# A FRAMEWORK FOR STRUCTURED KNOWLEDGE EXTRACTION AND REPRESENTATION FROM NATURAL LANGUAGE THROUGH DEEP SENTENCE ANALYSIS

Stefania Costantini, Niva Florio and Alessio Paolucci
*Dip. di Informatica, Università di L'Aquila, Via Vetoio 1, Coppito, L'Aquila, Italy*

Keywords: NLP, Reasoning, ASP, OOLOT, Semantic web, Deep analysis.

Abstract: We present a framework that allow to extract knowledge from natural language sentences using a deep analysis technique based on linguistic dependencies. The extracted knowledge is represented in OOLOT, an intermediate format inspired by the Language of Thought (LOT) and based on Answer Set Programming (ASP). OOLOT uses ontology oriented lexicon and syntax. Finally, it is possible to export the knowledge in OWL and native ASP.



## 1 INTRODUCTION

Many intelligent systems have to deal with knowledge expressed in natural language, either extracted from books, web pages and documents in general, or expressed by human users. Knowledge acquisition from these sources is a challenging matter, and many attempts are presently under way towards automatically translating natural language sentences into an appropriate knowledge representation formalism (Bos and Markert, 2005). Although this task is a classic Artificial Intelligence challenge (mainly related to Natural Language Processing and Knowledge Representation (Pereira and Shieber, 2002)), with the Semantic Web growth new interesting scenarios are opening. The Semantic Web aims at complementing the current text-based web with machine interpretable semantics; however, the manual population of ontologies is very tedious and time-consuming, and practically unrealistic at the web scale (Auer et al., 2007; Kasneci et al., 2008). Given the enormous amount of textual data that is available on the web, to overcome the knowledge acquisition bottleneck, the ontology population task must rely on the use of natural language processing techniques to extract relevant information from the Web and transforming it into a machine-processable representation.

In this paper we present our framework. It allows us to extract knowledge from natural language sentences using a deep analysis technique based on linguistic dependencies and phrase syntactic structure.

We also introduce OOLOT (Ontology Oriented Language of Thought). It is an intermediate language based on ASP, specifically designed for the representation of the distinctive features of the knowledge extracted from natural language. Since OOLOT is based on an ontology oriented lexicon, our framework can be easily integrated in the context of the Semantic Web.

Section 2 introduces the framework architecture. In Section 3 we analyse the sentence with a parser. Section 4 describes the context disambiguation and lexical item resolution methods. Section 5 introduces the intermediate format OOLOT, while Section 6.2 describes the translation methodology with the help of an example. Finally, Section 7 shows an exporting from OOLOT into OWL example, and in Section 8 we conclude with a brief rsum of achieved goals and future works.

## 2 THE FRAMEWORK ARCHITECTURE

The proposed framework aims at allowing automatically knowledge extraction starting from plain text, like a web page, and have a structured representation in OWL or ASP as output. Thus, the framework can be seen as a standalone system, or can be part of a wider workflow, e.g. a component of complex semantic web applications.

Starting from plain text written in natural lan-



Costantini S., Florio N. and Paolucci A..
A FRAMEWORK FOR STRUCTURED KNOWLEDGE EXTRACTION AND REPRESENTATION FROM NATURAL LANGUAGE THROUGH DEEP SENTENCE ANALYSIS.
DOI: 10.5220/0003663702740279
In *Proceedings of the International Conference on Knowledge Discovery and Information Retrieval* (KDIR-2011), pages 274-279
ISBN: 978-989-8425-79-9

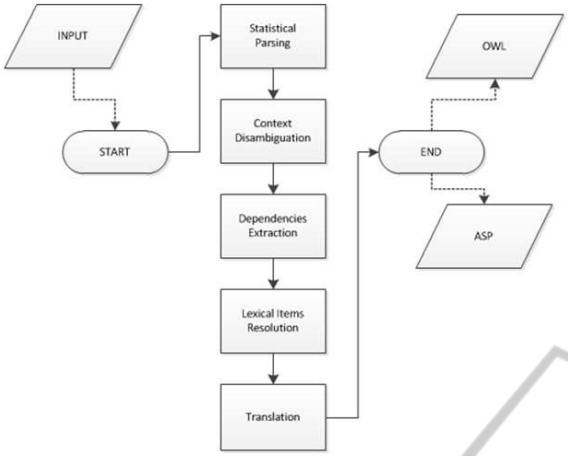


Figure 1: The framework architecture.

guage, as first step we process the sentence through the statistical parser (see Section 3). If we use a parser with embedded dependency extractor, we can perform a single step and have as output both the parse tree (constituents) and the dependency graph. Otherwise, if we use two different components, the workflow is that of Fig.1. For this step, we use a simple algorithm for context disambiguation (see Section 4). Then, each token is resolved w.r.t. popular ontologies including DBPedia and OpenCYC and the context is used for disambiguation in case of multiple choices. At this point we have enough information to translate the knowledge extracted from natural language sentence into our intermediate OOLOT format (see Section 5). For the translation process described in Section 6, we use the λ-calculus as engine that drives the translation into OOLOT, using information about the deep structure of the sentence extracted in the previous steps. From OOLOT it is possible to directly translate the encoded knowledge into OWL or to export the knowledge base in pure ASP.

## 3 PARSER ANALYSIS

Syntactic parser decomposes a text into a sequence of tokens (for example, words), and attributes them their grammatical functions and thematic or logical roles with respect to a given formal grammar, showing also the relations between the various elements of the sentence (Chomsky, 1956; Chomsky, 1957). Most of today's syntactic parsers are mainly statistical (Charniak, 1996; Charniak and Johnson, 2005; Collins, 1996; Collins, 1997) and Probabilistic Context Free Grammar (PCFG) (Collins, 1996; Collins, 1997; Charniak and Johnson, 2005; McClosky et al., 2006; Petrov et al., 2006; Petrov and Klein, 2007). Statistical parsing is useful to solve problems like ambiguity and efficiency, but with this kind of parsing we lose part of the semantic information; this aspect is recovered thanks to dependency representation (De Marneffe and Manning, 2008). Dependency grammars (DGs) were proposed by the French linguist Tesnière (Tesnière, 1959) and have recently received renewed attention (cfr. (Neuhaus and Bröker, 1997) and the references therein). In Dependency Grammars, words in a sentence are connected by means of binary, asymmetrical governor-dependent relationships. In fact, Tesnière assumes that each syntactic connection corresponds to a semantic relation.

It is difficult to evaluate parsers; we can compare them in many ways, such as the speed with which they examine a sentence or their accuracy in the analysis (e.g. (Cer et al., 2010)). The task based evaluation seems to be the best one (De Marneffe and Manning, 2008; Mollá and Hutchinson, 2003): we must choose whether to use a parser rather than another simply basing on our needs. At this stage of our ongoing research, we use the Stanford parser because it is more suited to our requirements, both for the analysis of the constituents and for that of the dependencies.

Stanford parser performs a dependency and constituent analysis (Klein and Manning, 2003a; Klein and Manning, 2003b). This parser provides us with different types of parsing: it can be used as an unlexicalized PCFG parser (Klein and Manning, 2003a) to analyse sentences, or as a lexicalized probabilistic parser (Klein and Manning, 2003b) combining the PCFG analysis with the lexical dependency analysis. The Stanford parser provides us a typed dependency and a phrase structure tree. The Stanford typed dependencies (cfr. (De Marneffe and Manning, 2008)) describe the grammatical relations in a sentence. The relations are binary and are arranged hierarchically and the head of a dependency can be any content words. Thanks to rules (De Marneffe et al., 2006) applied on phrase structure trees (also created by the Stanford parser), typed dependencies are generated.

We choose to analyse the sentence *"Many girls eat apples."*. Seeing Fig.2, we can notice that the parser attributes to each token its syntactic roles, and it provides us also the grammatical function of each word.

```
(ROOT
 (S
  (NP (JJ Many) (NNS girls))
  (VP (VBP eat)
   (NP (NNS apples)))
  (. .)))
```

Figure 2: Phrase structure produced by the Stanford parser for the sentence "Many girls eat apples".

Table 1: Lexical item resolution example.

| *Lexicon* | *Ontology* | *URI* |
|---|---|---|
| girls | DBPedia | http://dbpedia.org/resource/Girl |
| eat | DBPedia | http://dbpedia.org/class/Eating |
| apples | DBPedia | http://dbpedia.org/resource/Apple |

With regard to dependency analysis, the Stanford parser gives us two versions of this analysis: the typed dependency structure and the collapsed typed dependency structure (Fig.3) (De Marneffe and Manning, 2008).

amod(girls-2, Many-1)
nsubj(eat-3, girls-2)
dobj(eat-3, apples-4)

Figure 3: Collapsed typed dependency structure produced by the Stanford parser for the sentence "Many girls eat apples".

## 4 CONTEXT DISAMBIGUATION AND LEXICAL ITEM RESOLUTION

The context disambiguation task is a very important step in our workflow, as we need to assign each lexical unit to the correct meaning, and this is particularly hard due to the polysemy. For this task, we use a simple algorithm: we have a finite set of contexts (political, technology, sport, ...), and as first step we built a corpus of web pages for each context, and then we used each set as a training set to build a simple lexical model. Basically we build a matrix where for each row we have a lexical item, and for each column we have a context. The relation (lexical item, context) is the normalized frequency of each lexical item into the given context. The model is then used to assign the correct context to a given sentence. We use a $n \times m$ matrix, where $n$ is the number of lexical tokens (or items), and $m$ is the number of contexts. In other words, we give a score for each lexical token in relation to each context. To obtain the final score we perform a simple sum of the local values to obtain the global score, and thus to assign the final context to the sentence. Our method for context disambiguation can certainly be improved, in particular (Banerjee and Pedersen, 2002) seems to be a good development direction.

The context is crucial to choose the correct reference when a lexical item has multiple meanings, and thus, in an ontology can be resolved in multiple references. The context becomes the key factor for the resolution of each lexical item to the relative reference. We perform a lookup in the ontology for each token, or a set of them (using a sliding window of length k). For example, using DBPedia, for each token (or a set of tokens of length k), we perform a SPARQL query, assuming the existence of a reference to the lexical item: if this is true, we've found the reference, otherwise we go forward. If we found multiple references, we use the context to choose the most appropriate one.

With regard to the lexical item resolution, the steps are the following. Given the sentence *Many girls eat apples* and it's syntactic phrase structure (Fig.2) and dependencies structure (Fig.3), as first step we tokenize the sentence, obtaining:

*Many*, *girls*, *eat*, *apples*.

Before the lookup, we use the part of speech tagging from the parse tree to group the consecutive tokens that belong to the same class. In this case, such peculiar aspect of natural language is not present and thus the result is simply the following:

(*Many*), (*girls*), (*eat*), (*apples*).

Excluding the lexicon for which we have a direct form, for each other lexicon the reference ontology is resolved through a full text lookup; thus we obtain the lexical item resolution in Table 1.

## 5 FROM THE LANGUAGE OF THOUGHT TO OOLOT

The Language of Thought (LOT) is an intermediate format mainly inspired by (Kowalski, 2011). It has been introduced to represent the extracted knowledge in a way that is totally independent from original lexical items and, therefore, from original language.

Our LOT is itself a language, but its lexicon is ontology oriented, so we adopted the acronym OOLOT (Ontology Oriented Language Of Thought). This is a very important aspect: OOLOT is used to represent the knowledge extracted from natural language sentences, so basically the bricks of OOLOT (lexicons) are ontological identifier related to concepts (in the ontology), and they are not a translation at lexical level. It uses ASP as host environment that allows us for a native, high expressive knowledge representation and reasoning environment.

# 6 TRANSLATING INTO OOLOT

## 6.1 Background

(Costantini and Paolucci, 2010) describes a technique for extracting knowledge from natural language and automatically translate it into ASP. This translation method takes into accounts all words of the sentence is presented. This implies that the final representation is too dependant from original lexical structure, and this is sub-optimal if we want to export our knowledge base to a formalism like OWL. To translate into OOLOT, we built an extension of λ-calculus and we have introduced meta expressions to fully automate the translation process, originally inspired by (Baral et al., 2008). This is a key point in the process of representing knowledge extracted from natural language, and thus for using it into other contexts, e.g. the Semantic Web. The selection of a suitable formalism plays an important role, but under many aspects first-order logic would represent a natural choice, and it is actually not appropriate for expressing various kinds of knowledge, i.e., for dealing with default statements, normative statements with exceptions, etc. Recent work has investigated the usability of non-monotonic logics, like ASP (Baral et al., 2008) with great result in terms of dealing with the kind of knowledge represented through natural language.

OOLOT allows us to have native reasoning capabilities (using ASP) to support the syntactic and semantic analysis tasks. Embedded reasoning is of fundamental importance for the correct analysis of complex sentences, as shown in (Costantini and Paolucci, 2008), but having an ASP based host language is not limited to the previous aspects. In fact the integration of ASP and the Semantic Web is not limited on the Natural Language Processing side. Answer Set Programming fits very well with Semantic Web in the research efforts of integrating rule-based inference methods with current knowledge representation formalisms in the Semantic Web (Eiter, 2010; Schindlauer, 2006).

Ontology languages such as OWL and RDF Schema are widely accepted and successfully used for semantically enriching knowledge on the Web. However, these languages have a restricted expressivity if we have to infer new knowledge from existing. Semantic Web needs a powerful rule language complementing its ontology formalisms in order to facilitate sophisticated reasoning tasks. To overcome this gap, different approaches have been presented on how to combine Description Logics with rules, like in (Schindlauer, 2006).

Table 2: The λ-ASP-expression template.

| Lexicon | SemClass | $\lambda - ASP - expression\ T$ |
|---|---|---|
| - | noun | $\lambda x.\langle noun\rangle(x)$ |
| - | verb | $\lambda y.\langle verb\rangle(y)$ |
| - | transVerb | $\lambda y.\lambda w.\langle verb\rangle(y,w)$ |
| many | det | $\lambda u\lambda v.($ $v@X \leftarrow u@X,$ $not\ \neg v@X,$ $possible(v@X,u@X),$ $usual(v@X,u@X)$ $)$ |

## 6.2 Lambda Calculus translation

λ-calculus is a formal system designed to investigate function definition, function application and recursion. Any computable function can be expressed and evaluated via this formalism (Church, 1932). In (Costantini and Paolucci, 2010) we extended the λ-calculus introducing the λ-$ASP\text{-}Expression_T$ that allows a native support for ASP and, at the same time, permits to formally instantiate to λ-ASP-Expression (Baral et al., 2008; Costantini and Paolucci, 2010). For the purpose of our running example, the set of λ-$ASP\text{-}Expression_T$ is available in Table 2. The choice of the lambda calculus was made because it fully matches the specifications of the formal tool we need to drive the execution of the steps in the right way.

According to the workflow in Fig.1, the translation from plain text to the OOLOT intermediate format makes use of the information extracted in several steps. The information on the deep structure of the sentence is now used to drive the translation using the λ-calculus according to the λ- expression definitions in Table2. For each lexicon, we use the phrase structure in Fig.2 to determine the semantic class to which it belongs. In this way, we are able to fetch the correct *lambda*-ASP-expression template from the Table 2. For the running example, as result we have the λ-ASP-expressions of Table 3.

Now, differently from (Costantini and Paolucci, 2010), we use the dependencies, that is the deep structure information, to drive the translation. According to dependency in Fig.3, the first relation that we use is $amod(girls-2,\ many-1)$. Thus, for the λ-calculus definition, we apply the λ-ASP-expression for *girls* to the λ-ASP-expression for *many*, obtaining:

Table 3: The λ-ASP-expressions.

| Lexicon | $\lambda - ASP - expression$ |
|---|---|
| apples | $\lambda x.dbpedia : Apple(x)$ |
| eat | $\lambda y \lambda w.dbpedia : Eating(y,w)$ |
| girls | $\lambda z.dbpedia : Girl(x)$ |
| many | $\lambda u \lambda v.($<br>$v@X \leftarrow u@X,$<br>$not\ \neg v@X,$<br>$possible(v@X,u@X),$<br>$usual(v@X,u@X)$<br>$)$ |

$\lambda v.($
$v@X \leftarrow dbpedia : Girl(X),$
$not\ \neg v@X,$
$possible(v@X, dbpedia : Girl(X)),$
$usual(v@X, dbpedia : Girl(X))$
$)$

The second relation, $nsubj(eat-3,\ girls-2)$, drives the application of the λ-expression for *eat* to the expression for *girls* that we obtained in the previous step:

$$
\begin{aligned}
&dbpedia : Eating(X,W) \leftarrow \\
&\qquad dbpedia : Girl(X), \\
&\quad not\ \neg dbpedia : Eating(X,W), \\
&\quad possible(dbpedia : Eating(X,W), \\
&\qquad dbpedia : Girl(X)), \\
&\quad usual(dbpedia : Eating(X,W), \\
&\qquad dbpedia : Girl(X))
\end{aligned}
$$

Then, we apply *apple* to the expression we have seen, obtaining the final result:

$$
\begin{aligned}
&dbpedia : Eating(X, dbpedia : Apple) \leftarrow \\
&\qquad dbpedia : Girl(X), \\
&\quad not\ \neg dbpedia : Eating(X, dbpedia : Apple), \\
&\quad possible(dbpedia : Eating(X, \\
&\qquad\qquad dbpedia : Apple), \\
&\qquad dbpedia : Girl(X)), \\
&\quad usual(dbpedia : Eating(X, dbpedia : Apple), \\
&\qquad dbpedia : Girl(X))
\end{aligned}
$$

## 7 EXPORTING INTO OWL

Our framework has been designed to export the knowledge base from OOLOT into a target formalism. For now, we are working on a pure ASP and OWL exporter.

Exporting into OWL is a very important feature, because it allows endless possibilities due to its native Semantic Web integration. In this way, the framework as a whole become a power tool that starting from plain text produces the RDF/OWL representation of the sentences; through ASP it takes care of special reasoning and representation features of natural language. To complete the example, the resulting RDF/OWL representation is:

< http://dbpedia.org/resource/Girl,
http://dbpedia.org/ontology/Eating,
http://dbpedia.org/resource/Apple >

Figure 4: RDF.

Clearly, the exporting into OWL has, at this time, some drawbacks, including the loosing of some aspect of natural language that instead is perfectly managed in OOLOT. Exporting is an ongoing work, so there is room for improvement.



## 8 CONCLUSIONS

In this paper, we have proposed a comprehensive framework for extracting knowledge from natural language and representing the extracted knowledge in suitable formalisms so as to be able to reason about it and to enrich existing knowledge bases. The proposed framework is being developed and an implementation is under way and will be fully available in short time. The proposed approach incorporates the best aspects and results from previous related works and, although in the early stages, it exhibits a good potential.

Future improvements concern many aspects of the framework. On the OOLOT side, there is the need to better formalize the language itself, and better investigate the reasoning capabilities that it allows, and how to take the best advantage from them. The ontology-oriented integration is at a very early stage, and there is room for substantial improvements, including a better usage of the current reference ontologies, and the evaluation study about using an upper level ontology, in order to have a more homogeneous translation.

## ACKNOWLEDGEMENTS

This work was improved by conversations with Prof. Eng. Giovanni De Gasperis. We take immense pleasure in thanking him for discussions and his support.

## REFERENCES


Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., and Ives, Z. (2007). Dbpedia: A nucleus for a web of open data. *The Semantic Web*, pages 722–735.

Banerjee, S. and Pedersen, T. (2002). *An Adapted Lesk Algorithm for Word Sense Disambiguation Using WordNet*, volume 2276 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg.

Baral, C., Dzifcak, J., and Son, T. C. (2008). Using answer set programming and lambda calculus to characterize natural language sentences with normatives and exceptions. In *Proceedings of the 23rd national conference on Artificial intelligence - Volume 2*, pages 818–823. AAAI Press.

Bos, J. and Markert, K. (2005). Recognising textual entailment with logical inference. In *HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 628–635. Association for Computational Linguistics.

Cer, D., de Marneffe, M., Jurafsky, D., and Manning, C. (2010). Parsing to stanford dependencies: Trade-offs between speed and accuracy. *LREC 2010*.

Charniak, E. (1996). Tree-bank grammars. In *Proceedings of the National Conference on Artificial Intelligence*, pages 1031–1036.

Charniak, E. and Johnson, M. (2005). Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 173–180. Association for Computational Linguistics.

Chomsky, N. (1956). Three models for the description of language. *IEEE Transactions on Information Theory*, 2(3):113–124.

Chomsky, N. (1957). *Syntactic Structures*. The MIT Press.

Church, A. (1932). A set of postulates for the foundation of logic. *The Annals of Mathematics*, 33(2):346–366.

Collins, M. (1996). A new statistical parser based on bigram lexical dependencies. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 184–191. Association for Computational Linguistics.

Collins, M. (1997). Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, pages 16–23. Association for Computational Linguistics.

Costantini, S. and Paolucci, A. (2008). Semantically augmented DCG analysis for next-generation search engine. *CILC (July 2008)*.

Costantini, S. and Paolucci, A. (2010). Towards translating natural language sentences into asp. In *Proc. of the Intl. Worksh. on Answer Set Programming and Other Computing Paradigms (ASPOCP), Edimburgh*.

De Marneffe, M., MacCartney, B., and Manning, C. (2006). Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*, volume 6, pages 449–454. Citeseer.

De Marneffe, M. and Manning, C. (2008). The stanford typed dependencies representation. In *Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation*, pages 1–8. Association for Computational Linguistics.

Eiter, T. (2010). Answer set programming for the semantic web. *Logic Programming*, pages 23–26.

Kasneci, G., Ramanath, M., Suchanek, F., and Weikum, G. (2008). The YAGO-NAGA approach to knowledge discovery. *SIGMOD Record*, 37(4):41–47.

Klein, D. and Manning, C. (2003a). Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 423–430. Association for Computational Linguistics.

Klein, D. and Manning, C. (2003b). Fast exact inference with a factored model for natural language parsing. *Advances in neural information processing systems*, pages 3–10.

Kowalski, R. (2011). *Computational Logic and Human Thinking: How to be Artificially Intelligent - In Press*. Cambridge University Press.

McClosky, D., Charniak, E., and Johnson, M. (2006). Effective self-training for parsing. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 152–159. Association for Computational Linguistics.

Mollá, D. and Hutchinson, B. (2003). Intrinsic versus extrinsic evaluations of parsing systems. In *Proceedings of the EACL 2003 Workshop on Evaluation Initiatives in Natural Language Processing: are evaluation methods, metrics and resources reusable?*, pages 43–50. Association for Computational Linguistics.

Neuhaus, P. and Bröker, N. (1997). The complexity of recognition of linguistically adequate dependency grammars. In *Proc. of ACL-97/EACL-97*.

Pereira, F. and Shieber, S. (2002). *Prolog and natural-language analysis*. Microtome Publishing.

Petrov, S., Barrett, L., Thibaux, R., and Klein, D. (2006). Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 433–440. Association for Computational Linguistics.

Petrov, S. and Klein, D. (2007). Improved inference for unlexicalized parsing. In *Proceedings of NAACL HLT 2007*, pages 404–411.

Schindlauer, R. (2006). Answer-set programming for the Semantic Web.

Tesnière, L. (1959). *Elèments de syntaxe structurale*. Klincksieck, Paris. ISBN 2252018615.