

# Dependency Parsing

Lilja Øvrelid  
INF5830  
Fall 2015

With thanks to Sandra Kübler and Joakim Nivre

## Why?

- ▶ Increasing interest in dependency-based approaches to syntactic parsing in recent years
  - ▶ New methods emerging
  - ▶ Applied to a wide range of languages
  - ▶ CoNLL shared tasks (2006, 2007)

# What?

- ▶ Computational methods for dependency-based parsing
  - ▶ Syntactic representations
  - ▶ Parsing algorithms
  - ▶ Machine learning
- ▶ Available resources for different languages
  - ▶ Parsers
  - ▶ Treebanks

## Syntactic parsing

- ▶ automatically determining the syntactic structure for a given sentence
- ▶ Traditionally (for phrase-structure grammars):
  - ▶ search through all possible trees for a sentence
  - ▶ bottom-up vs top-down approaches

# Ambiguities

- ▶ more than one possible structure for a sentence
- ▶ natural languages are hugely ambiguous
- ▶ a very common problem

PoS-ambiguities				Attachment ambiguities		
		VB				
	VBZ	VBP	VBZ			
NNP	NNS	NN	NNS	CD	NN	
Fed	raises	interest	rates	0.5	%	in effort to control inflation

## Back in the days (90s)

- ▶ Parsers assigned linguistically detailed syntactic structures (based on linguistic theories)
- ▶ Grammar-driven parsing: possible trees defined by the grammar
- ▶ Problems with **coverage**
  - ▶ only around 70% of all sentences were assigned an analysis
- ▶ Most sentences were assigned very many analyses by a grammar
  - ▶ no way of choosing between them

## Enter data-driven (statistical) parsing

- ▶ Today data-driven/statistical parsing is available for a range of languages and syntactic frameworks
- ▶ Data-driven approaches: possible trees defined by the treebank (may also involve a grammar)
- ▶ Produce one analysis (hopefully the most likely one) for any sentence
- ▶ And get most of them correct
- ▶ Still an active field of research, improvements are still possible!

## Statistics in parsing

- ▶ classical NLP parsing:
  - ▶ symbolic grammar and lexicon
  - ▶ proof systems to prove parses from words
- ▶ ambiguity problem is very large
  - ▶ minimal grammar on previous sentence: 36 parses
  - ▶ large broad-coverage grammar: millions of parses
- ▶ use probabilities to pick the most likely parse

# Treebanks

- ▶ need data to estimate probabilities
- ▶ collection of sentences manually annotated with the correct parse  $\Rightarrow$  a **treebank**
- ▶ Penn Treebank: treebanks from Brown, Switchboard, ATIS og *Wall Street Journal* corpora
- ▶ Treebanks for other languages
  - ▶ Prague Dependency Treebank (czech)
  - ▶ Negra/Tuba-DZ (German)
  - ▶ Penn (Chinese)
  - ▶ Norwegian Dependency Treebank
  - ▶ the CoNLL treebanks (Project A)

# Text parsing

- ▶ Goal: parse unrestricted text in natural language
  - ▶ Given a text  $T = (x_1, \dots, x_n)$  in language  $L$ , derive the correct analysis for every sentence  $x_i \in T$ .
- ▶ Challenges:
  - ▶ robustness: at least one analysis
  - ▶ disambiguation: at most one analysis
  - ▶ accuracy: correct analysis (for every sentence)
  - ▶ efficiency: reasonable time-and memory usage
- ▶ Two different methodological strategies
  - ▶ grammar-driven
  - ▶ data-driven

## Grammar-driven parsing

- ▶ A formal grammar  $G$  defines
  - ▶ the language  $L(G)$  that can be parsed
  - ▶ the class of analyses returned by the parser
- ▶ robustness (analyze any input sentence)
  - ▶ some input sentences  $x_i$  are not in  $L(G)$
  - ▶ constraint relaxation, partial parsing
- ▶ disambiguation
  - ▶ number of analyses assigned by grammar may be very large
  - ▶ probabilistic extensions, e.g. PCFG
- ▶ accuracy: assumed advantage, but requires joint optimization of robustness and disambiguation

## Data-driven parsing

1. formal model  $M$  defining possible analyses for sentences in  $L$
  2. A sample of annotated text  $S = (x_1, \dots, x_m)$  from  $L$
  3. An inductive inference scheme  $I$  defining actual analyses for the sentences of a text  $T = (x_1, \dots, x_n)$  in  $L$ , relative to  $M$  and  $S$ .
- ▶  $S$  is the **training data**: contains representations satisfying  $M$
  - ▶ a **treebank**: manually annotated with correct analysis
  - ▶  $I$  based on **supervised** machine learning

## Data-driven parsing

- ▶ robustness: depends on  $M$  and  $I$ , but usually designed such that any input string is assigned at least one analysis.
- ▶ disambiguation: severe problem, solved by inductive inference scheme
- ▶ improved accuracy represents main challenge
- ▶ efficiency: variation

## Data-driven dependency parsing

- ▶  $M$  defined by formal conditions on dependency graphs (labeled directed graphs that are):
  - ▶ connected
  - ▶ acyclic
  - ▶ single-head
  - ▶ (projective)
- ▶  $I$  may be defined in different ways
  - ▶ parsing method (deterministic, non-deterministic)
  - ▶ machine learning algorithm, feature representations
- ▶ Two main approaches: **graph-based** and **transition-based** models [McDonald and Nivre 2007]

## Graph-based approaches

- ▶ Basic idea:
  - ▶ define a space of candidate dependency graphs for a sentence
  - ▶ **Learning**: induce a model for scoring an entire dependency graph for a sentence
  - ▶ **Parsing**: Find the highest scoring dependency graph, given the induced model
- ▶ Characteristics:
  - ▶ global training
  - ▶ exhaustive search

## Transition-based approaches

- ▶ Basic idea:
  - ▶ define a transition system for mapping a sentence to its dependency graph
  - ▶ **Learning**: induce a model for predicting the next state transition, given the transition history
  - ▶ **Parsing**: Construct the optimal transition sequence, given the induced model
- ▶ Characteristics:
  - ▶ local training
  - ▶ greedy search



# MSTParser: Maximum Spanning Trees

[McDonald et al. 2005a, McDonald et al. 2005b]

- ▶ Score of a dependency tree = sum of scores of dependencies
- ▶ Scores are independent of other dependencies.
- ▶ Finding the highest scoring dependency tree = finding the maximum spanning tree (MST) in a graph containing all possible graphs
- ▶ Two cases:
  - ▶ Projective: Use Eisner's parsing algorithm.
  - ▶ Non-projective: Use Chu-Liu-Edmonds algorithm for finding the maximum spanning tree in a directed graph [Chu and Liu 1965, Edmonds 1967].
- ▶ Use machine learning for determining weight vector  $\mathbf{w}$ : large-margin multi-class classification (MIRA)

# MaltParser: transition-based dependency parsing

- ▶ MaltParser is a language-independent system for data-driven dependency parsing which is freely available
- ▶ It is based on a **deterministic** parsing strategy in combination with treebank-induced **classifiers** for predicting parsing actions
- ▶ MaltParser employs a rich feature history in order to guide parsing
- ▶ May easily be extended to take into account new features of the parse history

# MaltParser

- ▶ Parsing as a set of transitions between parse configurations
- ▶ A parse configuration is a triple  $\langle S, I, G \rangle$ , where
  - ▶  $S$  represents the parse stack – a list of tokens which are candidates for dependency arcs
  - ▶  $I$  is the queue of remaining input tokens
  - ▶  $G$  represents the dependency graph under construction
- ▶ The parse *guide* predicts the next parse action (transition), based on the current parse configuration
- ▶ The guide is trained employing discriminative machine learning
- ▶ Recasts the learning problem as a classification problem: given a parse configuration, predict the next transition

# Deterministic Parsing

- ▶ Basic idea:
  - ▶ Derive a single syntactic representation (dependency graph) through a deterministic sequence of elementary parsing actions
  - ▶ Sometimes combined with backtracking or repair
- ▶ Motivation:
  - ▶ Psycholinguistic modeling
  - ▶ Efficiency
  - ▶ Simplicity

## Shift-Reduce Type Algorithms

- ▶ Data structures:
  - ▶ Stack  $[\dots, w_i]_S$  of partially processed tokens
  - ▶ Queue  $[w_j, \dots]_Q$  of remaining input tokens
- ▶ Parsing actions built from atomic actions:
  - ▶ Adding arcs ( $w_i \rightarrow w_j, w_i \leftarrow w_j$ )
  - ▶ Stack and queue operations
- ▶ Restricted to **projective** dependency graphs

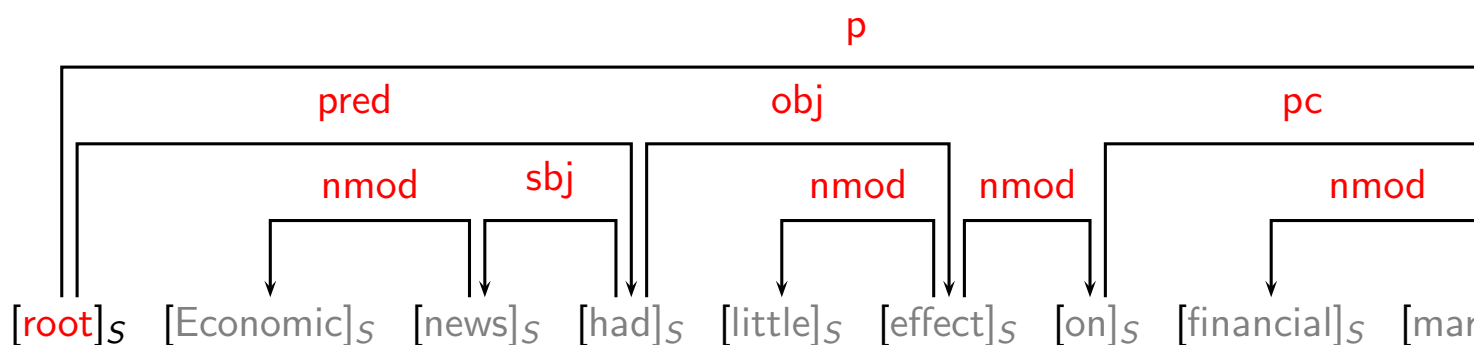
## Nivre's Algorithm

- ▶ Four parsing actions:

$$\begin{array}{l}
 \text{Shift} \quad \frac{[\dots]_S \quad [w_i, \dots]_Q}{[\dots, w_i]_S \quad [\dots]_Q} \\
 \text{Reduce} \quad \frac{[\dots, w_i]_S \quad [\dots]_Q \quad \exists w_k : w_k \rightarrow w_i}{[\dots]_S \quad [\dots]_Q} \\
 \text{Left-Arc}_r \quad \frac{[\dots, w_i]_S \quad [w_j, \dots]_Q \quad \neg \exists w_k : w_k \rightarrow w_i}{[\dots]_S \quad [w_j, \dots]_Q \quad w_i \xleftarrow{r} w_j} \\
 \text{Right-Arc}_r \quad \frac{[\dots, w_i]_S \quad [w_j, \dots]_Q \quad \neg \exists w_k : w_k \rightarrow w_j}{[\dots, w_i, w_j]_S \quad [\dots]_Q \quad w_i \xrightarrow{r} w_j}
 \end{array}$$

- ▶ Characteristics:
  - ▶ Integrated labeled dependency parsing
  - ▶ Arc-eager processing of right-dependents

## Example



Shift Left-Arc<sub>nmod</sub> Shift Left-Arc<sub>subj</sub> Right-Arc<sub>pred</sub> Shift Left-Arc<sub>nmod</sub>  
 Right-Arc<sub>obj</sub> Right-Arc<sub>nmod</sub> Shift Left-Arc<sub>nmod</sub> Right-Arc<sub>pc</sub> Reduce  
 Reduce Reduce Reduce Right-Arc<sub>p</sub>

## Classifier-Based Parsing

- ▶ Data-driven deterministic parsing:
  - ▶ Deterministic parsing requires an **oracle**.
  - ▶ An oracle can be approximated by a **classifier**.
  - ▶ A classifier can be trained using **treebank** data.
- ▶ Learning methods:
  - ▶ Support vector machines (SVM)  
[Kudo and Matsumoto 2002, Yamada and Matsumoto 2003, Isozaki et al. 2004, Cheng et al. 2004, Nivre et al. 2006]
  - ▶ Memory-based learning (MBL)  
[Nivre et al. 2004, Nivre and Scholz 2004]
  - ▶ Maximum entropy modeling (MaxEnt)  
[Cheng et al. 2005]

## Feature Models

- ▶ Learning problem:
  - ▶ Approximate a function from **parser configurations**, represented by feature vectors to **parser actions**, given a training set of gold standard derivations.
- ▶ Typical features:
  - ▶ Tokens:
    - ▶ Target tokens
    - ▶ Linear context (neighbors in  $S$  and  $Q$ )
    - ▶ Structural context (parents, children, siblings in  $G$ )
  - ▶ Attributes:
    - ▶ Word form (and lemma)
    - ▶ Part-of-speech (and morpho-syntactic features)
    - ▶ Dependency type (if labeled)
    - ▶ Distance (between target tokens)

## Feature Models

- ▶ Parse configurations are represented by a set of features, which focus on attributes of the *top* of the stack, the *next* input token and neighboring tokens in the stack, input queue and dependency graph

	form	pos	dep
$S:top$	+	+	+
$I:next$	+	+	
$G:head\ of\ top$	+		
$G:leftmost\ dependent\ of\ top$			+

## Non-Projective Dependency Parsing

- ▶ Many parsing algorithms are restricted to projective dependency graphs.
- ▶ Is this a problem?
- ▶ Statistics from CoNLL-X Shared Task [Buchholz and Marsi 2006]
  - ▶ NPD = Non-projective dependencies
  - ▶ NPS = Non-projective sentences

Language	%NPD	%NPS
Dutch	5.4	36.4
German	2.3	27.8
Czech	1.9	23.2
Slovene	1.9	22.2
Portuguese	1.3	18.9
Danish	1.0	15.6

## Two Main Approaches

- ▶ Algorithms for non-projective dependency parsing:
  - ▶ McDonald's spanning tree algorithm [McDonald et al. 2005b]
  - ▶ Covington's algorithm [Nivre 2006]
- ▶ Post-processing of projective dependency graphs:
  - ▶ Pseudo-projective parsing [Nivre and Nilsson 2005]

# Non-Projective Parsing Algorithms

► Complexity considerations:

- ▶ Projective (Proj)
- ▶ Non-projective (NonP)

Problem/Algorithm	Proj	NonP
Deterministic parsing [Nivre 2003, Covington 2001]	$O(n)$	$O(n^2)$
First order spanning tree [McDonald et al. 2005b]	$O(n^3)$	$O(n^2)$

## Post-Processing

► Two-step approach:

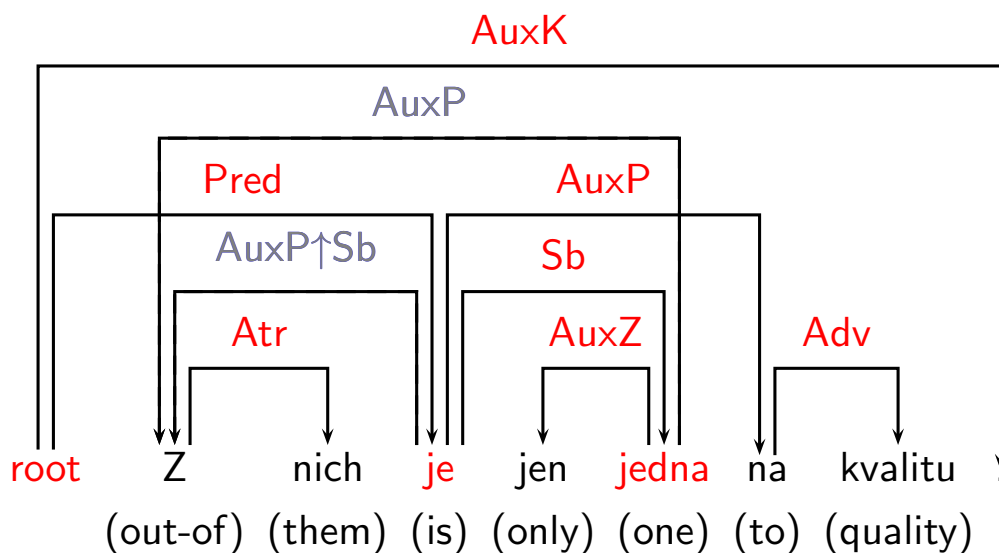
1. Derive the best projective approximation of the correct (possibly) non-projective dependency graph.
2. Improve the approximation by replacing projective arcs by (possibly) non-projective arcs.

► Rationale:

- ▶ Most “naturally occurring” dependency graphs are primarily projective, with only a few non-projective arcs.

## Pseudo-Projective Parsing

- ▶ Projectivize training data:
  - ▶ Projective head nearest permissible ancestor of real head
  - ▶ Arc label extended with dependency type of real head



## Pros and Cons of Dependency Parsing

- ▶ What are the advantages of dependency-based methods?
- ▶ What are the disadvantages?
- ▶ Four types of considerations:
  - ▶ Complexity
  - ▶ Transparency
  - ▶ Word order
  - ▶ Expressivity

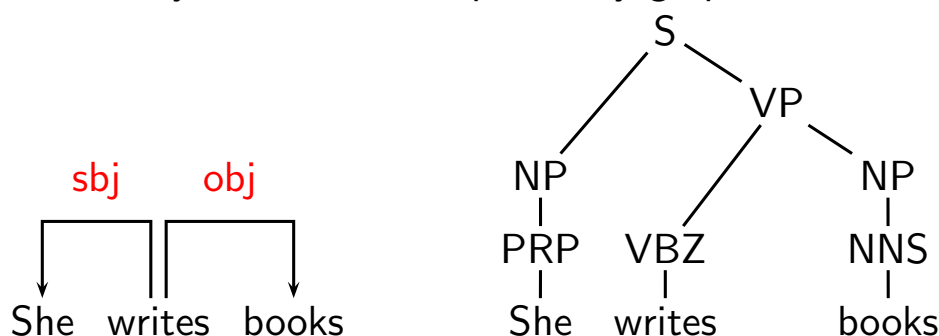


# Complexity

- ▶ Practical complexity:
  - ▶ Given the **Single-Head** constraint, parsing a sentence  $x = w_1, \dots, w_n$  can be reduced to labeling each token  $w_i$  with:
    - ▶ a **head word**  $h_i$ ,
    - ▶ a **dependency type**  $d_i$ .
- ▶ Theoretical complexity:
  - ▶ By exploiting the special properties of dependency graphs, it is sometimes possible to improve worst-case complexity compared to constituency-based parsing

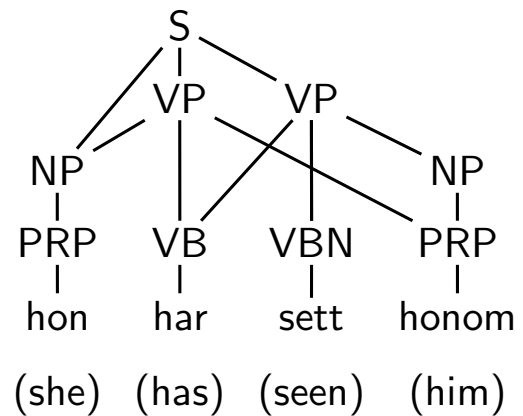
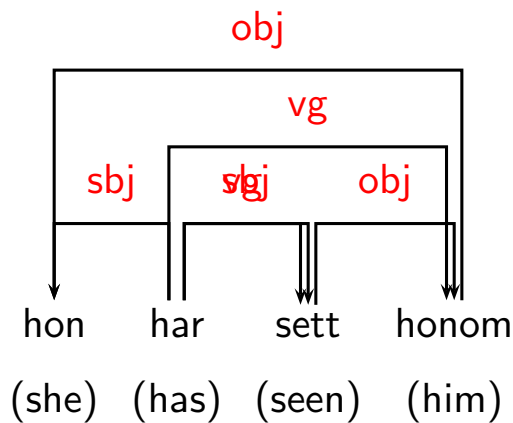
# Transparency

- ▶ Direct encoding of predicate-argument structure
- ▶ Fragments directly interpretable
- ▶ **But** only with **labeled** dependency graphs



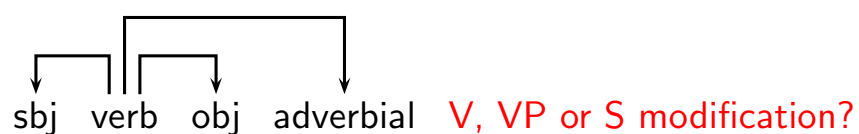
## Word Order

- ▶ Dependency structure independent of word order
- ▶ Suitable for free word order languages (cf. German results)
- ▶ **But** only with **non-projective** dependency graphs



## Expressivity

- ▶ Limited expressivity:
  - ▶ Every projective dependency grammar has a strongly equivalent context-free grammar, but not vice versa [Gaifman 1965].
  - ▶ Impossible to distinguish between phrase modification and head modification in unlabeled dependency structure [Mel'čuk 1988].



## Practical Issues

- ▶ Where to get the software?
  - ▶ Dependency parsers
  - ▶ Conversion programs for constituent-based treebanks
- ▶ Where to get the data?
  - ▶ Dependency treebanks
  - ▶ Treebanks that can be converted into dependency representation
- ▶ How to evaluate dependency parsing?
  - ▶ Evaluation scores

## Parsers

- ▶ Trainable parsers
  
- ▶ Concentrate on freely available parsers

# Trainable Parsers

- ▶ Ryan McDonald's **MSTParser**
  - ▶ Based on the algorithms of [McDonald et al. 2005a, McDonald et al. 2005b]
  - ▶ URL: [sourceforge.net/projects/mstparser](http://sourceforge.net/projects/mstparser)
  - ▶ Written in JAVA

# Trainable Parsers (2)

- ▶ Joakim Nivre's **MaltParser**
  - ▶ Inductive dependency parser with memory-based learning and SVMs
  - ▶ URL: <http://maltparser.org>
  - ▶ Executable versions are available for Solaris, Linux, Windows, and MacOS, open source
  - ▶ Written in JAVA

## Trainable Parsers (3)

- ▶ Many others
  - ▶ **Mate**: <https://code.google.com/p/mate-tools/>
  - ▶ **Turbo**: <http://www.cs.cmu.edu/~ark/TurboParser/>
  - ▶ **Spacy**: <http://spacy.io/>

## Trebanks

- ▶ Genuine dependency treebanks
- ▶ Treebanks for which conversions to dependencies exist
  
- ▶ See also CoNLL-X Shared Task  
URL: <http://nextens.uvt.nl/~conll/>
  
- ▶ Conversion strategy from constituents to dependencies

## Dependency Treebanks

- ▶ Arabic: Prague Arabic Dependency Treebank
- ▶ Czech: Prague Dependency Treebank
- ▶ Danish: Danish Dependency Treebank
- ▶ Portuguese: Bosque: Floresta sintá(c)tica
- ▶ Slovene: Slovene Dependency Treebank
- ▶ Turkish: METU-Sabancı Turkish Treebank

## Dependency Treebanks (2)

- ▶ Norwegian Dependency Treebank
  - ▶ Around 300 000 tokens of Bokmål and 300 000 tokens of Nynorsk, released in 2014
  - ▶ Freely downloadable (Språkbanken, Nasjonalbiblioteket)

## Constituent Treebanks

- ▶ English: Penn Treebank
- ▶ Bulgarian: BulTreebank
- ▶ Chinese: Penn Chinese Treebank, Sinica Treebank
- ▶ Dutch: Alpino Treebank for Dutch
- ▶ German: TIGER/NEGRA, TüBa-D/Z
- ▶ Japanese: TüBa-J/S
- ▶ Spanish: Cast3LB
- ▶ Swedish: Talbanken05

Conversions to dependency structures exist for all of these

## Conversion from Constituents to Dependencies

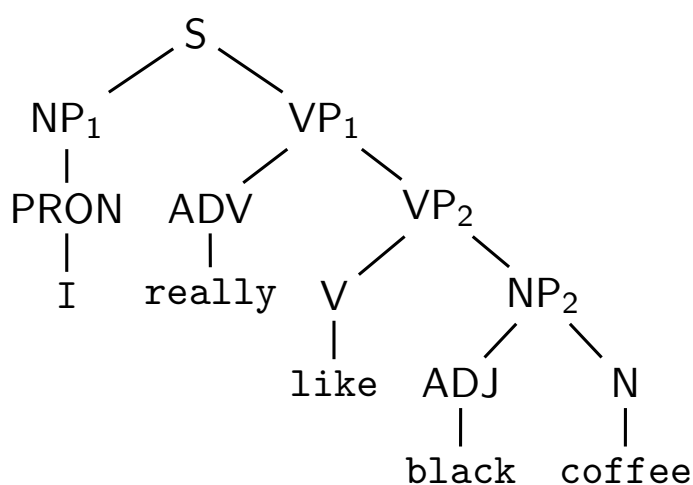
- ▶ Conversion from constituents to dependencies is possible
- ▶ Needs head/non-head information
- ▶ If no such information is given  $\Rightarrow$  heuristics
- ▶ Conversion for Penn Treebank to dependencies: e.g., Magerman, Collins, Lin, Yamada and Matsumoto ...
- ▶ Conversion restricted to structural conversion, no labeling
- ▶ Concentrate on Lin's conversion: [Lin 1995, Lin 1998]

## Lin's Conversion

- ▶ Idea: Head of a phrase governs all sisters.
- ▶ Uses **Tree Head Table**: List of rules where to find the head of a constituent.
- ▶ An entry consists of the node, the direction of search, and the list of possible heads.
- ▶ Sample entries:
  - (S right-to-left (Aux VP NP AP PP))
  - (VP left-to-right (V VP))
  - (NP right-to-left (Pron N NP))
- ▶ First line: The head of an S constituent is the first Aux daughter from the right; if there is no Aux, then the first VP, etc.

## Lin's Conversion - Example

- (S right-to-left (Aux VP NP AP PP))
- (VP left-to-right (V VP))
- (NP right-to-left (Pron N NP))

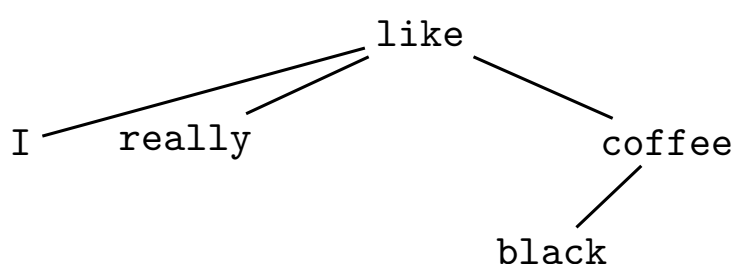


root	head	lex. head
S	VP <sub>1</sub>	??
VP <sub>1</sub>	VP <sub>2</sub>	??
S	VP <sub>1</sub>	like
VP <sub>1</sub>	VP <sub>2</sub>	like
VP <sub>2</sub>	V	like



## Lin's Conversion - Example (2)

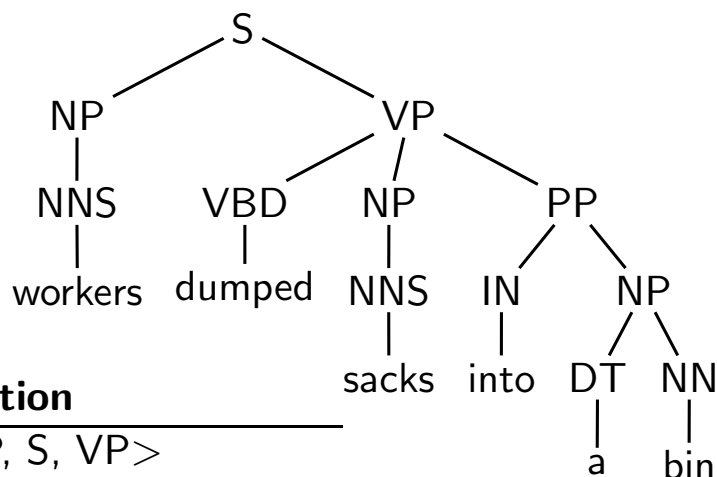
- ▶ The head of a phrase dominates all sisters.
- ▶  $VP_1$  governs  $NP_1 \Rightarrow$  *like* governs *I*
- ▶  $VP_2$  governs  $ADV \Rightarrow$  *like* governs *really*



## From Structural to Labeled Conversion

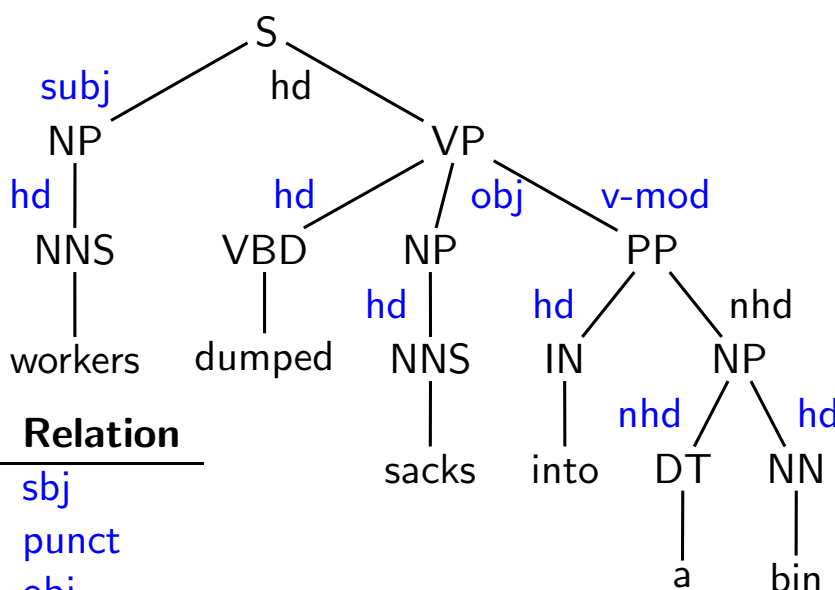
- ▶ Conversion so far gives only pure dependencies from head to dependent.
- ▶ Collins uses combination of constituent labels to label relation [Collins 1999]:
  - ▶ Idea: Combination of mother node and two subordinate nodes gives information about grammatical functions.
  - ▶ If  $headword(Y_h) \rightarrow headword(Y_d)$  is derived from rule  $X \rightarrow Y_1 \dots Y_n$ , the relation is  $\langle Y_d, X, Y_h \rangle$

# Collins' Example



Dependency	Relation
dumped → workers	<NP, S, VP>
dumped → root	<S, START, START>
dumped → sacks	<NP, VP, VBD>
dumped → into	<PP, VP, VBD>
into → bin	<NP, PP, IN>
bin → a	<DT, NP, NN>

# Example with Grammatical Functions



Dependency	Relation
dumped → workers	subj
dumped → root	punct
dumped → sacks	obj
dumped → into	v-mod
into → bin	nhd
bin → a	nhd

# Evaluation

- ▶ Internal evaluation: compare **accuracy** of model output to gold standard
- ▶ External evaluation (task-based evaluation):
  - ▶ quantify whether model output improves performance on a dependent task

## Evaluation: data-driven dependency parsing

evaluation scores:

- ▶ *Attachment score* percentage of words that have the correct head (and label)
- ▶ Labeled and unlabeled
- ▶ For single dependency types (labels):
  - ▶ *Precision*
  - ▶ *Recall*
  - ▶ *F measure*

## Part I: Data-driven dependency parsing

- ▶ Dependency grammar (last Monday)
- ▶ Dependency parsing (today)
- ▶ Project A released today
- ▶ Experimental methodology (Thursday)
- ▶ Project A (written report due **Oct. 23rd**):
  - ▶ training and evaluation of parsers for several languages
  - ▶ CoNLL-X (2006, 2007)
  - ▶ MaltParser: freely available software for data-driven dependency parsing

- ▶ Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*.
- ▶ Yuchang Cheng, Masayuki Asahara, and Yuji Matsumoto. 2004. Deterministic dependency structure analyzer for Chinese. In *Proceedings of the First International Joint Conference on Natural Language Processing (IJCNLP)*, pages 500–508.
- ▶ Yuchang Cheng, Masayuki Asahara, and Yuji Matsumoto. 2005. Machine learning-based dependency analyzer for Chinese. In *Proceedings of International Conference on Chinese Computing (ICCC)*, pages ?–?
- ▶ Y. J. Chu and T. J. Liu. 1965. On the shortest arborescence of a directed graph. *Science Sinica*, 14:1396–1400.
- ▶ Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.
- ▶ Michael A. Covington. 2001. A fundamental algorithm for dependency parsing. In *Proceedings of the 39th Annual ACM Southeast Conference*, pages 95–102.
- ▶ J. Edmonds. 1967.

- Optimum branchings. *Journal of Research of the National Bureau of Standards*, 71B:233–240.
- ▶ Haim Gaifman. 1965.  
Dependency systems and phrase-structure systems. *Information and Control*, 8:304–337.
  - ▶ Hideki Isozaki, Hideto Kazawa, and Tsutomu Hirao. 2004.  
A deterministic word dependency analyzer enhanced with preference learning. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING)*, pages 275–281.
  - ▶ Taku Kudo and Yuji Matsumoto. 2002.  
Japanese dependency analysis using cascaded chunking. In *Proceedings of the Sixth Workshop on Computational Language Learning (CoNLL)*, pages 63–69.
  - ▶ Dekang Lin. 1995.  
A dependency-based method for evaluating broad-coverage parsers. In *Proceedings of IJCAI-95*, pages 1420–1425.
  - ▶ Dekang Lin. 1998.  
A dependency-based method for evaluating broad-coverage parsers. *Natural Language Engineering*, 4:97–114.
  - ▶ Ryan McDonald and Joakim Nivre. 2007.

- Characterizing the errors of data-driven dependency parsing models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.
- ▶ Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005a.  
Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 91–98.
  - ▶ Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005b.  
Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the Human Language Technology Conference and the Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP)*, pages 523–530.
  - ▶ Igor Mel'čuk. 1988.  
*Dependency Syntax: Theory and Practice*. State University of New York Press.
  - ▶ Joakim Nivre and Jens Nilsson. 2005.  
Pseudo-projective dependency parsing. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 99–106.
  - ▶ Joakim Nivre and Mario Scholz. 2004.  
Deterministic dependency parsing of English text. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING)*, pages 64–70.
  - ▶ Joakim Nivre, Johan Hall, and Jens Nilsson. 2004.

- Memory-based dependency parsing. In Hwee Tou Ng and Ellen Riloff, editors, *Proceedings of the 8th Conference on Computational Natural Language Learning (CoNLL)*, pages 49–56.
- ▶ Joakim Nivre, Johan Hall, Jens Nilsson, Gülsen Eryiğit, and Svetoslav Marinov. 2006. Labeled pseudo-projective dependency parsing with support vector machines. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL)*.
  - ▶ Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In Gertjan Van Noord, editor, *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*, pages 149–160.
  - ▶ Joakim Nivre. 2006. Constraints on non-projective dependency graphs. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 73–80.
  - ▶ Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In Gertjan Van Noord, editor, *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*, pages 195–206.