Learning to Assign Grammatical Function Labels

Grzegorz Chrupała    Josef van Genabith

National Center for Language Technology
School of Computing
Dublin City University

NCLT Seminar Series, 15 Feb 2006
Outline

1 Motivation

2 Learning Function Labels

3 Evaluation
Outline

1. Motivation
2. Learning Function Labels
3. Evaluation
GramLab project

- Treebank-based Multilingual Lexical Functional Grammar induction for English, Chinese, Japanese, Arabic, German, French and Spanish
- Leverage successful data-driven approaches to parsing
- Core idea: automatic annotation of constituent trees (c-structures) to obtain LFG f-structures
What are f-structures

- F-structures encode grammatical relations such as Subject, Object, Sentential Complement etc.
- Less interlanguage variability than c-structure
- Explicit representation of
  - control and raising constructions
  - pro-drop
  - long distance dependencies
- Syntactic interface to predicate structure and semantics
C-structure to f-structure mapping

(↑SUBJ) = NH

NP

el gato
the cat

↑ = ↓

VG

quiere
wants

(↑XCOMP) = NH

(↑SUBJ) = (↓SUBJ)

S

↑ = ↓

VINF

comer
eat

↑ OBJ) = ↓

NP

pescado
fish
C-structure to f-structure mapping

\[
S 
\quad \downarrow \uparrow \text{SUBJ} = \downarrow \text{SUBJ} \\
\quad \downarrow \uparrow \text{XCOMP} = \downarrow \text{XCOMP} \\
\uparrow \downarrow \text{NP} \quad \uparrow \downarrow \text{VG} \\
\quad \text{el gato} \quad \text{quieres} \\
\quad \text{the cat} \quad \text{wants} \\
\quad \downarrow \uparrow \text{VINF} \quad \downarrow \uparrow \text{NP} \\
\quad \text{comer} \quad \text{pescado} \\
\quad \text{eat} \quad \text{fish} \]
C-structure to f-structure mapping

$\text{(↑SUBJ)} = \downarrow$

$\downarrow$

NP

el gato

the cat

(↓SUBJ) = (↑XCOMP) = (↑OBJ) = (↓SUBJ)

↑=↓

VG

quiere

wants

S

↑=↓

VINF

comer

eat

NP

el gato

the cat

(↓SUBJ) = (↑SUBJ)

↑=↓

S

↑=↓

NP

pescado

fish

SUBJ

[ ]

PRED 'gato'

PRED 'querer⟨SUBJ,XCOMP⟩'

XCOMP

SUBJ

[ ]

PRED 'comer⟨SUBJ,OBJ⟩'

OBJ

[ PRED 'pescado' ]
Automatic functional annotation

Initial work done on English Penn II Treebank

- Extract most common CFG rules from treebank
- Write annotation rules based on node context:
  - Node category
  - Node’s mother’s category
  - Position relative to head: head / left / right
- Apply to treebank trees or to parser output
- Collect and solve functional equations
Issues with Spanish

Difficulties in applying this approach to Spanish and Cast3LB

- flexible order of sentence constituents
- information encoded in terms of tree configurations in Penn, but in function tags in Cast3LB

```
S
  └─ neg
      ├─ NEG
      │  └─ no
      ├─ gv
      │    └─ espere
      │         └─ let-expect
      └─ sn-SUJ
          └─ el lector
              └─ the reader
      └─ sn-CD
          └─ una definición previa
              └─ a previous definition
```
Issues with Spanish

Difficulties in applying this approach to Spanish and Cast3LB

- flexible order of sentence constituents
- information encoded in terms of tree configurations in Penn, but in function tags in Cast3LB

```
S
  |
  v
neg-NEG
  | no
  v
no
  | gv
  v
espere
  | sn-SUJ
  v el lector
  v the reader
  v una definición previa
  v a previous definition
```
Issues with Spanish

Difficulties in applying this approach to Spanish and Cast3LB

- flexible order of sentence constituents
- information encoded in terms of tree configurations in Penn, but in function tags in Cast3LB

\[ S \]

\[ \begin{align*}
\text{neg-NEG} & \quad \text{gv} & \quad \text{sn-SUJ} & \quad \text{sn-CD} \\
\text{no} \quad \text{espere} & \quad \text{el lector} & \quad \text{una definición previa} \\
\text{no} & \quad \text{let-expect} & \quad \text{the reader} & \quad \text{a previous definition}
\end{align*} \]
Arguments vs adjuncts

A general problem: argument-adjunct distinction

1. can be difficult to make
2. sometimes is arbitrary
Arguments vs adjuncts

A general problem: argument-adjunct distinction

1. can be difficult to make
2. sometimes is arbitrary
Previous work on automatic functional annotation for Cast3LB

- O’Donovan et al. 2005 decided to rely on Cast3LB functional tags.
- Bikel’s parser trained to output function tags by treating compound labels as atomic
- Reasonable results, but maybe we can do better.
Outline

1. Motivation

2. Learning Function Labels

3. Evaluation
Adding function tags to parse trees

- Can better results be obtained if function tags are added in a postprocess? Advantages:
  - Smaller category label set resulting in reduction of data sparseness
  - Control over learning algorithm and feature set used

- Previous work, Penn Treebank, Charniak’s and Collins’s parsers
  - Blaheta and Charniak, Assigning function tags to parsed text, *NAACL 2000*
  - Jijkoun and de Rijke, Enriching the output of a parser using memory-based learning, *ACL 2004*
Machine-learning approach

- Adding function tags as a classification task
  - Classes: corresponding to the 24 Cast3LB tags plus the null class for untagged nodes
  - Candidate nodes: sisters to nodes labelled `gv` (Verb Group), `infinitiu` (Infinitive) and `gerundi` (Gerund)
- Experiments with two algorithms: memory-based (Timbl) and Support Vector Machines (R interface to LIBSVM)
Node Features

Nine features: configurational, morphological and lexical information about candidate node

- Position relative to head
- Node head lemma
- Alternative head lemma
- Node head POS
- Node label
- Is definite NP
- Is NP and agrees with head verb
- No. of terminal descendents
- Head lemma +human
Local and Context Features

Four **local** features (head verb and parent information)
- Head verb lemma,
- Head verb person,
- Head verb number,
- Parent node label

$8 \times 4$ **context** features ($=$ node features of two previous and two following nodes in linear order)

For SVM, feature values of frequency $< 20$ were replaced with a dummy value
Outline

1. Motivation

2. Learning Function Labels

3. Evaluation
Evaluation metrics

- Accuracy: proportion of candidate nodes that were assigned correct label
- Precision, Recall and F-score defined on sets of tuples: \( \langle GF, i, j \rangle \), where
  - \( GF \) – grammatical function
  - \( \langle w_i, \ldots, w_j \rangle \) – subsequence of tokens spanned by function-tagged node

Example:

\[ \text{[NEG No]} \text{ espere [Suj el lector]} \text{ [CD una definición previa]} \]

\( \{ \langle \text{NEG}, 1, 1 \rangle, \langle \text{Suj}, 3, 4 \rangle, \langle \text{CD}, 5, 7 \rangle \} \)
Evaluation metrics

- **Accuracy**: proportion of **candidate** nodes that were assigned correct label.

- **Precision, Recall and F-score** defined on sets of tuples: \( \langle GF, i, j \rangle \), where:
  - \( GF \) – grammatical function
  - \( \langle w_i, \ldots, w_j \rangle \) – subsequence of tokens spanned by **function-tagged node**.

Example:

\[
\text{[NEG No ] espere [S U J el lector ] [C D una definición previa ]}
\]

\( \text{no let-expect the reader a previous definition} \)

\( \{\langle NEG, 1, 1 \rangle, \langle SUJ, 3, 4 \rangle, \langle CD, 5, 7 \rangle\} \)
## Results for gold-standard trees

<table>
<thead>
<tr>
<th></th>
<th>Accuracy</th>
<th>Precision</th>
<th>Recall</th>
<th>F-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timbl</td>
<td>87.55</td>
<td>88.06</td>
<td>87.66</td>
<td>87.86</td>
</tr>
<tr>
<td>SVM</td>
<td>87.64</td>
<td>88.15</td>
<td>87.75</td>
<td>87.95</td>
</tr>
</tbody>
</table>
Results for parser output

Parser performance

<table>
<thead>
<tr>
<th></th>
<th>Labelled Bracket P</th>
<th>Labelled Bracket R</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>All</strong></td>
<td>79.52</td>
<td>79.10</td>
<td>79.30</td>
</tr>
<tr>
<td><strong>&lt; 40</strong></td>
<td>82.09</td>
<td>81.49</td>
<td>81.78</td>
</tr>
</tbody>
</table>

Labelling

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
<th>F-score</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Baseline</strong></td>
<td>51.97</td>
<td>51.68</td>
<td>51.82</td>
</tr>
<tr>
<td><strong>Timbl</strong></td>
<td>56.54</td>
<td>56.57</td>
<td>56.55</td>
</tr>
<tr>
<td><strong>SVM</strong></td>
<td>56.91</td>
<td>56.78</td>
<td><strong>56.84</strong></td>
</tr>
</tbody>
</table>

Baseline – parser treating compound category-function labels as atomic.
### Parser performance

<table>
<thead>
<tr>
<th></th>
<th>Labelled Bracket P</th>
<th>Labelled Bracket R</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>79.52</td>
<td>79.10</td>
<td>79.30</td>
</tr>
<tr>
<td>&lt; 40</td>
<td>82.09</td>
<td>81.49</td>
<td>81.78</td>
</tr>
</tbody>
</table>

### Labelling

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
<th>F-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>51.97</td>
<td>51.68</td>
<td>51.82</td>
</tr>
<tr>
<td>Timbl</td>
<td>56.54</td>
<td>56.57</td>
<td>56.55</td>
</tr>
<tr>
<td>SVM</td>
<td>56.91</td>
<td>56.78</td>
<td><strong>56.84</strong></td>
</tr>
</tbody>
</table>

Baseline – parser treating compound category-function labels as atomic.
Parse quality influence

- Features used refer not only to candidate node but to mother and sister nodes.
- Small constituency misconfiguration may influence classification decision of several nodes.
- But, small improvements in parse quality may give greater improvements in function tag assignment :-)
- In a recent experiment:
  - Evalb f-score went up to $80.68$ (+1.38).
  - Labelling f-score went up to $59.6$ (+2.76).
F-Structure evaluation

Cast3LB tags to LFG functions is not a one-to-one mapping, eg.:
- CD → \{OBJ, COMP, XCOMP\}
- \{CC, MOD, ET\} → ADJUNCT

F-structure evaluation less sensitive to slightly incorrect parse trees.

Results for parser output with SVM tags – 100 gold standard
f-structure-derived dependencies

<table>
<thead>
<tr>
<th>Precision</th>
<th>Recall</th>
<th>F-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>76.6</td>
<td>75.58</td>
<td>76.09</td>
</tr>
</tbody>
</table>
## Error analysis

### Simplified confusion matrix for SVM predictions on gold trees

<table>
<thead>
<tr>
<th></th>
<th>SUJ</th>
<th>CD</th>
<th>CI</th>
<th>CC</th>
<th>CREG</th>
<th>ATR</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUJ</td>
<td>464</td>
<td>29</td>
<td>1</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CD</td>
<td>29</td>
<td>410</td>
<td>5</td>
<td>22</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>CI</td>
<td>0</td>
<td>1</td>
<td>44</td>
<td>8</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CC</td>
<td>5</td>
<td>15</td>
<td>12</td>
<td>539</td>
<td>29</td>
<td>5</td>
</tr>
<tr>
<td>CREG</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>39</td>
<td>0</td>
</tr>
<tr>
<td>ATR</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>137</td>
</tr>
</tbody>
</table>
Subject vs Direct Object

- Over 55% of mistakenly tagging CD as SUJ are daughters of relative clause nodes (S.R), against 16% of all candidate nodes.
- Subject - Direct Object ambiguity in relative clauses
Subject vs Direct Object

- Over 55% of mistakenly tagging CD as SUJ are daughters of relative clause nodes (S.R), against 16% of all candidate nodes.
- Subject - Direct Object ambiguity in relative clauses

```
<table>
<thead>
<tr>
<th>grup.nom</th>
</tr>
</thead>
<tbody>
<tr>
<td>nc</td>
</tr>
<tr>
<td>sistemas</td>
</tr>
<tr>
<td>systems</td>
</tr>
</tbody>
</table>

S.R

- relatiu
  | que
  | which

- gv
  | usan
  | use

sn

el 95% de los ordenadores
95% of computers
In addition to cases of ambiguity

- Training set too small to learn many lexical dependencies
- Inherently fuzzy distinction
- Subtle semantic distinctions and/or a degree of arbitrariness

Compare

- Llegar [\textit{CREG} a un acuerdo ]
  
  \textit{arrive at an agreement}
- Llegar [\textit{cc} a Estados Unidos]
  
  \textit{arrive in the US}
Further work

- Unbounded dependencies between argument and verb
- Inter-annotator agreement for function would give us an idea of where the upper bound is.
- What to do about arbitrary distinctions and inconsistencies in human annotation?