A generic tool to generate a lexicon for NLP from Lexicon-Grammar tables

Matthieu Constant [1], Elsa Tolone[1]
Université Paris-Est

Abstract

Lexicon-grammar tables constitute a large-coverage syntactic lexicon but they cannot be directly used in Natural Language Processing (NLP) applications because they sometimes rely on implicit information. In this paper, we introduce a generic tool for generating a syntactic lexicon for NLP from the lexicon-grammar tables. It relies on a global table that contains undefined information and on a unique extraction script including all operations to be performed for all tables. We also show an experiment that has been conducted to generate a new lexicon of French verbs and nouns.

Keywords : syntactic lexicon, NLP, lexicon-grammar table.

1. Introduction

Symbolic approaches to deep parsing often require large-coverage and fine-grained lexical information, such as a syntactic lexicon. Lexicon-grammar tables (Gross 1975; Gross 1994), carefully developed by linguists since the 70s, constitute such a syntactic resource. Each table represents a class of predicates sharing some syntactic features. Each row corresponds to lexical entries (verbs, nouns, adjectives, adverbs, frozen expressions) and each column corresponds to syntactic features (constructions, argument distribution, and so on). However, they are not directly exploitable for NLP applications because pieces of information are not formally encoded although their informal descriptions are available in the literature.

Some projects such as (Hathout & Namer 1998; Gardent *et al.* 2006; Sagot & Fort 2007) attempted to reformat lexicon-grammar tables in a lexicon for NLP. In these projects, each class is assigned a specific configuration which encodes missing information and defines restructuration operations. For instance, each configuration in (Gardent *et al.* 2006) is represented by a graph that makes the class structure explicit and translates each column header in a feature structure. Nevertheless, lexicon-grammar tables are continually updated to be improved (e.g. addition and renaming of features) and this approach can be painful for maintenance. For example, if a same feature is added to several classes, all corresponding configurations have to be modified. In this paper, we describe a tool that uses a global approach. First, it relies on the so-called table of classes, which encodes pieces of information that are undefined in the original classes, especially features that are constant over a whole class. Next, as a syntactic feature has exactly one interpretation over the set of classes, each feature is assigned once a set of reformatting operations in an extraction script.

This paper is organized as follows. First, we briefly describe the lexicon-grammar classes and the table of classes, and their relevance to our work. Then, we present in detail our tool, illustrate it with a concrete example for French and evaluate it on the basis of practical uses.

2. Classes in the Lexicon-Grammar

While modern linguistics, under the generative influence, has been trying to model the human language on the basis of a rather small number of samples, scholars working in the lexicon-grammar framework have been concentrated on the construction of syntactic and lexical databases

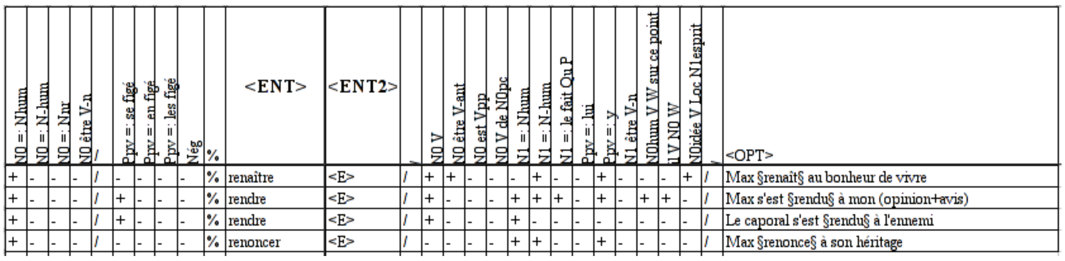
[1] Université Paris-Est, {mconstan,tolone}@univ-paris-est.fr

for more than thirty years (Gross 1975; Gross 1994; Boons *et al.* 1976; Guillet & Leclère 1992). The lexicon-grammar methodology consists in establishing a taxonomy of syntactic-semantic classes the lexical items of which share some syntactic features. For instance, class 33 contains verbs that enter the construction with a human nominal subject and one indirect complement introduced by preposition *à*. Each class is represented with a table including all the lexical items of the class. If a verb has two meanings, it is divided in two lexical items: in the verb class 33 (see figure 1), *se rendre* has two meanings, so two lexical items:

Jean s'est rendu à mon opinion (John finally accepted my opinion)
Vercingetorix s'est rendu à Cesar (Vercingetorix surrendered to Ceasar)

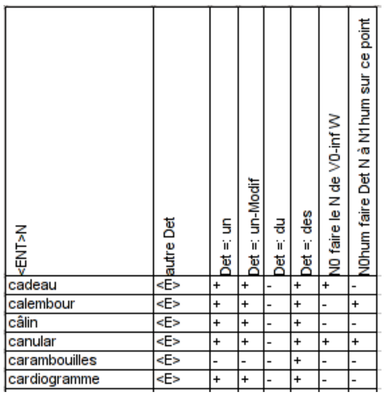
A selection of features are applied to all entries and their linguistic validity is checked. At the intersection of a row corresponding to a lexical item and a column corresponding to a feature, the cell is set to '+' or '-' whether it is valid or not. For instance, one meaning of *se rendre* (to accept) accepts a non human nominal complement in its canonical sentence: its feature `N1 =: N-hum` value is true ('+') while it is false ('-') for the other (to surrender). There are also some features the values of which are lexical items. For instance, the prepositional complements can require different prepositions according to the predicates: in class 1 which is enclosing auxiliary verbs followed by a preposition and an infinitive, *arrêter* (to stop) needs preposition *de* and *commencer* (to begin) needs preposition *à*.

In the classification of French verbs, for example, there are 13,300 verb entries grouped in 59 syntactic classes. The same principles have been applied to the classification of nominal predicates, with approximately 10,300 lexical entries, e.g. figure 2 shows a sample of a noun class from (Giry-Schneider 1987). In the same way, 42,400 frozen expressions have been described.



N0 =: Nhum	N0 =: N-hum	N0 =: Nnr	N0 être V-n	/	Ppv =: se figé	Ppv =: en figé	Ppv =: les figé	Nég	%	<ENT>	<ENT2>	/	N0 V	N0 être V-ant	N0 est Vpp	N0 V de N0pc	N1 =: Nhum	N1 =: N-hum	N1 =: le fait Qu P	Ppv =: lui	Ppv =: y	N1 être V-n	N0hum V W sur ce point	il V N0 W	N0idée V Loc N1esprit	/	<OPT>
+	-	-	-	/	-	-	-	-	%	renaître	<E>	/	+	+	-	-	-	+	-	-	+	-	-	-	+	/	Max §renaît§ au bonheur de vivre
+	-	-	-	/	+	-	-	-	%	rendre	<E>	/	+	-	-	-	+	+	+	-	+	-	+	+	-	/	Max s'est §rendu§ à mon (opinion+avis)
+	-	-	-	/	+	-	-	-	%	rendre	<E>	/	+	-	-	-	+	-	-	-	-	-	-	-	-	/	Le caporal s'est §rendu§ à l'ennemi
+	-	-	-	/	-	-	-	-	%	renoncer	<E>	/	-	-	-	-	+	+	-	-	+	-	-	-	-	/	Max §renonce§ à son héritage

Figure 1. sample of verb class 33



<ENT>N	autre Det	Det =: un	Det =: un-Modif	Det =: du	Det =: des	N0 faire le N de V0-inf W	N0hum faire Det N à N1hum sur ce point
cadeau	<E>	+	+	-	+	+	-
calembour	<E>	+	+	-	+	-	+
câlin	<E>	+	+	-	+	-	-
canular	<E>	+	+	-	+	+	+
carambouilles	<E>	-	-	-	+	-	-
cardiogramme	<E>	+	+	-	+	-	-

Figure 2. sample of noun class FNAN

3. Table of classes

Some basic pieces of information in the lexicon-grammar classification are left implicit in the current version of the Lexicon-Grammar, so they cannot be exploited by NLP tools. For instance, a feature is often explicitly recorded in the entries of a class if its values depend on entries. In particular, classes are defined on the basis of features which are not explicitly recorded in the lexicon. These definitions are only described in literature. To tackle this issue, the principle of table of classes has been established on the model of (Paumier 2003). It consists in assigning features according to the class because some features are constant over a class (e.g. class definition features). Each row stands for a class and each column stands for a feature. Each cell corresponds to the validity of a feature in a class. Two cases can occur:

- the values depend on the entries of the class and must be assigned for each entry; the cell is then filled with the symbol 'o';
- the values are uniform over the class and can be assigned in the cell.

For instance, the table of French verb classes being currently constructed by researchers at the Institut Gaspard Monge of Université Paris-Est is composed of 59 verb classes and 438 features. A sample of this table is given in figure 3. In this table, we can see that definitional features of class 33 are set: e.g. construction feature 'N0 V à N1' is true ('+'). Construction feature 'N0 V N1' is never valid ('-'). Non definitional feature 'N1 =: N-hum' is assigned 'o' because it depends on the lexical entries.

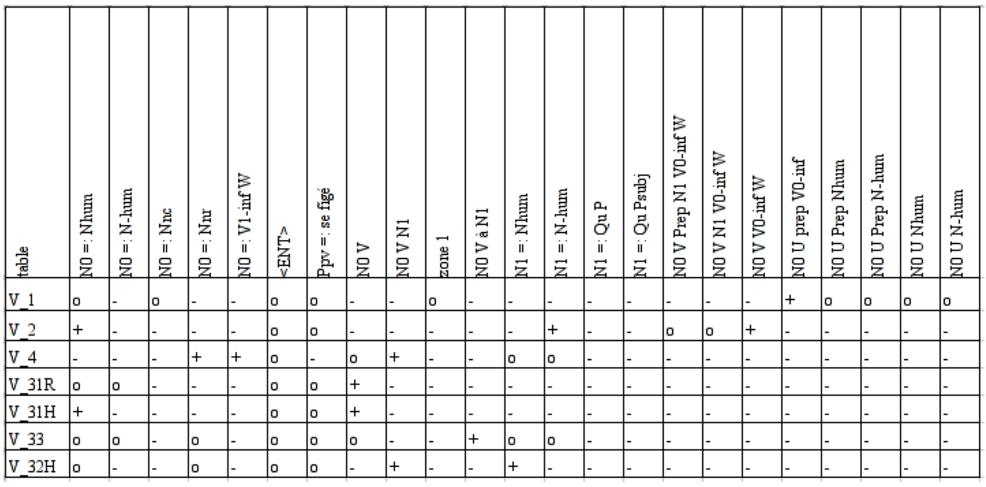


table	N0 =: Nhum	N0 =: N-hum	N0 =: Nnc	N0 =: Nnr	N0 =: V1-inf W	<ENT>	Ppv =: se figé	N0 V	N0 V N1	zone 1	N0 V à N1	N1 =: Nhum	N1 =: N-hum	N1 =: Qu P	N1 =: Qu Psubj	N0 V Prep N1 V0-inf W	N0 V N1 V0-inf W	N0 V V0-inf W	N0 U prep V0-inf	N0 U Prep Nhum	N0 U Prep N-hum	N0 U Nhum	N0 U N-hum
V_1	o	-	o	-	-	o	o	-	-	o	-	-	-	-	-	-	-	-	+	o	o	o	o
V_2	+	-	-	-	-	o	o	-	-	-	-	-	+	-	-	o	o	+	-	-	-	-	-
V_4	-	-	-	+	+	o	-	o	+	-	-	o	o	-	-	-	-	-	-	-	-	-	-
V_31R	o	o	-	-	-	o	o	+	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
V_31H	+	-	-	-	-	o	o	+	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
V_33	o	o	-	o	-	o	o	o	-	-	+	o	o	-	-	-	-	-	-	-	-	-	-
V_32H	o	-	-	o	-	o	o	-	+	-	-	+	-	-	-	-	-	-	-	-	-	-	-

Figure 3. sample of table of verb classes

4. Extracting an NLP lexicon with a simple script

Past proposals for reformatting lexicon-grammar tables into a lexicon for NLP consisted in making a specific setup for each class: selecting relevant features, providing information on the missing features and restructuring the data (Hathout & Namer 1998; Gardent *et al.* 2006). As the definition of the same reformatting operations can be repeated several times over the set of classes because some features occur in several classes, this approach can be painful for encoding and maintenance.

We propose a more global approach by using (a) a unique configuration available for all classes and (b) a table of classes to provide information undefined in the original classes. The global configuration is in the form of a script that is parsed with a parser generated by the tool Tatoo (Cervelle *et al.* 2006). Our program, implemented in Java, takes as input a configuration script and a table of classes. It outputs the set of lexical entries encoded in the classes of the table, formatted as described in the script. It is based on the two following points:

- information is encoded in linguistic objects defined in the script. They are represented by lists and feature structures, that can be combined together: for example, objects define syntactic constituents, distributions of syntactic constituents, constructions, predicate-argument representations, lexical rules; the objects can be parameterized by the syntatic features available in the table of classes;

- each feature of the table of classes is associated with a set of operations that combine linguistic objects together: for instance, when feature *N0 =: Nhum* is true for a given entry, an object defining a human noun phrase is added to the distribution of N0 (argument 0 of the predicate). If the feature is assigned true for a given lexical entry, the associated operations are activated;

This implies that each feature has one and only one interpretation over all classes. If a given feature has two different meanings in two different classes, our system cannot deal with it (as opposed with other systems).

A linguistic object is made of lists and feature structures. An instance of such object is defined by indicating its type, its name and its value. For example, the first instruction below instantiates a constituent (`const`) named `N-hum`, that is a non human noun phrase. These different objects can be combined together: e.g. a distribution is a set of syntactic constituents. In the last instruction below, `X0` contains the distribution of the argument 0 : a human noun phrase (Nhum) and a non human noun phrase (N-hum).

```
define const N-hum [cat="NP",nothum="true"];
define const Nhum [cat="NP",hum="true"];
define const inf [cat="VP",mood="inf"];
define dist X0 [dist=(Nhum,N-hum),pos="0"]
```

Like in every object-oriented programming language, there also exists an inheritance mechanism. For instance, an infinitive introduced by preposition *à* (object `a_inf`) inherits the features of the object `inf` (defining an infinitive), and has a new feature indicating the presence of preposition *à*.

```
define const a_inf inf[prep="à"];
```

All these objects can be parameterized with the features of the table of classes. The parameters are of two types: boolean or string. For example, the code below defines a verbal predicate named `predV`. Its lemma is the value of the feature `<ENT>` (i.e. the lexical value of the entry). The code also indicates that the lexical rule 'passivation transformation with preposition *par* (by)' is encoded with the feature *[passif par]*.

```
define pred predV [cat="verb",lemma="@<ENT>@"];
define lexicalRule passivePar {passivePar="@[passif par]@"};
```

For each lexical entry, the parameters of the linguistic objects are resolved as follows. Each parameter corresponding to a feature, is given a lexical or boolean value. The program first looks up the table of classes. If the feature has a constant value over the whole class the entry belongs to, the feature is assigned this value. If the feature value depends on the lexicon (feature value is 'o' for the line corresponding to the class), the program retrieves the value of the feature of the entry. For instance, the verb *aimer* (love) belongs to class 32H which contains transitive verbs with human subject. The feature '[passif par]' is always true over this class. The two parameterized objects shown above would then become:

```
define pred predV [cat="verb",lemma="aimer"];
define lexicalRule passivePar {passivePar="true"};
```

That means that a piece of information given in the table of classes has greater priority than one in the class of the entry. If a contradiction occurs between table of classes and classes, priority is given the encoding of table of classes.

For each lexical entry, the program can then apply reformatting operations for each features in the table of classes from these "lexicalized" objects. Operations are of one type: `add` an object in an other one. For instance, add an attribute-value pair or a list in a feature structure. The operations are independent of their order of application, i.e. they are non-destructive and do not depend on each other. For instance, when inserting an attribute-value pair *(a,v)* in a feature structure, if there already exists another value *ov* for attribute *a*, the new value is the disjunction of *v* and *ov*. The operation is therefore non-destructive. Lists are actually sets because the result of two additions must be independent of their order of application. Before inserting a new element in a list, the program checks whether it exists or not. If it exists, it is not inserted. For instance, the following code [1] indicates that, if feature 'N0 =: Nnc' is true (i.e. N0 is a free noun phrase), the program adds objects `Nhum` and `N-hum` in the distribution of `N0` and inserts `N0` in the list of constituents.

```
prop @N0 =: Nnc@{
    add N0 in constituents;
    add Nhum in N0.dist;
    add N-hum in N0.dist;
}
```

The resulting lexicon is generated in an XML format. Elements and attributes in XML can be defined by relating them with the linguistic objects in a script. The XML being hardly readable by a human, a compressed textual output has also been implemented (see examples in section 5.).

5. An example of generated lexicon

An example of French lexicon for NLP [2] has been generated from a selection of lexicon-grammar tables, i.e. all tables of verbs and nouns [3], that are freely available under the LGPL-LR license. It is composed of 8,341 verbal entries (from 35 tables) and 4,475 nominal entries (from 30 tables). The extraction script encodes a selection of features. Some have been discarded because they are not exploitable. For instance, we discarded features involving nouns derived from verbs with no explicit information on the derivation procedure. Some features involving body part nouns were not always relevant. The generated lexicon is also provided under the LGPL-LR license. Each entry of the lexicon includes four sections:

- section **Lexical information** identifies the predicate (e.g. verb *se rendre*) and its lexical constraints (e.g. determiner distribution for nouns, and prepositions in the constructions).
- section **Arguments** indicates the nature of the arguments of the predicates: for instance, the argument N0 of *canular* in class FNAN must be a human noun phrase.
- section **Constructions** enumerates identifiers of all constructions of the predicate: e.g. noun predicate *canular* enters the construction family 'N0 faire Det N à N1' (*Jean a fait un canular à Luc* = John made a joke to Luke).
- section **Lexical rules** lists lexical rules accepted by the predicate such as passivation transformation.

[1] It is also possible to factorize parts of the code by using functions.

[2] Several independent initiatives exist, such as Dicovalence (van den Eynde & Mertens 2006), Synlex (Gardent *et al.* 2006) or the *Lefff* (Sagot *et al.* 2006). Nevertheless, the two first ones do not include other classes than verbs and the latter sometimes lacks linguistic precision because it has been acquired semi-automatically.

[3] They can be found at the following url: `http://infolingu.univ-mlv.fr`

The example below shows the code that is generated for the verbal entry *se rendre* (to surrender) of class 33. Arguments 0 and 1 must be human noun phrases. It enters constructions labeled ’N0 V à N1’ and ’N0 V’ that should be described in a grammar such as family of trees that anchor the verb entry.

```
ID=V_33_129
lexical-info:[cat="verb",
              verb:[lemma="rendre",ppvse="true"]
]
args:(const:[pos="0",
             dist:(comp:[hum="true",cat="NP"])
            ],
      const:[pos="1",
             dist:(comp:[hum="true",cat="NP"])
            ]
     )
constructions:(construction="N0 V à N1",construction="N0 V")
...
```

Below is an example of the nominal entry *canular* (joke) in the noun class FNAN the definition construction of which is ’N0 faire Det N à N1’ with N0 a human noun phrase and *faire* (make) a light verb.

```
ID=N_fnan_29
lexical-info:[cat="noun",
              Vsup:[cat="verb",list:(value="faire")],
              noun:[noun1="canular"],
              list-det:(det:[value="un",modif="false"],
                        det:[value="un",modif="true"],
                        det:[value="des",modif="false"]
                       )
             ]
args:(const:[pos="0",
             dist:(comp:[hum="true",cat="NP"])
            ],
      const:[pos="1",
             dist:(comp:[hum="true",cat="NP"])
            ]
     )
constructions:(construction="N0 faire Det N à N1",construction="N0 faire Det N")
...
```

6. Evaluation

The construction of the lexicon mentionned above allowed us to clearly identify practical advantages and drawbacks of our tool. Its main advantage is the use of the table of classes. In practice, all missing information is gathered in one single file instead of as many files as classes in the approach of (Gardent *et al.* 2006). In addition, it brings a more global linguistic view: before, the method to generate an NLP lexicon from the lexicon-grammar tables was to find the definitional features of each class and make them explicit. Now, with the use of the table of classes, one wonders if each feature is of interest for each class. Some new linguistic questions within the lexicon-grammar framework may arise.

Moreover, the combination of the tool with the table of classes simplifies the maintenance of the NLP lexicon. First, all reformatting operations for each feature are encoded once in the script independently of the classes. Then, if it appears that a new feature is constant over a whole class, a '+' symbol simply needs to be added in the corresponding cell of the table of classes. The script does not require to be modified to add this information in the generated lexicon, because all reformatting operations corresponding to this feature have already been encoded.

The system requires that each feature has exactly one meaning in all classes. The use of the tool helps maintaining coherence in the table of classes. For instance, the feature 'zone' is a text zone in several classes but with different interpretations :

- in most classes, it provides the lexical value of prepositions introducing verb complements independently of their positions in the canonical construction.
- in class 38L0, it indicates the suffix to be added to the verb in order to obtain its derived noun.
- in class 35R, it gives an example of complement.

We had to add new features to have only one feature for each meaning. In particular, prepositions have been numbered such that it makes it possible to identify directly the complements they introduce.

However, some limitations appeared clearly. First, it was sometimes necessary to repeat tens of similar operations over sets of features. For instance, it was necessary to create manually for all construction feature linguistic objects differing solely in their label. It is due to the fact that the script does not allow for loops, functions with parameters, arrays and dynamic creation of linguistic objects. Moreover, the program is not able to process operations requiring order. For instance, in some compound noun classes, the different components of the noun are encoded in several columns, in linear order. As the program is independent of the order of feature application, it is impossible to generate the compound itself as a whole. It would therefore be interesting to implement macros that would allow such processes.

7. Conclusion and perspectives

In this paper, we have introduced a tool for generating NLP lexicons from lexicon-grammar tables. A table of classes is used to provide information missing in the classes: it makes explicit all implicit information underlying these classes. An extraction script associates each feature with a set of reformatting operations that are activated for each entry when the feature value is true. Applied on the lexicon-grammar tables for French, this tool produces a syntactic lexicon that is suitable for NLP applications such as parsing. The tool has also been experimented to generate a lexicon of nouns. It shows that it can be used for predicates other than verbs. We plan to use it for extracting a lexicon of frozen expressions.

We project to combine it with a Tree Adjoining Grammar (TAG). Indeed, each construction that is attested in this lexicon can be turned into a lexicalized TAG quasi-tree: a given lexical entry can anchor any quasi-trees that defines one of its possible constructions. The definition of these quasi-trees will be generated from a meta-grammar for French thanks to the XMG meta-grammar compiler (Crabbé 2005). The resulting lexicalized TAG will be given as input to the DyALog system (Villemonte de la Clergerie 2004) in order to output a parser for French.

References

BOONS J.-P., GUILLET A. et LECLÈRE C. (1976), *La structure des phrases simples en français: constructions intransitives*, Droz, Genève.

CERVELLE J., FORAX R. et ROUSSEL G. (2006), "Tatoo: an innovative parser generator", in *Proceedings of the conference on Principles and Practices of Programming in Java (PPPJ 2006)*,Mannheim, Germany.

CRABBÉ B. (2005), *Représentation informatique de grammaires fortement lexicalisées: Application à la grammaire d'arbres adjoints*, PhD thesis, University of Nancy.

GARDENT C., GUILLAUME B., PERRIER G. et FALK I. (2006), "Extraction d'information de sous-catégorisation à partir des tables du LADL", in *Actes de la Conférence sur le Traitement Automatique des Langues Naturelles*.

GIRY-SCHNEIDER J. (1987), *Les prédicats nominaux en français. Les phrases simples à verbe support.*, Droz, Genève.

GROSS M. (1975), *Méthodes en syntaxe: régimes des constructions complétives,* Hermann, Paris, France.

GROSS M. (1994), *Constructing Lexicon-Grammars,* in *Computational Approaches to the Lexicon*, Oxford University Press, Oxford.

GUILLET A. et LECLÈRE C. (1992), *La structure des phrases simples en français : constructions transitives locatives,* Droz, Genève.

HATHOUT N. et NAMER F. (1998), "Automatic Construction and Validation of French Large Lexical Resources: Reuse of Verb Theoretical Linguistic Descriptions", in *Proceedings of the Language Resources and Evaluation Conference (LREC'98).*

PAUMIER S. (2003), *De la reconnaissance de formes linguistiques à l'analyse syntaxique*, PhD thesis, Université Paris-Est Marne-la-Vallée.

SAGOT B., CLÉMENT L., VILLEMONTE DE LA CLERGERIE E. et BOULLIER P. (2006), "The Le*fff* 2 syntactic lexicon for French: architecture, acquisition, use", in *Proceedings of the Linguistic Resources and Evaluation Conference (LREC'06).*

SAGOT B. et FORT K. (2007), "Améliorer un lexique syntaxique á l'aide des tables du lexique-grammaire : adverbes en -ment", in *Proceedings of the Lexis and Grammar Conference*,Bonifacio, France.

VAN DEN EYNDE K. et MERTENS P. (2006), *Le dictionnaire de valence* DICOVALENCE *: manuel d'utilisation*.

VILLEMONTE DE LA CLERGERIE E. (2004), "DyALog Primer: Building tabular parsers and programs".