Dependency Parsing

Article *in* Synthesis Lectures on Human Language Technologies · January 2009

DOI: 10.2200/S00169ED1V01Y200901HLT002

2 authors:

Joakim Nivre
Uppsala University

Sandra Kübler
Indiana University Bloomington

Dependency Parsing

Tutorial at COLING-ACL, Sydney 2006

Joakim Nivre[1] Sandra Kübler[2]

[1]Uppsala University and Växjö University, Sweden
E-mail: nivre@msi.vxu.se

[2]Eberhard-Karls Universität Tübingen, Germany
E-mail: kuebler@sfs.uni-tuebingen.de

Why?

- Increasing interest in dependency-based approaches to syntactic parsing in recent years
  - New methods emerging
  - Applied to a wide range of languages
  - CoNLL-X shared task (June, 2006)
- Dependency-based methods still less accessible for the majority of researchers and developers than the more widely known constituency-based methods

For Whom?

- Researchers and students working on syntactic parsing or related topics within other traditions
- Researchers and application developers interested in using dependency parsers as components in larger systems

What?

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

Outline

Outline

Dependency Syntax

- The basic idea:
  - Syntactic structure consists of lexical items, linked by binary asymmetric relations called dependencies.
- In the words of Lucien Tesnière [Tesnière 1959]:
  - La phrase est un *ensemble organisé* dont les éléments constituants sont les *mots*. [1.2] Tout mot qui fait partie d'une phrase cesse par lui-même d'être isolé comme dans le dictionnaire. Entre lui et ses voisins, l'esprit aperçoit des *connexions*, dont l'ensemble forme la charpente de la phrase. [1.3] Les connexions structurales établissent entre les mots des rapports de *dépendance*. Chaque connexion unit en principe un terme *supérieur* à un terme *inférieur*. [2.1] Le terme supérieur reçoit le nom de *régissant*. Le terme inférieur reçoit le nom de *subordonné*. Ainsi dans la phrase *Alfred parle* [. . .], *parle* est le régissant et *Alfred* le subordonné. [2.2]

Dependency Syntax

- The basic idea:
  - Syntactic structure consists of lexical items, linked by binary asymmetric relations called dependencies.
- In the words of Lucien Tesnière [Tesnière 1959]:
  - The sentence is an *organized whole*, the constituent elements of which are *words*. [1.2] Every word that belongs to a sentence ceases by itself to be isolated as in the dictionary. Between the word and its neighbors, the mind perceives *connections*, the totality of which forms the structure of the sentence. [1.3] The structural connections establish *dependency* relations between the words. Each connection in principle unites a *superior* term and an *inferior* term. [2.1] The superior term receives the name *governor*. The inferior term receives the name *subordinate*. Thus, in the sentence *Alfred parle* [. . .], *parle* is the governor and *Alfred* the subordinate. [2.2]

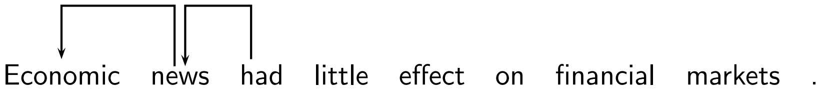
Dependency Structure

Economic news had little effect on financial markets .

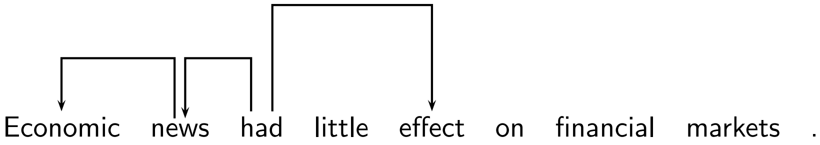
Dependency Structure

Economic news had little effect on financial markets .

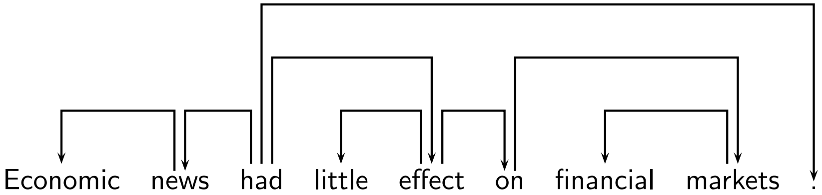
Dependency Structure



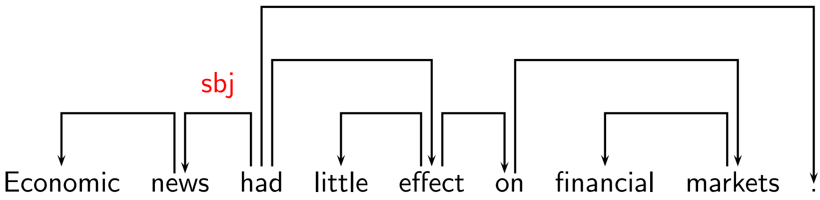
Dependency Structure



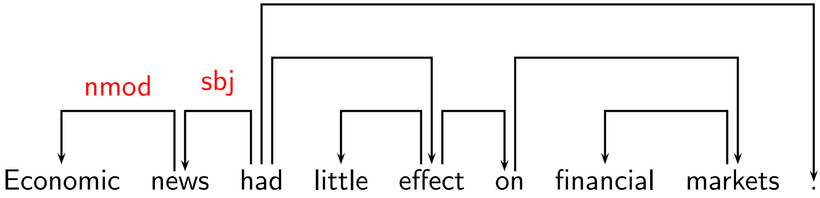
Dependency Structure



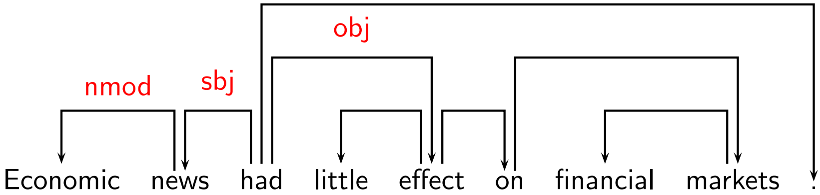
Dependency Structure



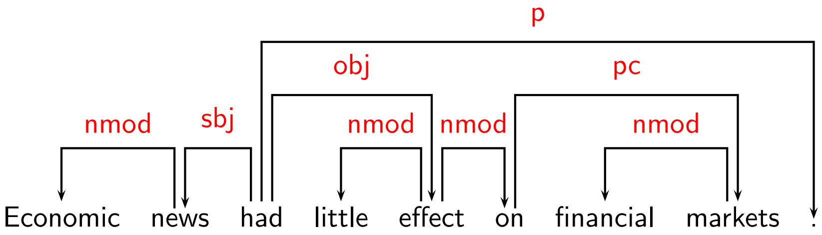
Dependency Structure



Dependency Structure



Dependency Structure



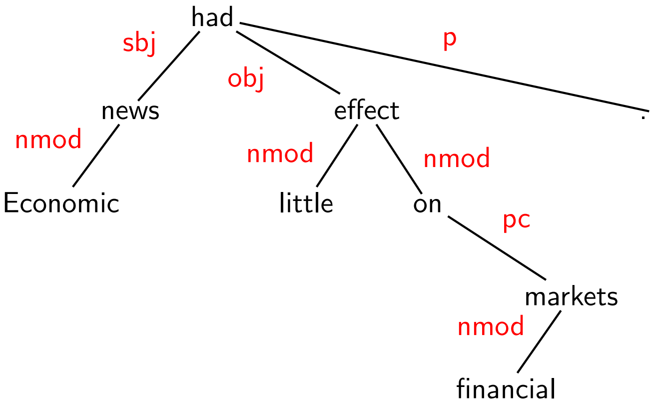
Terminology

Superior	Inferior
Head	Dependent
Governor	Modifier
Regent	Subordinate
⋮	⋮

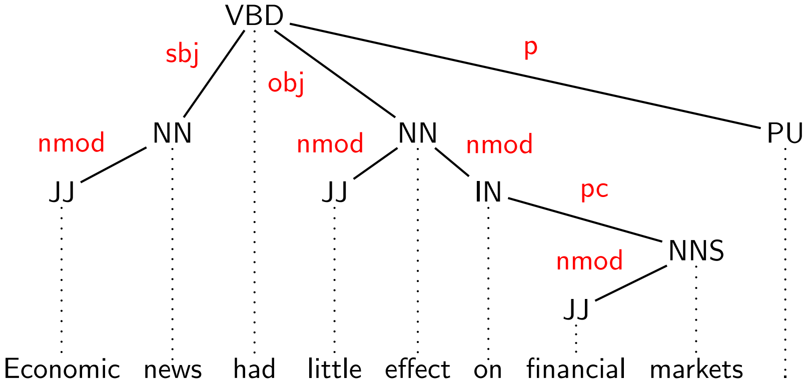
Terminology

Superior	**Inferior**
Head	Dependent
Governor	Modifier
Regent	Subordinate
⋮	⋮

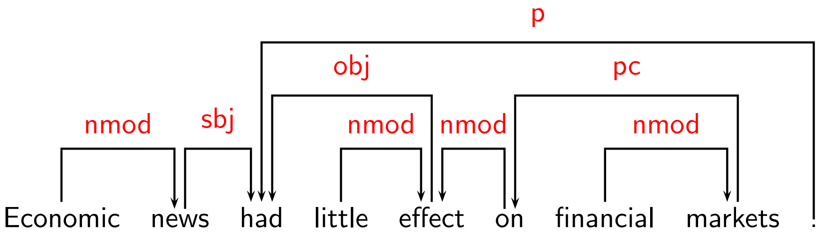
Notational Variants



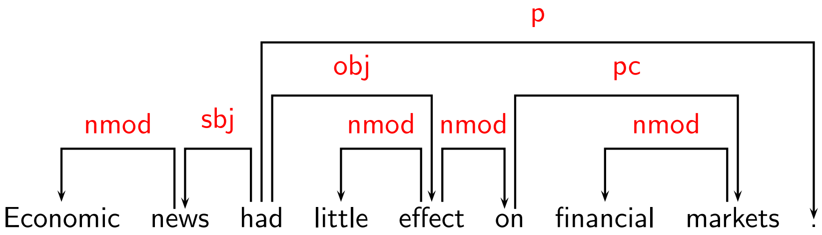
Notational Variants



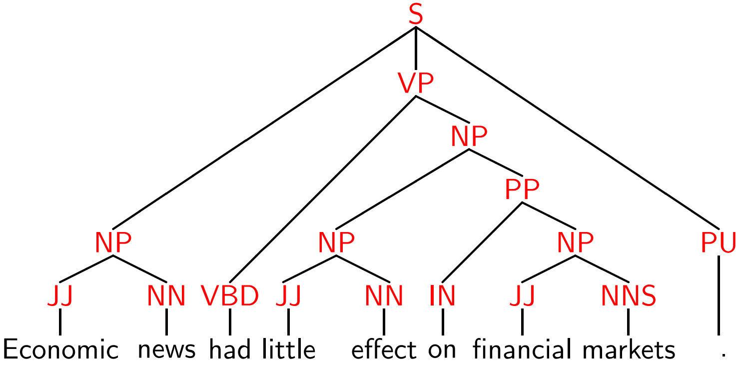
Notational Variants



Notational Variants



Phrase Structure



Comparison

- Dependency structures explicitly represent
  - head-dependent relations (directed arcs),
  - functional categories (arc labels),
  - possibly some structural categories (parts-of-speech).
- Phrase structures explicitly represent
  - phrases (nonterminal nodes),
  - structural categories (nonterminal labels),
  - possibly some functional categories (grammatical functions).
- Hybrid representations may combine all elements.

Some Theoretical Frameworks

- Word Grammar (WG) [Hudson 1984, Hudson 1990]
- Functional Generative Description (FGD) [Sgall et al. 1986]
- Dependency Unification Grammar (DUG)
  [Hellwig 1986, Hellwig 2003]
- Meaning-Text Theory (MTT) [Mel'čuk 1988]
- (Weighted) Constraint Dependency Grammar ([W]CDG)
  [Maruyama 1990, Harper and Helzerman 1995,
  Menzel and Schröder 1998, Schröder 2002]
- Functional Dependency Grammar (FDG)
  [Tapanainen and Järvinen 1997, Järvinen and Tapanainen 1998]
- Topological/Extensible Dependency Grammar ([T/X]DG)
  [Duchier and Debusmann 2001, Debusmann et al. 2004]

Some Theoretical Issues

- Dependency structure sufficient as well as necessary?
- Mono-stratal or multi-stratal syntactic representations?
- What is the nature of lexical elements (nodes)?
  - Morphemes?
  - Word forms?
  - Multi-word units?
- What is the nature of dependency types (arc labels)?
  - Grammatical functions?
  - Semantic roles?
- What are the criteria for identifying heads and dependents?
- What are the formal properties of dependency structures?

Some Theoretical Issues

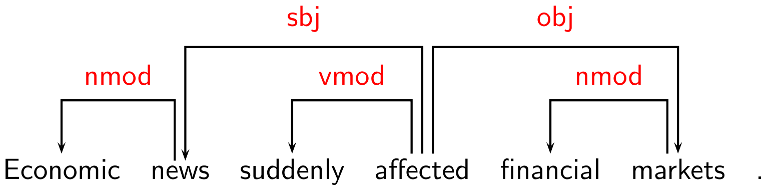
- Dependency structure sufficient as well as necessary?
- Mono-stratal or multi-stratal syntactic representations?
- What is the nature of lexical elements (nodes)?
  - Morphemes?
  - Word forms?
  - Multi-word units?
- What is the nature of dependency types (arc labels)?
  - Grammatical functions?
  - Semantic roles?
- What are the criteria for identifying heads and dependents?
- What are the formal properties of dependency structures?

Criteria for Heads and Dependents

- Criteria for a syntactic relation between a head H and a dependent D in a construction C [Zwicky 1985, Hudson 1990]:
  1. H determines the syntactic category of C; H can replace C.
  2. H determines the semantic category of C; D specifies H.
  3. H is obligatory; D may be optional.
  4. H selects D and determines whether D is obligatory.
  5. The form of D depends on H (agreement or government).
  6. The linear position of D is specified with reference to H.
- Issues:
  - Syntactic (and morphological) versus semantic criteria
  - Exocentric versus endocentric constructions

Some Clear Cases

Construction	Head	Dependent
Exocentric	Verb	Subject (sbj)
	Verb	Object (obj)
Endocentric	Verb	Adverbial (vmod)
	Noun	Attribute (nmod)



Some Tricky Cases

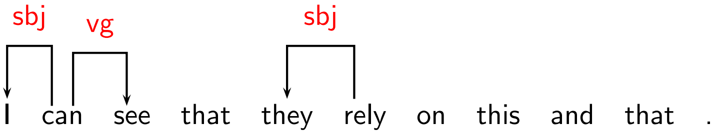
- Complex verb groups (auxiliary ↔ main verb)
- Subordinate clauses (complementizer ↔ verb)
- Coordination (coordinator ↔ conjuncts)
- Prepositional phrases (preposition ↔ nominal)
- Punctuation

?

I can see that they rely on this and that .

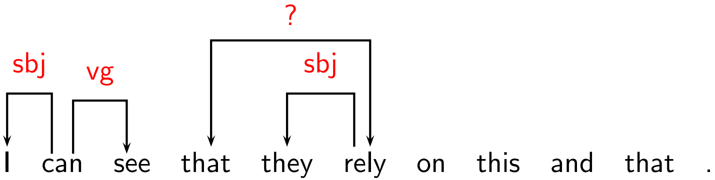
Some Tricky Cases

- Complex verb groups (auxiliary $\leftrightarrow$ main verb)
- Subordinate clauses (complementizer $\leftrightarrow$ verb)
- Coordination (coordinator $\leftrightarrow$ conjuncts)
- Prepositional phrases (preposition $\leftrightarrow$ nominal)
- Punctuation



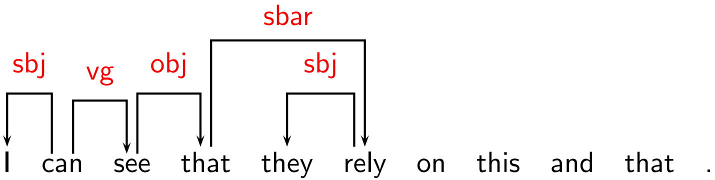
Some Tricky Cases

- Complex verb groups (auxiliary ↔ main verb)
- Subordinate clauses (complementizer ↔ verb)
- Coordination (coordinator ↔ conjuncts)
- Prepositional phrases (preposition ↔ nominal)
- Punctuation



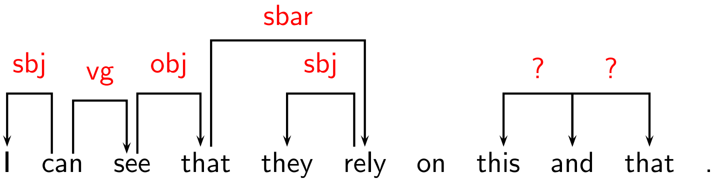
Some Tricky Cases

- Complex verb groups (auxiliary ↔ main verb)
- Subordinate clauses (complementizer ↔ verb)
- Coordination (coordinator ↔ conjuncts)
- Prepositional phrases (preposition ↔ nominal)
- Punctuation



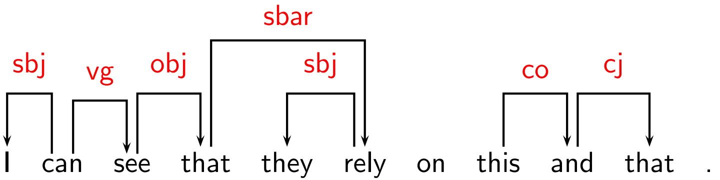
Some Tricky Cases

- Complex verb groups (auxiliary ↔ main verb)
- Subordinate clauses (complementizer ↔ verb)
- Coordination (coordinator ↔ conjuncts)
- Prepositional phrases (preposition ↔ nominal)
- Punctuation



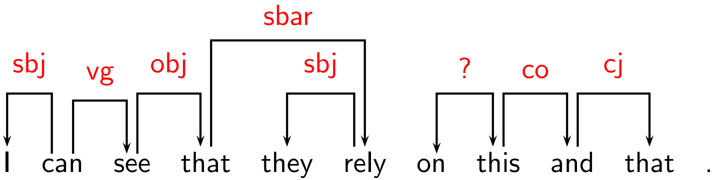
Some Tricky Cases

- Complex verb groups (auxiliary ↔ main verb)
- Subordinate clauses (complementizer ↔ verb)
- Coordination (coordinator ↔ conjuncts)
- Prepositional phrases (preposition ↔ nominal)
- Punctuation



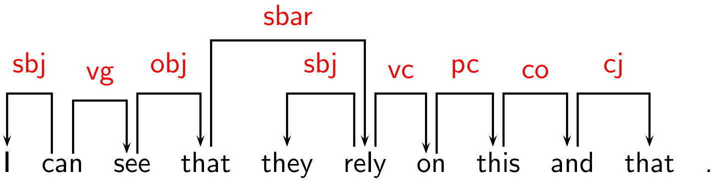
Some Tricky Cases

- Complex verb groups (auxiliary ↔ main verb)
- Subordinate clauses (complementizer ↔ verb)
- Coordination (coordinator ↔ conjuncts)
- Prepositional phrases (preposition ↔ nominal)
- Punctuation



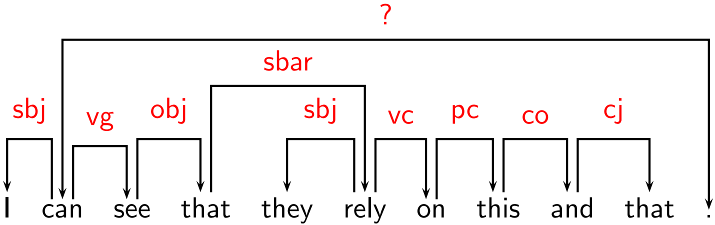
Some Tricky Cases

- Complex verb groups (auxiliary ↔ main verb)
- Subordinate clauses (complementizer ↔ verb)
- Coordination (coordinator ↔ conjuncts)
- Prepositional phrases (preposition ↔ nominal)
- Punctuation



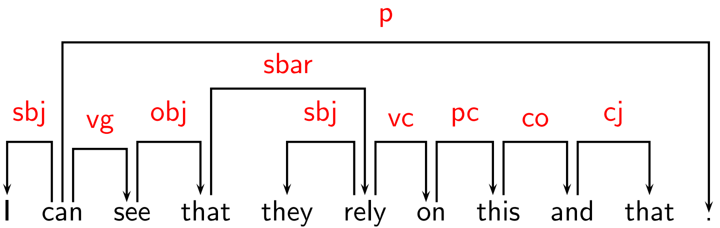
Some Tricky Cases

- Complex verb groups (auxiliary ↔ main verb)
- Subordinate clauses (complementizer ↔ verb)
- Coordination (coordinator ↔ conjuncts)
- Prepositional phrases (preposition ↔ nominal)
- Punctuation



Some Tricky Cases

- Complex verb groups (auxiliary ↔ main verb)
- Subordinate clauses (complementizer ↔ verb)
- Coordination (coordinator ↔ conjuncts)
- Prepositional phrases (preposition ↔ nominal)
- Punctuation



Dependency Graphs

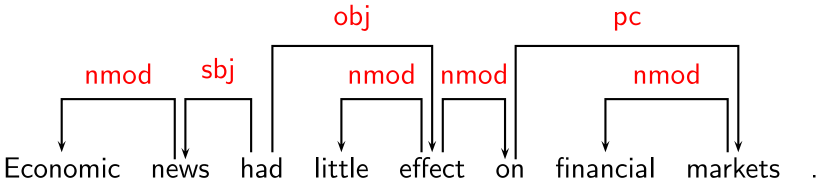
- A dependency structure can be defined as a directed graph G, consisting of
  - a set V of nodes,
  - a set E of arcs (edges),
  - a linear precedence order $<$ on V.
- Labeled graphs:
  - Nodes in V are labeled with word forms (and annotation).
  - Arcs in E are labeled with dependency types.
- Notational conventions ($i, j \in V$):
  - $i \rightarrow j \;\equiv\; (i, j) \in E$
  - $i \rightarrow^* j \;\equiv\; i = j \vee \exists k : i \rightarrow k,\ k \rightarrow^* j$

Formal Conditions on Dependency Graphs

- G is (weakly) connected:
  - For every node i there is a node j such that $i \rightarrow j$ or $j \rightarrow i$.
- G is acyclic:
  - If $i \rightarrow j$ then not $j \rightarrow^{*} i$.
- G obeys the single-head constraint:
  - If $i \rightarrow j$, then not $k \rightarrow j$, for any $k \neq i$.
- G is projective:
  - If $i \rightarrow j$ then $i \rightarrow^{*} k$, for any k such that $i < k < j$ or $j < k < i$.

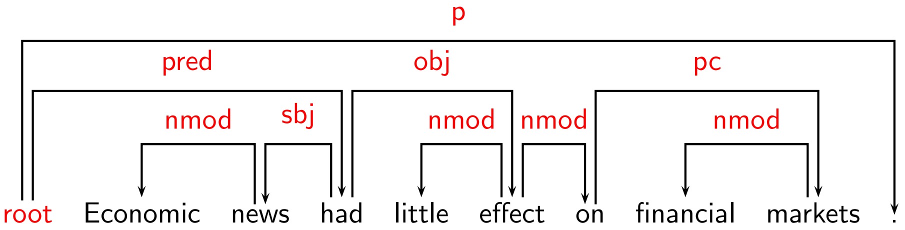
Connectedness, Acyclicity and Single-Head

- Intuitions:
  - Syntactic structure is complete (Connectedness).
  - Syntactic structure is hierarchical (Acyclicity).
  - Every word has at most one syntactic head (Single-Head).
- Connectedness can be enforced by adding a special root node.



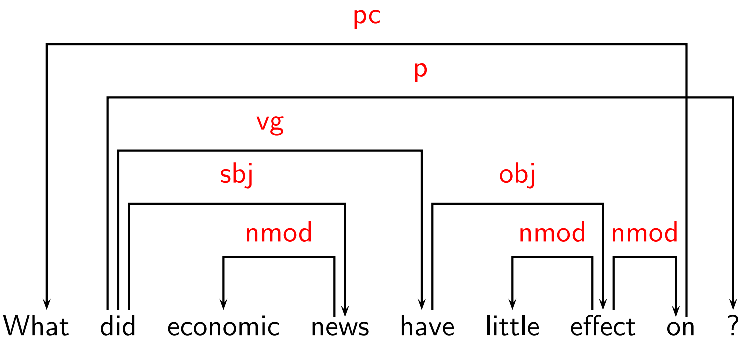
Connectedness, Acyclicity and Single-Head

- Intuitions:
  - Syntactic structure is complete (Connectedness).
  - Syntactic structure is hierarchical (Acyclicity).
  - Every word has at most one syntactic head (Single-Head).
- Connectedness can be enforced by adding a special root node.



Projectivity

- Most theoretical frameworks do not assume projectivity.
- Non-projective structures are needed to account for
  - long-distance dependencies,
  - free word order.



Scope of the Tutorial

- Dependency parsing:
  - Input: Sentence $x = w_1, \ldots, w_n$
  - Output: Dependency graph G
- Focus of tutorial:
  - Computational methods for dependency parsing
  - Resources for dependency parsing (parsers, treebanks)
- Not included:
  - Theoretical frameworks of dependency syntax
  - Constituency parsers that exploit lexical dependencies
  - Unsupervised learning of dependency structure

Parsing Methods

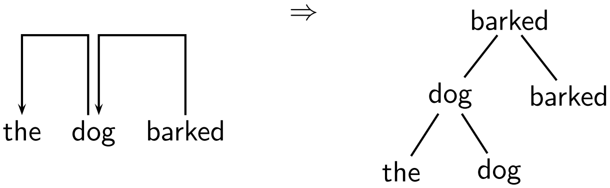
- Three main traditions:
  - Dynamic programming
  - Constraint satisfaction
  - Deterministic parsing
- Special issue:
  - Non-projective dependency parsing

Dynamic Programming

- Basic idea: Treat dependencies as constituents.
- Use, e.g., CYK parser (with minor modifications).
- Dependencies as constituents:

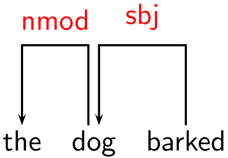
Dynamic Programming

- Basic idea: Treat dependencies as constituents.
- Use, e.g., CYK parser (with minor modifications).
- Dependencies as constituents:

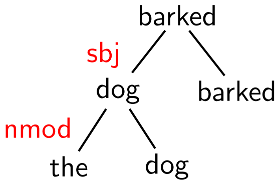


Dynamic Programming

- Basic idea: Treat dependencies as constituents.
- Use, e.g., CYK parser (with minor modifications).
- Dependencies as constituents:



$\Rightarrow$



Dependency Chart Parsing

- Grammar is regarded as context-free, in which each node is lexicalized.
- Chart entries are subtrees, i.e., words with all their left and right dependents.
- Problem: Different entries for different subtrees spanning a sequence of words with different heads.
- Time requirement: $O(n^5)$.

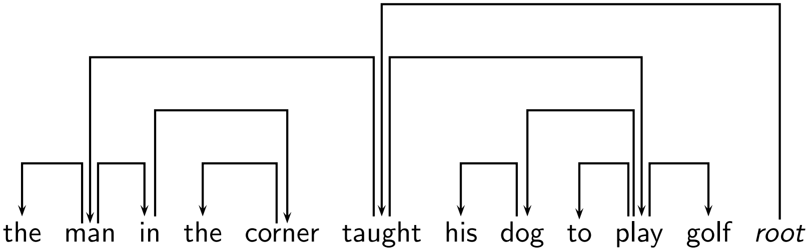
Dynamic Programming Approaches

- Original version: [Hays 1964]
- Link Grammar: [Sleator and Temperley 1991]
- Earley-style parser with left-corner filtering: [Lombardo and Lesmo 1996]
- Bilexical grammar: [Eisner 1996a, Eisner 1996b, Eisner 2000]
- Bilexical grammar with discriminative estimation methods: [McDonald et al. 2005a, McDonald et al. 2005b]

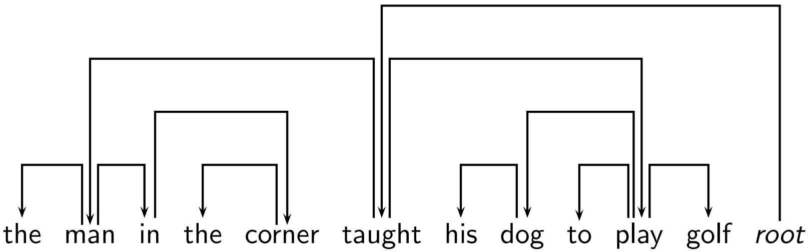
Eisner's Bilexical Algorithm

- Two novel aspects:
  - Modified parsing algorithm
  - Probabilistic dependency parsing
- Time requirement: $O(n^3)$.
- Modification: Instead of storing subtrees, store spans.
- Def. span: Substring such that no interior word links to any word outside the span.
- Underlying idea: In a span, only the endwords are active, i.e. still need a head.
- One or both of the endwords can be active.

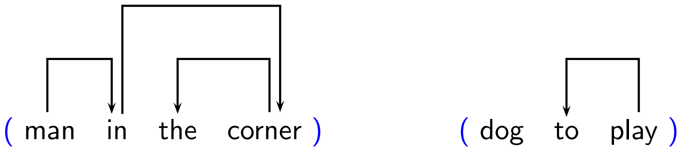
Example



Example



Spans:



Assembly of Correct Parse

Start by combining adjacent words to minimal spans:

(the man) (man in) (in the) . . .

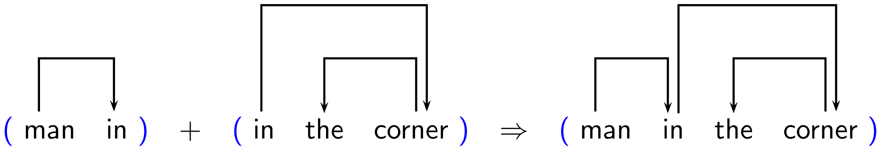
Assembly of Correct Parse

Combine spans which overlap in one word; this word must be governed by a word in the left or right span.

(in the) $+$ (the corner) $\Rightarrow$ (in the corner)

Assembly of Correct Parse

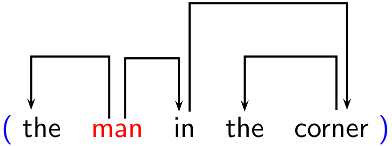
Combine spans which overlap in one word; this word must be governed by a word in the left or right span.



Assembly of Correct Parse

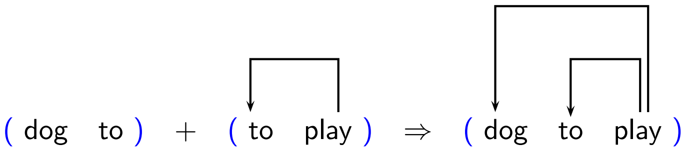
Combine spans which overlap in one word; this word must be governed by a word in the left or right span.

Invalid span:

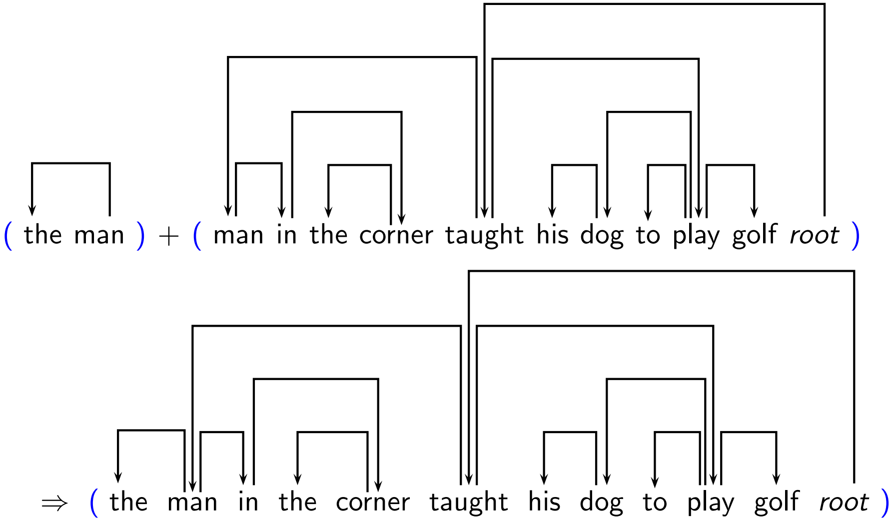


Assembly of Correct Parse

Combine spans which overlap in one word; this word must be governed by a word in the left or right span.



Assembly of Correct Parse



Eisner's Probability Models

- Model A: Bigram lexical affinities
  - First generates a trigram Markov model for POS tagging.
  - Decides for each word pair whether they have a dependency.
  - Model is leaky because it does not control for crossing dependencies, multiple heads, . . .

Eisner's Probability Models

- Model A: Bigram lexical affinities
  - First generates a trigram Markov model for POS tagging.
  - Decides for each word pair whether they have a dependency.
  - Model is leaky because it does not control for crossing dependencies, multiple heads, ...
- Model B: Selectional preferences
  - First generates a trigram Markov model for POS tagging.
  - Each word chooses a subcat/supercat frame.
  - Selects an analysis that satisfies all frames if possible.
  - Model is also leaky because last step may fail.

Eisner's Probability Models

- Model A: Bigram lexical affinities
  - First generates a trigram Markov model for POS tagging.
  - Decides for each word pair whether they have a dependency.
  - Model is leaky because it does not control for crossing dependencies, multiple heads, ...
- Model B: Selectional preferences
  - First generates a trigram Markov model for POS tagging.
  - Each word chooses a subcat/supercat frame.
  - Selects an analysis that satisfies all frames if possible.
  - Model is also leaky because last step may fail.
- Model C: Recursive Generation
  - Each word generates its actual dependents.
  - Two Markov chains:
    - Left dependents
    - Right dependents
  - Model is not leaky.

Eisner's Model C

$$Pr(words, tags, links) =$$

$$\prod_{1 \leq i \leq n} \left(\prod_{c} Pr(tword(dep_c(i)) \mid tag(dep_{c-1}(i)), tword(i)) \right)$$

$c = -(1 + \#left - deps(i)) \ldots 1 + \#right - deps(i),\ c \neq 0$

or: $dep_{c+1}(i)$ if $c < 0$

Eisner's Results

- 25 000 Wall Street Journal sentences
- Baseline: most frequent tag chosen for a word, each word chooses a head with most common distance
- Model X: trigram tagging, no dependencies
- For comparison: state-of-the-art constituent parsing, Charniak: 92.2 F-measure

Model	**Non-punct**	**Tagging**
Baseline	41.9	76.1
Model X	–	93.1
Model A	too slow	
Model B	83.8	92.8
Model C	86.9	92.0

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.
- If scores are available, parsing can be formulated as maximum spanning tree problem.
- 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 online learning for determining weight vector $\mathbf{w}$: large-margin multi-class classification (MIRA)

Maximum Spanning Trees (2)

- Complexity:
  - Projective (Eisner): $O(n^3)$
  - Non-projective (CLE): $O(n^2)$

$$score(sent, deps) = \sum_{(i,j) \in deps} score(i,j) = \sum_{(i,j) \in deps} \mathbf{w} \cdot f(i,j)$$

Online Learning

Training data: $\mathcal{T} = (sent_t, deps_t)_{t=1}^{\mathcal{T}}$

1. $\mathbf{w} = 0$; $\mathbf{v} = 0$; $i = 0$;
2. for $n : 1..\mathcal{N}$
3. for $t : 1..\mathcal{T}$
4. $\mathbf{w}^{(i+1)} =$ update $\mathbf{w}^{(i)}$ according to $(sent_t, deps_t)$
5. $\mathbf{v} = \mathbf{v} + \mathbf{w}^{(i+1)}$
6. $i = i + 1$
7. $\mathbf{w} = \mathbf{v}/(\mathcal{N} \cdot \mathcal{T})$

MIRA

MIRA weight update:

min $||\mathbf{w}^{(i+1)} - \mathbf{w}^{(i)}||$ so that

$$score(sent_t, deps_t) - score(sent_t, deps') \geq L(deps_t, deps')$$

$$\forall deps' \in dt(sent_t)$$

- $L(deps, deps')$: loss function
- $dt(sent)$: possible dependency parses for sentence

Results by McDonald et al. (2005a, 2005b)

- Unlabeled accuracy per word (W) and per sentence (S)

	English		Czech	
Parser	**W**	**S**	**W**	**S**
k-best MIRA Eisner	90.9	37.5	83.3	31.3
best MIRA CLE	90.2	33.2	84.1	32.2
factored MIRA CLE	90.2	32.2	84.4	32.3

- New development (EACL 2006):
  - Scores of dependencies are not independent any more
  - Better results
  - More later

Parsing Methods

- Three main traditions:
  - Dynamic programming
  - Constraint satisfaction
  - Deterministic parsing
- Special issue:
  - Non-projective dependency parsing

Constraint Satisfaction

- Uses *Constraint Dependency Grammar*.
- Grammar consists of a set of boolean constraints, i.e. logical formulas that describe well-formed trees.
- A constraint is a logical formula with variables that range over a set of predefined values.
- Parsing is defined as a constraint satisfaction problem.
- Parsing is an *eliminative* process rather than a *constructive* one such as in CFG parsing.
- Constraint satisfaction removes values that contradict constraints.

Examples for Constraints

- Based on [Maruyama 1990]

Examples for Constraints

- Based on [Maruyama 1990]
- Example 1:
  - $word(pos(x)) = DET \Rightarrow$
    $(label(X) = NMOD,\ word(mod(x)) = NN,\ pos(x) < mod(x))$
  - A determiner (DET) modifies a noun (NN) on the right with the label NMOD.

Examples for Constraints

- Based on [Maruyama 1990]
- Example 1:
  - $word(pos(x)) = DET \Rightarrow$
    $(label(X) = NMOD, word(mod(x)) = NN, pos(x) < mod(x))$
  - A determiner (DET) modifies a noun (NN) on the right with the label NMOD.
- Example 2:
  - $word(pos(x)) = NN \Rightarrow$
    $(label(x) = SBJ, word(mod(x)) = VB, pos(x) < mod(x))$
  - A noun modifies a verb (VB) on the right with the label SBJ.

Examples for Constraints

- Based on [Maruyama 1990]
- Example 1:
  - $word(pos(x)) = DET \Rightarrow$
    $(label(X) = NMOD, word(mod(x)) = NN, pos(x) < mod(x))$
  - A determiner (DET) modifies a noun (NN) on the right with the label NMOD.
- Example 2:
  - $word(pos(x)) = NN \Rightarrow$
    $(label(x) = SBJ, word(mod(x)) = VB, pos(x) < mod(x))$
  - A noun modifies a verb (VB) on the right with the label SBJ.
- Example 3:
  - $word(pos(x)) = VB \Rightarrow$
    $(label(x) = ROOT, mod(x) = nil)$
  - A verb modifies nothing, its label is ROOT.

Constraint Satisfaction Approaches

- Constraint Grammar: [Karlsson 1990, Karlsson et al. 1995]
- Constraint Dependency Grammar:
  [Maruyama 1990, Harper and Helzerman 1995]
- Functional Dependency Grammar: [Järvinen and Tapanainen 1998]
- Topological Dependency Grammar: [Duchier 1999, Duchier 2003]
- Extensible Dependency Grammar: [Debusmann et al. 2004]
- Constraint Dependency Grammar with defeasible constraints:
  [Foth et al. 2000, Foth et al. 2004, Menzel and Schröder 1998, Schröder 2002]

Constraint Satisfaction

- Constraint satisfaction in general is *NP complete*.
- Parser design must ensure practical efficiency.
- Different approaches to do constraint satisfaction:
  - Maruyama applies constraint propagation techniques, which ensure local consistency (arc consistency).
  - Weighted CDG uses transformation-based constraint resolution with anytime properties [Foth et al. 2000, Foth et al. 2004, Menzel and Schröder 1998, Schröder 2002].
  - TDG uses constraint programming [Duchier 1999, Duchier 2003].

Maruyama's Constraint Propagation

Three steps:

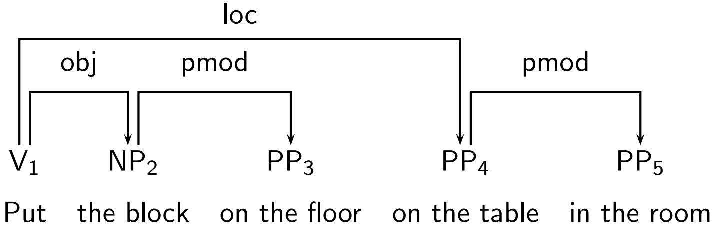
1. Form initial constraint network using a "core" grammar.
2. Remove local inconsistencies.
3. If ambiguity remains, add new constraints and repeat step 2.

Constraint Propagation Example

- Problem: PP attachment
- Sentence: *Put the block on the floor on the table in the room*
- Simplified representation: V_1 NP_2 PP_3 PP_4 PP_5

Constraint Propagation Example

- Problem: PP attachment
- Sentence: *Put the block on the floor on the table in the room*
- Simplified representation: V_1 NP_2 PP_3 PP_4 PP_5
- Correct analysis:



Initial Constraints

- - $word(pos(x))$=PP
    $\Rightarrow (word(mod(x)) \in \{PP, NP, V\}, mod(x) < pos(x))$
  - A PP modifies a PP, an NP, or a V on the left.

Initial Constraints

- - *word(pos(x))=PP*
    $\Rightarrow$ *(word(mod(x))* $\in \{$*PP, NP, V*$\}$, *mod(x)* $<$ *pos(x))*
  - A PP modifies a PP, an NP, or a V on the left.
- - *word(pos(x))=PP, word(mod(x))* $\in \{$*PP, NP*$\}$
    $\Rightarrow$ *label(x)=pmod*
  - If a PP modifies a PP or an NP, its label is pmod.

Initial Constraints

- - *word(pos(x))=PP*
    $\Rightarrow$ *(word(mod(x))* $\in$ {*PP, NP, V*}, *mod(x)* $<$ *pos(x))*
  - A PP modifies a PP, an NP, or a V on the left.
- - *word(pos(x))=PP, word(mod(x))* $\in$ {*PP, NP*}
    $\Rightarrow$ *label(x)=pmod*
  - If a PP modifies a PP or an NP, its label is pmod.
- - *word(pos(x))=PP, word(mod(x))=V* $\Rightarrow$ *label(x)=loc*
  - If a PP modifies a V, its label is loc.

Initial Constraints

- - $word(pos(x))=PP$
    $\Rightarrow (word(mod(x)) \in \{PP, NP, V\}, mod(x) < pos(x))$
  - A PP modifies a PP, an NP, or a V on the left.
- - $word(pos(x))=PP, word(mod(x)) \in \{PP, NP\}$
    $\Rightarrow label(x)=pmod$
  - If a PP modifies a PP or an NP, its label is pmod.
- - $word(pos(x))=PP, word(mod(x))=V \Rightarrow label(x)=loc$
  - If a PP modifies a V, its label is loc.
- - $word(pos(x))=NP$
    $\Rightarrow (word(mod(x))=V, label(x)=obj, mod(x) < pos(x))$
  - An NP modifies a V on the left with the label obj.

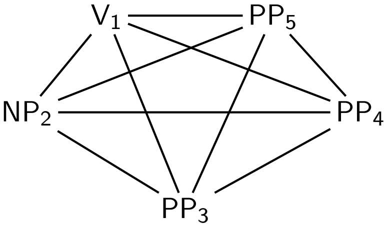
Initial Constraints

- - $word(pos(x)){=}PP$ $\Rightarrow (word(mod(x)) \in \{PP, NP, V\}, mod(x) < pos(x))$
  - A PP modifies a PP, an NP, or a V on the left.
- - $word(pos(x)){=}PP, word(mod(x)) \in \{PP, NP\}$ $\Rightarrow label(x){=}pmod$
  - If a PP modifies a PP or an NP, its label is pmod.
- - $word(pos(x)){=}PP, word(mod(x)){=}V \Rightarrow label(x){=}loc$
  - If a PP modifies a V, its label is loc.
- - $word(pos(x)){=}NP$ $\Rightarrow (word(mod(x)){=}V, label(x){=}obj, mod(x) < pos(x))$
  - An NP modifies a V on the left with the label obj.
- - $word(pos(x)){=}V \Rightarrow (mod(x){=}nil, label(x){=}root)$
  - A V modifies nothing with the label root.

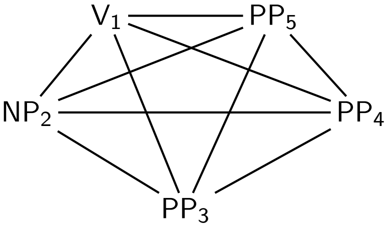
Initial Constraints

- - $word(pos(x))=PP$ $\Rightarrow (word(mod(x)) \in \{PP, NP, V\}, mod(x) < pos(x))$
  - A PP modifies a PP, an NP, or a V on the left.
- - $word(pos(x))=PP, word(mod(x)) \in \{PP, NP\}$ $\Rightarrow label(x)=pmod$
  - If a PP modifies a PP or an NP, its label is pmod.
- - $word(pos(x))=PP, word(mod(x))=V \Rightarrow label(x)=loc$
  - If a PP modifies a V, its label is loc.
- - $word(pos(x))=NP$ $\Rightarrow (word(mod(x))=V, label(x)=obj, mod(x) < pos(x))$
  - An NP modifies a V on the left with the label obj.
- - $word(pos(x))=V \Rightarrow (mod(x)=nil, label(x)=root)$
  - A V modifies nothing with the label root.
- - $mod(x) < pos(y) < pos(x) \Rightarrow mod(x) \leq mod(y) \leq pos(x)$
  - Modification links do not cross.

Initial Constraint Network



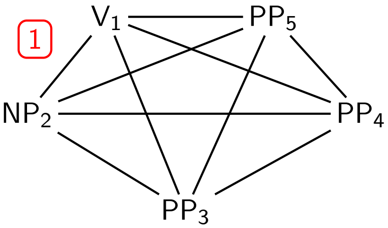
Initial Constraint Network



Possible values ⇐ unary constraints:

- V_1: <root, nil>
- NP_2: <obj, 1>
- PP_3: <loc, 1>, <pmod, 2>
- PP_4: <loc, 1>, <pmod, 2>, <pmod, 3>
- PP_5: <loc, 1>, <pmod, 2>, <pmod, 3>, <pmod,4>

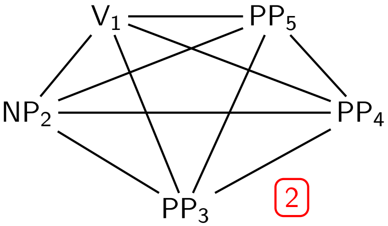
Initial Constraint Network



Each arc has a constraint matrix:
For arc 1:

$\downarrow$ V_1 \ NP_2 $\rightarrow$	<obj, 1>
<root, nil>	1

Initial Constraint Network



Each arc has a constraint matrix:
For arc 2:

↓ PP_3 \ PP_4 →	<loc, 1>	<pmod, 2>	<pmod, 3>
<loc, 1>	1	0	1
<pmod, 2>	1	1	1

Adding New Constraints

- Still 14 possible analyses.
- Filtering with binary constraints does not reduce ambiguity.
- Introduce more constraints:

Adding New Constraints

- Still 14 possible analyses.
- Filtering with binary constraints does not reduce ambiguity.
- Introduce more constraints:
-
  - $\mathit{word(pos(x))} = \mathit{PP}$, $\mathit{on\_table} \in \mathit{sem(pos(x))}$
    $\Rightarrow \neg(\mathit{floor} \in \mathit{sem(mod(x))})$
  - A floor is not on the table.

Adding New Constraints

- Still 14 possible analyses.
- Filtering with binary constraints does not reduce ambiguity.
- Introduce more constraints:
-   - $word(pos(x))=PP$, $on\_table \in sem(pos(x))$ $\Rightarrow \neg(floor \in sem(mod(x)))$
    - A floor is not on the table.
-   - $label(x)=loc$, $label(y)=loc$, $mod(x)=mod(y)$, $word(mod(x))=V$ $\Rightarrow x=y$
    - No verb can take two locatives.

Adding New Constraints

- Still 14 possible analyses.
- Filtering with binary constraints does not reduce ambiguity.
- Introduce more constraints:
- - $word(pos(x)){=}PP,\ on\_table \in sem(pos(x))$
    $\Rightarrow \neg(floor \in sem(mod(x)))$
  - A floor is not on the table.
- - $label(x){=}loc,\ label(y){=}loc,\ mod(x){=}mod(y),\ word(mod(x)){=}V$
    $\Rightarrow x{=}y$
  - No verb can take two locatives.
- Each value in the domains of nodes is tested against the new constraints.

Modified Tables

Old:

$\downarrow$ $PP_3 \setminus PP_4$ $\rightarrow$	<loc, 1>	<pmod, 2>	<pmod, 3>
<loc, 1>	1	0	1
<pmod, 2>	1	1	1

Modified Tables

Old:

↓ $PP_3 \setminus PP_4$ →	<loc, 1>	<pmod, 2>	<pmod, 3>
<loc, 1>	1	0	1
<pmod, 2>	1	1	1

violates first constraint

Modified Tables

Old:

$\downarrow$ $PP_3 \setminus PP_4$ $\rightarrow$	<loc, 1>	<pmod, 2>	<pmod, 3>
<loc, 1>	1	0	1
<pmod, 2>	1	1	1

After applying first new constraint:

$\downarrow$ $PP_3 \setminus PP_4$ $\rightarrow$	<loc, 1>	<pmod, 2>
<loc, 1>	1	0
<pmod, 2>	1	1

Modified Tables

Old:

$\downarrow \mathrm{PP}_3 \setminus \mathrm{PP}_4 \rightarrow$	<loc, 1>	<pmod, 2>	<pmod, 3>
<loc, 1>	1	0	1
<pmod, 2>	1	1	1

After applying first new constraint:

$\downarrow \mathrm{PP}_3 \setminus \mathrm{PP}_4 \rightarrow$	<loc, 1>	<pmod, 2>
<loc, 1>	1	0
<pmod, 2>	1	1

violates second constraint

Modified Tables

Old:

$\downarrow \mathrm{PP}_3 \setminus \mathrm{PP}_4 \rightarrow$	<loc, 1>	<pmod, 2>	<pmod, 3>
<loc, 1>	1	0	1
<pmod, 2>	1	1	1

After applying first new constraint:

$\downarrow \mathrm{PP}_3 \setminus \mathrm{PP}_4 \rightarrow$	<loc, 1>	<pmod, 2>
<loc, 1>	0	0
<pmod, 2>	1	1

Modified Tables

Old:

$\downarrow$ $PP_3 \setminus PP_4$ $\rightarrow$	<loc, 1>	<pmod, 2>	<pmod, 3>
<loc, 1>	1	0	1
<pmod, 2>	1	1	1

After applying first new constraint:

$\downarrow$ $PP_3 \setminus PP_4$ $\rightarrow$	<loc, 1>	<pmod, 2>
<loc, 1>	0	0
<pmod, 2>	1	1

After applying second new constraint:

$\downarrow$ $PP_3 \setminus PP_4$ $\rightarrow$	<loc, 1>	<pmod, 2>
<pmod, 2>	1	1

Weighted Constraint Parsing

- Approach by [Foth et al. 2004, Foth et al. 2000, Menzel and Schröder 1998, Schröder 2002]
- Robust parser, which uses soft constraints
- Each constraint is assigned a weight between 0.0 and 1.0
- Weight 0.0: hard constraint, can only be violated when no other parse is possible
- Constraints assigned manually (or estimated from treebank)
- Efficiency: uses a heuristic transformation-based constraint resolution method

Transformation-Based Constraint Resolution

- Heuristic search
- Very efficient
- Idea: first construct arbitrary dependency structure, then try to correct errors
- Error correction by transformations
- Selection of transformations based on constraints that cause conflicts
- Anytime property: parser maintains a complete analysis at any time $\Rightarrow$ can be stopped at any time and return a complete analysis

Menzel et al.'s Results

- Evaluation on NEGRA treebank for German
- German more difficult to parse than English (free word order)
- Constituent-based parsing: labeled F measure including grammatical functions: 53.4 [Kübler et al. 2006], labeled F measure: 73.1 [Dubey 2005].
- Best CoNLL-X results: unlabeled: 90.4, labeled: 87.3 [McDonald et al. 2006].

Data	Unlabeled	Labeled
1000 sentences	89.0	87.0
$<$ 40 words	89.7	87.7

Parsing Methods

- Three main traditions:
  - Dynamic programming
  - Constraint satisfaction
  - Deterministic parsing
- Special issue:
  - Non-projective dependency parsing

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

Covington's Incremental Algorithm

- Deterministic incremental parsing in $O(n^2)$ time by trying to link each new word to each preceding one [Covington 2001]:

```
PARSE(x = (w_1, ..., w_n))
    for i = 1 up to n
        for j = i − 1 down to 1
            LINK(w_i, w_j)
```

$$\textsc{Link}(w_i,\ w_j) = \begin{cases} E \leftarrow E \cup (i,j) & \text{if } w_j \text{ is a dependent of } w_i \\ E \leftarrow E \cup (j,i) & \text{if } w_i \text{ is a dependent of } w_j \\ E \leftarrow E & \text{otherwise} \end{cases}$$

- Different conditions, such as Single-Head and Projectivity, can be incorporated into the LINK operation.

Shift-Reduce Type Algorithms

- Data structures:
  - Stack $[\ldots, w_i]_S$ of partially processed tokens
  - Queue $[w_j, \ldots]_Q$ 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
- Left-to-right parsing in $O(n)$ time
- Restricted to projective dependency graphs

Yamada's Algorithm

- Three parsing actions:

$$\text{Shift} \quad \frac{[\ldots]_S \quad [w_i, \ldots]_Q}{[\ldots, w_i]_S \quad [\ldots]_Q}$$

$$\text{Left} \quad \frac{[\ldots, w_i, w_j]_S \quad [\ldots]_Q}{[\ldots, w_i]_S \quad [\ldots]_Q \quad w_i \rightarrow w_j}$$

$$\text{Right} \quad \frac{[\ldots, w_i, w_j]_S \quad [\ldots]_Q}{[\ldots, w_j]_S \quad [\ldots]_Q \quad w_i \leftarrow w_j}$$

- Algorithm variants:
  - Originally developed for Japanese (strictly head-final) with only the Shift and Right actions [Kudo and Matsumoto 2002]
  - Adapted for English (with mixed headedness) by adding the Left action [Yamada and Matsumoto 2003]
  - Multiple passes over the input give time complexity $O(n^2)$

Nivre's Algorithm

- Four parsing actions:

$$\text{Shift} \quad \frac{[\ldots]_S \quad [w_i, \ldots]_Q}{[\ldots, w_i]_S \quad [\ldots]_Q}$$

$$\text{Reduce} \quad \frac{[\ldots, w_i]_S \quad [\ldots]_Q \quad \exists w_k : w_k \rightarrow w_i}{[\ldots]_S \quad [\ldots]_Q}$$

$$\text{Left-Arc}_r \quad \frac{[\ldots, w_i]_S \quad [w_j, \ldots]_Q \quad \neg\exists w_k : w_k \rightarrow w_i}{[\ldots]_S \quad [w_j, \ldots]_Q \quad w_i \stackrel{r}{\leftarrow} w_j}$$

$$\text{Right-Arc}_r \quad \frac{[\ldots, w_i]_S \quad [w_j, \ldots]_Q \quad \neg\exists w_k : w_k \rightarrow w_j}{[\ldots, w_i, w_j]_S \quad [\ldots]_Q \quad w_i \stackrel{r}{\rightarrow} w_j}$$

- Characteristics:
  - Integrated labeled dependency parsing
  - Arc-eager processing of right-dependents
  - Single pass over the input gives time complexity $O(n)$

Example

$[$root$]_S$ $[$Economic news had little effect on financial markets .$]_Q$

Example

[root Economic]$_S$ [news had little effect on financial markets .]$_Q$

Shift

Example

nmod

[root]$_S$ Economic [news had little effect on financial markets .]$_Q$

Left-Arc$_{nmod}$

Example

nmod

[root Economic news]$_S$ [had little effect on financial markets .]$_Q$

Shift

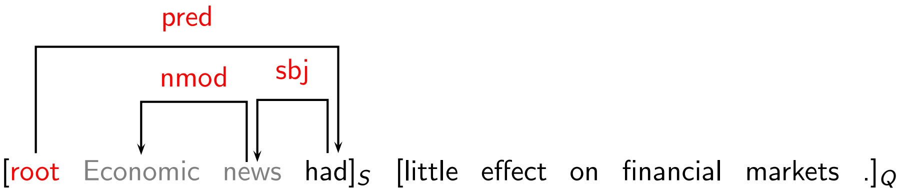
Example

nmod sbj

[root]$_S$ Economic news [had little effect on financial markets .]$_Q$

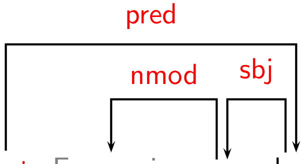
Left-Arc$_{sbj}$

Example



Right-Arc$_{pred}$

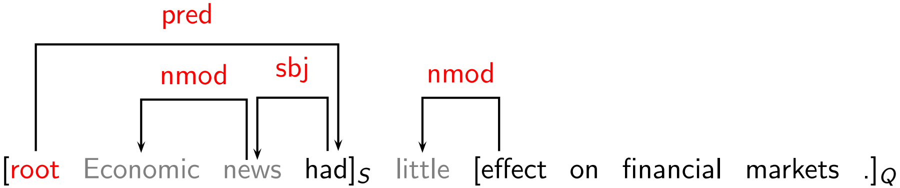
Example



[root Economic news had little]$_S$ [effect on financial markets .]$_Q$

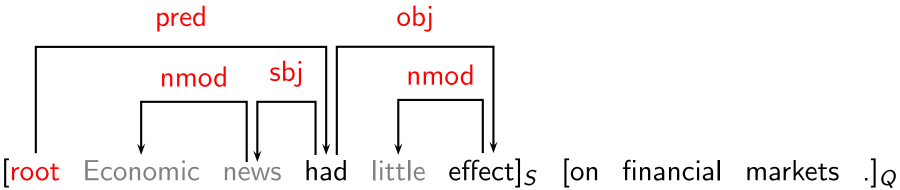
Shift

Example



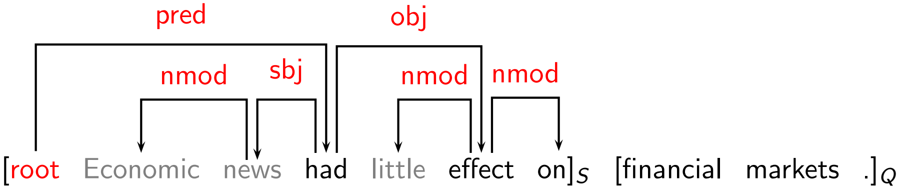
Left-Arc$_{nmod}$

Example



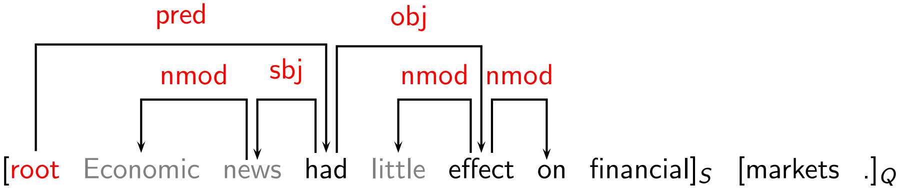
Right-Arc$_{obj}$

Example



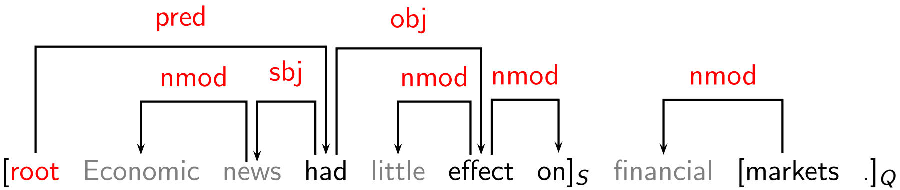
Right-Arc$_{nmod}$

Example



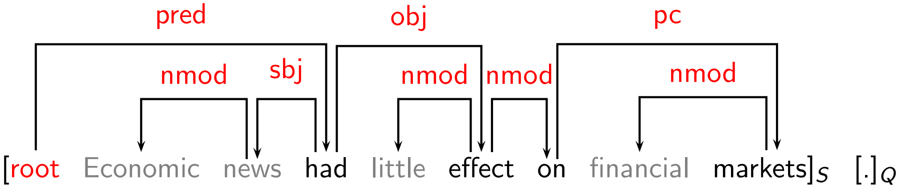
Shift

Example



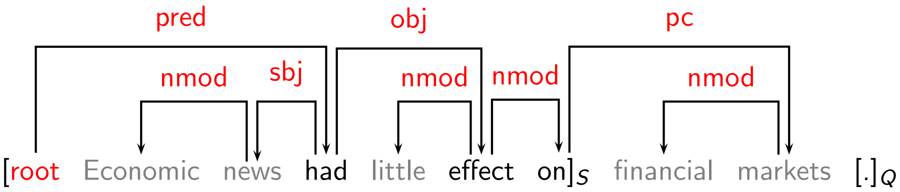
Left-Arc$_{nmod}$

Example



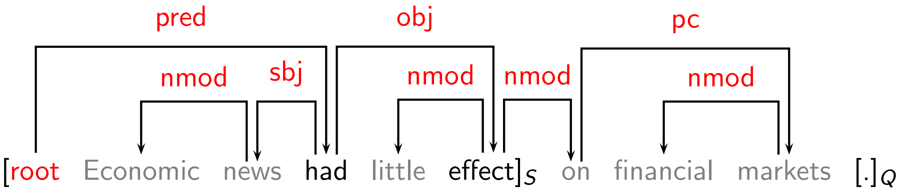
Right-Arc$_{pc}$

Example



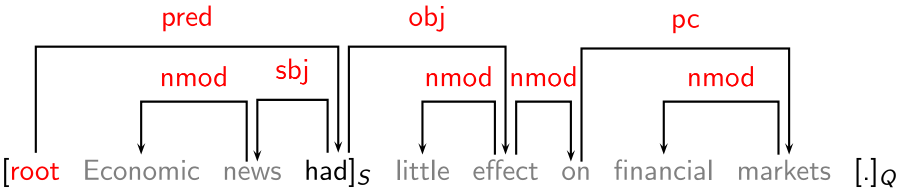
Reduce

Example



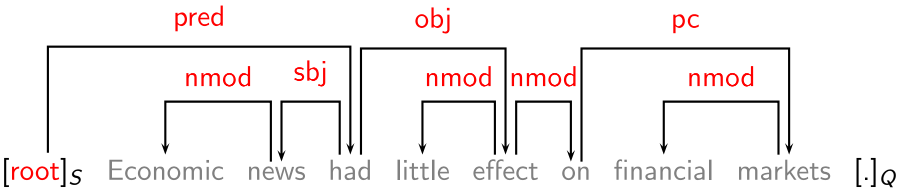
Reduce

Example



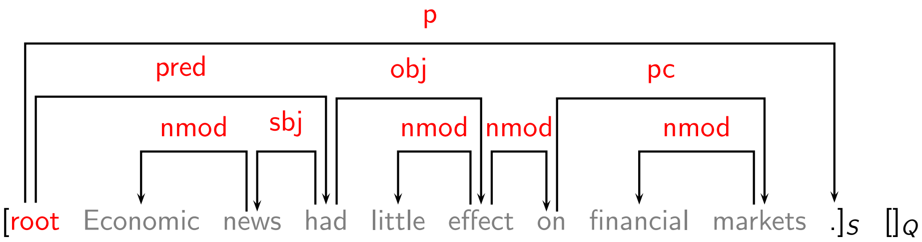
Reduce

Example



Reduce

Example



Right-Arc$_p$

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 states, 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)

State of the Art – English

- Evaluation:
  - Penn Treebank (WSJ) converted to dependency graphs
  - Unlabeled accuracy per word (W) and per sentence (S)
    - Deterministic classifier-based parsers
      [Yamada and Matsumoto 2003, Isozaki et al. 2004]
    - Spanning tree parsers with online training
      [McDonald et al. 2005a, McDonald and Pereira 2006]
    - Collins and Charniak parsers with same conversion

Parser	W	S
Charniak	92.2	45.2
Collins	91.7	43.3
McDonald and Pereira	91.5	42.1
Isozaki et al.	91.4	40.7
McDonald et al.	91.0	37.5
Yamada and Matsumoto	90.4	38.4

Comparing Algorithms

- Parsing algorithm:
  - Nivre's algorithm gives higher accuracy than Yamada's algorithm for parsing the Chinese CKIP treebank [Cheng et al. 2004].
- Learning algorithm:
  - SVM gives higher accuracy than MaxEnt for parsing the Chinese CKIP treebank [Cheng et al. 2004].
  - SVM gives higher accuracy than MBL with lexicalized feature models for three languages [Hall et al. 2006]:
    - Chinese (Penn)
    - English (Penn)
    - Swedish (Talbanken)

Parsing Methods

- Three main traditions:
  - Dynamic programming
  - Constraint satisfaction
  - Deterministic parsing
- Special issue:
  - Non-projective dependency parsing

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:
  - Constraint satisfaction methods [Tapanainen and Järvinen 1997, Duchier and Debusmann 2001, Foth et al. 2004]
  - 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]
  - Corrective modeling [Hall and Novák 2005]
  - Approximate non-projective parsing [McDonald and Pereira 2006]

Non-Projective Parsing Algorithms

- Complexity considerations:
  - Projective (Proj)
  - Non-projective (NonP)

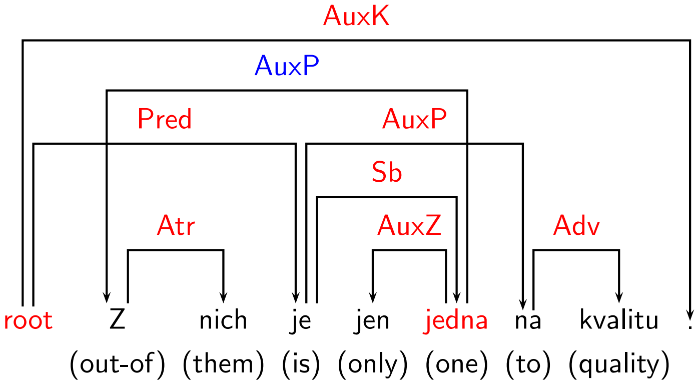
Problem/Algorithm	Proj	NonP
Complete grammar parsing [Gaifman 1965, Neuhaus and Bröker 1997]	P	NP hard
Deterministic parsing [Nivre 2003, Covington 2001]	$O(n)$	$O(n^2)$
First order spanning tree [McDonald et al. 2005b]	$O(n^3)$	$O(n^2)$
Nth order spanning tree ($N > 1$) [McDonald and Pereira 2006]	P	NP hard

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.
- Approaches:
  - Pseudo-projective parsing [Nivre and Nilsson 2005]
  - Corrective modeling [Hall and Novák 2005]
  - Approximate non-projective parsing [McDonald and Pereira 2006]

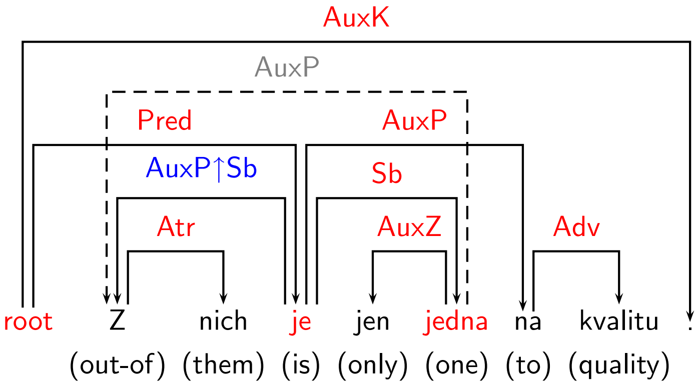
Pseudo-Projective Parsing

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



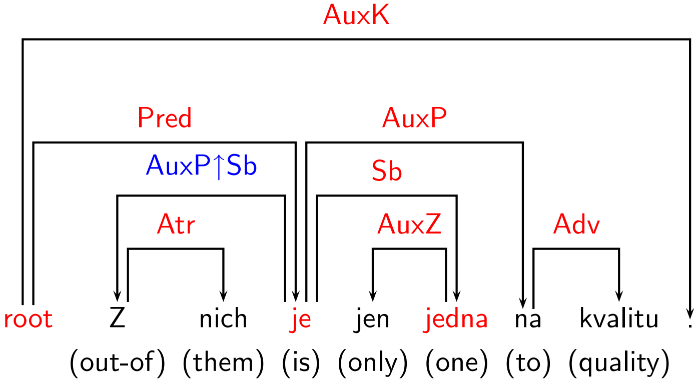
Pseudo-Projective Parsing

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



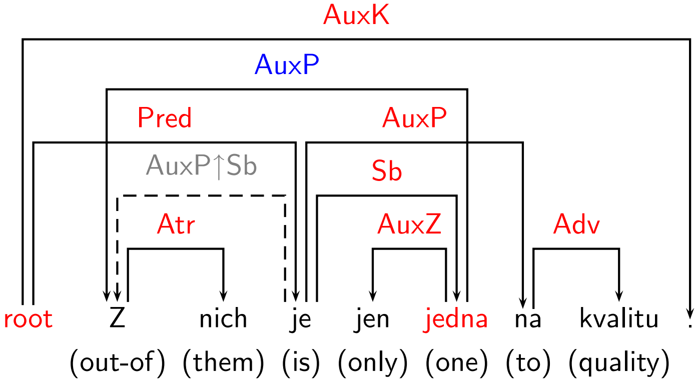
Pseudo-Projective Parsing

- Deprojectivize parser output:
  - Top-down, breadth-first search for real head
  - Search constrained by extended arc label



Pseudo-Projective Parsing

- Deprojectivize parser output:
  - Top-down, breadth-first search for real head
  - Search constrained by extended arc label



Corrective Modeling

- Conditional probability model

$$P(h'_i | w_i, N(h_i))$$

  for correcting the head h_i of word w_i to h'_i, restricted to the local neighboorhood $N(h_i)$ of h_i
- Model trained on parser output and gold standard parses (MaxEnt estimation)
- Post-processing:
  - For every word w_i, replace h_i by $\mathrm{argmax}_{h'_i} P(h'_i | w_i, N(h_i))$.

Second-Order Non-Projective Parsing

- The score of a dependency tree y for input sentence x is

$$\sum_{(i,k,j) \in y} s(i, k, j)$$

  where k and j are adjacent, same-side children of i in y.
- The highest scoring projective dependency tree can be computed exactly in $O(n^3)$ time using Eisner's algorithm.
- The highest scoring non-projective dependency tree can be approximated with a greedy post-processing procedure:
  - While improving the global score of the dependency tree, replace an arc $h_i \rightarrow w_i$ by $h'_i \rightarrow w_i$, greedily selecting the substitution that gives the greatest improvement.

State of the Art – Czech

- Evaluation:
  - Prague Dependency Treebank (PDT)
  - Unlabeled accuracy per word (W) and per sentence (S)
    - Non-projective spanning tree parsing [McDonald et al. 2005b]
    - Corrective modeling on top of the Charniak parser [Hall and Novák 2005]
    - Approximate non-projective parsing on top of a second-order projective spanning tree parser [McDonald and Pereira 2006]
    - Pseudo-projective parsing on top of a deterministic classifier-based parser [Nilsson et al. 2006]

Parser	W	S
McDonald and Pereira	85.2	35.9
Hall and Novák	85.1	—
Nilsson et al.	84.6	37.7
McDonald et al.	84.4	32.3
Charniak	84.4	–

State of the Art – Multilingual Parsing

- CoNLL-X Shared Task: 12 (13) languages
- Organizers: Sabine Buchholz, Erwin Marsi, Yuval Krymolowski, Amit Dubey
- Main evaluation metric: Labeled accuracy per word
- Top scores ranging from 91.65 (Japanese) to 65.68 (Turkish)
- Top systems (over all languages):
  - Approximate second-order non-projective spanning tree parsing with online learning (MIRA) [McDonald et al. 2006]
  - Labeled deterministic pseudo-projective parsing with support vector machines [Nivre et al. 2006]

Pros and Cons of Dependency Parsing

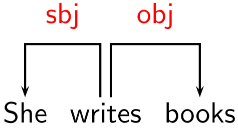
- What are the advantages of dependency-based methods?
- What are the disadvantages?
- Four types of considerations:
  - Complexity
  - Transparency
  - Word order
  - Expressivity

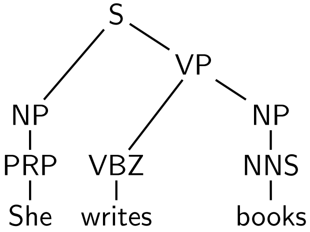
Complexity

- Practical complexity:
  - Given the Single-Head constraint, parsing a sentence $x = w_1, \ldots, w_n$ 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:
    - Lexicalized parsing in $O(n^3)$ time [Eisner 1996b]

Transparency

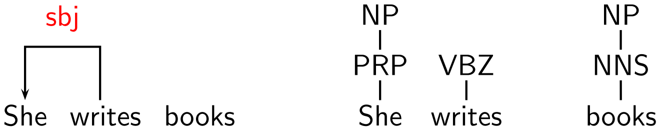
- Direct encoding of predicate-argument structure





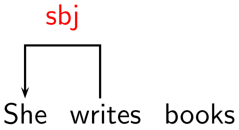
Transparency

- Direct encoding of predicate-argument structure
- Fragments directly interpretable



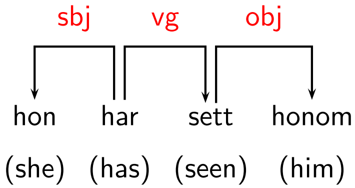
Transparency

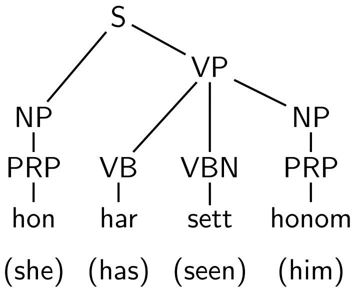
- Direct encoding of predicate-argument structure
- Fragments directly interpretable
- But only with labeled dependency graphs



Word Order

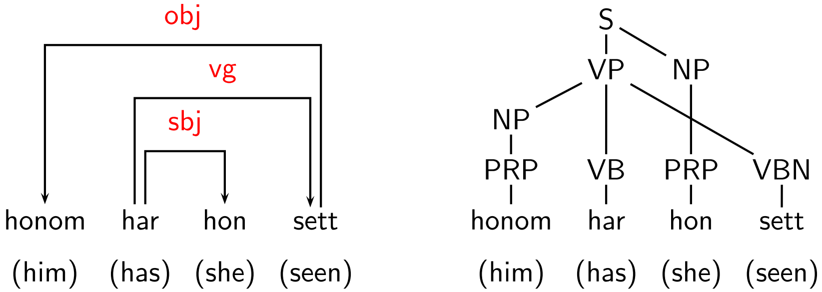
- Dependency structure independent of word order
- Suitable for free word order languages (cf. German results)





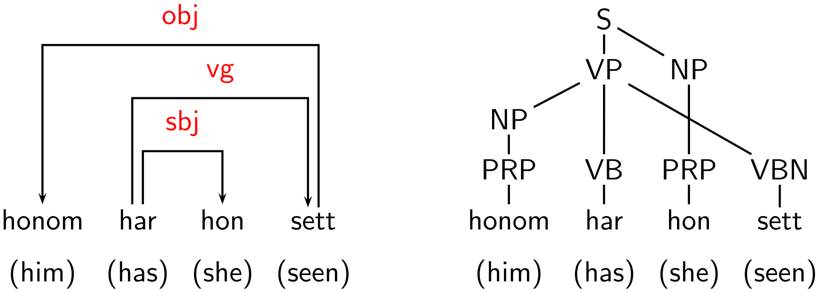
Word Order

- Dependency structure independent of word order
- Suitable for free word order languages (cf. German results)



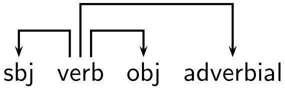
Word Order

- 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].



V, VP or S modification?

- What about labeled non-projective dependency structures?

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
- Where to get help and information?
  - Dependency parsing wiki

Parsers

- Trainable parsers
- Parsers with manually written grammars

Parsers

- Trainable parsers
- Parsers with manually written grammars

- Concentrate on freely available parsers

Trainable Parsers

- Jason Eisner's **probabilistic dependency parser**
  - Based on bilexical grammar
  - Contact Jason Eisner: jason@cs.jhu.edu
  - Written in LISP
- Ryan McDonald's **MSTParser**
  - Based on the algorithms of [McDonald et al. 2005a, McDonald et al. 2005b]
  - URL: `http://www.seas.upenn.edu/~ryantm/software/MSTParser/`
  - Written in JAVA

Trainable Parsers (2)

- Joakim Nivre's **MaltParser**
  - Inductive dependency parser with memory-based learning and SVMs
  - URL:
    `http://w3.msi.vxu.se/~nivre/research/MaltParser.html`
  - Executable versions are available for Solaris, Linux, Windows, and MacOS (open source version planned for fall 2006)

Parsers for Specific Languages

- Dekang Lin's **Minipar**
  - Principle-based parser
  - Grammar for English
  - URL: `http://www.cs.ualberta.ca/~lindek/minipar.htm`
  - Executable versions for Linux, Solaris, and Windows
- Wolfgang Menzel's **CDG Parser**:
  - Weighted constraint dependency parser
  - Grammar for German, (English under construction)
  - Online demo: `http://nats-www.informatik.uni-hamburg.de/Papa/ParserDemo`
  - Download: `http://nats-www.informatik.uni-hamburg.de/download`

Parsers for Specific Languages (2)

- Taku Kudo's **CaboCha**
  - Based on algorithms of [Kudo and Matsumoto 2002], uses SVMs
  - URL: `http://www.chasen.org/~taku/software/cabocha/`
  - Web page in Japanese
- Gerold Schneider's **Pro3Gres**
  - Probability-based dependency parser
  - Grammar for English
  - URL: `http://www.ifi.unizh.ch/CL/gschneid/parser/`
  - Written in PROLOG
- Daniel Sleator's & Davy Temperley's **Link Grammar Parser**
  - Undirected links between words
  - Grammar for English
  - URL: `http://www.link.cs.cmu.edu/link/`

Treebanks

- 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-Sabanci Turkish Treebank

Dependency Treebanks (2)

- Prague Arabic Dependency Treebank
  - ca. 100 000 words
  - Available from LDC, license fee
    (CoNLL-X shared task data, catalogue number LDC2006E01)
  - URL: `http://ufal.mff.cuni.cz/padt/`
- Prague Dependency Treebank
  - 1.5 million words
  - 3 layers of annotation: morphological, syntactical, tectogrammatical
  - Available from LDC, license fee
    (CoNLL-X shared task data, catalogue number LDC2006E02)
  - URL: `http://ufal.mff.cuni.cz/pdt2.0/`

Dependency Treebanks (3)

- Danish Dependency Treebank
  - ca. 5 500 trees
  - Annotation based on Discontinuous Grammar [Kromann 2005]
  - Freely downloadable
  - URL: `http://www.id.cbs.dk/~mtk/treebank/`
- Bosque, Floresta sintá(c)tica
  - ca. 10 000 trees
  - Freely downloadable
  - URL: `http://acdc.linguateca.pt/treebank/info_floresta_English.html`

Dependency Treebanks (4)

- Slovene Dependency Treebank
  - ca. 30 000 words
  - Freely downloadable
  - URL: `http://nl.ijs.si/sdt/`
- METU-Sabanci Turkish Treebank
  - ca. 7 000 trees
  - Freely available, license agreement
  - URL: `http://www.ii.metu.edu.tr/~corpus/treebank.html`

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

Constituent Treebanks (2)

- Penn Treebank
  - ca. 1 million words
  - Available from LDC, license fee
  - URL: `http://www.cis.upenn.edu/~treebank/home.html`
  - Dependency conversion rules, available from e.g. [Collins 1999]
  - For conversion with arc labels: Penn2Malt: `http://w3.msi.vxu.se/~nivre/research/Penn2Malt.html`
- BulTreebank
  - ca. 14 000 sentences
  - URL: `http://www.bultreebank.org/`
  - Dependency version available from Kiril Simov (`kivs@bultreebank.org`)

Constituent Treebanks (3)

- Penn Chinese Treebank
  - ca. 4 000 sentences
  - Available from LDC, license fee
  - URL: `http://www.cis.upenn.edu/~chinese/ctb.html`
  - For conversion with arc labels: Penn2Malt: `http://w3.msi.vxu.se/~nivre/research/Penn2Malt.html`
- Sinica Treebank
  - ca. 61 000 sentences
  - Available Academia Sinica, license fee
  - URL: `http://godel.iis.sinica.edu.tw/CKIP/engversion/treebank.htm`
  - Dependency version available from Academia Sinica

Constituent Treebanks (4)

- Alpino Treebank for Dutch
  - ca. 150 000 words
  - Freely downloadable
  - URL: `http://www.let.rug.nl/vannoord/trees/`
  - Dependency version downloadable at `http://nextens.uvt.nl/~conll/free_data.html`
- TIGER/NEGRA
  - ca. 50 000/20 000 sentences
  - Freely available, license agreement
  - TIGER URL:
    `http://www.ims.uni-stuttgart.de/projekte/TIGER/TIGERCorpus/`
    NEGRA URL:
    `http://www.coli.uni-saarland.de/projects/sfb378/negra-corpus/`
  - Dependency version of TIGER is included in release

Constituent Treebanks (5)

- TüBa-D/Z
  - ca. 22 000 sentences
  - Freely available, license agreement
  - URL: `http://www.sfs.uni-tuebingen.de/en_tuebadz.shtml`
  - Dependency version available from SfS Tübingen
- TüBa-J/S
  - Dialog data
  - ca. 18 000 sentences
  - Freely available, license agreement
  - Dependency version available from SfS Tübingen
  - URL: `http://www.sfs.uni-tuebingen.de/en_tuebajs.shtml` (under construction)

Constituent Treebanks (6)

- Cast3LB
  - ca. 18 000 sentences
  - URL: `http://www.dlsi.ua.es/projectes/3lb/index_en.html`
  - Dependency version available from Toni Martí (amarti@ub.edu)
- Talbanken05
  - ca. 300 000 words
  - Freely downloadable
  - URL: `http://w3.msi.vxu.se/~nivre/research/Talbanken05.html`
  - Dependency version also available

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.

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))

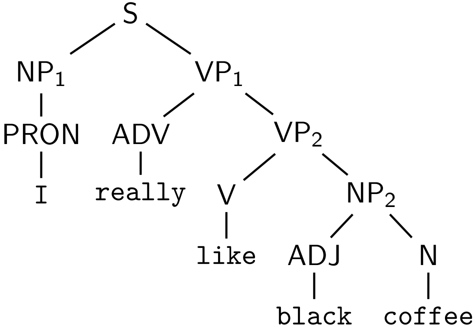
- First line: The head of an S constituent is the first Aux daughter from the right; if there is no Aux, then the first VP, etc.

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))

Lin's Conversion - Example

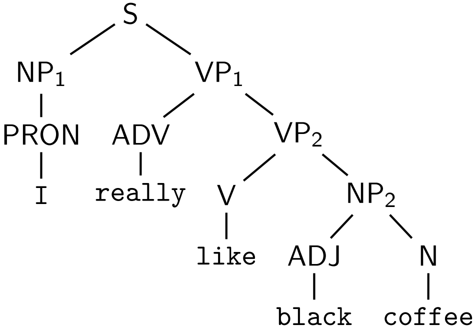
(S	right-to-left	(Aux VP NP AP PP))
(VP	left-to-right	(V VP))
(NP	right-to-left	(Pron N NP))



root	head	lex. head

Lin's Conversion - Example

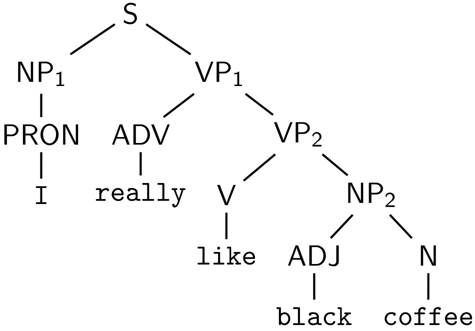
(S	right-to-left	(Aux VP NP AP PP))
(VP	left-to-right	(V VP))
(NP	right-to-left	(Pron N NP))



root	head	lex. head
S	VP_1	??

Lin's Conversion - Example

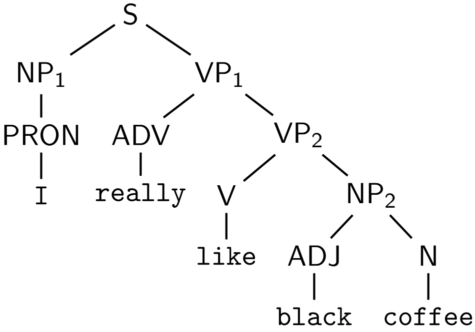
(S right-to-left (Aux VP NP AP PP))
(VP left-to-right (V VP))
(NP right-to-left (Pron N NP))



root	head	lex. head
VP_1	VP_2	??

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))



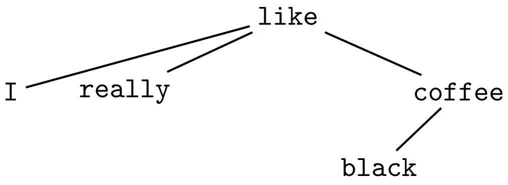
root	head	lex. head
S	VP_1	like
VP_1	VP_2	like
VP_2	V	like

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*

Lin's Conversion - Example (2)

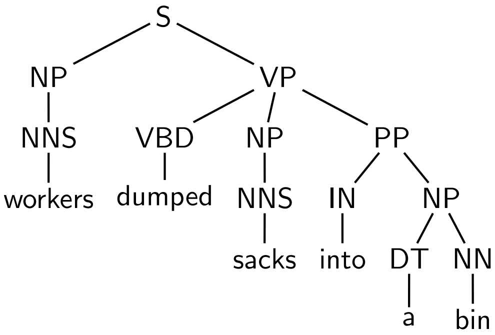
- The head of a phrase dominates all sisters.
- VP_1 governs NP_1 $\Rightarrow$ *like* governs *I*
- VP_2 governs ADV $\Rightarrow$ *like* governs *really*



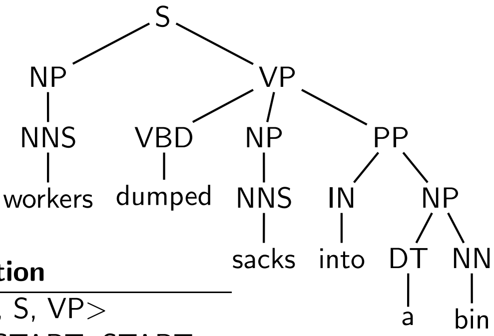
From Structural to Labeled Conversion

- 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 $\mathit{headword}(Y_h) \rightarrow \mathit{headword}(Y_d)$ is derived from rule $X \rightarrow Y_1 \ldots Y_n$, the relation is $<Y_d, X, Y_h>$

Collins' Example

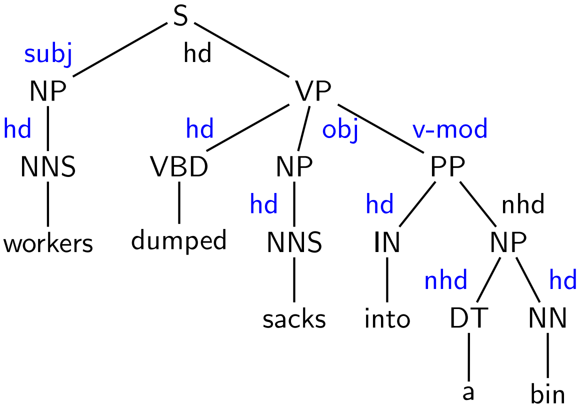


Collins' Example

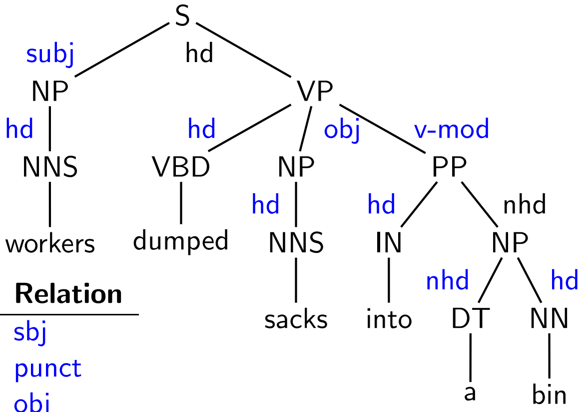


Dependency	Relation
dumped → workers	<NP, S, VP>
dumped → *root*	<S, START, START>
dumped → sacks	<NP, VP, VBD>
dumped → into	<PP, VP, VBD>
into → bin	<NP, PP, IN>
bin → a	<DT, NP, NN>

Example with Grammatical Functions



Example with Grammatical Functions



Dependency	Relation
dumped $\longrightarrow$ workers	sbj
dumped $\longrightarrow$ *root*	punct
dumped $\longrightarrow$ sacks	obj
dumped $\longrightarrow$ into	v-mod
into $\longrightarrow$ bin	nhd
bin $\longrightarrow$ a	nhd

Evaluation

evaluation scores:

- *Exact match* (= S)
  percentage of correctly parsed sentences
- *Attachment score* (= W)
  percentage of words that have the correct head
- For single dependency types (labels):
  - *Precision*
  - *Recall*
  - F_β *measure*
- *correct root*
  percentage of sentences that have the correct root

Evaluation

evaluation scores:

- *Exact match* (= S)
  percentage of correctly parsed sentences
- *Attachment score* (= W)
  percentage of words that have the correct head
- For single dependency types (labels):
  - *Precision*
  - *Recall*
  - *F_β measure*
- *correct root*
  percentage of sentences that have the correct root

Evaluation

evaluation scores:

- *Exact match* (= S)
  percentage of correctly parsed sentences
- *Attachment score* (= W)
  percentage of words that have the correct head
- For single dependency types (labels):
  - *Precision*
  - *Recall*
  - F_β *measure*
- *correct root*
  percentage of sentences that have the correct root

Evaluation

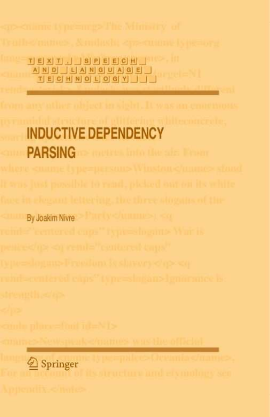
evaluation scores:

- *Exact match* (= S)
  percentage of correctly parsed sentences
- *Attachment score* (= W)
  percentage of words that have the correct head
- For single dependency types (labels):
  - *Precision*
  - *Recall*
  - F_β *measure*
- *correct root*
  percentage of sentences that have the correct root

- All labeled and unlabeled

Further Information

- Dependency parsing wiki
  `http://depparse.uvt.nl`
- Book by Joakim: *Inductive Dependency Parsing*



Outlook

- Future trends (observed or predicted):
  - Multilingual dependency parsing
    - CoNLL Shared Task
    - Comparative error analysis
    - Typological diversity and parsing methods
  - Non-projective dependency parsing
    - Non-projective parsing algorithms
    - Post-processing of projective approximations
    - Other approaches
  - Global constraints
    - Grammar-driven approaches
    - *N*th-order spanning tree parsing
    - Hybrid approaches [Foth et al. 2004]
  - Dependency and constituency
    - What are the essential differences?
    - Very few theoretical results

- 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. Determinstic 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)*.
- 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.
- Ralph Debusmann, Denys Duchier, and Geert-Jan M. Kruijff. 2004. Extensible dependency grammar: A new methodology. In *Proceedings of the Workshop on Recent Advances in Dependency Grammar*, pages 78–85.

- Amit Dubey. 2005. What to do when lexicalization fails: Parsing German with suffix analysis and smoothing. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, Ann Arbor, MI.
- Denys Duchier and Ralph Debusmann. 2001. Topological dependency trees: A constraint-based account of linear precedence. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 180–187.
- Denys Duchier. 1999. Axiomatizing dependency parsing using set constraints. In *Proceedings of the Sixth Meeting on Mathematics of Language*, pages 115–126.
- Denys Duchier. 2003. Configuration of labeled trees under lexicalized constraints and principles. *Research on Language and Computation*, 1:307–336.
- J. Edmonds. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards*, 71B:233–240.
- Jason M. Eisner. 1996a. An empirical comparison of probability models for dependency grammar. Technical Report IRCS-96-11, Institute for Research in Cognitive Science, University of Pennsylvania.
- Jason M. Eisner. 1996b. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING)*, pages 340–345.

- Jason M. Eisner. 2000. Bilexical grammars and their cubic-time parsing algorithms. In Harry Bunt and Anton Nijholt, editors, *Advances in Probabilistic and Other Parsing Technologies*, pages 29–62. Kluwer.
- Kilian Foth, Ingo Schröder, and Wolfgang Menzel. 2000. A transformation-based parsing technique with anytime properties. In *Proceedings of the Sixth International Workshop on Parsing Technologies (IWPT)*, pages 89–100.
- Kilian Foth, Michael Daum, and Wolfgang Menzel. 2004. A broad-coverage parser for German based on defeasible constraints. In *Proceedings of KONVENS 2004*, pages 45–52.
- Haim Gaifman. 1965. Dependency systems and phrase-structure systems. *Information and Control*, 8:304–337.
- Keith Hall and Vaclav Novák. 2005. Corrective modeling for non-projective dependency parsing. In *Proceedings of the 9th International Workshop on Parsing Technologies (IWPT)*, pages 42–52.
- Johan Hall, Joakim Nivre, and Jens Nilsson. 2006. Discriminative classifiers for deterministic dependency parsing. In *Proceedings of COLING-ACL*.
- Mary P. Harper and R. A. Helzerman. 1995. Extensions to constraint dependency parsing for spoken language processing. *Computer Speech and Language*, 9:187–234.

- David G. Hays. 1964. Dependency theory: A formalism and some observations. *Language*, 40:511–525.
- Peter Hellwig. 1986. Dependency unification grammar. In *Proceedings of the 11th International Conference on Computational Linguistics (COLING)*, pages 195–198.
- Peter Hellwig. 2003. Dependency unification grammar. In Vilmos Agel, Ludwig M. Eichinger, Hans-Werner Eroms, Peter Hellwig, Hans Jürgen Heringer, and Hening Lobin, editors, *Dependency and Valency*, pages 593–635. Walter de Gruyter.
- Richard A. Hudson. 1984. *Word Grammar*. Blackwell.
- Richard A. Hudson. 1990. *English Word Grammar*. Blackwell.
- 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.
- Timo Järvinen and Pasi Tapanainen. 1998. Towards an implementable dependency grammar. In Sylvain Kahane and Alain Polguère, editors, *Proceedings of the Workshop on Processing of Dependency-Based Grammars*, pages 1–10.
- Fred Karlsson, Atro Voutilainen, Juha Heikkilä, and Arto Anttila, editors. 1995. *Constraint Grammar: A language-independent system for parsing unrestricted text*. Mouton de Gruyter.

- Fred Karlsson. 1990. Constraint grammar as a framework for parsing running text. In Hans Karlgren, editor, *Papers presented to the 13th International Conference on Computational Linguistics (COLING)*, pages 168–173.
- Matthias Trautner Kromann. 2005. *Discontinuous Grammar: A Dependency-Based Model of Human Parsing and Language Learning*. Doctoral Dissertation, Copenhagen Business School.
- Sandra Kübler, Erhard W. Hinrichs, and Wolfgang Maier. 2006. Is it really that difficult to parse German? In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing, EMNLP 2006*, Sydney, Australia.
- 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.
- Vincenzio Lombardo and Leonardo Lesmo. 1996. An Earley-type recognizer for dependency grammar. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING)*, pages 723–728.

- Hiroshi Maruyama. 1990. Structural disambiguation with constraint propagation. In *Proceedings of the 28th Meeting of the Association for Computational Linguistics (ACL)*, pages 31–38.
- Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 81–88.
- 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.
- Ryan McDonald, Kevin Lerman, and Fernando Pereira. 2006. Multilingual dependency analysis with a two-stage discriminative parser. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL)*.
- Igor Mel'čuk. 1988. *Dependency Syntax: Theory and Practice*. State University of New York Press.

- Wolfgang Menzel and Ingo Schröder. 1998. Decision procedures for dependency parsing using graded constraints. In Sylvain Kahane and Alain Polguère, editors, *Proceedings of the Workshop on Processing of Dependency-Based Grammars*, pages 78–87.
- Peter Neuhaus and Norbert Bröker. 1997. The complexity of recognition of linguistically adequate dependency grammars. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL) and the 8th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 337–343.
- Jens Nilsson, Joakim Nivre, and Johan Hall. 2006. Graph transformations in data-driven dependency parsing. In *Proceedings of COLING-ACL*.
- 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.
- Ingo Schröder. 2002. *Natural Language Parsing with Graded Constraints*. Ph.D. thesis, Hamburg University.
- Petr Sgall, Eva Hajičová, and Jarmila Panevová. 1986. *The Meaning of the Sentence in Its Pragmatic Aspects*. Reidel.
- Daniel Sleator and Davy Temperley. 1991. Parsing English with a link grammar. Technical Report CMU-CS-91-196, Carnegie Mellon University, Computer Science.
- Pasi Tapanainen and Timo Järvinen. 1997. A non-projective dependency parser. In *Proceedings of the 5th Conference on Applied Natural Language Processing*, pages 64–71.

- Lucien Tesnière. 1959. *Éléments de syntaxe structurale*. Editions Klincksieck.
- 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.
- A. M. Zwicky. 1985. Heads. *Journal of Linguistics*, 21:1–29.