Syntactic Formalisms for Parsing Natural Languages

Aleš Horák, Miloš Jakubíček, Vojtěch Kovář
(based on slides by Juyeon Kang)

ia161@nlp.fi.muni.cz

Autumn 2013

Outline

- HPSG Parser : **Enju**
  - Parsing method
  - Description of parser
  - Result
- CCG Parser : **C&C Tools**
  - Parsing method
  - Description of parser
  - Result

Theoretical backgrounds

Lecture 3 about HPSG Parsing

Lecture 6 & 7 about CCG Parsing and Combinatory Logic

Enju (Y. Miyao, J.Tsujii, 2004, 2008)

- Syntactic parser for English
- Developed by Tsujii Lab. Of the University of Tokyo
- Based on the wide-coverage probabilistic HPSG
  - HPSG theory [Pollard and Sag, 1994]
- Useful links to Enju
  - http://www-tsujii.is.s.u-tokyo.ac.jp/enju/demo.html
  - http://www-tsujii.is.s.u-tokyo.ac.jp/enju/

Motivations

Parsing based on a proper linguistic formalism is one of the core research fields in CL and NLP.

But!

a monolithic, esoteric and inward looking field, largely dissociated from real world application.

Motivations

So why not!

The integration of linguistic grammar formalisms with statistical models to propose an robust, efficient and open to eclectic sources of information other than syntactic ones

Motivations

Two main ideas

- Development of wide-coverage linguistic grammars
- Deep parser which produces semantic representation (predicate-argument structures)

Parsing method

- Application of probabilistic model in the HPSG grammar and development of an efficient parsing algorithm
  - Accurate deep analysis
  - Disambiguation
  - Wide-coverage
  - High speed
  - Useful for high level NLP application

Parsing method

1. Parsing based on HPSG
   - Mathematically well-defined with sophisticated constraint-based system
   - Linguistically justified
   - Deep syntactic grammar that provides semantic analysis

Parsing method

Difficulties in parsing based on HPSG

- Difficult to develop a broad-coverage HPSG grammar
- Difficult to disambiguate
- Low efficiency: very slow

Parsing method

Solution:
Corpus-oriented development of an HPSG grammar

- The principal aim of grammar development is treebank construction
- Penn treebank is coverted into an HPSG treebank
- A lexicon and a probabilistic model are extracted from the HPSG treebank

Parsing method

Approach:

- develop grammar rules and an HPSG treebank
- collect lexical entries from the HPSG treebank

How to make an HPSG treebank?

Convert Penn Treebank into HPSG and develop grammar by restructuring a treebank in conformity with HPSG grammar rules

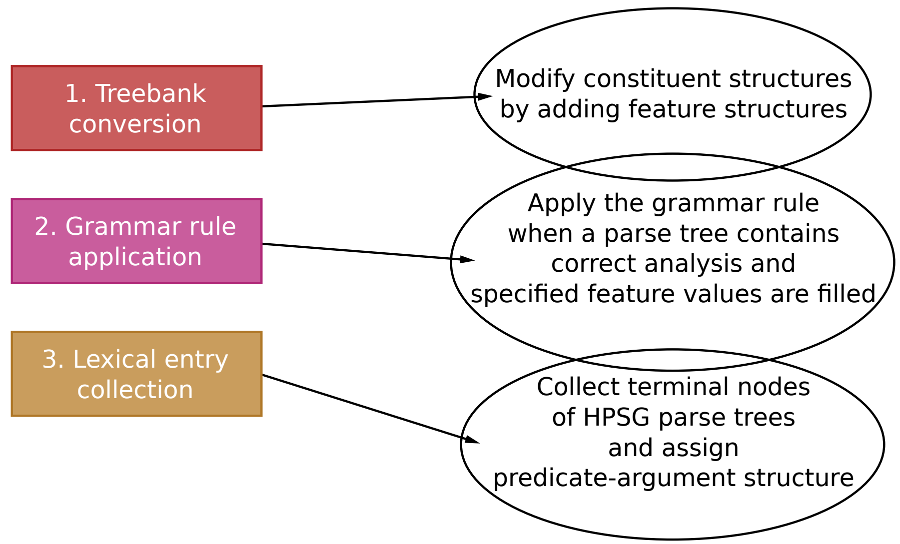
Parsing method

HPSG = lexical entries and grammar rules
Enju grammar has 12 grammar rules and
3797 lexical entries for 10,536 words

(Miyao *et al.* 2004)

Parsing method

Overview of grammar development



Parsing method

2 Probabilistic model and HPSG:

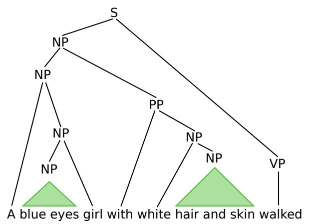
Log-linear model for unification-based grammars

(Abney 1997, Johnson et al. 1999, Riezler et al. 2000, Miyao et al. 2003, Malouf and van Noord 2004, Kaplan et al. 2004, Miyao and Tsujii 2005)

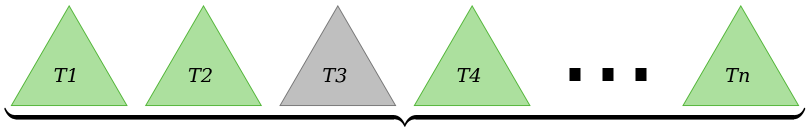
$$p(\mathrm{T}|w)$$

$w =$ *"A blue eyes girl with white hair and skin walked*

$T =$



Parsing method



All possible parse trees derived from w with a grammar.
For example, $p(T3|\mathrm{w})$ is the probability of selecting $T3$ from $T1$, $T2$, ..., and Tn.

Parsing method

Log-linear model for unification-based grammars

- Input sentence: w

$w = w_1/P_1, w_2/P_2, \dots w_n/P_n$

- Output parse tree T

$$p(T|\mathbf{w}) = \frac{1}{Z} \exp\left(\sum_u \lambda_u f_u(T)\right)$$

Normalization factor

Weight for a feature function

Feature function

Description of parser

Description of parser

parsing proceeds in the following steps:

1. preprocessing

Preprocessor converts an input sentence into a word lattice.

2. lexicon lookup

Parser uses the predicate to find lexical entries for the word lattice

3. kernel parsing

Parser does phrase analysis using the defined grammar rules in the kernel parsing process.

Description of parser

- Chart
  - data structure
  - two dimensional table
  - we call each cell in the table ‘CKY cell.’

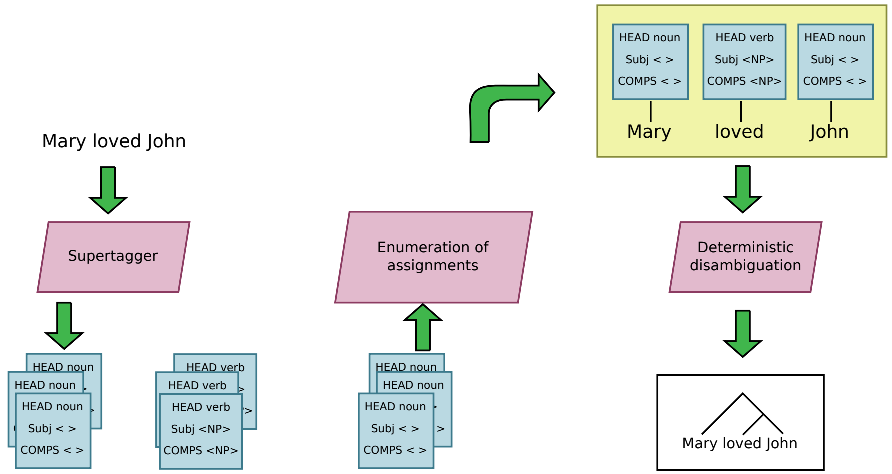
Example

Let an input sentence $s(= w1, w2, w3, ..., wn), w1 =$ ”I”, $w2 =$ ”saw”, $w3 =$ ”a”, $w4 =$ ”girl”, $w5 =$ ”with”, $w6 =$ ”a”, $w7 =$ ”telescope” for the sentence “*I saw a girl with a telescope*”, the chart is arranged as follows.

0,7						
0,6	1,7					
0,5	1,6	2,7				
0,4	1,5	2,6	3,7			
0,3	1,4	2,5	3,6	4,7		
0,2	1,3	2,4	3,5	4,6	5,7	
0,1	1,2	2,3	3,4	4,5	5,6	6,7
I	saw	a	girl	with	a	telescope

Description of parser

System overview



Demonstration

http://www-tsujii.is.s.u-tokyo.ac.jp/enju/demo.html

Results

- Fast, robust and accurate analysis
  - Phrase structures
  - Predicate argument structures
- **Accurate deep analysis** – the parser can output both phrase structures and predicate-argument structures. The accuracy of predicate-argument relations is around 90% for newswire articles and biomedical papers.
- **High speed** – parsing speed is less than 500 msec. per sentence by default (faster than most Penn Treebank parsers), and less than 50 msec when using the highspeed setting (”mogura”).

C&C tools

- Developed by Curran and Clark [Clark and Curran, 2002, Curran, Clark and Bos, 2007], University of Edinburgh
- Wide-coverage statistical parser based on the CCG: CCG Parser
- Computational semantic tools named **Boxer**
- Useful links
  - http://svn.ask.it.usyd.edu.au/trac/candc
  - http://svn.ask.it.usyd.edu.au/trac/candc/wiki/Demo

CCG Parser [Clark, 2007]

Statistical parsing and CCG

Advantages of CCG

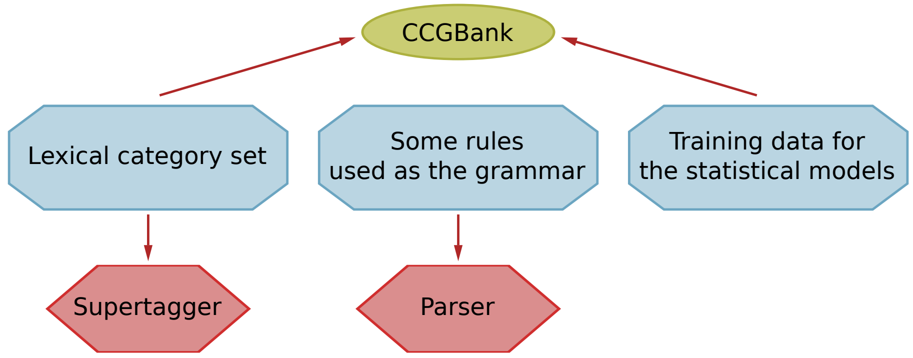
- providing a compositional semantic for the grammar

→completely transparent interface between syntax and semantics

- the recovery of long-range dependencies can be integrated into the parsing process in a straightforward manner

Parsing method

- Penn Treebank conversion : TAG, LFG, HPSG and CCG
- **CCGBank** [Hockenmaier and Steedman, 2007]
  - CCG version of the Penn Treebank
  - Grammar used in CCG parser



Parsing method-CCG Bank

- Corpus translated from the Penn Treebank, CCGBank contains
  - Syntactic derivations
  - Word-word dependencies
  - Predicate-argument structures

Parsing method-CCG Bank

- Semi automatic conversion of phrase-structure trees in the Penn Treebank into CCG derivations
- Consists mainly of newspaper texts
- Grammar:

Lexical category set

Combinatory rules

Unary type-changing rules

Normal-form constraints

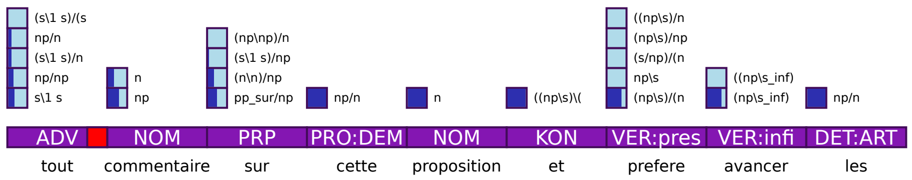
Punctuation rules

Parsing method

Supertagging [Clark, 2002]

uses conditional maximum entropy models

implement a maximum entropy supertagger



Parsing method-Supertagger

- Set of 425 lexical categories from the CCGbank
- The per-word accuracy of the Supertagger is around 92% on unseen WSJ text.

$\rightarrow$ Using the multi-supertagger increases the accuracy significantly – to over 98% – with only a small cost in increased ambiguity.

Parsing method-Supertagger

- **Log-linear models in NLP applications:**
  - POS tagging
  - Name entity recognition
  - Chunking
  - Parsing

$\rightarrow$ referred as **maximum entropy models** and **random fields**

Parsing method-Supertagger

Log-linear parsing models for CCG

1. the probability of a dependency structure
2. the normal-form model: the probability of a single derivation

$\rightarrow$ modeling 2) is simpler than 1)

1. defined as $P(\pi|S) = \sum_{d \in \Delta(\pi)} P(d, \pi|S)$
2. defined using a log-linear form as follows: $P(w|S) = \frac{1}{Z_S} e^{\lambda . f(w)}$

$$Z_S = \sum_{w \in p(S)} e^{\lambda . f(w')}$$

Parsing method-Supertagger

Features common to the dependency and normal-form models

Feature type	Example
LexCat + word	$(S/S)/NP$ + Before
LexCat + POS	$(S/S)/NP$ + IN
RootCat	$S[dcl]$
RootCat + World	$S[dcl]$ + was
RootCat + POS	$S[dcl]$ + VBD
Rule	$S[dcl] \rightarrow NP\ S[dcl]\backslash NP$
Rule + Word	$S[dcl] \rightarrow NP\ S[dcl]\backslash NP$ + bought
Rule + POS	$S[dcl] \rightarrow NP\ S[dcl]\backslash NP$ + VBD

Parsing method-Supertagger

Predicate-argument dependency features for the dependency model

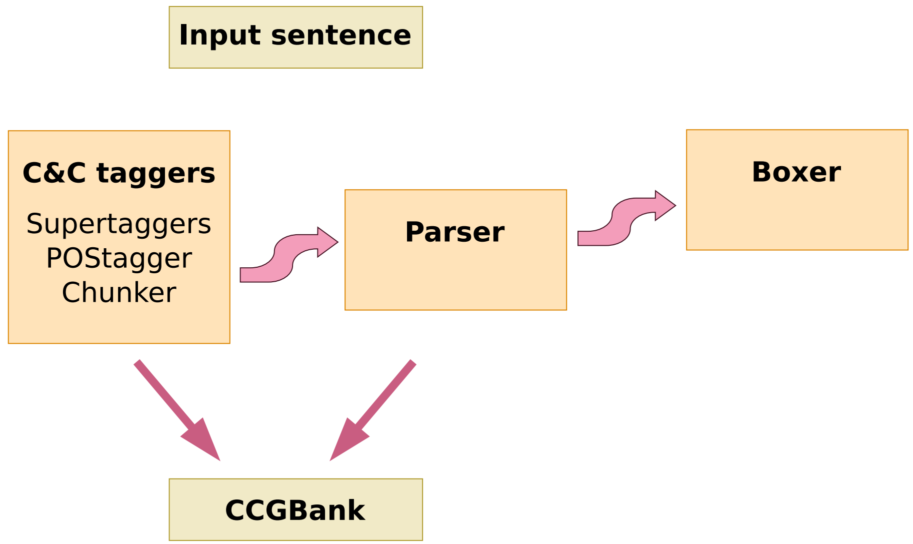
Feature type	Example
Word-Word	$\langle bought, (S\backslash NP_1)/NP_2, 2, stake, (NP\backslash NP)/(S[dcl]/NP)\rangle$
Word-POS	$\langle bought, (S\backslash NP_1)/NP_2, 2, NN, (NP\backslash NP)/(S[dcl]/NP)\rangle$
POS-Word	$\langle VBD, (S\backslash NP_1)/NP_2, 2, stake, (NP\backslash NP)/(S[dcl]/NP)\rangle$
POS-POS	$\langle VBD, (S\backslash NP_1)/NP_2, 2, NN, (NP\backslash NP)/(S[dcl]/NP)\rangle$
Word + Distance(words)	$\langle bought, (S\backslash NP_1)/NP_2, 2, (NP\backslash NP)/(S[dcl]/NP)\rangle + 2$
Word + Distance(punct)	$\langle bought, (S\backslash NP_1)/NP_2, 2, (NP\backslash NP)/(S[dcl]/NP)\rangle + 0$
Word + Distance(verbs)	$\langle bought, (S\backslash NP_1)/NP_2, 2, (NP\backslash NP)/(S[dcl]/NP)\rangle + 0$
POS + Distance(words)	$\langle VBD, (S\backslash NP_1)/NP_2, 2, (NP\backslash NP)/(S[dcl]/NP)\rangle + 2$
POS + Distance(punct)	$\langle VBD, (S\backslash NP_1)/NP_2, 2, (NP\backslash NP)/(S[dcl]/NP)\rangle + 0$
POS + Distance(verbs)	$\langle VBD, (S\backslash NP_1)/NP_2, 2, (NP\backslash NP)/(S[dcl]/NP)\rangle + 0$

Parsing method-Supertagger

Rule dependency features for the normal-form model

Feature type	Example
Word-Word	$\langle company, S[dcl] \rightarrow NP\ S[dcl]\backslash NP, bought\rangle$
Word-POS	$\langle company, S[dcl] \rightarrow NP\ S[dcl]\backslash NP, VBD\rangle$
POS-Word	$\langle NN, S[dcl] \rightarrow NP\ S[dcl]\backslash NP, bought\rangle$
POS-POS	$\langle NN, S[dcl] \rightarrow NP\ S[dcl]\backslash NP, VBD\rangle$
Word + Distance(words)	$\langle bought, S[dcl] \rightarrow NP\ S[dcl]\backslash NP\rangle + > 2$
Word + Distance(punct)	$\langle bought, S[dcl] \rightarrow NP\ S[dcl]\backslash NP\rangle + 2$
Word + Distance(verbs)	$\langle bought, S[dcl] \rightarrow NP\ S[dcl]\backslash NP\rangle + 0$
POS + Distance(words)	$\langle VBD, S[dcl] \rightarrow NP\ S[dcl]\backslash NP\rangle + > 2$
POS + Distance(punct)	$\langle VBD, S[dcl] \rightarrow NP\ S[dcl]\backslash NP\rangle + 2$
POS + Distance(verbs)	$\langle VBD, S[dcl] \rightarrow NP\ S[dcl]\backslash NP\rangle + 0$

Description of parser



Demonstration

http://svn.ask.it.usyd.edu.au/trac/candc/wiki/Demo

Results

Supertagger ambiguity and accuracy on section00

β	k	CATS/WORD	ACC	SENT ACC	ACC(POS)	SENT ACC
0.075	20	1.27	97.34	67.43	96.34	60.27
0.030	20	1.43	97.92	72.87	97.05	65.50
0.010	20	1.72	98.37	77.73	97.63	70.52
0.005	20	1.98	98.52	79.25	97.86	72.24
0.001	150	3.57	99.17	87.19	98.66	80.24

Results

Parsing accuracy on DepBank

	CCG parser			CCGbank			
Relation	Prec	Rec	F	Prec	Rec	F	# GRs
dependent	84.07	82.19	83.12	88.83	84.19	86.44	10,696
aux	95.03	90.75	92.84	96.47	90.33	93.30	400
conj	79.02	75.97	77.46	83.07	80.27	81.65	595
ta	51.52	11.64	18.99	62.07	12.59	20.93	292
det	95.23	94.97	95.10	97.27	94.09	95.66	1,114
arg_mod	81.46	81.76	81.61	86.75	84.19	85.45	8,295
mod	71.30	77.23	74.14	77.83	79.65	78.73	3,908
ncmod	73.36	78.96	76.05	78.88	80.64	79.75	3,550
xmod	42.67	53.93	47.64	56.54	60.67	58.54	178
cmod	51.34	57.14	54.08	64.77	69.09	66.86	168
pmod	0.00	0.00	0.00	0.00	0.00	0.00	12
arg	85.76	80.01	82.78	89.79	82.91	86.21	4,387

DepBank: Parc Dependency Bank
[King et al. 2003]

Results

subj_or_dobj	86.08	83.08	84.56	91.01	85.29	88.06	3,127
subj	84.08	75.57	79.60	89.07	78.43	83.41	1,363
nesubj	83.89	75.78	79.63	88.86	78.51	83.37	1,354
xsubj	0.00	0.00	0.00	50.00	28.57	36.36	7
csubj	0.00	0.00	0.00	0.00	0.00	0.00	2
comp	86.16	81.71	83.88	89.92	84.74	87.25	3,024
obj	86.30	83.08	84.66	90.42	85.52	87.90	2,328
dobj	87.01	88.44	87.71	92.11	90.32	91.21	1,764
obj2	68.42	65.00	66.67	66.67	60.00	63.16	20
iobj	83.22	65.63	73.38	83.59	69.81	76.08	544
clausal	77.67	72.47	74.98	80.35	77.54	78.92	672
xcomp	77.69	74.02	75.81	80.00	78.49	79.24	381
ccomp	77.27	70.10	73.51	80.81	76.31	78.49	291
pcomp	0.00	0.00	0.00	0.00	0.00	0.00	24
macroaverage	65.71	62.29	63.95	71.73	65.85	68.67	
microaverage	81.95	80.35	81.14	86.86	82.75	84.76	