Intertwining Deep Syntactic Processing and Named Entity Detection

Caroline Brun and Caroline Hagège

Xerox Research Centre Europe
6, Chemin de Maupertuis
38210 Meylan France
{Caroline.Brun,Caroline.Hagege}@xrce.xerox.com

Abstract. In this paper, we present a robust incremental architecture for natural language processing centered around syntactic analysis but allowing at the same time the description of specialized modules, like named entity recognition. We show that the flexibility of our approach allows us to intertwine general and specific processing, which has a mutual improvement effect on their respective results: for example, syntactic analysis clearly benefits from named entity recognition as a pre-processing step, but named entity recognition can also take advantage of deep syntactic information.

1 Introduction

The robust system presented in this article performs deep syntactic analysis associated with the detection and categorization of named entities that are present in texts. This system is robust as it takes any kind of text as input and always gives an output in a short time (about 2000 words/second). At the same time, we show that robustness is not synonymous with shallowness, as our system is able to handle fine-grained syntactic phenomena (like control and raising). Furthermore, our system is flexible enough to enable the integration of specialized modules, as we did for named entity recognition. We first describe our system, then focus on the entity recognition module we developed and show how it is integrated in the general processing chain. We then give some examples of the benefit of having these two modules developed together: syntactic analysis benefits from named entity recognition and the task of named entity recognition benefits from a fine-grained syntactic analysis. Finally we conclude by giving some hints of our future work.

2 Description of Our System

2.1 Robust and Deep Syntactic Analysis Using XIP

XIP (Xerox Incremental Parser) (see Aït et al.[2]) is the tool we use to perform robust and deep syntactic analysis. Deep syntactic analysis consists for us in the construction of a set of syntactic relations from an input text. Although dependency grammars (see Mel'čuk [12] and Tesnière [16]) inspired us, we prefer calling the syntactic output of our system syntactic relations as we do not obey principles like projectivity and we take liberties with the above-mentioned syntactic paradigms. These relations[1] link lexical units of the input text and/or more complex syntactic domains that are constructed during the processing (mainly chunks, see Abney [1])). These relations are labelled, when possible, with deep syntactic functions. More precisely, we try to link a predicate (verbal or nominal) with what we call its deep subject, its deep object, and modifiers. When the deep subjects and deep objects are not found, the general syntactic relations are still available. For instance, for the sentence *The escheat law cannot be enforced now because it is almost impossible to locate such property, Daniel declared.*, the parser produces the following relations:

```
DETD(law,The)              MOD_POST_INFINIT(impossible,locate)
MOD_PRE(law,escheat)              MOD_PRE(impossible,almost)
NUCL_VLINK_MODAL(cannot,be)       EMBED_INFINIT(locate,is)
NUCL_VLINK_PASSIVE(be,enforced)   OBJ-N(locate,property)
OBJ-N(enforced,law)               MOD_PRE(property,such)
TIME(enforced,now)                SUBJ-N(declared,Daniel)
EMBED(is,enforced)                MAIN(declared)
NUCL_SUBJCOMPL(is,impossible)     SUBJ-N(is,it)
```

It is important to notice that, in this example, the passive form *The escheat law cannot be enforced* has been recognized as such and then normalized, as we obtain the relation *OBJ-N(enforce,law).*

We now briefly explain below how these relations are obtained.

XIP and General Syntactic Analysis. XIP is a tool that integrates different steps of NLP, namely: tokenization, POS tagging (combination of HMM and hand-made rules), chunking and the extraction of syntactic relations. Chunking is not compulsory for the syntactic relation extraction, but we decided to apply this first stage of processing in order to find the boundaries of non-recursive phrases. This preliminary analysis will then facilitate the latter processing stage (See Giguet's work [6] for more detailed indications of the interest of finding chunks in order to ease the extraction of dependencies).

A chunking rule can be expressed in two ways:

– by *sequence rules* which define a list of categories;
– by *ID (immediate dominance) rules* defining sets of categories that are combined with LP (linear precedence) constraints.

In both cases, contextual information can be given.

For instance, the following sequence rule (in which no specific context is given) expresses that a Nominal Chunk (NP) starts with a lexical unit bearing

[1] We consider binary and more generally n-ary relations.

the feature det:+ (i.e. is a determiner), can be followed by 0 or more adjectives which are followed by a lexical unit having the feature noun:+

```
NP = ?[det:+], (adj)*, ?[noun:+] .
```

This rule could also have been expressed by the following ID rule and LP constraints[2].

```
NP -> ?[noun:+], ?[det:+], (adj)* .
[det:+] < [adj:+]
[adj:+] < [noun:+]
```

In our approach, after chunking is performed, the system calculates syntactic relations through what we call deduction rules. These rules apply on a chunk tree (that can be completely flat if no chunks have been previously calculated) and consist in three parts: **context**, **condition** and **extraction**.

Context is a regular expression on chunk tree nodes that has to match with a syntactic construction.

Condition is a boolean condition on dependencies, on linear order between nodes of the chunk tree, or on a comparison of features associated with nodes.

Extraction corresponds to a list of dependencies to be created if the contextual description matches and the condition is true.

For instance, the following rule establishes a *SUBJ* relation between the head of a nominal chunk and a finite verb:

```
| NP{?*,#1[last:+]},  ?*[verb:~],  VP{?*, #2[last:+]}|
if (~SUBJ(#2,#1))
SUBJ(#2,#1).
```

The first line of the rule correspond to context and describe a nominal chunk in which the last element is assigned to the variable #1, followed by any thing but a verb, followed by a verbal chunk in which the last element is assigned to the variable #2. The second line checks wether a *SUBJ* relation exists between the lexical nodes corresponding to the variable #2 (the verb) and #1 (the head of the nominal chunk). The test is true if the *SUBJ* relation does not exist. If both context and condition are verified, then a relation *SUBJ* is created between the verb and the noun (last line).

An important feature is that our parser always provides a unique analysis (it is deterministic), this analysis being potentially underspecified.

XIP and Deep Syntactic Analysis. Together with surface syntactic relations handled by a general English grammar, we calculate more sophisticated and complex relations using derivational morphology properties, deep syntactic properties (subject and object of infinitives in the context of control verbs), and some limited lexical semantic coding (Levin's verb class alternations). These deep syntactic relations correspond roughly to the agent-experiencer roles that

[2] The symbol $<$ expresses linear precedence.

is subsumed by the *SUBJ-N* relation and to the patient-theme role subsumed by the *OBJ-N* relation. Not only verbs bear these relations but also deverbal nouns with their corresponding arguments (for more details on deep syntactic analysis using XIP see Hagge and Roux[7]).

For instance the following rule establishes that the surface subject of a verb in passive form is in fact the deep object of this verb while the surface object of this verb corresponds to the deep subject.

```
if ( SUBJ(#1[passive:+],#2) & OBJ(#1,#3) )
    SUBJ-N(#1,#3),
    OBJ-N(#1,#2) .
```

At the end of the deep syntactic analysis stage, deep syntactic relations together with surface syntactic relations are available.

2.2 A Basic System for Named Entity Categorization

Named entity recognition and categorization is a fundamental task for a wide variety of natural language processing applications, such as question answering, information management, text mining and business intelligence, lexical acquisition, etc. Therefore, the NLP community shows a great interest concerning this issue. For example, the MUC conferences defined a task of named entity recognition using annotated corpora, and enabled the comparison of different methods for the task (see Sekine and Eryguchi[14] for an interesting state of the art of the different methodologies, and Poibeau [13] for an analysis of the evaluation criteria). More recently, the ACE [3] project (Automatic Content Extraction) has a specific task concerning named entities as well.

This task is also a useful step towards achieving fined-grained syntactic and semantic analysis. For these reasons, it seemed useful to integrate such functionality into the XIP parser. Moreover, the overall parsing process should benefit from the integration of this module.

The system we built for named entity categorization focuses on the following predefined classes:

– percentages, e.g. *10%*, *10 per cent*
– dates, e.g. *March 4, 1991*, and temporal expressions, e.g. *Tuesday*
– expressions denoting an amount of money, e.g. *$26 billion*
– locations, e.g. *West Bank*, *Mount Everest*
– person names e.g. *President Saddam Hussein*, *Jacques Chirac*, *Edward III of England*
– organizations e.g. *British Aiways*, *Bang & Olufsen*, *Bank of Brazil*
– events e.g. *World War II*, *America's Cup*
– legal documents *Warsaw Pact*, *Maastricht Treaty*

This list is non-exhaustive, but corresponds to the most common types of entities generally recognized by dedicated systems. This "basic" system is built

[3] http://www.ldc.upenn.edu/Projects/ACE/intro.html

within the XIP parser presented above, on top of a part-of-speech tagger. This system is purely rule-based. It consists in a set of ordered local rules that use lexical information combined with contextual information about part-of-speech, lemma forms and lexical features. These rules detect the sequence of words involved in the entity and assign a feature (loc, org, etc.) to the top node of the sequence, which is a noun in most of the cases. In the incremental parsing process, these rules are applied in the pre-syntactic component, before the chunking and dependency rules, therefore no syntax is used at this stage. For example, the following rule is used to detect and categorize organization names consisting in a sequence of nouns starting with a capital letter (feature *cap*) and finishing with a typical organisation marker like *Ltd.*, *Corp*, *Inc.*, etc. These markers bear the feature *orgEnd* within the lexicon.

```
noun[org=+] -> noun+[cap=+],noun[orgEnd=+]
```

The rule enables the detection of the organization name in the following example:

<ORG>Apple Computer Inc. </ORG> said its chief financial officer, Fred Anderson, will retire june 1.

At this stage of the processing, one can already use contextual information in these rules. This is illustrated on the following rule:

```
noun[person=+] = noun[cap=+] |part, noun[famtie=+]|
```

This rule transforms (overriding sign =) a noun starting with a capital letter in a person name, when it is followed by an element of category part (*'s*) and by a noun bearing the feature *famtie*, like *brother*, *mother*, etc. It enables the detection of the person name in the following example, *They were flying to Miami for a bash hosted by <PERS>Graf</PERS>'s brother.*

These rules are combined with a propagation system integrated into the parser, that allows subparts of the entities to be marked, and then allows new occurrences of these subparts to be categorized when encountered later on in the text. For example if the word *Washington*, which is ambiguous between a person name and a city name, is encountered in some part of the text in the string *George Washington*, then the system tags all remaining occurrences of *Washington* as a person, using feature propagation on this entity part. This functionality is also very useful when proper nouns are truncated, which is very common for (business) organisation names:

<ORG>Allied Supermarkets Inc</ORG> said it filed with the Securities and Exchange Commission ... <ORG>Allied</ORG> named Drexel Burnham Lambert Inc and Donaldson, Lufkin and Jenrette Securities Corp as co-underwriters of both offerings.

At the current stage of development, the basic system contains about 300 local grammar rules for entity detection.

Since the entity recognition system is embedded in a syntactic parser, the corresponding rules have been built in order to maintain a high precision, which attempt to prevent any deterioration of the parsing results. We have conducted a preliminary evaluation on a short corpus (about 500 sentences) from the Reuters news agency, on the location, person, and organisation named entities. It led to 90.2% precision and 75.5% recall.

2.3 Complete Architecture

The complete architecture of the parsing system, including entity processing is shown on figure 1. Parsing and named entity categorization can interact one with another (bold arrows).

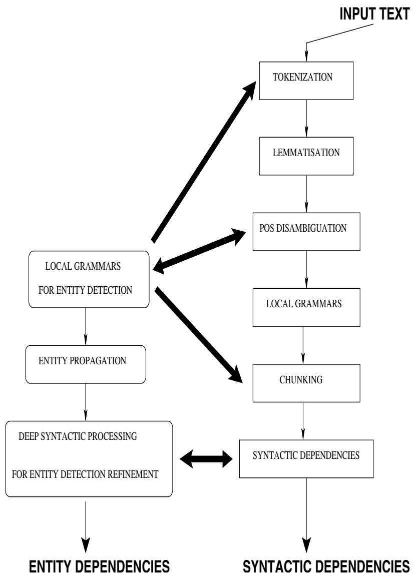

Fig. 1. Architecture of the System

3 Parsing and Entity Categorization as Interleaved Processes

3.1 Syntax and Entities

As presented before, our entity recognition system is embedded in our syntactic parser. Apart from being a useful task by itself, entity recognition improves the overall quality of the parsing output.

Entities for Better Tokenization. The first straightforward improvement concerns tokenization problems. Treating numbers, dates, monetary expressions, etc., as syntactic units avoids many word segmentation problems, since these expressions are often formed with punctuation signs that could be misinterpreted at a deeper syntactic level. Consider the following sentence:

George Unzelman, president of HyOx, Inc., said these proposals "will place pressure on small refining operations and promote further industry consolidation."

Analysing *HyOx, Inc.* as a unit (organization), allows the nominal apposition of *George Unzelman* to be delimited properly: the interpretation of the commas needs to be different when it marks the boundaries of a nominal apposition or when it is employed within an organisation name. This is what is reflected by our incremental multi-layer analysis.

Entities and POS Tagging. Still in pre-syntactic processing, at the level of POS disambiguation, the previous detection of entities can enable to avoid POS tagging errors that will have then consequences in the rest of processing. Take for instance the following example *Seat and Porsche had fewer registrations in July 1996.* Having *Seat* detected as an organization prevents it being interpreted as a verb which will have important consequences in the syntactic processing.

Entities and Syntactic Analysis. More interestingly, another kind of improvement concerns syntactic analysis directly. In fact, entity categorization gives some semantic information that can benefit the syntactic analysis. We can take a concrete case that represents one of the difficult points in the syntactic processing, namely the treatment of coordination.

Consider for instance the two following utterances where geographical entities are marked up with <LOC> and organizations with <ORG>:

He will be replaced by Eliahu Ben-Elissar, a former Israeli envoy to <LOC> Egypt < /LOC> and <LOC> Jordan < /LOC>

and

He will be replaced by Eliahu Ben-Elissar, a former Israeli envoy to <LOC> Egypt < /LOC> and <ORG> Likud party < /ORG> politician.

In the first example, the fact that *Jordan* and *Egypt* are both geographical entities enables us to consider that they are coordinated together. In the contrary, in the second example, as *politician* is not of the same type than *Egypt*, we will prefer to coordinate it with the name *envoy.*

Entities : A Step Towards Semantics. Finally, when an entity name is dependent on a governor, knowing the semantic type of the entity can help determining the kind of relationship that exists between the entity and its governor. It can help the labelling of semantic-oriented relations. Take the following example:

They met in Baghdad.

- Knowing that *Baghdad* is marked up as a geographical entity,
- having the syntactic relation *MODIFIER* between *met* and *Baghdad* where *met* is the governor and *Baghdad* a dependent,
- and knowing that *Baghdad* is introduced by the preposition *in* enables the MODIFIER syntactic relation to be further specified as a LOCALIZATION relation.

3.2 Entities and Deep Syntax

Entity Metonymy. Recently, the ACE project had focused on *Entity Detection and Tracking*, *Relation Detection and Characterization* and *Event Detection and Characterization.* In the context of this project, a particular emphasis is placed on named entity metonymy, namely the fact that a given entity may have different "senses", and therefore should be categorized differently according to the context. For instance, the word *China* can be marked as a geographical entity (LOC), but when used in the following context: *"China on Wednesday called on Japan to acknowledge its wartime past..."*, it is obvious that *China* should not be considered as a geographical unit but as a human organization. In a similar way, the word *Rolex* in *"If you are a real Rolex lover like I am, purchase the book"* should not be typed as an organization (Swiss watch designer company) but as a common noun (watches made by the Rolex company). This phenomenon is distinct from "basic ambiguity", as for *Washington*, which is either a location or a person. Therefore ACE focuses on semantic analysis whereas previous project in the same line, like MUC, focussed on linguistic analysis.

In this context, Maynard et al.[11] show how they adapt their "standard" entity detection system to the ACE requirement, in a very efficient way, taking advantage of the modularity of the GATE architecture. However, the adaptation of the system to take into account the phenomena of metonymy is not described. Furthermore, in the work described by Maynard et al., parsing is not one of the stages in the processing chain.

Along these lines, we decided to do an experiment on the contribution of deep syntactic parsing to named entity metonymy detection, making use of the flexibility of our architecture.

The system of named entity extraction we presented above can be enriched and improved using the results of robust deep syntactic processing with some limited lexical semantic coding. As claimed in McDonald[10], richer contextual information is necessary for high accuracy, and in our case this richer information consists in deep parsing results combined with lexical knowledge.

The enrichment, provided by an independent module, can be applied to any kind of named entity extraction system (rule-based in our case but it could also be used on top of a learning system or a mixed approach). This enables better semantic categorization of entities and also the discovery of named entities that are not easily detectable within a restricted context. Moreover, this information will allow entity categorization to be overridden or more precisely specified when some strings that could denote entity names are used as common nouns (e.g. *"I drive an Alfa Romeo"* where *Alfa Romeo* is here an artefact, i.e. a car of brand Alfa Romeo). To a certain extent, the task we want to perform is similar to Word Sense Disambiguation (see for example Ide and Veronis[8]), in the context of named entity categorization: we use prototypical information about subcategorization frames to disambiguate named entities.

In the following subsections we describe our module for the refinement of semantic categorization of named entities, which is based on deep syntactic processing.

This deep syntactic processing produces a normalized syntactic analysis (taking advantage of morphological properties of words and of a minimal lexical semantic information). At the end of the analysis process, the labeling of previously detected entities is refined. Some categories attached to entities that have been detected may be overridden by others or simply discarded. Moreover, it is important to observe that our methodology improves entity detection for other kind of problems:

– It helps characterize highly ambiguous entities, since they are syntactically related to words which bear some semantic features (e.g. "<PERS> Turner </PERS> says..." vs. "<LOC> Turner </LOC> is located in the hills of Western Maine")
– It enables entities to be typed even when gazette er information is missing (e.g. "<ORG> DELFMEMS </ORG> will be a new company created in 2004").

It is important to notice that we do not follow the ACE guidelines (ACE[5]) exactly, in particular for the Geopolitical entities (GPE, i.e. "composite entities comprised of a population, a government, a physical location and a nation"), for which we have a less fine-grained tagset. Indeed, we do not keep the distinction between:

– *France signed a treaty with Germany last week* : GPE with role ORG;
– *France likes to eat cheese* : GPE with role PERS;
– *The world leaders meet in France yesterday* : GPE with role LOC;
– *France produces better wine than New Jersey*: GPE with role GPE;

Since for us, an organisation is a basically a group of people, we simply aim to distinguish the location sense of GPE from the other senses, which we consider as organization.

Moreover, our system focuses on proper noun categorization: it won't attempt to spot common noun like *the house painters*, as in ACE.

How Deep Syntax Can Help. The contextual rules of the entity recognition module described above enable us to catch and categorize named entities with reasonable precision. In this section we show how deep syntax can help and improve the named entity recognition task in the following ways:

– refine a rough but correct categorization done in a previous step,
– detect some entities that have not been detected previously,
– override an incorrect entity categorization that has been previously made.

As said before, our system can extract deep syntactic relations between predicates and their arguments. Having some information about selectional restriction of predicates appearing in the text thus enables us to check and possibly correct the tagging of named entities which are arguments of these predicates. In this paper, what especially interests us is the *SUBJ-N* relation that links a predicate to its normalized subject. If a predicate denotes an activity that is typically performed by human beings, then we can be sure that the normalized subject of this predicate has to bear a human feature. Having this in mind, when entities are found to be involved in a *SUBJ-N* dependency with a predicate that expresses a human activity, then we know that this entity cannot be a product or or a location but something that denotes one or a group of human beings. This enables the refinement of named entity tags found in a previous step or possibly the detection of some entities that have not been previously detected because of a lack of context. For instance in the example *China declared ...* as *declared* is typically a verb whose deep subject is a human or a group of human beings, we can easily that the *LOC* interpretation for the entity *China* is not correct and that *China* is here an organisation.

Furthermore, simple syntactic contexts can be clues to override erroneous named entity categorization. For instance in the example *He won the race with his Mitsubishi*, as we have a possessive *his* determining the word *Mitsubishi*, it is very unlikely that *Mitsubishi* has the status of a company name and hence is not a named entity even if starting with upper-case and present in some gazetteer.

The limited lexical coding that we performed in order to refine our named entity categorization module consists mostly in the categorization of a set of verbs expecting a human (or at least a living) being or a group of human beings as normalized subject.

As much as possible, we use pre-defined semantic verb classes, such as Levin's classes (see Levin [9]). Interesting classes that we found in Levin are the following:

– "Learn verbs", for instance *learn, memorize, study*, etc.
– "Social interaction verbs", for instance *agree, argue, struggle, debate*, etc.
– "Communication verbs", for instance *dictate, write, recite, telephone*, etc.
– "Ingestion verbs", for instance *eat, gobble, swallow, devour*, etc.
– "Killing verbs", for instance *murder, massacre, poison, strangle*, etc.

Together with these significant Levin classes, we also use verbs that introduce indirect speech which were already marked in our system, as they possess specific syntactic properties (inversion of the subject for instance). This class of verb was extracted from the COMLEX lexicon [4] and consists in verbs like *say, declare, utter*, etc.

We give below an example of a XIP rule showing that if we find that the deep subject of a verb of human activity is tagged as a named entity denoting a location, then, we retract this interpretation and we type this entity as a person or human organisation (PERS_OR_ORG unary relation).

```
if (  SUBJ-N(#1[human_activ],#2) & ^LOCATION(#2) )
        PERS_OR_ORG(#2)
```

The next section shows that this limited and straighforward classification, even if modest, refines some of the entity categorization performed by the general system.

3.3 About Evaluation

In this section we describe a small experiment we performed in order to evaluate the impact of deep syntactic analysis on the named entity detection and categorization task.

In this experiment, the challenge is different than what we expect from the general named entity detection system. We want to be able to distinguish here when an entity like *China* denotes a place and when it denotes an organization (see ACE and GPE distinctions above). Once again, we use a small corpus of about 500 sentences and annotate it manually, taking metonymy into account.

Results we obtained with the general entity detection system on the refined catogories set are the following:

- Precision : 81 %
- Recall : 75 %

Using deep syntax and the limited lexical information we mentionned above, we obtained the following results:

- Precision : 85 %
- Recall : 74 %

These results show that with a minimal effort in lexical coding and with the use of only two kinds of grammatical relations (namely deep-subject and possessive), precision increases. We expect that a deeper study on the impact of syntactic properties on entity categorization will enable us to go further in that direction.

4 Conclusions and Future Work

In this paper, we present a robust architecture for deep syntactic analysis, enriched by a named entity detection module. Named entity detection is often seen as an independent NLP task, but we show that a syntactic parser can benefit from it as a pre-processing step. Moreover, since recent trends in named entity categorization focus on semantic analysis (e.g. metonymy), we think that deep syntactic information is necessary to bridge the gap between a linguistic analysis and a deeper semantic analysis of named entities. We thus propose a system that interleaves parsing and entity detection. First results are encouraging, and we plan to pursue our work with a deeper linguistic study of the syntactic information needed to improve our system.

In addition to that, we think that Word Sense Disambiguation is a task that should make the most of entity categorization. We developed previously a rule-based Word Sense Disambiguation system of which one the main components is our syntactic parser (Brun and Segond[3]). Since the integration of named entity categorization results is handled directly by our architecture, it could be worthwhile to evaluate the consequences on the Word Sense Disambiguation task.

References

1. Abney S. Parsing by Chunks. In *Robert Berwick, Steven Abney and Carol Tenny (eds.). Principle-Based Parsing*, Kluwer Academic Publishers (1991).
2. Aït-Mokhtar S., Chanod J-P and Roux C. Robustness beyond shallowness: incremental dependency parsing. Special issue of the NLE Journal (2002).
3. Brun C., Segond F. Semantic encoding of electronic documents. In *International Journal of Corpus Linguistic, vol. 6, no 1, 2001.*
4. Grishman R., Macleod C., Meyers A. COMLEX: building a computational lexicon. In *Proceedings of the 15th International Conference on Computational Linguistics* (1994).
5. EDT Guidelines for English V4.2.6. In *ACE Current Annotation Effort: http://www.ldc.upenn.edu/Projects/Ace/Annotation.*
6. Giguet E. Mthodes pour l'analyse automatique de structures formelles sur documents multilingues. Thse de Doctorat en Informatique de l'Universit de Caen. (1998)
7. Hagge C., Roux C. Entre syntaxe et smantique: Normalisation de l'analyse syntaxique en vue de l'amlioration de l'extraction d'information partir de textes. In *Actes de TALN 2003, Batz-sur-Mer, France.* (2003)
8. Ide N., Veronis J. Introduction to the Special Issue on Word Sense Disambiguation: The State of the Art In *Computational Linguistics, vol. 24, no 1, 1998.*
9. Levin B. English Verb Classes and Alternations - A Preliminary Investigation. The University of Chicago Press (1993).
10. MacDonald. Internal and External Evidence in the Identification and Semantic Categorization of Proper Names. In *B. Boguraev and J.Pustejovsky editors, Corpus Processing For Lexical Acquisition, pp 21-29, Mit Press.*
11. Maynard D., Bontcheva K., Cunningham H. Towards a semantic extraction of named entity recognition. In *Proceedings of RANLP 2003, Borovets, Bulgaria. (2003).*
12. Mel'čuk I. Dependency Syntax. State University of New York, Albany, (1988)
13. Poibeau, Thierry.: Deconstructing Harry, une valuation des systmes de reprage d'entits nommes. In *Revue de la socit d'lectronique, d'lectricit et de traitement de l'information, 2001.*
14. Sekine S., Eryguchi Y. Japanese Named Entity Extraction and Evaluation - Analysis of Results. In *Proceedings of Coling'2000,Saarbrucken, Germany, pp. 25-30.*
15. Sekine S., Kiyoshi S., Chikashi N. Extended Named Entity Hierarchy. In *Proceedings of LREC2002, Las Palmas, Spain, 2002.*
16. Tesnière L. Eléments de syntaxe structurale. Editions Klincksieck, Deuxième édition revue et corrigée Paris (1969)