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

Matthieu Constant, Elsa Tolone
Université Paris-Est, IGM
{mconstan,tolone}@univ-paris-est.fr

Lexicon Grammar Conference, L'Aquila
September 10, 2008

Motivations

- Lexicon-Grammar tables (or **classes**) are not directly exploitable for NLP applications
  - pieces of information are kept undefined (e.g. definitional properties)
  - described only in the literature
- Our work :
  - encode implicit properties in a global table
  - implement a generic tool that generates NLP syntactic lexicons thanks to this table

Related Work

- Some works to convert Lexicon-grammar classes into syntactic lexicons (e.g. Hathout and Namer, 98 ; Gardent et al., 06 ; Danlos and Sagot, 07 ; Sagot and Fort, 07)
- In general, for each class, use of a specific configuration defining the implicit properties and the structure of the output
  - e.g. (Gardent et al. 06) uses a configuration graph for each class
  - lexicon-grammar classes are continually updated => approach can be painful for maintenance
- Our approach :
  - implicit information encoded in a table of classes
  - a unique configuration for all classes of a given part-of-speech, where each property (or **feature**) is assigned a set of reformatting operations

Outline

- Lexicon-Grammar Classes
- Table of Classes
- LGExtract, a Generic Tool
- Example of a Generated Lexicon
- Evaluation
- Conclusions and Future Work

Lexicon-Grammar

- A taxonomy of syntactic-semantic classes
  - lexical items (or entries) can be verbs, nouns, adjectives, ...
  - the lexical items of each class share some syntactic features
  - each item has a specific meaning
- Application of a selection of features for each entry
  - encoding in the form of a table (row = entry, column = feature)
  - each feature is tested for each entry
  - binary encoding (+ : accepted feature ; - : forbidden feature)
  - lexical encoding (e.g. required prepositions)

Example of verb class

N0 = :Nhum	N0 = :N-hum	N0 = :Nnr	N0 être V-n	/	Pprv = :se figé	Pprv = :en figé	Pprv = :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	Pprv = :lui	Pprv = :y	N1 être V-n	N0hum V W sur ce point	à 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

FIG.: sample of verb class 33

Example of noun class

<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	-	+	+	-	+	+	-
calembour	-	+	+	-	+	-	+
câlin	-	+	+	-	+	-	-
canular	-	+	+	-	+	+	+
carambouilles	-	-	-	-	+	-	-
cardiogramme	-	+	+	-	+	-	-

FIG.: sample of noun class FNAN

Table of Classes

- Current version of Lexicon-Grammar
  - basic pieces of information are left implicit
  - e.g. constant definitional features of a class are only mentioned informally in literature
- Use of a table of classes
  - for each class, all features are taken into account, not only a selection! (Paumier, 03)
  - encoding in the form of a table
  - 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

Table of Classes (cont.)

- Encoding of a cell
  - **case 1 :** the value depends on the entries of the class ; the cell is then filled with the symbol ’o’, i.e. the information can be found in the class for each entry ;
  - **case 2 :** the value is uniform over the class and can be assigned in the cell with the constant symbols ’+’ or ’-’.
- Practical construction of tables of classes at the Université Paris-Est
  - verbs : Laporte, Tolone, Constant, Leclère, Nakamura, Paumier
  - nouns : Tolone
    (all definitional features have been encoded for all classes but other features are not encoded yet)

Example

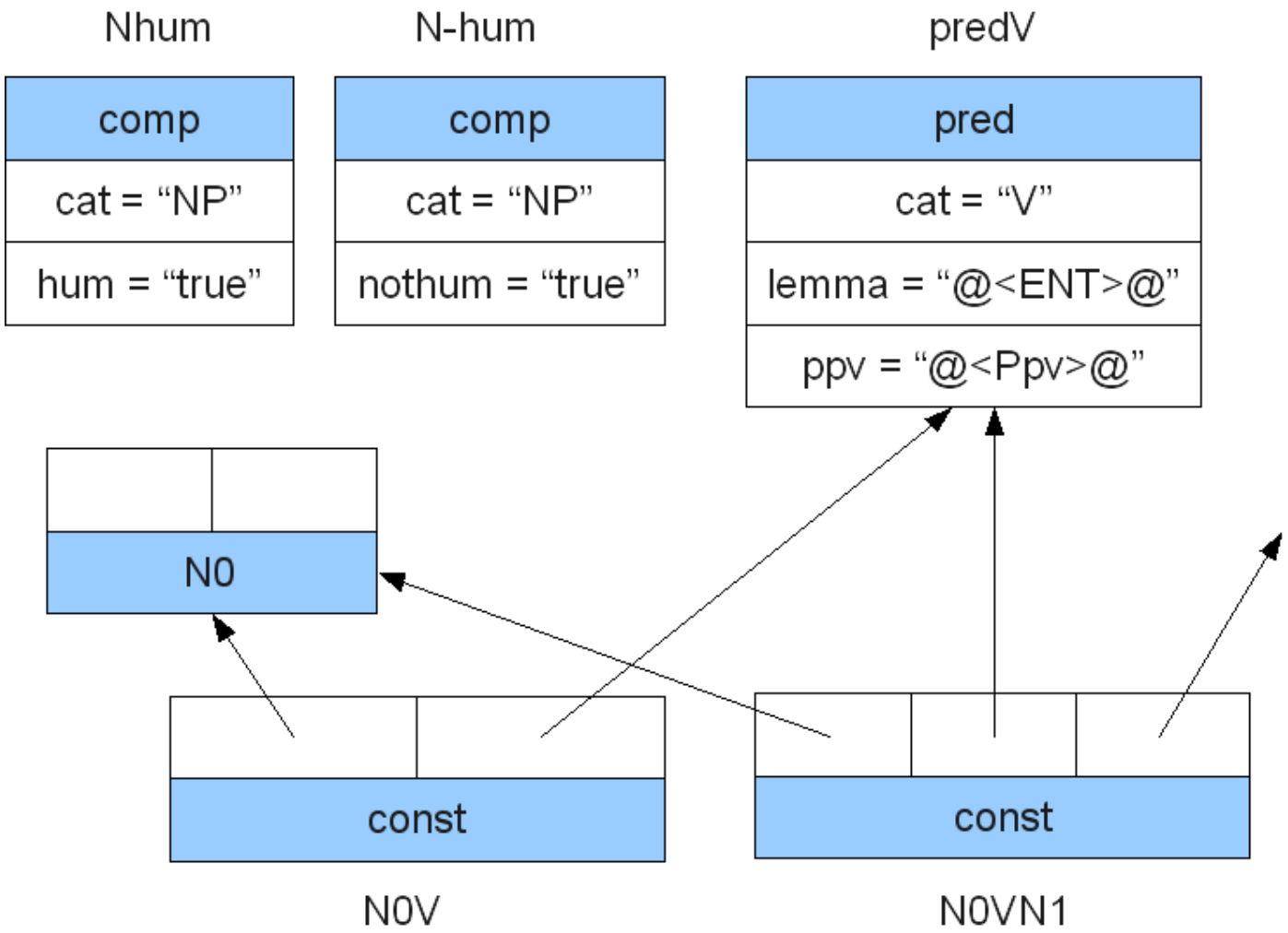
table	N0 =: Nhum	N0 =: N-hum	N0 =: Nnc	N0 =: Nnr	N0 =: V1-inf W	<ENT>	Ppv =: se figé	N0 V	N0 V N1	zone 1	N0 V à N1	N1 =: Nhum	N1 =: N-hum	N1 =: Qu P	N1 =: Qu Psubj	N0 V Prep N1 V0-inf W	N0 V N1 V0-inf W	N0 V V0-inf W	N0 U prep V0-inf	N0 U Prep Nhum	N0 U Prep N-hum	N0 U Nhum	N0 U N-hum
V_1	o	-	o	-	-	o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

FIG.: sample of the table of verb classes

LGExtract

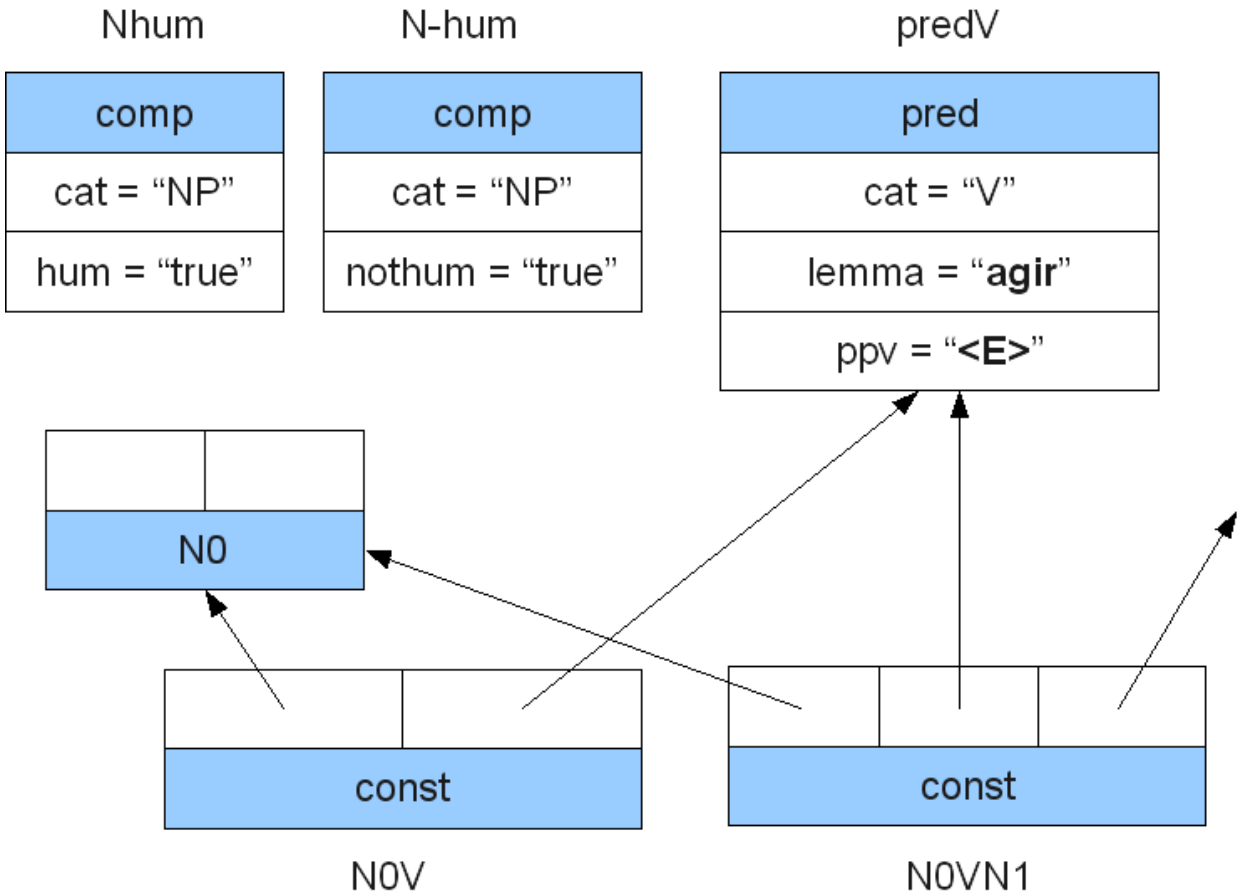
- LGExtract : a generic tool to generate NLP syntactic lexicons from Lexicon-Grammar classes
  - input : a table of classes, lexicon-grammar classes, a configuration script
  - output : a syntactic lexicon in XML or in a raw text format
- Given a set of linguistic objects that can be parameterized by features
- For each lexical item,
  - combination of a selection of linguistic objects according to feature encoding
  - resolution of parameters in the objects
  - generation of the content of the resulting objects in the output

Example


Nhum
comp
cat = “NP”
hum = “true”
N-hum
comp
cat = “NP”
nothum = “true”
predV
pred
cat = “V”
lemma = “@<ENT>@”
ppv = “@<Ppv>@”
N0
const
N0V
const
N0VN1

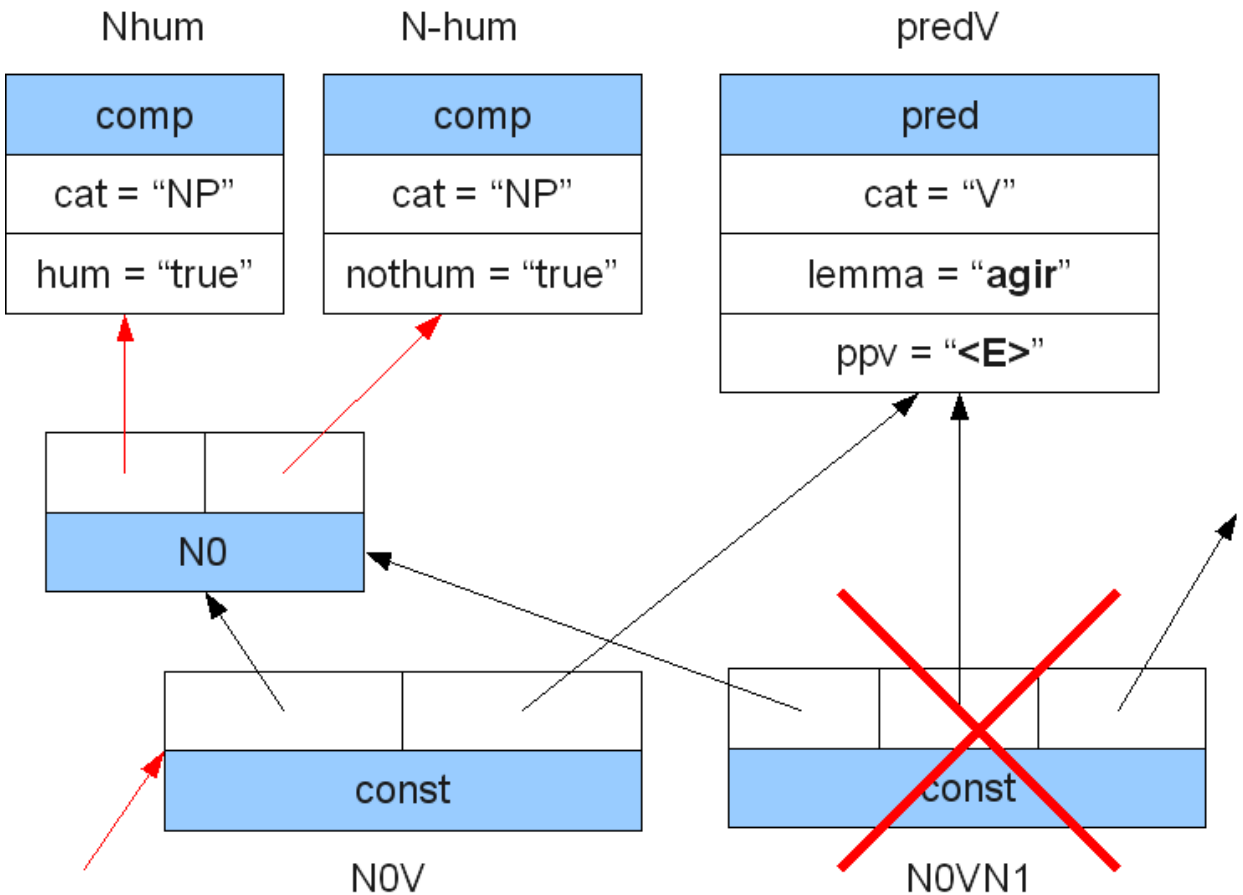
Example - entry *agir*

- @<ENT>@ = agir
- @<Ppv>@ = <E>



Example - entry *agir*

- @N0 = : Nhum@ = + (add Nhum in N0)
- @N0 = : N-hum@ = + (add N-hum in N0)
- @N0 V@ = + (add N0V in constructions)



Configuration script

- Definition of a set of linguistic objects
  - they are in the form of lists and feature structures

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

  - they can be parameterized by features

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

- For each feature, declaration of a set of operations combining objects together

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

Misc.

- Output format :
  - the resulting lexicon can be generated in an XML format
  - elements and attributes in XML can be defined by relating them with the linguistic objects
- Technical characteristics :
  - implemented in Java
  - configuration script parsed with a parser generated from Tatoo (Cervelle et al., 06)

Example of a generated lexicon

- Input :
  - a selection of lexicon-grammar tables : all tables of verbs and nouns that are freely available under the LGPL-LR license
  - two tables of classes : verbs and nouns (incomplete)
  - two configuration scripts (encoded for a selection of features)
- Output :
  - 8,341 verbal entries (from 35 tables) and 4,475 nominal entries (from 30 tables)
  - available under the LGPL-LR license
  - url : `http ://infolingu.univ-mlv.fr`

A lexical entry

```
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")
...
```

Evaluation

- Advantages :
  - a more global linguistic view of classes
  - maintenance simplification (two files !)
- Drawbacks :
  - defining similar linguistic objects can be boring : no dynamic creation of objects !
  - the program cannot deal with operations requiring order : e.g. concatenating components of compound nouns

Concluding remarks

- implementation of a generic tool to produce lexicons for NLP from lexicon-grammar classes
  - use of table of classes to encode definitional features of the classes
  - definition of a unique configuration script for all classes of a given part-of-speech
- Generation of an example lexicon for verbs and nouns
  - available under the LGPL-LR license
  - `http ://infolingu.univ-mlv.fr`

Future Work

- Improve LGExtract : use of macros and integrate dynamic creation of linguistic objects
- Continue encoding of tables of classes
- Generate a lexicon of frozen expressions
- Plug the lexicon in a parser !

THANK YOU !

http ://infolingu.univ-mlv.fr

Resolution of parameters

- Resolution of parameters *@feat@* for the lexical entry *entry* in class *c*
- Two cases according to the value of the cell for class *c* and feature *feat* in table of classes :
  - **case 1 :** if the value is 'o', the result is the value of the cell in class *c* for entry *entry* and feature *feat*
  - **case 2 :** if the value is constant, the result is this value