# Comparing a Rule-Based and a Machine Learning Approach for Semantic Analysis

Francois-Xavier Desmarais
École Polytechnique
de Montréal
Montréal, PQ, Canada
Francois-Xavier.Desmarais@polymtl.ca

Michel Gagnon
École Polytechnique
de Montréal
Montreal, PQ, Canada
Michel.gagnon@polymtl.ca

Amal Zouaq
Royal Military College of Canada
Department of Mathematics and Computer Science
Kingston, ON, Canada
Amal.Zouaq@rmc.ca

***Abstract*—Semantic analysis is a very important part of natural language processing that often relies on statistical models and machine learning approaches. However, these approaches require resources that are costly to acquire. This paper describes our experiments to compare Anasem, a Prolog rule-based semantic analyzer, with the best system of the Conference on Natural Language Learning (CoNLL) shared task dedicated to a sub-task of semantic analysis: Semantic Role Labeling. Both CoNLL best system and Anasem are based on a dependency grammar, but the major difference is how the two systems extract their semantic structures (rules versus machine learning). Our results show that a rule-based approach might still be a promising solution able to compete with a machine learning system under certain conditions.**

***Keywords-Semantic role labeling; evaluation; rule-base systems; machine learning.***

## I. Introduction

One of the most challenging tasks of natural language processing is semantic analysis (SA), which aims at discovering semantic structures in texts. Two schools of thought try to tackle this hard task:

**The Computational Semantics approach**: semantic analysis is often built on top of grammars describing lexical items through feature structures [3]. The aim is to extract a logical representation such as first-order logic and discourse representation structures (DRS). These types of grammars are often hard to build and maintain but they offer a wide coverage of various linguistic phenomena (e.g., co-reference resolution, negations, and long-distance dependencies). To our knowledge, such a wide coverage is only handled through this type of systems. Another problem is that very few if any datasets enable the comparison of these types of systems.

**The Machine Learning approach**: semantic analysis is decomposed into various tasks such as semantic role labeling (SRL) [4], co-reference resolution [11] and named entity extraction [1]. While machine learning, especially supervised approaches, proved to be successful in some of these tasks, it suffers from well-known shortcomings: Firstly, the algorithms depend highly on the availability of training corpora, which take a lot of resources to be developed. Secondly, the learned models often do not scale well on different datasets and domains, thus necessitating other training corpora.

The two aforementioned approaches involve a non-negligible effort either in terms of software development (computational semantics) or in terms of data availability (machine learning). The two obvious questions are whether one of these approaches is less costly than the other and whether one is more successful than the other. Trying to address the second aspect, this paper aims at providing insights on the following research question: **Can a rule-based semantic analyzer reach the same performance of a machine learning one?**

Despite the fact that machine learning systems have become prominent in some tasks such as syntactic parsing, there is no clear evidence that they are more efficient and less costly for the semantic analysis task.

In this paper, we compare two systems on a specific sub-task of semantic analysis, which is the identification of predicates and their arguments:

- *ANASEM*, our Prolog-based semantic analysis system which outputs Discourse Representations Structures based on dependency grammar patterns. We consider that ANASEM falls within the computational semantics approach;
- The *LTH Parser* [4], which is the winner of CoNLL (The Twelfth Conference on Computational Natural Language Learning) shared task, dedicated to SRL. This system is based on dependency parsing and machine learning.

These two systems are run on a subset of the CoNLL gold standard [12]. The paper explains in detail how we handled this comparison.

The paper is organized as follows. Section 2 provides a brief overview of the state of the art in semantic analysis. Section 3 is a description of our rule-based semantic analyzer Anasem. Section 4 presents the core of our methodology by explaining the adaptations we had to perform on our system to compare our results with CoNLL winner. Section 5 details the results obtained by Anasem. Finally, section 6 discusses the limitations of our approach.

## II. State of the art

As aforementioned, the term “semantic analysis” might take various meanings depending on the targeted community. In this paper, we consider SA as the process of extracting predicates and arguments. There have been considerable efforts these last years in areas such as semantic role labeling [12], dependency-based representations [14] and machine learning [4] to extract these semantic structures. Two main approaches are pervasive to state-of-the-art Natural Language Processing systems: statistical and machine learning techniques and rule-based techniques. Syntactic analysis seems to have evolved essentially towards statistical parsers [8]. However, rule-based approaches have proven successful in others tasks. For example, the best-performing system at the CoNLL 2011 shared task for co-reference resolution [6] is a rule-based system. Similarly, in semantic analysis, the STEP 2008 shared task [2] reported on various systems among which Boxer [3] used a categorical grammar approach. A formal comparison of these systems, using a gold standard, is missing. To our knowledge, CoNLL 2008 Shared task is among the very few which offer such a gold standard. The participants at this competition were essentially machine learning systems including the first-ranked system, the LTH Parser [4], which relied on dependency analysis and classifiers for SRL. In this paper, our objective is to compare the performance of ANASEM with the LTH parser. To our knowledge, there was no previous tentative in recent semantic analyzers to compare a symbolic rule-based approach to a machine learning approach on the same corpus.

## III. ANASEM, A PROLOG-BASED SEMANTIC ANALYZER

Anasem [14] is a rule-based system written in Prolog and built on a modular pipeline made of 3 functionalities: syntactic parsing, canonical tree generation and pattern recognition.

### A. Syntactic Analysis

The syntactic analysis is the first step in the pipeline, and like [4] it is based on dependency parsing. Anasem uses the Stanford parser [5], its dependency module [7] and its part-of-speech tagger [13] to perform the syntactic analysis. For instance, the sentence *They drank brandy in the lounge* returns the following result, where part-of-speech tags and dependencies are given (note that each word is given with its position in the sentence.)

Part of speech:

```
They/PRP drank/VBD brandy/NN in/IN the/DT
lounge/NN ./.
```

Syntactic analysis:

```
nsubj(drank-2, They-1)
dobj(drank-2, brandy-3)
prep(drank-2, in-4)
det(lounge-6, the-5)
pobj(in-4, lounge-6)
```

### B. Canonical Tree generation

The second step of the pipeline is to generate a canonical tree from the syntactic analysis to facilitate the subsequent step of pattern recognition. The dependency parse is coupled with parts-of-speech to create a Prolog term. This Prolog term represents a unified structure that can be processed recursively based on the principle of compositionality. Using our previous example, we obtain the following representation:

```
root/tree(token(drank, 2)/v,
 [nsubj/tree(token(they, 1)/prp, []),
  dobj/tree(token(brandy, 3)/n, []),
  prep/tree(token(in, 4)/prep,
   [pobj/tree(token(lounge, 6)/n,
     [det/tree(token(the,5)/d,[])])])])
```

A final step is to modify the generated tree to facilitate patterns identification. Some important modifications are related to coordination and negation. Dependencies involving a coordinated form are duplicated and attached to every member of the coordination. For example, the parse tree for the sentence *John visited Paris and Roma* would be translated into a tree that corresponds to the sentence *John visited Paris and John visited Roma*. Another important transformation achieved at this step concerns negation. Instead of being dependent of the main verb of the clause, the negation is moved at the root of the clause.

### C. Pattern recognition

An Anasem pattern represents a syntactic rule that can be mapped to a semantic representation. Anasem contains about 60 patterns. Each part of the Prolog tree is analyzed in a recursive manner, thus implementing a pattern hierarchy (based on the rules appearance in Prolog). The output is a discourse representation structure [14]. Using the previous example, we obtain the following DRS:

```
---------------------------------
[id1,id2,e1,id3]
--------------------------------
entity(id1,they)
entity(id2,brandy)
event(e1,drank,id1,id2)
entity(id3,lounge)
in(e1,id3)
---------------------------------
```

This DRS introduces three entities and one event. Event `e1` is a drinking event that involves two entities (the brandy and the persons who are drinking it). The DRS also expresses a relation between the event and its location (the lounge).

## IV. METHODOLOGY

This section describes the methodology followed to compare Anasem with LTH Parser.

### A. CoNLL Corpus and Terminology Description

CoNLL Shared Task provided a corpus based on a subset of the Penn Treebank II [7] [12]. Two types of corpora were made available: a training corpus that contained a structured output with parts of speech, syntactic analysis and semantic representations, and a test corpus. In our case, the major problem with these corpora is the lack of compatibility with Anasem's output format (DRS, grammatical relationships, grammatical categories and semantic categories). Table I shows the subset of CoNLL format that was used in our adaptations. Each term is related to a part-of-speech, the position of its head in the sentence (value is 0 for the root of the sentence), and a grammatical relationship. The semantic representation starts with the semantic predicate (and its frame in PropBank [10] and NomBank [9]). Finally the last columns indicate semantic arguments in the form of semantic categories labeled A0, A1, AM-TMP, etc. For every predicate there is a corresponding column, in the same order. For example, in Table I, predicate *happen.01* has arguments A1 and AM-TMP that correspond to *accident* and *as*, respectively.

TABLE I. AN EXAMPLE OF CONLL FORMAT

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | The | DT | 2 | NMOD | _ | _ | _ |
| 2 | accident | NN | 3 | SBJ | _ | A1 | _ |
| 3 | happened | VBD | 0 | ROOT | happen.01 | _ | _ |
| 4 | as | IN | 3 | TMP | _ | AM-TMP | _ |
| 5 | the | DT | 6 | NMOD | _ | _ | _ |
| 6 | night | NN | 7 | SBJ | _ | _ | A1 |
| 7 | was | VBD | 4 | SUB | _ | _ | _ |
| 8 | falling | VBG | 7 | VC | fall.01 | _ | _ |
| 9 | . | . | 3 | P | _ | _ | _ |

## B. *Anasem Adaptation to CoNLL*

Given the different terminology adopted by CoNLL, we had to modify two major modules of Anasem, namely the canonical tree generator and the semantic patterns that were using the Stanford nomenclature.

### 1) *The Canonical Tree Generator*

As aforementioned, Anasem uses the Stanford parser [5] to generate the canonical trees. To exploit our patterns, we had to keep Anasem's canonical tree representation while using CoNLL lexico-syntactic representations. These representations were available in the shared task corpora [12] designated hereafter as the gold standard (GS). We extracted the syntactic relations, the parts of speech and the head of each word from the GS (see Table II) and replaced Anasem’s dependency relationships and parts of speech.

The sentence *The accident happened as the night was falling* was transformed into the trees illustrated in Table II.

As can be noticed, although there were similarities between the initial tree and the obtained tree, there were also some major differences. For example, some root nodes changed as shown by comparing the node `advcl falling/v` in our initial tree with the node `tmp as/prep` in the obtained tree. These differences can be explained by the fact that the Stanford parser and CoNLL have different syntactic representations. For example, the "auxiliary" is represented by the Stanford Parser with its head as the verb and the syntactic relation as "aux", while in CoNLL, the auxiliary is the head and its syntactic relation is called a "verb chain" (vc).

TABLE II. CANONICAL TREE TRANSFORMATION

```
Initial Canonical Tree
root happened/v
  nsubj accident/n
       det the/d
  advcl falling/v
       mark as/prep
       nsubj night/n
          det the/d
       aux was/v
Tree using CoNLL Terminology
root happened/v
  sbj accident/n
       nmod the/d
  tmp as/prep
       sub was/v
       sbj night/n
            nmod the/d
       vc falling/v
Modified Canonical Tree
root happened/v
  sbj accident/n
       nmod the/d
  tmp falling/v
       sbj night/n
              nmod the/d
       aux was/v
       complm as/prep
```

We classified these differences into two major categories:

a. Structural differences, which happen when word positions and heads inside the tree are different.
b. Nominal differences, which happen when the terminology of the grammatical relations is different but the meaning is the same.

We had to adapt the canonical tree generator to deal with these differences. Most problems caused by structural differences were solved by creating a set of rules that we applied to the canonical tree (for instance `AUX` in our previous example). Nominal differences were then resolved by providing a mapping between Stanford grammatical relations and CoNLL relations and updating Anasem patterns accordingly.

### 2) *Patterns Adaptation*

Almost all the patterns had to be adapted to use the CoNLL grammatical terminology. Many patterns needed a nomenclature modification, for example the noun subject tagged `NSUBJ` in Stanford had to be renamed to `SBJ` to match CoNLL terminology. There were few exceptions such as the negative form which did not necessitate a change. Apart from the terminological changes, we experienced some mapping problems due to differences in granularities between the Stanford grammatical relationships hierarchy (which seems more fine-grained) and the CoNLL one. The grammatical relationship `NMOD` is a good example of this problem. For example, in CoNLL, determiner and adjectives are both classified as a `NMOD`, while Stanford has specific categories (`DET`, `AMOD`). In this case, we were able to perform mappings with the Stanford hierarchy by using parts of speech to differentiate the various possibilities.

Certain patterns were not used because their grammatical relations were not identified in CoNLL. For example, clausal complements, represented by the `CCOMP` relation in Stanford parser, are interpreted as generic complements in CoNLL.

Although there were many differences between Anasem's output (DRS) and CoNLL's output (Table I), attempts were made to automate the process, but they were unsuccessful due to too many special cases. Therefore, we had to select a subset of the original corpora, manually identify the mappings and finally check the obtained tree transformations before being able to parse the obtained trees using the pattern recognition module.

## C. Corpus Selection and Comparison Methodology

We selected a subset of sentences from the original corpora to analyze them with our system and compare the results with LTH Parser. We established few rules to avoid some specific problems with few characters, which are not processed by the current version of Anasem, and to deal with the way CoNLL handle hyphenated words. These rules are as follow:

- A sentence must not contain the following characters (-`&$%()_:\/) as Anasem is not robust in front of this type of input.
- A sentence must not contain hyphenated words: this rule was due to the way CoNLL processes these words. The GS separates the words and the hyphen and considers each word independently while Anasem considers them as a single entity.
- A sentence must have between 5 and 30 words.
- A sentence must have at least 1 verb.

In particular, the last two rules were used to focus on the most representative and declarative sentences (e.g., with at least one verb). For instance, a sentence such as : *"At law school, the same"* was excluded from our evaluation.

Using these filtering rules, we extracted sentences from the CoNLL training corpus (dev set). Then, to avoid any bias, we randomized all the sentences with "www.random.org" and extracted the first 51. These sentences (dev set) were used to compare Anasem results with the gold standard semantic representations. We repeated the same process on the test corpus to extract a set of sentences to be used for a fair comparison between Anasem results and LTH Parser, which was trained on CoNLL training corpus. 50 sentences were extracted from the test corpus, with an overall of 101 sentences.

To be able to compare Anasem with CoNLL semantic representation, we had to establish a comparison methodology. Due to the differences between Anasem semantic representation (DRS containing entities, events, attributes, etc.), this comparison was essentially based on the unlabeled extraction of the semantic representations.

We ran Anasem on the test and dev corpora and the LTH Parser on the test corpus.

To evaluate ANASEM DRS, we extracted the predicates from the gold standard and we verified if these predicates were available in the generated DRS either as an `entity` or an `event`. Then for each of these predicates, we compared the arguments indicated in the gold standard with those in the DRS to correlate them with either an `event` argument (e.g., `event (e2,falling,id2))` or a complement argument (e.g., `time(e1,e2)` or `in (id2,id3)`). This allowed us to compute recall for Anasem.

For the comparison between ANASEM and LTH Parser, we used the evaluation tool of the CoNLL competition shared task. Since we aimed mainly at the identification of predicates and arguments (without providing a label), we slightly modified this tool to display the presentation of unlabelled arguments and predicates.

## V. RESULTS

In this section, we present the results of our experiments.

## A. Sentence-based results

We categorized our results into the following:

**All sentences**: These results include both fully and partially covered sentences on both corpora (development and test). Since we do not cover all the possible patterns yet in Anasem, it was anticipated that some sentences would be partially analyzed.

**Sentences with full predicate coverage**: These results describe the performance of Anasem over sentences that were successfully parsed at the syntactic level and whose predicates have all been discovered at the semantic analysis level. These sentences are qualified as fully covered sentences in our results.

All the results based on the GS are reported as unlabeled recall values only. In fact, even though we consider CoNLL corpus as a GS, our analysis is that this GS is not very well adapted for a fair computation of Anasem precision. In fact, Anasem covers semantic relationships that are not available in the GS but that are still valid. One case is that Anasem extracts all attributes when CoNLL GS seems to neglect some. For example in the phrase *"…his sweaty armpits…"* , CoNLL considers only "*his*" as an attribute of "*armpits*" (A1) while Anasem identifies also "*sweaty*" as an attribute, which seems reasonable. Another case is that CoNLL restricts the analysis to predicates with arguments from PropBank [10] and NomBank [9], and cannot identify predicates without arguments (such as Shipyard(X)), which might be criticized in the context of a global semantic analysis. Computing precision over this GS would affect negatively the precision of Anasem. However, to give the reader an idea about the performance of our system, we manually computed the precision over the DRS extracted from the test corpus (see Table III).

TABLE III. PRECISION RESULTS FOR ANASEM

| | |
|---|---|
| Predicates | 92 % |
| Arguments | 80 % |
| Predicates and arguments | 86 % |

**All sentences**

These results are obtained on the 101 sentences from the test and development corpora. Each sentence is broken down into predicates and arguments to allow for a more focused analysis of the results. Among these 101 sentences, there were 53 fully covered sentences, 37 partially covered sentences and 11 empty outputs. These outputs are due to failure at the semantic or syntactic level (e.g., an unknown pattern or an illegal Prolog term generation) with a total of

229 predicates and 404 arguments. For the same set of sentences, the gold standard indicated 298 predicates and 672 arguments. Overall, we obtained a recall of 77 % for the identification of predicates and 61 % for the arguments.

**Sentences with full predicate coverage**

As mentioned previously, we tagged sentences with missing predicates as partially analyzed sentences. The missing items often resulted from some unimplemented pattern in our Prolog-based semantic analyzer. Thus, we decided to present the results of fully covered sentences separately, as we wanted to evaluate our success on identified patterns. There were 53 fully covered sentences in our corpus with 337 possible arguments among which we identified 271 arguments. This resulted in a significant improvement of 19% over the previous results with a recall of 80%.

Table IV summarizes the results obtained for the extraction of unlabeled predicates and arguments.

TABLE IV. RECALL VALUES FOR THE DETECTION OF PREDICATES AND THEIR ARGUMENTS (UNLABELED).

| | Predicates | Arguments |
|---|---|---|
| All sentences | 77% | 61% |
| Fully covered sentences | 100% | 80% |

We can notice that the recall for predicates and arguments identification is much better on fully covered sentences.

## B. *Arguments recognition*

These results focus on arguments rather than on the predicates. The objective is to assess the success of Anasem in correctly extracting arguments when the predicate is identified.

Table V shows the recall values for argument detection when we consider the correctly identified predicates, independently of the sentences (All detected predicates). In the 50 sentences corpus, 94 predicates were found. In the gold standard, there were 218 arguments associated to these predicates, from which 162 were found by Anasem. That is a 74 % rate of success.

TABLE V. RECALL VALUES FOR ARGUMENT DETECTION IF WE CONSIDER ONLY CASES WHERE PREDICATES WERE DETECTED.

| | Arguments |
|---|---|
| All detected predicates | 74% |
| Predicates detected in fully covered sentences | 78% |

Taking only the predicates that were detected in fully covered sentences, the figures are slightly better (106 out of 136 arguments), that is, a 78 % rate of success. This shows that choosing arguments from partially analyzed sentences tends to give worst results than with fully-covered sentences.

## C. *Comparison with CoNLL Shared Task Best System*

We also executed LTH Parser on the 50 sentences from the test corpus. LTH results were as follow for the unlabeled predicates and arguments: 136 predicates out of 142 and 250 arguments out of the 314 that were in the gold standard. This represents a 95 % recall rate for the predicates and 80 % recall value for the identification of the arguments (see Table VI). Results are significantly lower with Anasem especially for argument detection. However, we see that by considering only completely parsed sentences, the difference seems to vanish.

TABLE VI. COMPARISON OF RECALL VALUES IN % FOR ANASEM AND LTH.

| | Predicates | Arguments |
|---|---|---|
| Anasem (All sentences) | 72% | 57% |
| Anasem (fully covered sentences) | - | 79% |
| LTH | 95% | 80% |

Since Anasem makes a distinction between core arguments (that correspond to A0, A1…A4 in CoNLL) and modifier arguments (all other arguments in CoNLL), we were interested to see whether the recall values are the same for both categories. Table VII shows the results, including LTH evaluation on the test corpus. Interestingly, we note that the difference between core and modifiers is greater in LTH Parser than in Anasem.

TABLE VII. RECALL VALUES FOR ARGUMENT DETECTION, DISTINGUISHING CORE ARGUMENTS AND MODIFIER ARGUMENTS.

| | Core | Modifiers |
|---|---|---|
| All sentences results on test corpus - Anasem | 57% | 53% |
| Fully covered sentences results on test corpus - Anasem | 81% | 76% |
| Results on test corpus - LTH | 84% | 69% |

## VI. DISCUSSION

### A. *General Observations*

There are a few things that we need to clarify regarding our results. In the CoNLL shared task, the participating systems were based on external lexicons, namely PropBank [10] and NomBank [9] to identify the arguments types. These lexicons identify arguments based on a word considered as a predicate. Each word-predicate is related to a frame that assigns generic categories such as A0 or A1 to the arguments.

The peculiarity of Anasem is that it is a standalone system, which does not rely on any external resource to identify predicates and arguments. Only dependency parses are used, which means that Anasem is able to extract these predicates and arguments but cannot annotate them with particular categories or types. Therefore, we relied on the unlabeled extraction task of the competition and compared unlabeled results. This means that whenever we compare our output with the gold standard, we only compare the presence of the arguments and the predicates, regardless of the type of the arguments.

The results for all the sentences in the combined corpora were not outstanding (77 % for the predicates and 61 % for the arguments). On the one hand, this was not really surprising, considering the limited amount of patterns implemented in Anasem. On the other hand, with the fully covered sentences the results rivaled the winner's of the shared task of CoNLL (79% versus 80% argument wise). Our conclusion is that Anasem, though not complete yet, has a good potential to perform as well as a machine learning approach, while staying independent from a training corpus and domain. These results will have to be confirmed in future experiments on bigger corpora.

### B. *Limitations*

There are limitations in our experiments. To begin with, we used a sub-corpus of the original corpus of CoNLL, and it was a rather small part of it with only 101 sentences. This small number was largely due to the complexity of comparing our adapted output to the CoNLL format and to the time-consuming effort required to make the manual comparison. This manual comparison might have been biased due to possible potential errors by the expert performing the comparison. However, a clear methodology for comparing the representations was established upfront and was closely followed.

We are conscious that our limited set of sentences might affect positively our results if well-analyzed sentences are selected. However, these sentences were randomly selected as previously explained. Moreover, the opposite phenomenon might occur and amplify analysis errors if the wrong sentences (i.e., not well-analyzed sentences) are selected from the gold standard.

Another important point is that there are a number of errors that we identified in the CoNLL gold standard, and in general these errors affected negatively our experiments, and probably more strongly than if we had thousands of sentences. This is the main reason of dividing the results into fully covered and partially covered sentences.

## VII. CONCLUSION

This paper presented the results of a rule-based system for semantic analysis and compared it to the winner of the CoNLL shared task [12]. It shows that using a modular system with syntactic analysis based on dependency grammar can have comparable results with a machine-learning based analysis when the sentences are fully covered. It also demonstrates the difficulty of comparing systems based on various formalisms and lexicons. In future work, we plan to add new rules to our pattern recognition analyzer and to repeat the same types of experiments using a wider range of sentences. We also want to find a way to measure the precision and the F1 scores. Our preliminary conclusion is that semantic analysis with ruled-based systems has its place among statistical and machine-learning approaches.

ACKNOWLEDGMENTS

The authors would like to thank the Royal Military College of Canada (RMC) for its financial support through the Start-up Fund Program.

## REFERENCES

1] Baluja, S., Mittal, V. O. and Sukthankar, R., 2000. Applying Machine Learning for High-Performance Named-Entity Extraction.. *Computational Intelligence,* 16(4), pp. 586-595.

2] Bos, J., 2008. Introduction to the shared task on Comparing Semantic Representations. In: *Proceedings of the 2008 Conference on Semantics in Text Processing.* Venice: ACL, pp. 257-261.

3] Bos, J., 2008. Wide-Coverage Semantic Analysis with Boxer. In: J. Bos and R. Delmonte, eds. *Semantics in Text Processing. STEP 2008 Conference Proceedings*:College Publications., pp. 277-286.

4] Johansson, R. and Nugues, P., 2008. *Dependency-based syntactic-semantic analysis with PropBank and NomBank.* ACL, pp. 183--187.

5] Klein, D. and Manning, C. D., 2003. *Accurate Unlexicalized Parsing.* s.l., s.n., pp. 423-430.

6] Lee, H., Peirsman, Y., Chang, A., Chambers, N., Surdeanu, M. and Jurafsky, D., 2011. Stanford's multi-pass sieve coreference resolution system at the CoNLL-2011 shared task. In: *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*:ACL, pp. 28-34.

7] Marcus, M., Santorini, B. and Marcinkiewicz, M. A., 1993. Building a large annotated corpus of English: the Penn Treebank.. *Computation Linguistics,* p. 19(2).

8] Marneffe, M.-C. d., MacCartney, B. and Manning, C. D., 2006. *Generating Typed Dependency Parses from Phrase Structure Parses.* s.l., s.n.

9] Meyers, A. et al., 2004. *The NomBank Project: An Interim Report.* Boston, ACL, pp. 24-31.

10] Palmer, M., Gildea, D. and Kingsbury, P., 2005. The Proposition Bank: A Corpus Annotated with Semantic Roles. *Computational Linguistics Journal,* p. 31:1.

11] Soon, W. M., Ng, H. T. and Lim, D. C. Y., 2001. A Machine Learning Approach to Coreference Resolution of Noun Phrases. *Computational Linguistics,* 27(4), pp. 521-544.

12] Surdeanu, M., Johansson, R., Meyers, A., Marquez, L. and Nivre, J., 2008. *The CoNLL-2008 Shared Task on Joint Parsing of Syntactic and Semantic Dependencies.* s.l., s.n.

13] Toutanova, K., Klein, D., Manning, C. and Singer, Y., 2003. *Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network.* s.l., s.n., pp. 252-259.

14] Zouaq, A., Gagnon, M. and Ozell, B., 2010. *Semantic Analysis using Dependency-based Grammars.* s.l., Bahri Publications, pp. 85-101.