ACL Tutorial: Wide-coverage NLP with Expressive Grammars

Josef van Genabith (Dublin City University)
Julia Hockenmaier (University of Illinois)
Yusuke Miyao (National Institute of Informatics)
Why this tutorial?

• A lot of progress in robust, wide-coverage NLP with expressive grammars in the last 10 years
  – Expressive grammars and machine learning go together
  – Expressive grammars are needed for semantics
  – Expressive grammars are being used in real applications

But:

• Research is limited to small number of groups
• Each group works within their own formalism
  – What are the commonalities and differences?
  – How can others get started?
Overview

• Part 1: Introduction to expressive grammars
  – Why expressive grammars?
  – Tree-Adjoining Grammar
  – Combinatory Categorial Grammar
  – Lexical-Functional Grammar
  – Head-Driven Phrase Structure Grammar

• Part 2: NLP with expressive grammars
  – Grammar extraction: obtaining the grammar
  – Wide-coverage parsing: using the grammar
  – Other applications: using the grammar
I. Why Expressive Grammars?
Why grammar?

Surface string: Mary saw John

Grammar:
- Parsing
- Generation

Meaning representation:
- Logical form: saw(Mary, John)
- Pred-arg structure: (PRED saw (AGENT Mary (PATIENT John))
- Dependency graph: saw Mary John
Grammar formalisms

- Formalisms provide a **language** in which linguistic theories can be expressed and implemented.

- Formalisms define **elementary objects** (trees, strings, feature structures) and **recursive operations** which generate complex objects from simple objects.

- Formalisms may impose **constraints** (e.g. on the kinds of dependencies they can capture).
How do grammar formalisms differ?

Formalisms define different representations

- **Tree-adjoining Grammar (TAG):**
  Fragments of phrase-structure trees

- **Lexical-functional Grammar (LFG):**
  Annotated phrase-structure trees (c-structure)
  linked to feature structures (f-structure)

- **Combinatory Categorial Grammar (CCG):**
  Syntactic categories paired with meaning representations

- **Head-Driven Phrase Structure Grammar (HPSG):**
  Complex feature structures (Attribute-value matrices)
How do grammar formalisms differ?

Weak generative capacity:
– What languages (sets of strings) can be defined?
  • $a^n b^m$ is regular, $a^n b^n$ is context-free
– Expressive grammars can represent more languages

Strong generative capacity:
– What structures can be defined?
– Expressive grammars can represent more structures
Different types of dependencies

**Head-Argument:** e.g. verb-subject
- Arguments are subcategorized for
- Arguments have to be realized, but only once

**Head-Adjunct:** e.g. noun-adj., verb-adverb
- Adjuncts are not subcategorized for
- There can be an arbitrary number of adjuncts

**Coordination:**
- Conjuncts may be standard constituents
  *John and Mary; live or die*
- Conjuncts may be nonstandard constituents
  *((John will) and (Mary may want)) to go*
Context-free grammars

• CFGs capture only **nested** dependencies
  – The dependency graph is a **tree**
  – The dependencies **do not cross**
Beyond CFGs: Nonprojective dependencies

Dependencies: tree with crossing branches

Arise in the following constructions

- (Non-local) **scrambling** (free word order languages)
  
  *Die Pizza hat Klaus versprochen zu bringen*

- **Extraposition** *(The guy is coming who is wearing a hat)*

- **Topicalization** *(Cheeseburgers, I thought he likes)*
Beyond CFGs: Nonlocal dependencies

- Dependencies form a **DAG**
  (a node may have **multiple incoming edges**)

- Arise in the following constructions:
  - **Control** (*He has promised me to go*), **raising** (*He seems to go*)
  - **Wh-movement** (*the man who you saw yesterday is here again*),
  - **Non-constituent** coordination
    (right-node raising, gapping, argument-cluster coordination)
Unbounded non-local dependencies

Extraction:
- Wh-movement:
  the articles which (you believed he saw that…) I filed
- Tough-movement:
  the articles are easy to file
- Parasitic gaps:
  the articles that I filed without reading

Non-standard coordination:
- Right-node raising:
  [[Mary ordered] and [John ate]] the tapas.
- Argument cluster coordination:
  Mary ordered [[tapas for herself] and [wine for John]].
- Sentential gapping:
  [[Mary ordered tapas] and [John beer]].
Commonalities and differences: Lexicalization

**No lexicalization:** (CFG)
- The lexicon contains little syntactic information (e.g. just POS-tags)
- Recursion is entirely defined by *language-specific* grammar rules

**Weak lexicalization:** (LFG)
- The lexicon (and lexical rules) specify *some* language-specific information (e.g. subcategorization, semantics, control, binding theory, passivization)
- Recursion is defined by language-specific grammar rules (but lexical information may constrain which rules can be used in which context)

**Strong lexicalization:** (TAG, CCG, HPSG)
- The lexicon (and lexical rules) specifies *all* language-specific information (e.g. word order, subcategorization, semantics, control, binding theory)
- The lexicon pairs words with complex elementary objects
  These objects may have an *extended domain of locality* (i.e. capture structure beyond a single CFG rule)
- Recursion is defined by completely universal operations
II. TREE-ADJOINING GRAMMAR
Tree-Adjoining Grammar

TAG is a tree-rewriting formalism:
- TAG’s elementary objects are trees (not strings)
- TAG’s operations (substitution, adjunction) work on trees.
- TAG requires a linguistic theory which specifies the shape of these elementary trees.

TAG is mildly context-sensitive:
- can capture Dutch crossing dependencies
- but is still efficiently parseable
TAG: the machinery

Elementary trees:
- Initial trees: combine via substitution
- Auxiliary trees: combine via adjunction

Derived trees:
- The output of substitution and adjunction

Derivation trees:
- A record of the derivation process
A small TAG lexicon

\[ \alpha_1: \]
\[ S \rightarrow NP \quad VP \]
\[ NP \rightarrow VBZ \quad NP \]
\[ VBZ \rightarrow eats \]

\[ \alpha_2: \]
\[ NP \rightarrow John \]

\[ \alpha_3: \]
\[ NP \rightarrow tapas \]

\[ \beta_1: \]
\[ VP \rightarrow RB \quad VP^* \]
\[ RB \rightarrow always \]
A TAG derivation: arguments

\[ \alpha_1: S \rightarrow \text{NP} \rightarrow V \rightarrow \text{NP} \rightarrow v_{\text{BZ}} \rightarrow \text{NP} \rightarrow \text{eats} \]

Substitution:
\[ \alpha_2: \text{NP} \rightarrow \text{John} \]
\[ \beta_1: V \rightarrow \text{NP} \rightarrow \text{RB} \rightarrow \text{NP} \rightarrow v_{\text{P}} \rightarrow \text{NP} \rightarrow \text{always} \]
\[ \alpha_3: \text{NP} \rightarrow \text{NP} \rightarrow \text{NP} \rightarrow \text{tapas} \]

Derived tree:
\[ \alpha_1: \text{S} \rightarrow \text{NP} \rightarrow \text{VP} \rightarrow \text{NP} \rightarrow v_{\text{BZ}} \rightarrow \text{NP} \rightarrow \text{eats} \]

Substitution!
A TAG derivation: arguments

Derived tree

\(\alpha_1\):
\[\text{S} \rightarrow \text{NP} \rightarrow \text{VP} \rightarrow \text{VBZ} \rightarrow \text{NP} \]
\(\alpha_2\):
\(\text{John} \rightarrow \text{eats} \rightarrow \text{tapas}\)
\(\alpha_3\):

\(\beta_1\):
\[\text{VP} \rightarrow \text{RB} \rightarrow \text{always} \]

\(\beta_1^*\):
\[\text{VP} \rightarrow \text{VP}^* \]
A TAG derivation: adjuncts

Derived tree

\[ \alpha_1 \]
\[ \alpha_2 \]
\[ \alpha_3 \]

\[ \alpha_1: S \]
\[ NP \]
\[ John \]
\[ VBZ \]
\[ eats \]
\[ NP \]
\[ tapas \]

\[ \beta_1: VP \]
\[ RB \]
\[ always \]

\[ VP \]
\[ VP^* \]
A TAG derivation: adjuncts

\[ \alpha_1 \]

Derived tree

\[ \alpha_2 \beta_1 \alpha_3 \]

\[ S \]

\[ \text{NP} \]

\[ \text{RB} \]

\[ \text{always} \]

\[ \text{VP} \]

\[ \text{NP} \]

\[ \text{eats} \]

\[ \text{VP}^* \]

\[ \text{tapas} \]
Nonlocal dependencies in TAG

Use different elementary trees

Use obligatory adjunction
TSG, TIG, and TAG

• **Tree Substitution Grammar:**
  – only substitution

• **Tree Insertion Grammar:**
  – only substitution and sister adjunction

• **Tree Adjoining Grammar:**
  – substitution, sister adjunction and wrapping adjunction
Extensions and variants of TAG

Multicomponent TAG
- Elementary trees can be *sets* of trees
- More expressive than standard TAG

Spinal TAG
- Elementary trees have only a spine
- Leaves subcategorization and argument/adjunct distinction underspecified
III. COMBINATORY CATEGORIAL GRAMMAR
Properties of CCG

• CCG rules are **type-driven**, not structure-driven
  – Types = functions
  – Intransitive verbs and VPs are indistinguishable

• CCG’s **syntax-semantics interface** is transparent
  – Lexicon pairs syntactic categories with interpretations
  – Every syntactic rule has a semantic counterpart
  – CCG rules are monotonic (no movement/traces)

• CCG has a **flexible constituent structure**
  – Simple, unified treatment of extraction and coordination

• CCG is **mildly context-sensitive**
CCG: the machinery

**Syntactic categories:**
specify subcategorization; define word order

**Semantic interpretations:**
specify logical forms (pred.-arg. structure)

**Combinatory rules:**
specify how constituents can combine.

**Derivations:**
spell out process of combining constituents.
CCG categories

Simple (atomic) categories: NP, S, PP

Complex categories (functions): Return a result when combined with an argument

<table>
<thead>
<tr>
<th>Category</th>
<th>Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>VP, intransitive verb</td>
<td>S\NP</td>
</tr>
<tr>
<td>Transitive verb</td>
<td>(S\NP)/NP</td>
</tr>
<tr>
<td>Adverb</td>
<td>(S\NP)(S\NP)</td>
</tr>
<tr>
<td>Prepositions</td>
<td>((S\NP)(S\NP))/NP</td>
</tr>
<tr>
<td></td>
<td>(NP\NP)/NP</td>
</tr>
<tr>
<td></td>
<td>PP/NP</td>
</tr>
</tbody>
</table>
Function application

Forward application (>):

(S\NP)/NP  NP  ⇒>  S\NP
eats  tapas  eats tapas

Backward application (<):

NP  S\NP  ⇒<  S
John  eats tapas  John eats tapas

Used in all variants of categorial grammar
A (C)CG derivation

```
NP    (S\NP)/NP
       NP
        S\NP
         S
```

John eats tapas
CCG: semantics

- Every syntactic category and rule has a semantic interpretation
- Semantic interpretations are functions of the \textbf{same arity} as the syntactic category
- Semantics often written as \textit{\(\lambda\)-expressions}

\[
\begin{array}{ccc}
\text{John} & \text{eats} & \text{tapas} \\
\text{NP : } John' & (S\backslash NP)/NP : \lambda x.\lambda y.eats'x y & \text{NP : tapas'} \\
\hline
\text{S\backslash NP} : \lambda y.eats'tapas'y & \Rightarrow & \text{S : eats'tapas'John'}
\end{array}
\]
Function composition

Harmonic forward composition ($\succ_B$):

\[
\begin{align*}
X / Y & \quad Y / Z \quad \Rightarrow_{\succ_B} \quad X / Z \\
\lambda x. f(x) & \quad \lambda y. g(y) \quad \Rightarrow_{\succ_B} \quad \lambda z. f(g(z))
\end{align*}
\]

Harmonic backward composition ($\prec_B$):

\[
\begin{align*}
Y \setminus Z & \quad X \setminus Y \quad \Rightarrow_{\prec_B} \quad X \setminus Z \\
\lambda y. g(y) & \quad \lambda x. f(x) \quad \Rightarrow_{\prec_B} \quad \lambda z. f(g(z))
\end{align*}
\]

Forward crossing composition ($\succ_B^x$):

\[
\begin{align*}
X / Y & \quad Y \setminus Z \quad \Rightarrow_{\succ_B^x} \quad X \setminus Z \\
\lambda x. f(x) & \quad \lambda y. g(y) \quad \Rightarrow_{\succ_B^x} \quad \lambda z. f(g(z))
\end{align*}
\]

Backward crossing composition ($\prec_B^x$):

\[
\begin{align*}
Y / Z & \quad X \setminus Y \quad \Rightarrow_{\prec_B^x} \quad X / Z \\
\lambda y. g(y) & \quad \lambda x. f(x) \quad \Rightarrow_{\prec_B^x} \quad \lambda z. f(g(z))
\end{align*}
\]
# Type-raising

## Forward typeraising ($\Rightarrow_{>T}$):

<table>
<thead>
<tr>
<th>$X$</th>
<th>$\Rightarrow_{&gt;T}$</th>
<th>$T / (T \setminus X)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td></td>
<td>$\lambda f. f(a)$</td>
</tr>
</tbody>
</table>

## Backward typeraising ($\Rightarrow_{<T}$):

<table>
<thead>
<tr>
<th>$X$</th>
<th>$\Rightarrow_{&lt;T}$</th>
<th>$T \setminus (T / X)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td></td>
<td>$\lambda f. f(a)$</td>
</tr>
</tbody>
</table>
The CCG lexicon

Pairs words with their syntactic categories (and semantic interpretation):

\[
\begin{align*}
\textit{eats} & \quad (\text{S}\text{\textbackslash NP})/\text{NP} \\
& \quad \lambda x \lambda y. \textit{eats}'xy \\
\text{S}\text{\textbackslash NP} & \quad \lambda x. \textit{eats}'x
\end{align*}
\]

The main bottleneck for wide-coverage CCG parsing
The CCG lexicon: bounded dependencies

Bounded dependencies are captured in the lexicon through **coindexation** in the **syntactic** category and **copied variables** in the **semantic** interpretation.

**Auxiliaries**

*may*: $\frac{(S\backslash NP_i)/(S\backslash NP_i)}{\lambda P \lambda x. \text{may'}(x, P(x))}$

**Subject control**

*promise*: $\frac{((S\backslash NP_i)/(S\backslash NP_i))/NP}{\lambda y \lambda P \lambda x. \text{promise'}(x, y, P(x))}$

**Object control**

*persuade*: $\frac{((S\backslash NP)/(S\backslash NP_i))/NP_i}{\lambda y \lambda P \lambda x. \text{persuade'}(x, y, P(y))}$
Another CCG derivation

\[
\begin{array}{c}
\text{John} \\
\underline{NP} \\
\text{eats} \\
(S\backslash NP)/NP \\
\text{tapas} \\
\underline{NP}
\end{array}
\]

\[
S/(S\backslash NP) \\
\xrightarrow{T} \\
(S\backslash NP)/NP \\
\Rightarrow B \\
S/\text{NP} \\
\Rightarrow S
\]

- Function composition and type-raising create “spurious ambiguity”.
- Normal form derivations use composition and type-raising when only necessary
Non-local dependencies: Type-raising and composition
IV. Lexical Functional Grammar
Lexical-Functional Grammar

• LFG is constraint-based (Bresnan & Kaplan ‘81, Bresnan ‘01, Dalrymple ‘01)

• Two (basic) levels of representation:
  – **C-structure:**
    • Represents *surface grammatical configurations*: constituency, word order
    • Represented as annotated CFG trees
  – **F-structure:**
    • Represents *abstract syntactic functions, morphological + semantic information*
      – SUBJ(ject), OBJ(ect), OBL(ique), PRED(icate), COMP(lement), ADJ(unct)…
      – TENSE, ASPECT, NUM(ber), PERS(on), …
    • F-structure ≈ *basic predicate-argument structure*, dependency representation, logical form, … (van Genabith and Crouch, ’96;’97)
    • Represented as attribute-value matrices (AVMs; DAGs)
Lexical-Functional Grammar LFG
Lexical-Functional Grammar LFG

S
  /   |
NP    VP
  / |
NNP  V NP
   / |
Annan signed the NN

[SUBJ
  / |
  NUM sg
  PERS 3

[PRED Annan]

[PRED sign(↑SUBJ,↑OBJ)]

[TENSE past]

[OBJ
  / |
  NUM sg
  PERS 3

[SPEC
  / |
  DET [PRED the] ]]
Lexical-Functional Grammar LFG

```
S -> NP VP
VP -> V NP
NP -> DT NNP
NNP -> Annan
NN -> deal
V -> signed
DT -> the
```

```
NP
  NNP Annan
    V signed
      DT the
      NN deal
  VP
    NP
      NNP
        V
          NP
```

LFG Grammar Rules and Lexical Entries

S → NP
   ↑SUBJ=↓
   ↑=↓

VP → V
   ↑=↓
   ↑OBJ=↓

NP → DT
   ↑SPEC DET=↓
   ↑=↓

NP → NNP
   ↑=↓

NNP → Annan
   ↑PRED=Annan
   ↑NUM=sg
   ↑PERS=3

NN → deal
   ↑PRED=deal
   ↑NUM=sg
   ↑PERS=3

V → signed
   ↑PRED=sign(↑SUBJ,↑OBJ)
   ↑TENSE=past

DT → the
LFG Parse Tree (with Equations/Constraints)
LFG Constraint Resolution (1/3)
LFG Constraint Resolution (2/3)
LFG Constraint Resolution (3/3)

\[\begin{align*}
F-\text{Str} \models \land \left\{ \right. & \begin{array}{l}
2: \text{PRE}D= \text{Annan},
2: \text{NU}M=s\text{g},
2: \text{PE}RS=3\text{rd},
1=2,
0: \text{SUB}J=1,
4: \text{PRE}D= \text{sign} \langle 4: \text{SUB}J, 4: \text{OBJ} \rangle,
4: \text{TENSE}= \text{past},
3=4,
6: \text{PRE}D= \text{the},
5: \text{SPEC} \ \text{DE}T=6,
7: \text{PRE}D= \text{deal},
7: \text{NU}M=s\text{g},
7: \text{PE}RS=3\text{rd},
5=7,
3: \text{OBJ}=5,
0=3
\end{array}
\left. \right\}
\end{align*}\]

\[
\{ 0=3, 5=7, 1=2, 3=4 \} \Rightarrow \{ 3, 4 \} \rightarrow 0, 7 \rightarrow 5 \text{ and } 2 \rightarrow 1
\]

\[
F-\text{Str} \models \land \left\{ \right. & \begin{array}{l}
0: \text{SUB}J=1,
1: \text{PRE}D= \text{Annan},
1: \text{NU}M=s\text{g},
1: \text{PE}RS=3\text{rd},
0: \text{PRE}D= \text{sign} \langle 0: \text{SUB}J, 0: \text{OBJ} \rangle,
0: \text{TENSE}= \text{past}
0: \text{OBJ}=5,
5: \text{PRE}D= \text{deal},
5: \text{NU}M=s\text{g},
5: \text{PE}RS=3\text{rd},
5: \text{SPEC} \ \text{DE}T=6,
6: \text{PRE}D= \text{the}
\end{array}
\left. \right\}
\]

\[\begin{array}{c}
\land, \neg, \lor
\end{array}\]

\[\downarrow \in \uparrow \text{GF, /}\]
LFG Subcategorisation & Long Distance Dependencies

Subcategorisation:
- Semantic forms (subcat frames): `sign<SUBJ, OBJ>`
- Completeness:
  all GFs in semantic form present at local f-structure
- Coherence:
  only GFs in semantic form present at local f-structure

Long Distance Dependencies (LDDs):
- Resolved at f-structure with **Functional Uncertainty Equations** (regular expressions specifying paths in f-structure).
LFG LDDs: Complement Relative Clause

Penn-II

LFG
LFG LDDs: Complement Relative Clause
V. **Head-Driven Phrase Structure Grammar**
Head-Driven Phrase Structure Grammar

• HPSG (Pollard and Sag 1994, Sag et al. 2003) is a unification-/constraint-based theory of grammar
• HPSG is a lexicalized grammar formalism
• HPSG aims to explain generic regularities that underlie phrase structures, lexicons, and semantics, as well as language-specific/-independent constraints
• Syntactic/semantic constraints are uniformly denoted by signs, which are represented with feature structures
• Two components of HPSG
  – Lexical entries represent word-specific constraints (corresponding to elementary objects)
  – Principles express generic grammatical regularities (corresponding to grammatical operations)
Sign

- **Sign** is a formal representation of combinations of phonological forms, syntactic and semantic constraints.
Lexical entries express word-specific constraints

We use simplified notations in this tutorial:

- **PHON** "loves"
- **HEAD** *verb*
- **SUBJ** <NP₁>
- **COMPS** <NP₂>
- **CONT** `love(₁,₂)`

- **HEAD** *noun*
- **SUBJ** < >
- **COMPS** < >
- **CONT** [₁]
Lexical entries
Lexical entries represent word-specific constraints
→ Difference in lexical entry
  = difference in grammatical characteristics

Subcategorization frames
e.g. sentential complement

Syntactic alternation
e.g. passive

```
[ PHON "think" 
  HEAD verb 
  SUBJ <NP₁ > 
  COMPS <S₂ > 
  CONT think(₁, ₂ ) ]

[ PHON "loved" 
  HEAD verb 
  SUBJ <NP₁ > 
  COMPS <PP by ₂ > 
  CONT love(₂, ₁ ) ]
```
Principles

Principles describe **generic regularities** of grammar

Do not correspond to construction rules

- **Head Feature Principle**
  The value of HEAD must be percolated from the head daughter

  \[
  \begin{array}{c}
  \text{HEAD} \\
  1
  \end{array}
  \quad \rightarrow \quad \cdots \quad \begin{array}{c}
  \text{HEAD} \\
  1
  \end{array}
  \quad \cdots \quad \begin{array}{c}
  \text{head daughter}
  \end{array}
  \]

- **Valence Principle**
  Subcats not consumed are percolated to the mother

- **Immediate Dominance (ID) Principle**
  A mother and her immediate daughters must satisfy one of *immediate dominance schemas*

Many other principles: percolation of NONLOCAL features, semantics construction, etc.
Schemas

Schemas correspond to construction rules in CFGs and other grammar formalisms

– For subject-head constructions (ex. “John runs”)

\[
\begin{array}{c}
\text{SUBJ} \\
\hline
\text{1} \\
\text{SUBJ < 1 >}
\end{array}
\]

– For head-complement constructions (ex. “loves Mary”)

\[
\begin{array}{c}
\text{COMPS} \\
\hline
\text{2} \\
\text{COMPS < 1 | 2 >} \\
\text{1}
\end{array}
\]

– For filler-head constructions (ex. “what he bought”)

\[
\begin{array}{c}
\text{SLASH} \\
\hline
\text{2} \\
\text{1} \\
\text{SLASH < 1 | 2 >}
\end{array}
\]
Example: HPSG parsing

• Lexical entries determine syntactic/semantic constraints of words

Lexical entries

- Head noun
  - Subject (John)
  - Complements (Mary)

- Head verb
  - Subject (John)
  - Complements (Mary)

- Head noun
  - Subject (John)
  - Complements (Mary)
Example: HPSG parsing

Principles determine generic constraints of grammar

[Diagram showing HPSG parsing with examples:]
- John: HEAD noun, SUBJ <> 1, COMPS <>
- Mary: HEAD noun, SUBJ <> 2, COMPS <>
- saw: HEAD verb, SUBJ <HEAD noun>, COMPS <HEAD noun>
- Unification: HEAD 1, SUBJ 2, COMPS [3 4]
Example: HPSG parsing

Principle application produces phrasal signs

```
[HEAD noun
  SUBJ <>
  COMPS <>
][HEAD noun
  SUBJ <>
  COMPS <>
][HEAD noun
  SUBJ <>
  COMPS <>
]
```

John

HEAD verb
  SUBJ <HEAD noun>
  COMPS <>

saw

HEAD noun
  SUBJ <>
  COMPS <>

Mary
Recursive applications of principles produce syntactic/semantic structures of sentences

```
[HEAD verb
  SUBJ <>
  COMPS <>]

[HEAD verb
  SUBJ <HEAD noun>
  COMPS <>]

[HEAD noun
  SUBJ <>
  COMPS <>]  
John

[HEAD noun
  SUBJ <HEAD noun>
  COMPS <HEAD noun>]  
saw

[HEAD noun
  SUBJ <>
  COMPS <>]  
Mary
```
Example: Control verbs

*I persuaded* him to quit the trip.
→ He quit the trip  (object control)

*I promised* him to quit the trip.
→ I quit the trip   (subject control)

\[
\begin{align*}
\text{PHON} & \text{ “persuade”} \\
\text{HEAD} & \text{ verb} \\
\text{SUBJ} & \text{ <NP} \_1 > \\
\text{COMPS} & \text{ <NP} \_2 \text{, VP}_3 > \\
\text{CONT} & \text{ persuade(1, 2, 3(2,...))}
\end{align*}
\]

\[
\begin{align*}
\text{PHON} & \text{ “promise”} \\
\text{HEAD} & \text{ verb} \\
\text{SUBJ} & \text{ <NP} \_1 > \\
\text{COMPS} & \text{ <NP} \_2 \text{, VP}_3 > \\
\text{CONT} & \text{ promise(1, 2, 3(1,...))}
\end{align*}
\]

persuade(I, he, quit(he, trip))

promise(I, he, quit(I, trip))
Nonlocal dependencies

- NONLOCAL features (SLASH, REL, etc.) explain long-distance dependencies
  - WH movements
  - Topicalization
  - Relative clauses etc...

```
(1) [HEAD det
    SUBJ <->
    COMPS <->
    SPR <->]

the

(2) [HEAD noun
    SUBJ <->
    COMPS <->
    SPR <1>]

prices

(3) [HEAD noun
    SUBJ <->
    COMPS <->
    SPR <2>]

(4) [HEAD verb
    SUBJ <3>
    COMPS <4>
    SLASH <2>]

we

(5) [HEAD verb
    SUBJ <3>
    COMPS <2>
    SLASH <2>]

were

(6) [HEAD verb
    SUBJ <3>
    COMPS <2>
    SLASH <2>]

charged
```
HPSG resources

• Enju: an English HPSG grammar extracted from Penn Treebank
• Hand-crafted grammars
  – LinGO ERG (English)
  – JaCY (Japanese)
  – GG (German)
  – Alpino (Dutch)
  – Grammars for other languages are underdevelopment in the DELPH-IN community
• Grammar Matrix
  – A framework for the rapid start-up of new grammars
  – The framework provides principles/structures shared among all grammars
VI. INDUCING EXPRESSIVE GRAMMARS FROM CORPORA
Obtaining wide-coverage grammars

• Extracting grammars from treebanks:
  • Leverage the effort that went into original annotation
  • Requires a formalism (and treebank-)specific algorithm to translate existing treebank into desired target

• Handwritten grammars:
  • Require substantial manual effort
  • Difficult to reuse grammars across formalisms
  • Examples: XLE (LFG), ERG (HPSG), XTAG (TAG),...
Grammar extraction

Source Treebank

Source Grammar (Treebank manual)

Translation

Target Treebank

Target Grammar (TAG, CCG, HPSG, LFG)
Treebanks...

... contain arbitrary text:
  – arbitrarily long sentences:
    • parentheticals, speech repairs, complex coordinations...
  – arbitrarily short sentences:
    • fragments, headlines,...

... contain arbitrary descriptions:
  – arbitrarily complex descriptions:
    • coindexation, null elements, secondary edges...
  – arbitrarily simplified/shallow descriptions:
    • compound nouns, fragments, argument-adjunct distinction
Grammar formalisms...

....provide analyses for well-studied constructions
  – It may be unclear how to analyze less well-studied constructions

... may provide constrained expressivity
  – *Mildly context-sensitive* formalisms (TAG/CCG) cannot capture arbitrary (e.g. anaphoric) dependencies

... may require complete analyses
  – *Lexicalized* formalisms need lexical entries for every word
Research questions

• Are the treebank descriptions sufficient to obtain the desired ‘deep’ analyses?

• Can the grammar formalism account for the descriptions provided in the treebank?
What do we need to extract grammars from treebanks?

Source treebank needs to have an explicit representation of:

- heads
- arguments
   \[ \{ \text{core dependencies} \} \]
- modifiers
- conjuncts
- nonlocal dependencies

Extraction algorithms need to distinguish each dependency type.

\[ \Rightarrow \text{special treatment!} \]
What do treebanks capture?

Local dependencies and phrase structure
- Head-argument, head-modifier, simple coordination
- Core of any annotation;
  but argument/modifier distinction not always clear

Nonprojective dependencies
- Extraposition, scrambling
- Captured directly in dependency banks;
  with null elements in treebanks

Nonlocal dependencies
- Raising, control; wh-extraction, topicalization;
  non-standard coordination
- Require other means of representations
  (traces, secondary edges) – often ideosyncratic
- Annotation sometimes missing
Challenges for grammar extraction

Differences in analysis
– may require **systematic changes** to treebank

Treebank uses underspecified analyses
– may require **additional annotation or heuristics**

Noise in treebank analysis
– may require **ad-hoc changes** to treebank
The mapping is not a function

- TB contains dependencies the grammar can’t capture.
- TB doesn’t contain enough information to define a single target analysis.
- TB makes distinctions which the grammar does not care about (inconsistencies?)
– Phrase-structure treebank
  requires head-finding and arg/adjunct distinction heuristics

– Non-local dependencies:
  null elements, traces, and coindexation

  *-null elements: passive, PRO
  *T*-traces: wh-movement, tough movement
  *RNR*-traces: right-node raising

Other null elements:
  *EXP*: expletive,
  *ICH* (“insert constituent here”): extraposition
  *U* (units): $ 500 *U*
  *PPA* (permanent predictable ambiguity)

=--coindexation: argument cluster coordination and gapping
Wh-extraction in the Penn Treebank

Coindexed traces indicate non-local dependencies
– Explicit annotation of heads, arguments, modifiers, conjuncts

– Non-local dependencies: discontinuous constituents (or secondary edges)
GRAMMAR EXTRACTION
General procedure

1. **Cleanup/preprocessing** (optional)
   a) Eliminate noise and inconsistencies
   b) Change unwanted analyses; use heuristics to add information

2. **Parse treebank**
   a) Identify local dependencies: heads, args, modifiers, conjuncts
   b) Identify non-local dependencies: extraction, non-stand. coordination.

3. **Translate treebank**
   a) Basic case: local dependencies
      each type may require different treatment
   b) Special cases: non-local dependencies
      each type may require different treatment

4. **Postprocessing** (optional)
   a) clean-up
   b) translate syntactic analysis into semantics
Evaluating extracted grammars or lexicons

• Grammar/lexicon size
  – How many entries does each word have?
  – How many kinds of entries (e.g. different categories)?
  – Depends on heuristics used and on granularity of analysis

• Coverage and convergence
  – How many lexical entries required to parse unseen data are missing?

• Distribution of types of lexical entries
  – How many different kinds of rare categories?

• Quality?
  – Inspection, comparison with manual grammar
EXTRACTING TAGs
TAG extraction: head + arguments
EXTRACTING CCGs
The basic translation algorithm

1. Identify heads, arguments, adjuncts
2. Binarize tree
3. Read off CCG categories
4. Get dependency structure
CCGbank derivations

that ((NP\NP)/(S[dcl]\NP)) funds are, will
are ((S[dcl]\NP)/(S[pss]\NP)) funds listed
soon ((S\NP)/(S\NP)) will
will ((S[dcl]\NP)/(S[b]\NP)) funds be
be ((S[b]\NP)/(S[pss]\NP)) listed
listed (S[pss]\NP) funds
in (((S\NP)/(S\NP))/NP) listed York, London
Wh-extraction in CCGbank

- The trace is cut out, but the dependency is captured.
- The relative pronoun subcategorizes for an incomplete sentence.
- This derivation requires type-raising and composition.
Right-node raising

- are
- conj
- or
- soon will be
- funds
- listed

listed in London

are  ((S[dcl]\NP)/(S[pss]\NP)) funds listed
soon  ((S\NP)/(S\NP)) will
will  ((S[dcl]\NP)/(S[b]\NP)) funds be
be  ((S[b]\NP)/(S[pss]\NP)) listed
listed  (S[pss]\NP) funds
in  (((S\NP)/(S\NP))/NP) listed York, London
• Coverage of the translation algorithm: 99.44% of all sentences in the Treebank (main problem: sentential gapping)

• The lexicon (sec.02-21):
  – 74,669 entries for 44,210 word types
  – 1286 lexical category types (439 appear once, 556 appear 5 times or more)

• The grammar (sec. 02-21):
  – 3262 rule instantiations (1146 appear once)
The most ambiguous words

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>as</td>
<td>130</td>
<td>4237</td>
<td>of</td>
<td>59</td>
<td>22782</td>
</tr>
<tr>
<td>is</td>
<td>109</td>
<td>6893</td>
<td>that</td>
<td>55</td>
<td>7951</td>
</tr>
<tr>
<td>to</td>
<td>98</td>
<td>22056</td>
<td>-LRB-</td>
<td>52</td>
<td>1140</td>
</tr>
<tr>
<td>than</td>
<td>90</td>
<td>1600</td>
<td>not</td>
<td>50</td>
<td>1288</td>
</tr>
<tr>
<td>in</td>
<td>79</td>
<td>15085</td>
<td>are</td>
<td>48</td>
<td>3662</td>
</tr>
<tr>
<td>–</td>
<td>67</td>
<td>2001</td>
<td>with</td>
<td>47</td>
<td>4214</td>
</tr>
<tr>
<td>’s</td>
<td>67</td>
<td>9249</td>
<td>so</td>
<td>47</td>
<td>620</td>
</tr>
<tr>
<td>for</td>
<td>66</td>
<td>7912</td>
<td>if</td>
<td>47</td>
<td>808</td>
</tr>
<tr>
<td>at</td>
<td>63</td>
<td>4313</td>
<td>on</td>
<td>46</td>
<td>5112</td>
</tr>
<tr>
<td>was</td>
<td>61</td>
<td>3875</td>
<td>from</td>
<td>46</td>
<td>4437</td>
</tr>
</tbody>
</table>

Many frequent words have a lot of categories
Frequency distribution of categories

<table>
<thead>
<tr>
<th>Category frequency $f$</th>
<th>#Cats.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$100,000 \leq f &lt; 220,000$</td>
<td>2</td>
</tr>
<tr>
<td>$10,000 \leq f &lt; 100,000$</td>
<td>13</td>
</tr>
<tr>
<td>$1,000 \leq f &lt; 10,000$</td>
<td>49</td>
</tr>
<tr>
<td>$100 \leq f &lt; 1,000$</td>
<td>108</td>
</tr>
<tr>
<td>$10 \leq f &lt; 100$</td>
<td>253</td>
</tr>
<tr>
<td>$5 \leq f &lt; 10$</td>
<td>131</td>
</tr>
<tr>
<td>$2 \leq f &lt; 5$</td>
<td>291</td>
</tr>
<tr>
<td>$0 &lt; f \leq 1$</td>
<td>440</td>
</tr>
</tbody>
</table>
Boxer: from CCG to DRT

Translates CCG derivations (output of C&C parser) to Discourse Representation Theory

The events of April through June damaged the respect and confidence which most Americans previously had for the leaders of China.

\[
\begin{align*}
\text{timex}(x_0) &= \text{XXXX06xx} \\
\text{event}(x_1) \\
\text{timex}(x_2) &= \text{XXXX04xx} \\
\text{through}(x_2, x_0) \\
\text{of}(x_1, x_2) \\
\text{named}(x_4, \text{china, loc}) \\
\text{leader}(x_5) \\
\text{of}(x_5, x_4)
\end{align*}
\]

\[
\begin{align*}
\text{have}(x_7) \\
\text{agent}(x_7, x_6) \\
\text{patient}(x_7, x_3) \\
\text{named}(x_6, \text{americans, nam}) \\
\text{previously}(x_7) \\
\text{event}(x_7) \\
\text{for}(x_7, x_5) \\
\text{damage}(x_8) \\
\text{event}(x_8) \\
\text{agent}(x_8, x_1) \\
\text{patient}(x_8, x_3)
\end{align*}
\]
Reanalyzing the Penn Treebank

- Propbank and Nombank add information to the original Penn Treebank
- Vadas & Curran (ACL’08) add internal structure to compound nouns in Penn Treebank
- Honnibal, Curran & Bos (ACL’10) integrate this information into CCGbank
Extracting a CCG from Tiger

– We translate 92.4% of all trees into CCG (more work required...)
– >2500 lexical categories
COFFEE BREAK
EXTRACTING LFGs
Treebank Annotation: what we have

S
  └── S-TPC-1
      ├── NP
      │   └── VP
      │       └── DT the
      │           └── NN headline
      │               └── VBD said
      │                   └── S
      │                       └── *T*-1
      └── VP
          └── NP
              └── NNP U.N.
                  └── VBZ signs
                      └── NN treaty
Treebank Annotation: what we want

S

S-TPC-1
(\uparrow \text{TOPIC}) = \downarrow
\downarrow = F1

NP
(\uparrow \text{SUBJ}) = \downarrow

VP
\uparrow = \downarrow

NP
(\uparrow \text{SUBJ}) = \downarrow

NN
\uparrow = \downarrow

DT
(\uparrow \text{DPEC:DET}) = \downarrow

NN
\uparrow = \downarrow

VBD
\uparrow = \downarrow
(\uparrow \text{COMP}) = F1

S

*\text{T*}-1

U.N.
\uparrow \text{PRED}=U.N.
\uparrow \text{NUM}=sg
\uparrow \text{PERS}=3

VBZ
\uparrow = \downarrow

| signs
\uparrow \text{PRED}=\text{sign}
\uparrow \text{TENSE}=\text{pres}
| treaty
\uparrow \text{PRED}=\text{treaty}
\uparrow \text{NUM}=sg
\uparrow \text{PERS}=3

the
\uparrow \text{PRED}=\text{the}

headline
\uparrow \text{PRED}=\text{headline}
\uparrow \text{NUM}=sg
\uparrow \text{PERS}=3

said
\uparrow \text{PRED}=\text{say}
\uparrow \text{TENSE}=\text{past}
Treebank Annotation: what we want
Treebank Annotation: what we have

S
  /\  \
S-TPC-1  NP
  /\  \
  VP  NP
    /\  \
   NNP  VBZ  NP
      /\  \ |
     U.N. signs NN  treaty

VBD  S
  /\  \
   the headline  said  *T*-1
Treebank Annotation: what we have

S

NP

S-TPC

↓

F1

(↑TOPIC)=↓

NNP

↑=↓

U.N.

↑PRED=U.N.

↑NUM=sg

↑PERS=3

VBZ

↑=↓

signs

↑PRED=sign

↑TENSE=pres

NP

↑=↓

(↑OBJ)=↓

NN

treaty

↑PRED=treaty

↑NUM=sg

↑PERS=3

VP

↑=↓

DT

(↑DPEC:DET)=↓

↑PRED=the

NN

↑=↓

headline

↑PRED=headline

↑NUM=sg

↑PERS=3

NP

(↑SUBJ)=↓

VBD

↑=↓

said

↑PRED=say

↑TENSE=past

S

(↑COMP)=F1

↑=↓
Treebank Annotation: what we want

[Diagram of a tree structure with nodes labeled for PRED, TENSE, TOPIC, SUBJ, OBJ, and COMP, showing the sentence 'say sign U.N. treaty' and 'SPEC DET the headline']
Treebank Annotation: Penn-II & LFG

- Head-Lexicalisation [Magerman, 1994]
- Left-Right Context Annotation Principles
- Coordination Annotation Principles
- Catch-All and Clean-Up
- Traces

Proto F-Structures

Proper F-Structures
Treebank Annotation: Traces

Long Distance Dependencies:
• Topicalisation
• Questions
• Wh- and wh-less relative clauses
• Passivisation
• Control constructions
• ICH (interpret constituent here)
• RNR (right node raising)
• ...

Translate Penn-II traces and coindexation into corresponding reentrancy in f-structure
Treebank Annotation: Control & Wh-Rel. LDD

NP

NP

S

SHAP

| RELMOD= |

WHNP

| TOPICREL= |

the energy and ambitions

IN

that

NP-SBJ

| SUBJ= |

| =F3 |

NNS

| =F2 |

reformers

VB

| = |

wanted

S

| XCOMP= |

VP

| = |

NP-SBJ

| SUBJ= |

| =F2 |

-NONE-

TO

| TO-INF=+ |

to

VP

| = |

VB

| = |

NP

| OBJ= |

| =F3 |

-NONE-
Treebank Annotation: Right Node Raising
Treebank Annotation: Right Node Raising

apply<subj,obl:for>
win<subj,obj>
LFG grammar acquisition for parsing and generation
- **Spanish**: Cast3LB (Grzegorz Chrupala)
- **German**: TiGer Treebank (Ines Rehbein)
- **French**: P7T and MFT (Natalie Schluter)
- **Chinese**: CTB6 (Yuqing Guo)
- **Arabic**: ATB (Jafa Al’Raheb, Lamia Tounsi, Mohammed Attia, Hann Bchara)
- **Japanese**: Kyoto Text Corpus (Massanori Oya)

• Typologically very different languages
• Morpologically rich/poor
• Semi-free word order – strongly configurational languages
• Drop: pro, anything …
Other Treebanks and Dependency Banks

As a consequence:

• Very different LFG f-str annotation algorithms
• Original f-str annotation algorithm for English (configurational, not much morphology) and Penn-II (“X-bar, traces ….)
• More recent f-str annotation algorithms:
  • Use richer treebank labels
  • “Translate” to f-structures
  • More machine learning
not want look-for train have potential DE new writer
‘(People) don’t want to look for and train the new writers who have potential.’
Extracting HPSG
Translating Penn Treebank into HPSG

• Convert Penn-style phrase structure trees into HPSG-style structures
  – Converting tree structures
    • Small clauses, passives, NP structures, auxiliary/control verbs, LDDs, etc.
  – Mapping into HPSG-style representations
    • Head/argument/modifier distinction, schema name assignment
    • Mapping into HPSG signs
  – Applying HPSG principles/schemas
    • Fully specified HPSG structures are obtained
Overview

Principle/schema application

Mapping into HPSG-style representation
Tree structure conversion

• Coordination, quotation, insertion, and apposition
• Small clauses, “than” phrases, quantifier phrases, complementizers, etc.
• Disambiguation of non-/pre-terminal symbols (TO, etc.)
• HEAD features (CASE, INV, VFORM, etc.)
• Noun phrase structures
• Auxiliary/control verbs
• Subject extraction
• Long distance dependencies
• Relative clauses, reduced relatives
Passive

• “be + VBN” constructions are assigned “VFORM passive”
Auxiliary/control verbs

• Reentrancies are annotated for representing shared arguments

```
S
  NP-1
  "they"
  VP
  "did"
  n't
  have

S
  NP-1
  "they"
  VP
  "did"
  n't
  have

S
  VP
  "to"
  choose

S
  VP
  "to"
  choose
```

```
NP
  "this particular moment"
```

```
NP
  "this particular moment"
```
LDDs: Object relative

- SLASH represents moved arguments
- REL represents relative-antecedent relations

```
NP[REL <> SLASH <>]
  NP[REL <2> SLASH <>]
    NP[2]
    the energy and ambitions
    NP[1][REL <2> that]
    WHNP-3
  SBAR[REL <2> SLASH <>]
    S[SLASH <1>]
    NP-2
    VP[SLASH <1>]
    S[SLASH <1>]
    VP[SLASH <1>]
    NP
    VP[SLASH <1>]
    *-2 to
    reward
    NP
    *T*-3
```
Mapping into HPSG-style representations

• Convert pre-/non-terminal symbols into HPSG-style categories

  NN  \[\text{HEAD} \quad \text{noun}\]
      \[\text{AGR} \quad 3sg\]

  VBD  \[\text{HEAD} \quad \text{verb}\]
       \[\text{VFORM} \quad \text{finite}\]
       \[\text{TENSE} \quad \text{past}\]

• Assign schema names to internal nodes
Category mapping
& schema name assignment

• Example: “NL is officially making the offer”
Principle/schema application

- Subject-head
  - Subject
    - Verb: making
    - Head-comp
      - Verb
        - Head-mod: officially
        - Head-comp
          - Verb
            - Subject: the offer
            - COMPS

- Verb: is
  - Subject: the offer
  - COMPS

- Verb: is officially making the offer
Complicated example
Extracting lexical entries

Collect leaf nodes

Generalize & assign predicate argument structures

make:

make:

make:
Generalization

• Remove unnecessary feature values
• Convert lexical entries of inflected words into lexical entries of lexemes using *inverse* lexical rules
  
  – Derivational rules: Ex. passive rule

  \[
  \begin{align*}
  \text{HEAD} & \quad \text{verb} \\
  \text{SUBJ} & \quad <\text{HEAD: noun}> \\
  \text{COMPS} & \quad <\text{HEAD: prep by}> \\
  \end{align*}
  \]

  \[
  \begin{align*}
  \text{HEAD} & \quad \text{verb} \\
  \text{SUBJ} & \quad <\text{HEAD: noun}> \\
  \text{COMPS} & \quad <\text{HEAD: noun}> \\
  \end{align*}
  \]

  – Inflectional rules: Ex. past-tense rule

  \[
  \begin{align*}
  \text{HEAD} & \quad \begin{bmatrix}
  \text{verb} \\
  \text{VFORM} \\
  \text{TENSE}
  \end{bmatrix}
  \text{finite past} \\
  \end{align*}
  \]

  \[
  \begin{align*}
  \text{HEAD} & \quad \begin{bmatrix}
  \text{verb} \\
  \text{VFORM}
  \end{bmatrix}
  \text{base} \\
  \end{align*}
  \]
Predicate argument structures

• Create mappings from syntactic arguments into semantic arguments

Ex. lexical entry for “make”
Results

• Conversion coverage: 96% of sentences from Penn Treebank 02-21 were converted

• Lexicon:
  – Lexical entries are extracted for 45,236 word types
  – 1136 lexical entry types for base forms, 2289 types for expanded forms

• Parsing coverage: >99%
VII. WIDE-COVERAGE PARSING WITH EXPRESSIVE GRAMMARS
Wide-coverage parsing with expressive grammars

• Wide-coverage grammars are necessary for wide-coverage parsing ← solved!
• Wide-coverage grammars are a halfway to wide-coverage parsing
  – All grammatical structures do not necessarily correspond to “natural interpretation”
  – High parsing accuracy = accurate selection of the “correct” one from possible grammatical structures

Time flies like an arrow.

Which corresponds to the correct interpretation?
LFG/CCG/HPSG parsing ≈ CFG parsing

- LFG/CCG/HPSG parsing is essentially phrase structure parsing
- Conventional methods for CFG parsing can be applied
  - Chart parsing
  - Statistical models for disambiguation (PCFG, machine learning, etc.)
  - Search techniques (Viterbi, beam search, etc.)
Added benefits

- Semantic structures are output as a result of parsing
- Expressive grammars restrict search space
  - Ungrammatical structures are excluded by hard constraints
- Expressive grammars provide additional information for statistical disambiguation
  - Lexical categories, lexical entries → supertagging
  - Predicate argument structures → semantic features
  - f-structures
Available wide-coverage parsers and basic architectures

• CCG
  – C&C parser: supertagging + discriminative model for phrase structure parsing
  – StatCCG: generative parser

• HPSG
  – Enju parser: supertagging + discriminative model for phrase structure parsing

• LFG
  – DCU-LFG: pipeline architecture, integrated architecture
PARSING WITH CCG/HPSG
Basic architecture

• Supertagging + phrase structure parsing
  Looks like chart parsing
  • Terminal symbol: lexical category, lexical entry
  • Production rule: combinatory rule, principle/schema

• TAG, CCG and HPSG are lexicalized
  • Lexical categories/entries encode rich grammatical constraints
  • Terminal symbol selection (=supertagging) plays a crucial role
Supertagging

• Supertag = lexical category, lexical entry
• Supertagging = assign supertags to each word without parsing

P: large

P: small

I

like

it
Supertagging is “almost parsing”

- When a supertag is determined, the structure that will be constructed is almost determined.
- Supertagging greatly reduces the search space → boosts parsing speed and accuracy.

When gold supertags are given, random choice from a parse forest achieves >95% accuracy.
Machine learning for supertagging

• Supertagging is a sequence labeling task
  → Machine learning methods can be applied
    – Log-linear models, perceptron, etc.
• Simple machine learning works: in many cases, supertags can be determined by local contexts

... man **forced** his friend to ...
... **NN VBD PRP$ NN TO** ...
Effect of supertagging

- Experiments on HPSG parsing
- Evaluation metrics:
  - Labeled accuracy of predicate argument relations
  - Average parsing time per sentence

<table>
<thead>
<tr>
<th></th>
<th>LP(%)</th>
<th>LR(%)</th>
<th>F1(%)</th>
<th>Avg. time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chart parsing w/o supertagging</td>
<td>84.96</td>
<td>84.25</td>
<td>84.60</td>
<td>674ms/sent.</td>
</tr>
<tr>
<td>Chart parsing w/ supertagging</td>
<td>87.35</td>
<td>86.29</td>
<td>86.81</td>
<td>183ms/sent.</td>
</tr>
<tr>
<td>Supertagging + CFG filtering</td>
<td>86.90</td>
<td>86.71</td>
<td>86.80</td>
<td>19ms/sent.</td>
</tr>
</tbody>
</table>
Probabilistic grammars

Generative models: \( P(w, T) \)
- Joint distribution over all strings \( w \) and trees \( T \)
- Use Bayes Rule: \( \arg\max_T P(T \mid w) = \arg\max_T P(w,T) \)
- Advantage: easy to estimate (rel. frequencies)
- Disadvantages: difficult to capture complex features

Discriminative models: \( P(T \mid w) \)
- Use loglinear models to define distributions \( P(T \mid w) \)
- Advantage: can use complex features
- Disadvantage: more difficult to train
Generative models for expressive grammars

TAG/CCG:

- Very similar to (lexicalized) probabilistic CFGs
- Lexical entries are treated as atomic units. Since coindexation/reentrancies are properties of lexical elements (TAG: trees with traces; CCG: categories with coindexation), this does not cause any problems for generative models

LFG/HPSG:

- Reentrancies in feature structures cannot be modeled with generative models (Abney 2000)
- LFG: can use any (P)CFG parser for c-structure alone
Discriminative models

Probability $p(T)$ of parse tree $T$ given sentence $w$:

$$p(T \mid w) \propto \exp(\lambda \cdot f(T))$$

Non-probabilistic models can also been applied:
– SVM, averaged perceptron, etc.
– Sufficient for choosing the best parse
Design of features

- Feature engineering is essential for high accuracy
- Features should capture syntactic/semantic characteristics of structures
  - Syntactic categories, lexical heads, POSs, constituent size, distance, etc.
Example: syntactic features

Features for the Head-Modifier construction for “saw a girl” and “with a telescope”

\[
h = \left\{ \begin{array}{l}
\text{head\_modifier\_schema, distance = 3,} \\
\text{leftspan = 3, VP, saw, VBD, transitive\_verb,} \\
\text{rightspan = 3, PP, with, IN, vp\_modifying\_prep}
\end{array} \right\}
\]
Example: semantic features

Features for the predicate argument relation between “he” and “saw”

\[
f = \begin{cases} 
\text{label = ARG1, distance = 1,} \\
\text{saw, VBD, transitive_verb,} \\
\text{he, PRP, pronoun}
\end{cases}
\]
Long distance dependencies

TAG, CCG, HPSG:
The lexicon captures long-distance dependencies

– TAG, HPSG: LDDs require different lexical entries
  ⇒ Supertagging is crucial

... do you like ...

Which lexical entry should be assigned?
PARSING WITH LFGs
Basic architectures

LFG has two levels of representation
  – c-structure
  – f-structure

• Pipeline architecture:
  – Strategy: c-structure first, then f-structure
  – Advantage: existing PCFG parsers can be used

• Integrated architecture:
  – Strategy: compute both structures at the same time
  – Advantage: c-/f-structures may effectively constrain ungrammatical structures during parsing
LFG Parsing Architectures

Pipeline:
- Penn Treebank
  - Trees
  - PCFGs
  - History-Based Parsers
  - Trees
  - PCFG Parsers
  - Automatic F-Structure Annotation

Integrated:
- Annotated Trees
  - Automatic F-Structure Annotation
  - F-Structures
  - A-PCFGs
  - Parser
  - Constraint Solver
  - LDD Resolution
  - FU-Equations
  - Frames
  - Subcategorisation
Parsing: LFG and LDD Resolution

• Penn-II tree: traces and co-indexation for LDDs

“U.N. signs treaty, the paper said”

Proper f-structure
“PCFG” Parse tree without traces:

“U.N. signs treaty, the paper said”
Parsing: LFG and LDD Resolution

- Require:
  - functional uncertainty equations
  - subcat frames
- How? From f-str annotated Penn-II …
- Previous Example:
  - \( \uparrow \text{TOPIC} = \uparrow \text{COMP}^* \text{COMP} \) (search along a path of 0 or more comps)
  - say<SUBJ,COMP>
Parsing: LFG and LDD Resolution

- Previous Example:
  - \( \uparrow \text{TOPIC} = \uparrow \text{COMP}^* \text{COMP} \)
  - say\(<\text{SUBJ,COMP}>\)
EFFICIENCY AND ACCURACY
Efficiency and accuracy

Is parsing with expressive grammars slow?
- It was very slow more than ten years ago
- Various techniques have been proposed (details omitted)
  - Supertagging
  - Beam search techniques: iterative, global thresholding
  - CFG filtering
- Latest systems are faster than shallow parsers

Which parser is more accurate?
- How to compare parsing accuracy of different parsers?
## Efficiency comparison

<table>
<thead>
<tr>
<th>Parser</th>
<th>Framework</th>
<th>Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>MST parser</td>
<td>dependency</td>
<td>4.5 sent/sec</td>
</tr>
<tr>
<td>Sagae’s parser</td>
<td>dependency</td>
<td>21.6 sent/sec</td>
</tr>
<tr>
<td>Berkeley parser</td>
<td>CFG</td>
<td>4.7 sent/sec</td>
</tr>
<tr>
<td>Charniak’s parser</td>
<td>CFG</td>
<td>2.2 sent/sec</td>
</tr>
<tr>
<td>Charniak’s parser + reranker</td>
<td>CFG</td>
<td>1.9 sent/sec</td>
</tr>
<tr>
<td>Enju parser</td>
<td>HPSG</td>
<td>2.6 sent/sec</td>
</tr>
<tr>
<td>Fast Enju parser</td>
<td>HPSG</td>
<td>18.9 sent/sec</td>
</tr>
</tbody>
</table>
Accuracy evaluation

- Across-framework accuracy comparison
  - PS Parser
  - Dep. Parser
  - HPSG Parser
  - CCG Parser
  - LFG Parser
  - Common parse representation
    - Conversion
    - Evaluate accuracy

- Task-oriented evaluation (mentioned later)
  - PS Parser
  - Dep. Parser
  - HPSG Parser
  - CCG Parser
  - LFG Parser
  - NLP task (IE, etc.)
    - Features
    - Observe accuracy improvements
Across-framework accuracy comparison

• How do treebank-based constraint grammars/parsers compare to deep hand-crafted grammars/parsers like XLE and RASP?

• How do treebank–based CCG, LFG and HPSG compare with each other?

(Joint work with Aoife Cahil and Grzegorz Chrupala)
Parsers and data

• Parsers
  – Treebank-based LFG, CCG, HPSG parsers
  – RASP (version 2) (Briscoe & Carroll 2006)

• Data
  – PARC 700 Dependency Bank gold standard (King et al. 2003), Penn-II Section 23-based
  – DepBank (Briscoe & Carroll 2006) reannotated version of PARC 700 with CBS 500–style GRs
  – CBS 500 Dependency Bank gold standard (Carroll, Briscoe and Sanfillippo 1999), Susanne-based
Cross Comparison

Fig. 7. PARC700 dependencies for *But stocks kept falling*. Non-*pred* dependencies are indicated by dashed edges.

Fig. 8. DepBank dependencies for *But stocks kept falling*
Cross Comparison

• Lots of pain points:
  – Different tokenisation Penn-II and PARC700 and DepBank
  – Punctuation changed in DepBank => strings /= Penn-II
  – Different labels
  – Different analyses
  – Different granularity
  – Lots of fun

• Mapping ....
Treebank-Based LFG, CCG and HPSG

Dependency Evaluation Results against DepBank:

<table>
<thead>
<tr>
<th></th>
<th>Micro-average</th>
<th></th>
<th>Macro-average</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P</td>
<td>R</td>
<td>F</td>
<td>P</td>
</tr>
<tr>
<td>LFG</td>
<td>84.29</td>
<td>80.11</td>
<td>82.15</td>
<td>69.26</td>
</tr>
<tr>
<td>HPSG-Enju</td>
<td>83.57</td>
<td>81.73</td>
<td>82.64</td>
<td>77.87</td>
</tr>
<tr>
<td>CCG-C&amp;C</td>
<td>82.44</td>
<td>81.28</td>
<td>81.86</td>
<td>65.61</td>
</tr>
<tr>
<td>RASP-(v2)</td>
<td>77.66</td>
<td>74.98</td>
<td>76.29</td>
<td>61.12</td>
</tr>
</tbody>
</table>

Table 1: Results of LFG parsing resources against DepBank

<table>
<thead>
<tr>
<th></th>
<th>Micro-average</th>
<th></th>
<th>Macro-average</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P</td>
<td>R</td>
<td>F</td>
<td>P</td>
</tr>
<tr>
<td>LFG</td>
<td>86.06</td>
<td>83.96</td>
<td>85.00</td>
<td>71.42</td>
</tr>
<tr>
<td>HPSG-Enju</td>
<td>87.49</td>
<td>86.79</td>
<td>87.14</td>
<td>81.19</td>
</tr>
<tr>
<td>CCG-C&amp;C</td>
<td>86.86</td>
<td>82.75</td>
<td>84.76</td>
<td>71.73</td>
</tr>
</tbody>
</table>

Table 2: Upper Bound results of deep parsing resources against DepBank
Comparison against XLE and RASP

Labelled dependency f-scores
(Burke et al. 2004, Cahill et al. 2008):

**PARC 700**
- 80.55% XLE
- 82.73% DCU-LFG (+2.18%)
- 84.00% DCU-LFG now (+3.45%)

**CBS 500**
- 76.57% RASP
- 80.23% DCU-LFG (+3.66%)

Results statistically significant at ≥ 95% level (Noreen 1989)
VIII. APPLICATIONS
Applications of expressive grammars

• Parsing with expressive grammars is robust, accurate, ready to be applied to real-world problems
• Expressive grammars have shown competitive or state-of-the-art performance in several NLP tasks
  – Sentence realization (generation)
    • Grammars are necessary to bridge semantic representation to its sentence realization
  – Information extraction
    • Predicate argument relations are used like dependencies, with deeper information
  – Machine translation
    • Expressive syntactic/semantic structures are effectively combined with statistical MT
GENERATION
Sentence Realization

Sentence realization (generation):
Semantic representation $\rightarrow$ sentence

syntactic structure

Sentence realization:

Mary saw John

$parsing$

NP

VP

spring has come

see($Mary$, $John$)

semantic representation

grammar

PHON “saw”
HEAD verb
SUBJ <NP>
COMPS <NP>

...
Chart generation

• Chart parsing → chart generation
• Many parsing techniques can be applied to generation
  – Supertagging (*hypertagging*)
  – Beam search

He bought a book.

0                1              2              3

chart parsing

{0,1,2,3} {0,1,3} {0,2,3} {1,2,3}

{0,1} {0,2} {0,3} {1,2} {1,3} {2,3}

{0} {1} {2} {3}

he(x) buy(e) a(z) book(z)

0                1              2              3

chart generation
LFG Generation

Two architectures for generation from f-structures:

• Chart & Rule-Based Generation: use f-structure annotated CFG rules from Integrated Parsing Architecture + chart generator + \textit{probabilities conditioned on input f-structure} (!!)

• Dependency-Based Generation: linearize dependencies directly by learning \textit{n-gram models over dependencies} (NOT strings)!
LFG Generation: Chart & F-Str. Annotated Rule-Based

Probability Model

$$\arg\max_{\text{Tree}} P(\text{Tree}|\text{F-Str})$$

$$P(\text{Tree}|\text{F-Str}) := \prod_{X \rightarrow Y \text{ in Tree}} P(X \rightarrow Y | X, \text{Feats}) \quad (1)$$

$$\phi(X) = \text{Feats}$$
LFG Generation

NP
(↑SUBJ) = ↓
/  
|  
|  
NNP
(↑SUBJ) = ↓
|  
|  
V
(↑PRED) = ‘believe’
|  
|  
SBAR
(↑COMP) = ↓
|  
|  
S
|  
|  
PRED
|  
|  
‘BELIEVE((↑SUBJ)(↑COMP))’
|  
|  
SUBJ
|  
|  
f1:
NUM PL
PERS 3
|  
|  
COMP
|  
|  
f2:
|  
|  
f3:
SUBJ
|  
|  
f4:
NUM SG
PERS 3
|  
|  
TENSE
PRESENT
PAST
|  
|  
V
(↑PRED) = ‘resigned’
|  
|  
|  
|  
|  
|  
|  
|  
|  

They believe

NP
(↑SUBJ) = ↓
/  
|  
|  
John
(↑PRED) = ‘John’
(↑NUM) = SG
(↑TENSE) = PAST
(↑PERS) = 3

NP
(↑SUBJ) = ↓
/  
|  
|  

VP
(↑PRED) = ‘resigned’
(↑NUM) = PL
(↑TENSE) = present
(↑PERS) = 3

VP
(↑PRED) = ‘pro’
(↑NUM) = PL
(↑TENSE) = present
(↑PERS) = 3
LFG Generation: Dependency-Based

(a.) c-structure

(b.) f-structure

(c.) linearised grammatical functions / bilexical dependencies

Figure 1: C- and f-structures for the sentence We believe in the law of averages.
LFG Generation: Dependency-Based

\[ P(GF_1^m) = P(GF_1 \ldots GF_m) = \prod_{k=1}^{m} P(GF_k | GF_{k-n+1}^{k-1}) \] (1)

<table>
<thead>
<tr>
<th>Model</th>
<th>N-grams</th>
<th>Cond.</th>
</tr>
</thead>
<tbody>
<tr>
<td>basic ( (P) )</td>
<td>SPEC PRED ADJ</td>
<td></td>
</tr>
<tr>
<td>gf ( (P^g) )</td>
<td>SPEC PRED ADJ</td>
<td></td>
</tr>
<tr>
<td>pred ( (P^p) )</td>
<td>SPEC PRED ADJ</td>
<td>OBL ‘law’</td>
</tr>
<tr>
<td>lex ( (P^l) )</td>
<td>SPEC PRED[‘law’] ADJ[‘of’]</td>
<td></td>
</tr>
</tbody>
</table>
# Results

Table 6. *Cross system comparison of results for English WSJ section 23*

<table>
<thead>
<tr>
<th>System</th>
<th>Coverage</th>
<th>Complete</th>
<th>ExMatch</th>
<th>BLEU</th>
<th>SSA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Callaway (2003)</td>
<td>98.7%</td>
<td></td>
<td>49.0%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Langkilde (2002)&lt;sup&gt;a&lt;/sup&gt;</td>
<td>82.7%</td>
<td>28.2%</td>
<td>0.757</td>
<td>0.696</td>
<td></td>
</tr>
<tr>
<td>Nakanishi et al. (2005)&lt;sup&gt;b&lt;/sup&gt;</td>
<td>90.75%</td>
<td></td>
<td></td>
<td>0.7733</td>
<td></td>
</tr>
<tr>
<td>Cahill and van Genabith (2006)</td>
<td>98.05%</td>
<td>89.49%</td>
<td>0.6651</td>
<td>0.6808</td>
<td></td>
</tr>
<tr>
<td>Hogan et al. (2007)</td>
<td>99.96%</td>
<td></td>
<td></td>
<td>0.6882</td>
<td>0.7092</td>
</tr>
<tr>
<td>Rajkumar et al. (2009)</td>
<td>94.8%</td>
<td>85.04%</td>
<td>33.74%</td>
<td>0.8173</td>
<td></td>
</tr>
<tr>
<td>White and Rajkumar (2009)</td>
<td>97.06%</td>
<td>83.88%</td>
<td>40.45%</td>
<td>0.8506</td>
<td></td>
</tr>
<tr>
<td>Guo et al. (2008)</td>
<td>100%</td>
<td>100%</td>
<td>19.83%</td>
<td>0.7440</td>
<td>0.7534</td>
</tr>
<tr>
<td>This article LFG</td>
<td>100%</td>
<td>100%</td>
<td>31.54%</td>
<td>0.8065</td>
<td>0.7871</td>
</tr>
<tr>
<td>This article CoNLL</td>
<td>100%</td>
<td>100%</td>
<td>47.76%</td>
<td>0.8820</td>
<td>0.8596</td>
</tr>
</tbody>
</table>

<sup>a</sup> The results are for the “permute, no dir” type experiment in Langkilde (2002), where the inputs are most comparable to our f-structures in regard to the level of specification.

<sup>b</sup> The results are for sentences with a length limitation of 20 words.
INFORMATION EXTRACTION
Relation extraction

- Extracting relations expressed in texts
  - Protein-protein interactions
  - Gene-disease associations
  - Network of biological reactions (BioNLP’09 shared task)

- Train a machine learning classifier using parser output as features
  - Classification problem

\[
\begin{align*}
<\text{XPG}, \text{CSB}> & \quad \text{positive} \\
<\text{XPG}, \text{TFIIH}> & \\
<\text{TFIIH}, \text{CSB}> & \quad \text{negative}
\end{align*}
\]

XPG protein interacts with multiple subunit of TFIIH and with CSB protein.
BioNLP’09 shared task

• Finding **biological events** from abstracts
  – Protein annotations are given

... In this study we hypothesized that the phosphorylation of TRAF2 inhibits binding to the CD40 cytoplasmic domain. ...

**negative_regulation**

CAUSE:

**phosphorylation**

THEME: TRAF2

THEME:

**binding**

THEME: TRAF2

THEME2: CD40

SITE2: cytoplasmic domain
... the phosphorylation of TRAF2 inhibits binding to the CD40 cytoplasmic domain. ...

Trigger word detection

... the phosphorylation of TRAF2 inhibits binding to the CD40 cytoplasmic domain. ...

Event edge detection

... the phosphorylation of TRAF2 inhibits binding to the CD40 cytoplasmic domain. ...

Complex event detection

... the phosphorylation of TRAF2 inhibits binding to the CD40 cytoplasmic domain. ...

Entity

Theme

Cause

Negative_regulation

Binding
Event extraction system

- Event extraction by three modules
  - Trigger word detection
  - Event edge detection
  - Complex event detection
- Each module is a linear SVM with features on parsing output
  - Shortest dependency paths
  - Dependent/argument words
- Evaluate contributions from parsers and parse representation formats
Parsers & Formats

• Dependency parser
  – Gdep

• Phrase structure parsers
  – Stanford parser
  – McClosky’s self-trained parser (MC)

• Deep parser
  – C&C parser
  – Enju

• Parse representation formats
  – CoNLL-X
  – Stanford dependency (SD)
  – Predicate Argument Structure (PAS)
Parse representation formats

- CoNLL-X
  - root NFAT/AP-1 complex formed only with P and P2
  - NMOD
  - VMOD
  - PMOD
  - PMOD COORD

- Stanford (SD)
  - nn
  - prep
  - pobj
  - cc
  - nsubj
  - dep
  - conj
  - noun_arg1
  - arg1
  - prep_arg12
  - arg1
  - arg2
  - NFAT/AP-1 complex formed only with P and P2

- Enju PAS
  - verb_arg1
  - arg1
  - adj_arg1
  - arg1
  - coord_arg12
  - coord_arg12
  - arg1
  - arg2
  - NFAT/AP-1 complex formed only with P and P2
Format conversion

GDep
C&C
McClosky-Charniak
Bikel
Stanford
Enju

CCG
Treebank Converter
PTB
Syntactic Tree + PAS

CoNLL-X
C&C Tools
SD
Stanford tools
Enju2PTB
Results

- Parsers always help
- GDep, MC, C&C, and Enju are comparable
- Best results are close to results with gold parses

<table>
<thead>
<tr>
<th></th>
<th>SD</th>
<th>CoNLL</th>
<th>PAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>No parse</td>
<td>51.05</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GDep</td>
<td></td>
<td>55.70</td>
<td></td>
</tr>
<tr>
<td>Stanford</td>
<td>55.02</td>
<td>53.66</td>
<td></td>
</tr>
<tr>
<td>MC</td>
<td>55.60</td>
<td>56.01</td>
<td></td>
</tr>
<tr>
<td>C&amp;C</td>
<td>56.09</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Enju</td>
<td>55.48</td>
<td>55.74</td>
<td>56.57</td>
</tr>
<tr>
<td>Gold parse</td>
<td>56.34</td>
<td>56.09</td>
<td>57.94</td>
</tr>
</tbody>
</table>
Parser combination

• Combination helps in most cases
  – Different parsers/formats help a lot

<table>
<thead>
<tr>
<th></th>
<th>C&amp;C SD</th>
<th>MC CoNLL</th>
<th>Enju CoNLL</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC CoNLL</td>
<td>57.44 (+1.35)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Enju CoNLL</td>
<td>56.47 (+0.38)</td>
<td>56.24 (+0.23)</td>
<td></td>
</tr>
<tr>
<td>Enju PAS</td>
<td>57.20 (+0.63)</td>
<td>57.78 (+1.21)</td>
<td>56.59 (+0.02)</td>
</tr>
</tbody>
</table>

For more details, refer to Miwa et al. (COLING 2010)
Search engine for biomedical papers

- NLP tools are applied to 19 million abstracts in MEDLINE
  - HPSG parsing
  - Term recognition (proteins, diseases, etc.)
  - Event expression recognition

- HPSG parsing allows us to search for predicate argument relations rather than cooccurrences
  \[\rightarrow\] improves precision
Search for predicate argument relations

• “p53 activates something”

In this report, we demonstrated that human AMID gene promoter was activated by p53 in reporter gene assays.

The p53 protein integrates multiple upstream signals and functions as a tumor suppressor by activating distinct downstream genes.

Although p53 has been shown to directly activate transcriptional bax gene and to inhibit expression of bcl-2 gene during radiation-induced apoptosis, it is poorly understood how the Bcl-2 family changes in p53-deficient cells during radiation-induced apoptosis.

Since p21 is known to be transcriptionally activated by p53, these results suggest that TS downregulation of p21 may be occurring through a p53-independent mechanism in this in vitro cell system.

The DDATHF-stabilized p53 bound to the p21 promoter in vitro and in vivo but did not activate histone acetylation over the p53 binding sites in the p21 promoter that is an integral part of the transcriptional response mediated by the DNA damage pathway.
• Subject/predicate/object specification is matched with predicate argument structures
• Synonymous term/event expressions are matched

ERK2 activation is required for the MHBs (t) effect because ERK2 inhibition by its inhibitor PD98059 significantly reversed TRAIL-induced apoptosis of MHBs (t) -transfected cells.

In conclusion, we demonstrated for the first time that activation of phosphatidylinositol-3-kinase (PI-3K) -Akt and ERK2 pathways significantly contributed to cardioprotective effects of a Ca (2+) antagonist and a beta-adrenergic receptor blocker.

Recently, we found that all-trans retinoic acid (atRA) triggers the activation of extracellular-signal-regulated kinase 2 (ERK2), which phosphorylates TR2 and stimulates its partitioning to promyelocytic leukemia (PML) nuclear bodies, thereby converting the activator function of TR2 into repression (Gupta et al. 2008; Park et al. 2007).

Publicly available at: http://www-tsujii.is.s.u-tokyo.ac.jp/medie/
System Architecture

Huge text (e.g. MEDLINE)

Off-line

Parser

Term recognizer

Event expression recognizer

Search engine for structured text

On-line

Annotated text

Query

Results
MACHINE TRANSLATION
HPSG for syntax-based SMT

• HPSG works with syntax-aware SMT methods
  – Tree-to-string
  – String-to-tree
  – Forest-to-string
• HPSG structures provide rich syntactic/semantic information as features
  – Phrase structure
  – Construction type (i.e. schema name)
  – Syntactic/semantic head
  – Tense, aspect, voice
  – Lexical entry name
  – Predicate argument relations
Extracting translation rules from predicate argument relations

Minimum Covering Tree

Linear-time training & decoding
## En-to-Jp translation results

<table>
<thead>
<tr>
<th>System</th>
<th>BLEU (%)</th>
<th>Decoding time (Sec./Sentence)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joshua</td>
<td>21.79</td>
<td>0.486 (N=200)</td>
</tr>
<tr>
<td>Phrase-based</td>
<td>18.40</td>
<td>0.013 (N=10)</td>
</tr>
<tr>
<td>+ HPSG features</td>
<td>21.90</td>
<td>0.589</td>
</tr>
<tr>
<td>+ PAS-based rules</td>
<td>22.12</td>
<td>0.031</td>
</tr>
<tr>
<td>+ PAS-based rules, HPSG features</td>
<td>22.73</td>
<td>0.632</td>
</tr>
<tr>
<td>+ HPSG features (composed rules)</td>
<td>24.12</td>
<td>2.753</td>
</tr>
<tr>
<td>+ PAS-based rules, HPSG features (composed)</td>
<td>24.13</td>
<td>2.930</td>
</tr>
</tbody>
</table>

For more details, refer to Wu et al. (ACL 2010)
Most automatic MT evaluation metrics (BLEU, NIST) are string (n-gram) based.

Punish perfectly legitimate syntactic and lexical variation:
- *Yesterday* John resigned.
- John resigned yesterday.
- *Yesterday* John quit.

Legitimate lexical variation: WordNet synonyms into the string match

What about syntactic variation?
MT evaluation

• Idea: use labelled dependencies for MT evaluation
• Why: dependencies abstract away from some particulars of surface realisation
• Adjunct placement, order of conjuncts in a coordination, topicalisation, ...

\[
\begin{align*}
&\text{c-structure level:} \\
&\begin{aligned}
S &\rightarrow NP \rightarrow V \rightarrow NP-TMP \\
&\text{John} \rightarrow \text{resigned} \rightarrow \text{yesterday}
\end{aligned} \\
&\begin{aligned}
S &\rightarrow NP \rightarrow NP \rightarrow V \\
&\text{Yesterday} \rightarrow \text{John} \rightarrow \text{resigned}
\end{aligned}
\end{align*}
\]

\[
\begin{align*}
&\text{f-structure level:} \\
&\begin{aligned}
\text{SUBJ} &\rightarrow \text{PRED john} \\
&\text{NUM} \rightarrow \text{sg} \\
&\text{PERS} \rightarrow 3
\end{aligned} \\
&\begin{aligned}
\text{PRED} &\rightarrow \text{resign} \\
\text{TENSE} &\rightarrow \text{past} \\
\text{ADJ} &\rightarrow \{ \text{PRED yesterday} \}
\end{aligned}
\end{align*}
\]

\[
\begin{align*}
&\begin{aligned}
\text{SUBJ} &\rightarrow \text{PRED john} \\
&\text{NUM} \rightarrow \text{sg} \\
&\text{PERS} \rightarrow 3
\end{aligned} \\
&\begin{aligned}
\text{PRED} &\rightarrow \text{resign} \\
\text{TENSE} &\rightarrow \text{past} \\
\text{ADJ} &\rightarrow \{ \text{PRED yesterday} \}
\end{aligned}
\end{align*}
\]

\[
\begin{align*}
&\text{subj(resign, john)} \\
&\text{pers(john, 3)} \\
&\text{num(john, sg)} \\
&\text{tense(resign, past)} \\
&\text{adj(resign, yesterday)} \\
&\text{pers(yesterday, 3)} \\
&\text{num(yesterday, sg)}
\end{align*}
\]
Dependency-based MT evaluation

• Need a robust parser that can parse MT output 😊
  – Treebank-induced parsers parse (almost) anything …!

• To make this work, throw in:
  – n-best parsing
  – WordNet synonyms
  – partial matching
  – training weights

• Compare against string-based methods

• Compare (correlation) with human judgement
  – Why: humans not fooled by legitimate syntactic variation
IX. SUMMARY
Conclusions

Expressive grammars and robust, wide-coverage NLP are not a contradiction:

– Treebank-based grammar acquisition provides wide coverage
– Effective statistical parsing methods provide efficient and robust processing
– These grammars can also be used in other applications, e.g.: IE, generation and MT
X. *(PARTIAL) BIBLIOGRAPHY*
TAG references

TAG:

TAG extraction:
J. Chen, S. Bangalore, K. Vijaj-Shanker. Automated Extraction of Tree-Adjoining Grammars from Treebanks, Natural Language Engineering
TAG references

Supertagging:

Parsing with extracted TAGs:
L. Shen and A.K. Joshi. Incremental LTAG parsing, HLT/EMNLP 2005
Libin Shen and Aravind K. Joshi/ TAG Dependency Parsing with Bidirectional Incremental Construction EMNLP 2008
LFG references

Grammar Extraction: English

M. Burke, *Automatic Treebank Annotation for the Acquisition of LFG Resources*, Ph.D. Thesis, School of Computing, Dublin City University, Dublin 9, Ireland. 2005


LFG references

Arabic:


LFG references

French:


Natalie Schluter and Josef van Genabith, Treebank-Based Acquisition of LFG Parsing Resources for French, Proceedings of the Sixth International Language Resources and Evaluation (LREC'08), pp.2909-2916, Marrakech, Morocco, May 28-30, 2008, ISBN 2-9517408-4-0

LFG references

Chinese:


LFG Selected Publications

German:


Ines Rehbein and Josef van Genabith. Treebank Annotation Schemes and Parser Evaluation for German. in Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Natural Language Learning (EMNLP-CoNLL 2007), Prague, Czeck Republic, pp. 630-639

LFG Selected Publications

Spanish:

LFG references

Parsing:


LFG references

Loglinear models:
LFG references

Generation:


LFG references

Question Bank

MT evaluation
HPSG references

HPSG theory

HPSG grammars
HPSG references

HPSG treebanks


HPSG references

Supertagging
HPSG references

Parsing models for HPSG
HPSG references

Efficient parsing techniques for HPSG


HPSG references

Parser evaluation


HPSG references

Sentence realization


HPSG references

Applications
CCG references

CCG:

CCG implementations:
OpenCCG: http://openccg.sourceforge.net/
CCG references

CCG grammar extraction:
CCG references

CCGbank modifications:

CCG supertagging:
CCG references

Statistical CCG parsing:
CCG references

Parser evaluation:
Stephen Clark; James R. Curran. Comparing the Accuracy of CCG and Penn Treebank Parsers. ACL 2009

CCG generation:
Dominic Espinosa; Michael White; Dennis Mehay (2008). Hypertagging: Supertagging for Surface Realization with CCG
Michael White; Rajakrishnan Rajkumar (2009). Perceptron Reranking for CCG Realization. EMNLP 2009
CCG references

Boxer


CCG references

Semantic Role Labeling:

CCG applications:
CCG references

Machine translation
Alexandra Birch, Miles Osborne and Philipp Koehn (2007). CCG Supertags in factored translation models. In SMT Workshop at ACL
Treebank references

Penn Treebank:

Tiger Treebank:
Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, George Smith (2002) *The Tiger Treebank*. Workshop on Treebanks and Linguistic Theories

Parc700 and DepBank:
Stefan Riezler, Tracy H. King, Richard Crouch, Mary Dalrymple, Ronald M. Kaplan (2003) *The PARC 700 Dependency Bank*. Workshop on "Linguistically Interpreted Corpora" (LINC'03)