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

<table>
<thead>
<tr>
<th></th>
<th>&lt;ENT&gt;</th>
<th>&lt;ENT2&gt;</th>
<th>&lt;OPT&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>renaitre</td>
<td>&lt;E&gt;</td>
<td>Max $\square$ au bonheur de vivre</td>
</tr>
<tr>
<td>-</td>
<td>rendre</td>
<td>&lt;E&gt;</td>
<td>Max s'est $\square$ à mon (opinion+avis)</td>
</tr>
<tr>
<td>-</td>
<td>rendre</td>
<td>&lt;E&gt;</td>
<td>Le caporal s'est $\square$ à l'ennemi</td>
</tr>
<tr>
<td>-</td>
<td>renoncer</td>
<td>&lt;E&gt;</td>
<td>Max $\square$ à son héritage</td>
</tr>
</tbody>
</table>

**Fig.**: sample of verb class 33
Example of noun class

<table>
<thead>
<tr>
<th></th>
<th>autre Det</th>
<th>Det =: un</th>
<th>Det =: un-Modif</th>
<th>Det =: du</th>
<th>Det =: des</th>
<th>N0 faire le N de V0-inf W</th>
<th>NOnhum faire Det N à N1hum sur ce point</th>
</tr>
</thead>
<tbody>
<tr>
<td>cadeau</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td></td>
<td></td>
</tr>
<tr>
<td>calembour</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td></td>
<td></td>
</tr>
<tr>
<td>câlin</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>canular</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>carambouilles</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>cardiogramme</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

**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
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>
<thead>
<tr>
<th>Table</th>
<th>NO : N-hum</th>
<th>NO : N-hum</th>
<th>NO : Nuc</th>
<th>NO : Nuc</th>
<th>NO : V1-inf W</th>
<th>Ppv : se figé</th>
<th>NO V</th>
<th>NO V N1</th>
<th>zone 1</th>
<th>NO V a N1</th>
<th>N1 : Nhum</th>
<th>N1 : N-hum</th>
<th>N1 : Qu.P</th>
<th>N1 : Qu.Pobj</th>
<th>NO V Prep N1 V0-inf W</th>
<th>NO V V0-inf W</th>
<th>NO U prep N1 V0-inf W</th>
<th>NO U prep N Uhm</th>
<th>NO U Prep N-hum</th>
<th>NO U N-hum</th>
<th>NO U N-hum</th>
</tr>
</thead>
<tbody>
<tr>
<td>V 1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>V 2</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>V 4</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>V 31R</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>V 31H</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>V 33</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>+</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>V 32H</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**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 ;
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
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
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