations and predicates indicating if an entity has already been encountered in the discourse or if it is a new entity. Relationships between predicates are represented through their arguments and referential variables are assigned to the instantiated knowledge model elements.

Following our example sentence, the resulting logical representation is: `outside(e1, id3), of(id3, id4), entity(id4, city), resolve_e(id4), entity(id3, walls), resolve_e(id3), in(e1, id2), entity(id2, wind), resolve_e(id2), event(e1, flap, id1), entity(id1, banners), new_e(id1).`

This formula states that there are a number of entities (*city*, *wind*, *etc.*), an event (*flap*) involving the entity *banner* and two relationships "*outside*" and "*of*". "*Outside*" involves the event of *flapping e1* and the entity *walls*. The predicates *new_e* and *resolve_e* are used to indicate if the entity has already been encountered in previous sentences (*resolve_e*) or not (*new_e*). This will help us in anaphora resolution.

An example of a Prolog rule for the *nsubj-dobj* pattern is shown below. The rule involves the discovery of the two relationships of interest (*nsub*j and *dobj*) and calls the *semparse* procedure. This procedure creates a logical representation for the *nsubj* and the *dobj* sub-trees and finally produces an instance of an event object that combines these two outputs.

```
semparseMainPattern(tree(Node/v,Children),tree(Node
/v,Rest),
Id,SemIn,[event(Id,Node,IdAgent,IdObject)|SemOut]):
-
  select(nsubj/tree(N1/_,C1),Children, R1),
  select(dobj/tree(N2/_,C2),R1, Rest),
  semparse(tree(N1/n,C1),_,IdAgent,SemIn,Sem1),
  semparse(tree(N2/n,C2),_,IdObject,Sem1,SemOut),
  gensym(e,Id).
```

3.3 The Semantic Annotator

The obtained logical representation elements should then be assigned a sense. One of the tasks of a SRL system is to adequately segment predicates and their arguments before their classification into a specific set of roles. Due to the logical analysis, argument and predicate segmentation is already done and the semantic annotator should then focus on assigning an appropriate role to these representations. Here, we mainly focus on entities and events in the logical representations but further work should explore the whole structure.

3.3.1 The SUMO Upper-Level Ontology

One of the difficulties in semantic role labelling is that most of the approaches use very specific subsets of semantic roles and do not agree on the roles to be used. Using an upper-level ontology enables a high-level and formal definition of these roles. Moreover, the interest of using an ontology instead of a flat set of roles is the ability to use its hierarchical and conceptual structure in order to help the WSD process, ascertain the correctness of the identified roles, or reason about the annotated roles. In the context of the Semantic Web, this last point is very important, as the annotated texts will be meaningful to multiple SRL systems which should foster reusability and interoperability.

The Suggested Upper Merged Ontology (SUMO) [11] is an ontology composed of about 1000 terms and 4000 definitional statements. It has been extended with a Mid-Level Ontology (MILO), and a number of domain ontologies, which enable coverage of various application domains. SUMO has also gone through various developments stages and experimentations, which make it stable.

One interesting feature of SUMO is that its various sub-ontologies (base, structural, MILO, and domain ontologies) are independent and can be used alone or in combination. Here, we only exploit the upper level, meaning that we take into account only the SUMO ontology itself (merge.kif). Another point is its mapping from lexical items (terms) to general high-level concepts. In fact, SUMO [11] is mapped to the widely used WordNet computational lexicon [7]. The SUMO-WordNet mapping links each synset in WordNet to its SUMO sense through three types of relationships: equivalent links, instance links and subsumption links. Despite the fact that this mapping can be error-prone, we believe that it represents an excellent demonstration of how various state-of-the-art resources can be used in a modular pipeline. The other point is that choosing this upper-level ontology is not a limitation and can be extended by a domain ontology if this is required.

3.3.2 Word Sense Disambiguation

Choosing the appropriate role involves the use of WSD algorithms. At this point of our work, we have implemented a number of standard knowledge-based unsupervised WSD methods. The choice of unsupervised methods is guided by the same motivation as for the whole pipeline: avoiding costly and hard-to-build language resources.

The interest of the pipeline at the level of WSD is that the predicates and arguments to be disambiguated are already clearly identified in the logical representations. One step further would be to use the whole logical representation itself as a way to direct the disambiguation process. We are currently working on this.

Among the WSD methods, we used a number of Lesk-derived algorithms namely the Simplified and Original Lesk. We also implemented a version of the [2] algorithm which is based on a semantic network extracted from WordNet to build a context feature vector for the term to be disambiguated. We relied also on a baseline widely used in SRL evaluations: the most frequent sense. Finally, we used an algorithm that relies on co-occurrences frequencies extracted from SEMCOR to determine the number of overlapping terms between these co-occurring terms and the context of the term to disambiguate.

In all these implementations, if the algorithm fails to identify a particular sense, it then backs off to the most frequent sense. Below are the annotated entities and events in our example sentence. Here we only show the SUMO-based annotations but we also keep the WordNet-based annotations in the knowledge base.

This results into the following SUMO-based semantic representation of the example sentence: `outside(e1, id3), of(id3, id4), entity(id4, SUMO:City), resolve_e(id4), entity(id3, SUMO: StationaryArtifact), resolve_e(id3), in(e1,` `id2), entity(id2, SUMO: Breathing), resolve_e(id2), event(e1, SUMO: Motion, id1), entity(id1, SUMO: Fabric), new_e(id1).`

Of course, the system can also produce the WordNet-based semantic representation: `outside(e1, id3), of(id3, id4), entity(id4, WN: city%1:15:00::), resolve_e(id4), entity(id3, WN: wall%1:06:01::), resolve_e(id3), in(e1, id2), entity(id2, WN: wind%1:04:01::), resolve_e(id2), event(e1, WN: flap%2:38:00::, id1), entity(id1, WN: banner%1:06:00::), new_e(id1).`

4 EVALUATION

Evaluating such a rich pipeline is a challenge in itself. In fact, it involves evaluating the syntactic, logical and semantic annotation. Based on current reviews of the Stanford parser which describe a good performance [16], we decided to focus on the logical and semantic annotation evaluations. Two types of experiments were conducted using the well-known precision and recall metrics:

- A first experiment involving a small corpus of three descriptive texts (185 sentences) manually annotated using SUMO senses in order to build a SUMO gold standard. This corpus helped us in performing the logical form evaluation as well as the semantic annotation;
- A second experiment on the Senseval 3 dataset for the English lexical sample task [9] which enables to test the chosen WSD algorithms on a standard dataset and to compare the results with similar systems. This second experiment does not rely on the previous logical form extraction.

For comparison purposes, we used the most frequent sense baseline in both experiments. Precision and recall are calculated as follows:

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)

4.1 *Logical Analyzer Results*

This section tests the logical analyzer and the accuracy of the resulting logical formula by measuring the precision and recall of the extracted entities and events in the first corpus using our patterns. These results are summarized in Table 4.

Table 4. The logical analysis results in terms of entities and events

	Precision %	**Recall %**
Entities	94.98	80.45
Events	94.87	85.5

From these experiments, it is clear that our semantic analyzer yields promising results. Most of the time, the incorrect entities and events are due to a wrong syntactic parsing from the Stanford Parser. There are also some patterns that are not yet identified which make the recall lower. These results should be later completed with an evaluation on a bigger corpus, they should be detailed in terms of correctness of predicates, arguments and whole logical formulas [13] and finally, they should include the whole logical representation and not be limited to entities and events.

4.2 Semantic Annotator Results

Semantic annotation (as an isolated module) was tested over the first corpus as well as the Senseval data (English lexical sample task). Various algorithms were tested mainly using knowledge-based methods, including:

- The Simplified and Original Lesk algorithms as well as derivatives such as [2].
- An algorithm computing the most frequent sense based on the WordNet frequencies (extracted from SEMCOR) for the first corpus, and based on term frequencies in the training data for the Senseval dataset.
- An algorithm, dubbed *frequency of co-occurrence,* computing the overlap between the context of the term to be disambiguated and a vector of frequently co-occurring terms for each sense of the term together with their frequency. In the case of the first corpus, these co-occurrences frequencies are extracted from the SEMCOR corpus whereas they are extracted from the Senseval training data in the second corpus.

Many context sizes were tested for all these algorithms including all previous sentences and various words and sentence windows (from 0 to 4) as well as the logical graph structure obtained in the logical analysis.

Due to a lack of space and to the fact that WSD in itself does not represent our contribution in this paper, we simply present the results of the best performing algorithm together with the most frequent sense baseline without entering into the details of each implemented algorithm (the reader is referred to references and state-of-the-art literature). We did not obtain the best performance using only one algorithm on the three texts (first corpus) but Banerjee and Pedersen algorithm [2] was always among the top-ranking algorithms for entity annotation. Events were best disambiguated using frequency of co-occurrence (text 1), most frequent sense (text 2) and Simplified Lesk (text 3). The following table shows the mean of the precision and recall obtained for the three texts. As can be shown, the algorithm outperforms slightly the most frequent sense baseline. We are seeking better results and future work will explore graph-based WSD disambiguation based on the logical analysis form. Further experiment should also explore the impact of the corpus characteristics on the performance and the choice of WSD algorithms.

Table 5. A comparison of the precision/recall results for the two algorithms (WSD and most frequent sense)

	Best Algorithm (Precision)	Best Algorithm (Recall)	Most Frequent (Precision)	Most Frequent (recall)
SUMO entities	87.08	73.76	84.67	71.65
SUMO events	75.69	68.16	71.54	64.29

Regarding the Senseval corpus, we were able to obtain a precision/recall of 64.1 % (Fine-grained) and 69% (coarse-grained). Based on the overall results of the competition [9], we were able to exceed the most frequent sense performance which was listed as 55.2% (fine-grained) and 64.5% (coarse-grained) using a variant of [2] coupled with frequencies of co-occurrences. We used a two-sentence window around the word to be disambiguated and a cosine similarity. Our results rank us second among the unsupervised algorithms of the competition (although we consider the approach as minimally supervised).

5 CONCLUSION

Current semantic analysis techniques are generally in need of lot of training data, depend on resources such lexicons for their semantic interpretation and lack a uniform way to define roles or labels that should be assigned to sentences constituents. This paper presented a modular pipeline for semantic analysis that relies on state-of-the-art dependency parsing, logical analysis using a dependency-based grammar and finally semantic annotation. This annotation is performed using the upper-level ontology SUMO and the WordNet lexicon, which could be considered as standard resources. Choosing a dependency grammar instead of other formalisms is guided by a practical point of view: it is argued that state-of-the-art analyzers have reached a certain maturity, which makes them a good starting point for a semantic analysis. Moreover, dependency grammars provide an intuitive solution to the identification of logical forms from text as outlined by [13]. The proposed solution does not require costly training or data resources, except some standard resources well-known in the NLP community. The modular nature of the pipeline makes it more easily adaptable and updatable from a software engineering point of view. Finally, the system presented here is intended as a demonstration of what could be a semantic analysis with current methods and tools. Future work includes the enrichment of the dependency-based grammar with new patterns, a better handling of the meaning of conjunctions and other specific constructions (such as idioms), the processing of ambiguous structures and eventive nominalizations as well as the use of a bigger corpus for the evaluation of the logical analysis results. We are currently working on WSD algorithms that could benefit from the logical form not only for argument selection as proposed here, but also for argument annotation.

ACKNOWLEDGEMENTS

The authors would like to thank Prompt Quebec and UnimaSoft Inc. for their financial support.

REFERENCES

1. Allen, J. F., M. Swift, and W. de Beaumont.: Deep Semantic Analysis of Text. In *STEP 2008 Conference Proceedings*, pp. 343-354. College Publications (2008).

2. Banerjee, S. and Pedersen, T.: Extended gloss overlaps as a measure of semantic relatedness. In *Proc. of the 18th Int. Joint Conference on Artificial Intelligence*, Acapulco, Mexico, pp. 805-810 (2003)
3. Bos, J.: Introduction to the Shared Task on Comparing Semantic Representations. In *STEP 2008 Conference Proceedings*, pp 257-261 (2008).
4. Copestake, A., Flickinger, D., Pollard, C. and Sag, I.A.: Minimal Recursion Semantics - an Introduction, *Research on Language and Computation* 3:281–332, Springer (2005).
5. De Marneffe, M-C, and Manning. C.D: Stanford typed dependencies manual, http://nlp.stanford.edu/software/dependencies_manual.pdf (2008)
6. 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)
7. Fellbaum, C.: WordNet: An Electronic Lexical Database. MIT Press (1998)
8. Màrquez, L., Carreras, X., Litkowski, K. and Stevenson, S.: Semantic Role Labeling: An Introduction to the Special Issue. *Computational Linguistics*, 34(2), 145-159 (2008).
9. 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)
10. Moortgat, M.: Categorial grammar and formal semantics. In L. Nagel (ed.) Encyclopedia of Cognitive Science, Vol. 1, pp. 435-447. London, Nature Publishing Group, (2002)
11. Niles, I., and Pease, A.: Linking Lexicons and Ontologies: Mapping WordNet to the Suggested Upper Merged Ontology, *Proc. of the IEEE International Conference on Information and Knowledge Engineering*, pp 412--416 (2003)
12. Pease, A., Niles, I., and Li, J.: The Suggested Upper Merged Ontology: A Large Ontology for the Semantic Web and its Applications. In *Working Notes of the AAAI-2002 Workshop on Ontologies and the Semantic Web*, Canada (2002)
13. Rus, V.: A First Evaluation of Logic Form Identification Systems, in *Proc. of Senseval-3: Third Int Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, pp. 37--40, Spain (2004)
14. Sag, I., Wasow, T. and Bender, E.: Syntactic Theory: A Formal Introduction, 2nd Edition, Center for the Study of Language and Information - Lecture Notes (2003)
15. Steedman M., *The Syntactic Process*, MIT Press (2000)
16. Stevenson, M. and Greenwood, M.A.: Dependency Pattern Models for Information Extraction, Journal of Research on Language & Computation, Springer (2009)

17. 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)
18. Zouaq, A. and Nkambou, R.: Evaluating the Generation of Domain Ontologies in the Knowledge Puzzle Project, *IEEE Transactions on Knowledge and Data Engineering*, 21(11): 1559-1572 (2009).

AMAL ZOUAQ
ECOLE POLYTECHNIQUE DE MONTRÉAL,
C.P. 6079, SUCC. CENTRE-VILLE, MONTRÉAL (QUÉBEC), H3C 3A7
E-MAIL: <AMAL.ZOUAQ@POLYMTL.CA>

MICHEL GAGNON
ECOLE POLYTECHNIQUE DE MONTRÉAL,
C.P. 6079, SUCC. CENTRE-VILLE, MONTRÉAL (QUÉBEC), H3C 3A7
E-MAIL: <MICHEL.GAGNON@POLYMTL.CA>

BENOÎT OZELL
ECOLE POLYTECHNIQUE DE MONTRÉAL,
C.P. 6079, SUCC. CENTRE-VILLE, MONTRÉAL (QUÉBEC), H3C 3A7
E-MAIL: <BENOIT.OZELL@POLYMTL.CA>