A Comparison of Chinese Parsers for Stanford Dependencies

Wanxiang Che,[†] Valentin I. Spitkovsky[‡] and Ting Liu[†]

[†]Harbin Institute of Technology
[‡]Stanford University

ACL 2012

July 11, 2012

Outline

Outline

Stanford Dependencies

- A simple description of relations between pairs of words in a sentence
- A kind of semantically-oriented dependency representation
- Converted from constituent trees by rules
- 53 binary relations for English, 46 for Chinese

Stanford Dependencies

- A simple description of relations between pairs of words in a sentence
- A kind of semantically-oriented dependency representation
- Converted from constituent trees by rules
- 53 binary relations for English, 46 for Chinese

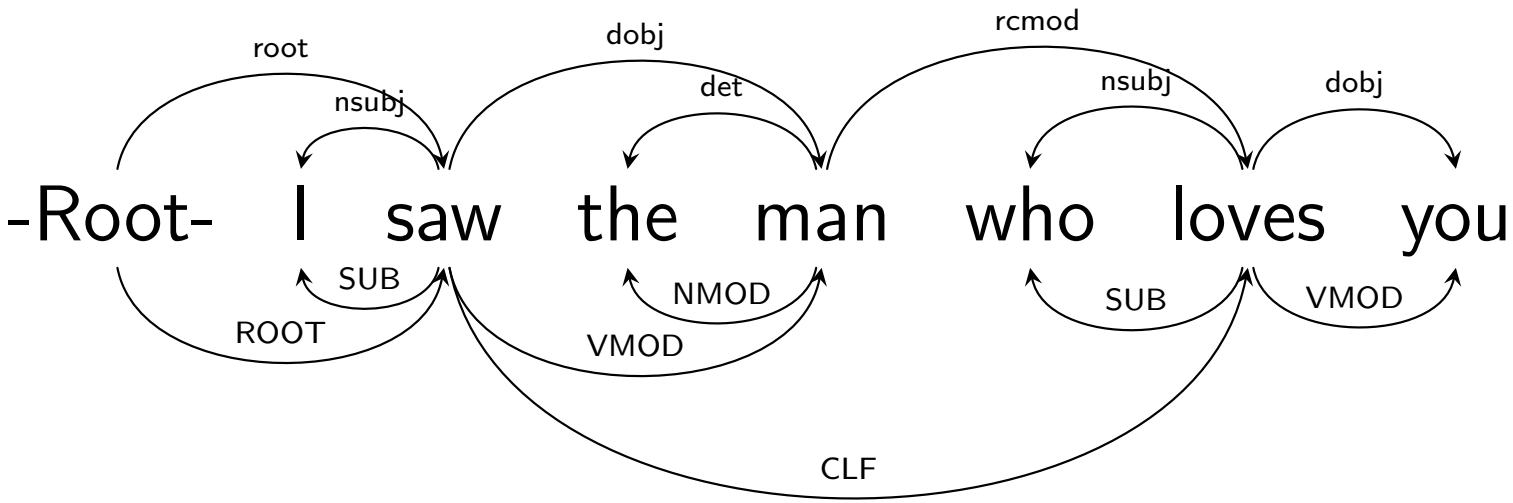


Figure: Stanford dependencies (above) vs. CoNLL style (below)

Stanford Dependencies Applications

- Intuitive and easy to apply, requires little linguistic expertise
  - Biomedical text mining (Kim et al., 2009)
  - Textual entailment (Androutsopoulos and Malakasiotis, 2010)
  - Information extraction (Wu and Weld, 2010; Banko et al., 2007)
  - Sentiment analysis (Meena and Prabhakar, 2007; Wu et al., 2011)

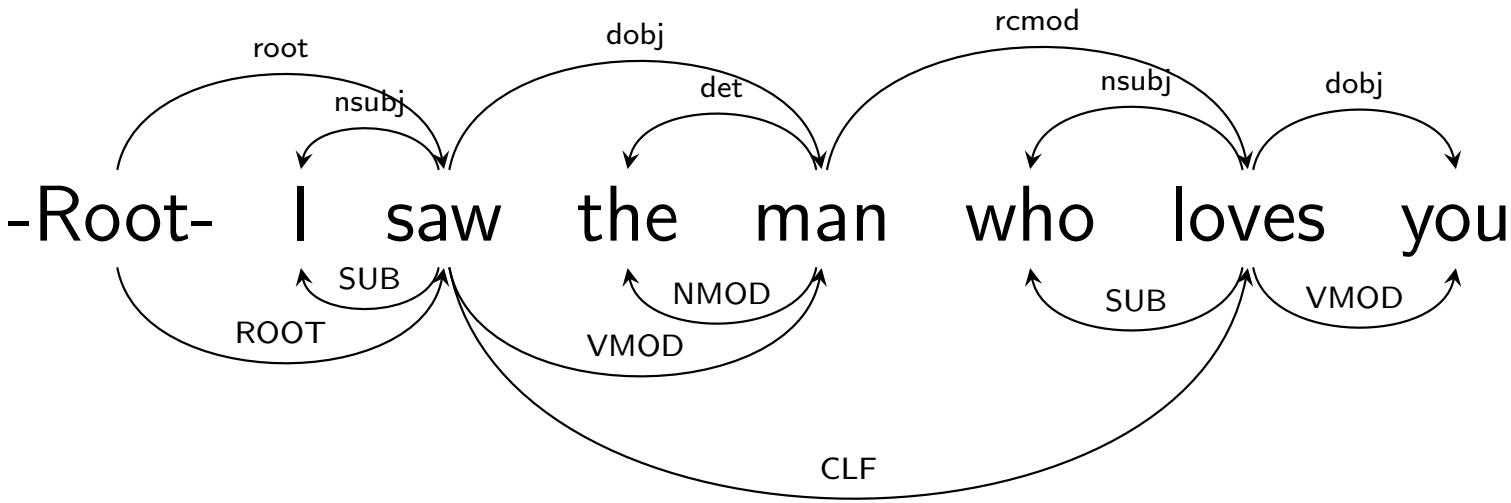


Figure: Stanford dependencies (above) vs. CoNLL style (below)

Parsing Methods

Parsing Methods

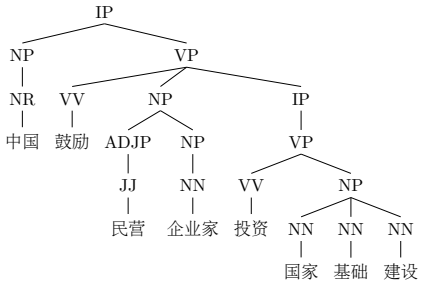
- Constituent Parsing (indirect)

Parsing Methods

- Constituent Parsing (indirect)
  - *Sentence*

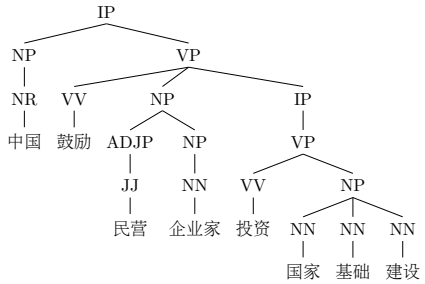
Parsing Methods

- Constituent Parsing (indirect)
  - *Sentence* $\Rightarrow$

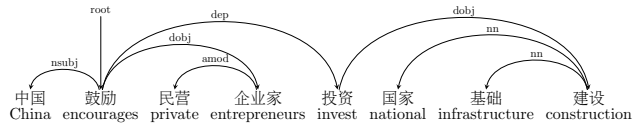


Parsing Methods

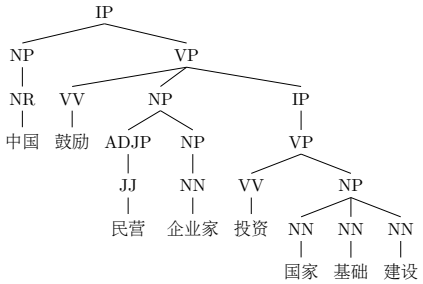
- Constituent Parsing (indirect)
  - *Sentence* $\Rightarrow$

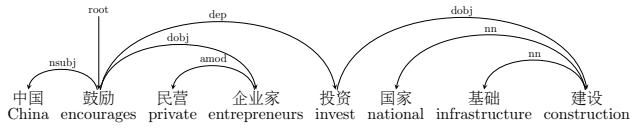


$\Rightarrow$



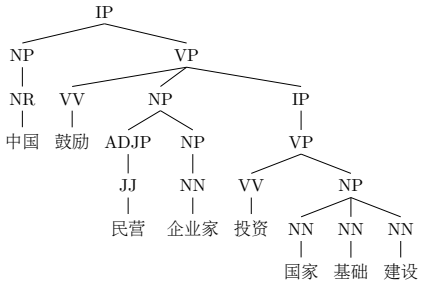
Parsing Methods

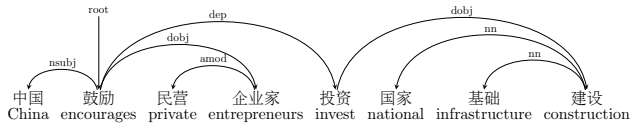
- Constituent Parsing (indirect)
  - *Sentence* $\Rightarrow$ 

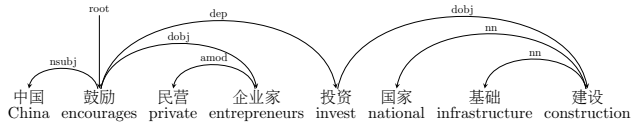
$\Rightarrow$ 

  - Stanford dependency parser's original implementation

Parsing Methods

- Constituent Parsing (indirect)
  - *Sentence* ⇒ 

⇒ 

  - Stanford dependency parser's original implementation
- Dependency Parsing (direct)
  - *Sentence* ⇒ 

Motivation

- Which method is better for Chinese Stanford Dependencies?

Motivation

- Which method is better for Chinese Stanford Dependencies?
- Comparison for English (Cer et al., 2010)

Motivation

- Which method is better for Chinese Stanford Dependencies?
- Comparison for English (Cer et al., 2010)
  - Constituent parsers systematically outperform direct methods

Motivation

- Which method is better for Chinese Stanford Dependencies?
- Comparison for English (Cer et al., 2010)
  - Constituent parsers systematically outperform direct methods
  - Did not explore more sophisticated (higher-order) dependency parsers
  - Did not explore more consistent (*n*-way jackknifing of) POS tags
  - Small bug in evaluation of MSTParser

Outline

Parsers Information

Open Source Parsers

Type	*Parser*	*Version*	*Algorithm*
Constituent	Berkeley	1.1	PCFG
	Bikel	1.2	PCFG
	Charniak	Nov. 2009	PCFG
	Stanford	2.0	Factored
Dependency	MaltParser	1.6.1	Arc-Eager
	Mate	2.0	2nd-order MST
	MSTParser	0.5	MST

Settings

Corpus

- Latest Chinese TreeBank (CTB) 7.0

Number of \ in	*Train*	*Dev*	*Test*	*Total*
files	2,083	160	205	2,448
sentences	46,572	2,079	2,796	**51,447**
tokens	1,039,942	59,955	81,578	1,181,475

Settings

Corpus

- Latest Chinese TreeBank (CTB) 7.0

Number of \in	*Train*	*Dev*	*Test*	*Total*
files	2,083	160	205	2,448
sentences	46,572	2,079	2,796	**51,447**
tokens	1,039,942	59,955	81,578	1,181,475

Software and Hardware

- Parsers: all default options
- Hardware: Intel's Xeon E5620 2.40GHz CPU and 24GB RAM

Features for Dependency Parsers

POS tags

- Stanford POS tagger
- Automatic tags for training data (via 10-way jackknifing)

Features for Dependency Parsers

POS tags

- Stanford POS tagger
- Automatic tags for training data (via 10-way jackknifing)

Lemmas

- The last character of each Chinese word
  - E.g., *bicycle* (自行车), *car* (汽车) and *train* (火车) are all various kinds of *vehicle* (车)

Outline

Chinese Results

		Dev		*Test*		
Type	*Parser*	UAS	LAS	UAS	LAS	*Time*
Constituent	Berkeley	*82.0*	*77.0*	*82.9*	*77.8*	*45:56*
	Bikel	79.4	74.1	80.0	74.3	*6,861:31*
	Charniak	77.8	71.7	78.3	72.3	128:04
	Stanford	*76.9*	*71.2*	*77.3*	*71.4*	330:50
Dependency	MaltParser (*liblinear*)	*76.0*	*71.2*	*76.3*	*71.2*	**0:11**
	MaltParser (*libsvm*)	77.3	72.7	78.0	73.1	*556:51*
	Mate (2nd-order)	**82.8**	**78.2**	**83.1**	**78.1**	87:19
	MSTParser (1st-order)	78.8	73.4	78.9	73.1	12:17

Bold: best results.

Dark Red: worst results.

Blue: best results of constituent parsers.

Outline

Comparison between Mate and Berkeley parsers

- Mate is slightly better than Berkeley (but not significantly, $p > 0.05$)

Comparison between Mate and Berkeley parsers

- Mate is slightly better than Berkeley (but not significantly, $p > 0.05$)
- Performance (F_1) comparison on different relations

Relation	*Count*	Mate	Berkeley
nn	7,783	**91.3**	89.3
dep	4,651	69.4	**70.3**
nsubj	4,531	**87.1**	85.5
advmod	4,028	**94.3**	93.8
dobj	3,990	**86.0**	85.0
conj	2,159	**76.0**	75.8
prep	2,091	**94.3**	94.1
root	2,079	81.2	**82.3**
nummod	1,614	**97.4**	96.7
assmod	1,593	**86.3**	84.1

More Analysis

Feature Effect

- 10-way jackknifing POS tags for training data

	Gold	Jackknifing
Mate	75.4	**78.2**
Berkeley	**77.0**	76.5

More Analysis

Feature Effect

- 10-way jackknifing POS tags for training data

	Gold	Jackknifing
Mate	75.4	**78.2**
Berkeley	**77.0**	76.5

- Lemmas for Mate
  - 77.8 (w/o) vs. **78.2** (with)

More Analysis

Feature Effect

- 10-way jackknifing POS tags for training data

	Gold	Jackknifing
Mate	75.4	**78.2**
Berkeley	**77.0**	76.5

- Lemmas for Mate
  - 77.8 (w/o) vs. **78.2** (with)

English vs. Chinese

	Chinese	English
Berkeley	77.0	87.9
Charniak	71.7	87.8
CJ (Charniak + Reranking)	—	**89.1**
Mate	**78.2**	88.6

Outline

Conclusion

- For Chinese, direct approach comparable to using constituents

Conclusion

- For Chinese, direct approach comparable to using constituents
- Which parser to use in practice?

Conclusion

- For Chinese, direct approach comparable to using constituents
- Which parser to use in practice?
  - Most accurate: Mate parser
  - Fastest: MaltParser (*liblinear*)
  - Trade-off: Berkeley parser

Conclusion

- For Chinese, direct approach comparable to using constituents
- Which parser to use in practice?
  - Most accurate: Mate parser
  - Fastest: MaltParser (*liblinear*)
  - Trade-off: Berkeley parser
- We prefer dependency parsers which more easily admit richer features

Conclusion

- For Chinese, direct approach comparable to using constituents
- Which parser to use in practice?
  - Most accurate: Mate parser
  - Fastest: MaltParser (*liblinear*)
  - Trade-off: Berkeley parser
- We prefer dependency parsers which more easily admit richer features
  - *n*-way jackknifing of POS tags and lemma features can help

Thanks and QA