Lost in the Woods?

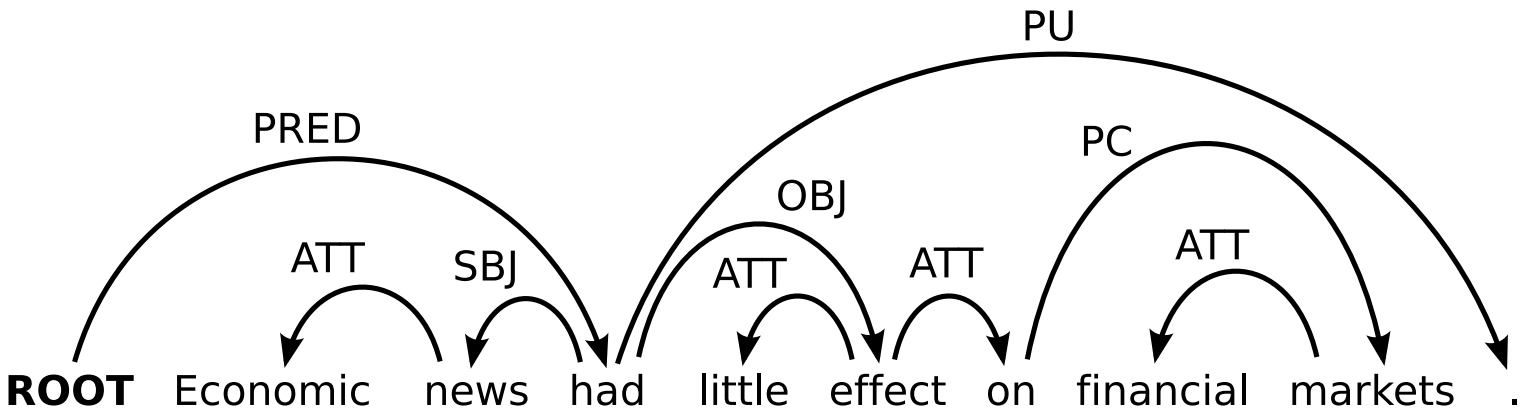
Transition-Based Dependency Parsing with Non-Projective Trees

Joakim Nivre

Uppsala University
Department of Linguistics and Philology
joakim.nivre@lingfil.uu.se

Introduction

- Syntactic parsing of natural language
  - Who does what to whom?
- Dependency-based syntactic representations
  - Long tradition in descriptive and theoretical linguistics
  - Increasingly popular in computational linguistics

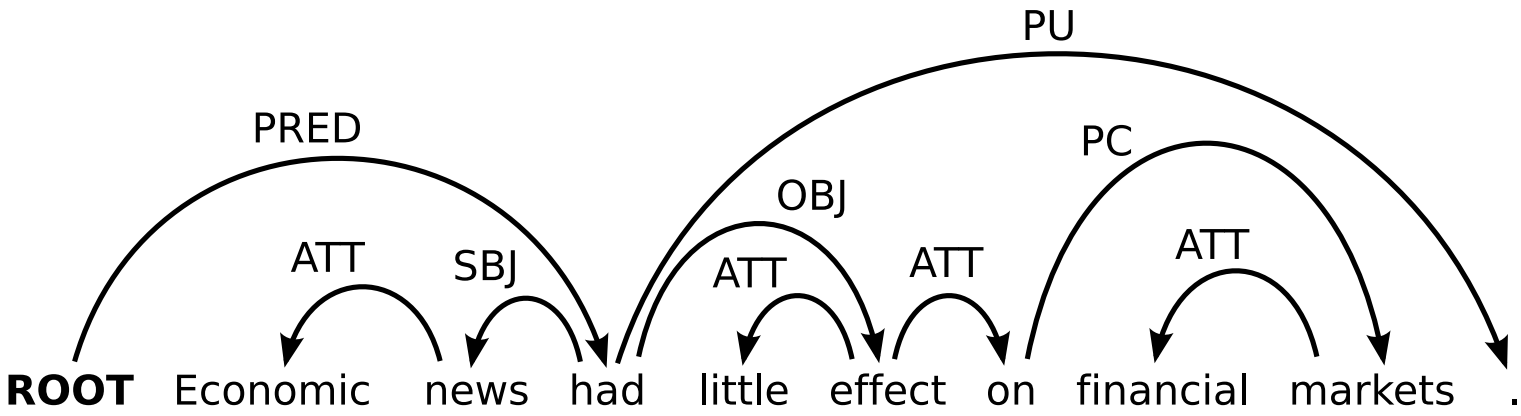


Why Popular?

- Usefulness of dependency structures in applications
  - Transparent encoding of predicate-argument structure
  - Interface from parser to downstream application
- Availability of dependency treebanks
  - Dependency annotation natural for many languages
  - Prague Dependency Treebank as a model
  - Data sets from CoNLL shared tasks (2006–2009)
- Advancement of data-driven dependency parsing
  - Robust and efficient parsing methods
  - High accuracy for many languages

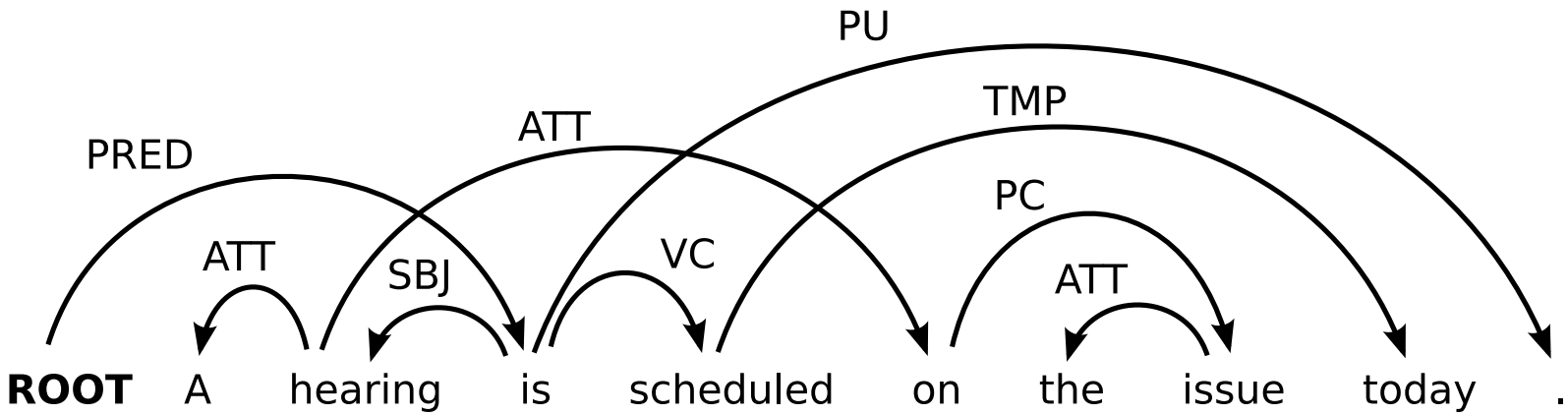
Dependency Syntax

- The basic idea:
  - Syntactic structure consists of lexical items, linked by binary asymmetric relations called dependencies.
- Many different theoretical frameworks



Projectivity

- Discontinuous constructions?
  - A dependency tree is projective if every subtree covers a contiguous substring of the sentence.
- Most theoretical frameworks do not assume projectivity (but many parsers do).



Non-Projectivity in Natural Language

Language	Sentences	Dependencies
Arabic [Hajič et al. 2004]	11.2%	0.4%
Basque [Aduriz et al. 2003]	26.2%	2.9%
Czech [Hajič et al. 2001]	23.2%	1.9%
Danish [Kromann 2003]	15.6%	1.0%
Greek [Prokopidis et al. 2005]	20.3%	1.1%
Russian [Boguslavsky et al. 2000]	10.6%	0.9%
Slovene [Džeroski et al. 2006]	22.2%	1.9%
Turkish [Oflazer et al. 2003]	11.6%	1.5%

Plan for the Talk

- Part 1:
  - Transition-based dependency parsing
  - Robust and efficient parsing for projective trees
- Part 2:
  - Dealing with non-projective trees
  - Comparison of three methods:
    - Pseudo-projective parsing
    - Extended arc transitions
    - Online reordering

Transition-Based Dependency Parsing

Overview of the Approach

- The basic idea:
  - Define a transition system for dependency parsing
  - Train a model for scoring possible transitions
  - Parse by searching for the optimal transition sequence
- Advantages:
  - Highly efficient parsers with low complexity
  - Rich history-based feature models for disambiguation

Dependency Trees Formally

- A dependency tree for a sentence $S = w_1, \ldots, w_n$, given a set L of labels, is a labeled directed tree T, consisting of
  - a set V of nodes, labeled with words (including **ROOT**),
  - a set $A \subseteq V \times L \times V$ of arcs, labeled with dependency types,
  - a linear precedence order $<$ on V,

  with **ROOT** as the unique root

- Note:
  - An arc (w_i, l, w_j) has head w_i, label l and dependent w_j
  - Alternative notation: $w_i \xrightarrow{l} w_j$ (or $w_i \rightarrow w_j$)

Transition System: Configurations

- A parser configuration is a triple $c = (S, Q, A)$, where
  - S = a stack $[\ldots, w_i]_S$ of partially processed nodes,
  - Q = a queue $[w_j, \ldots]_Q$ of remaining input nodes,
  - A = a set of labeled arcs (w_i, l, w_j).
- Initialization:

  $([w_0]_S, [w_1, \ldots, w_n]_Q, \{\})$

  NB: w_0 = **ROOT**
- Termination:

  $([w_0]_S, []_Q, A)$

Transition System: Transitions

- Left-Arc(l)

$$\frac{([\ldots, w_i, w_j]_S, \quad Q, \quad A)}{([\ldots, w_j]_S, \quad Q, \quad A \cup \{(w_j, l, w_i)\})} \quad [i \neq 0]$$

- Right-Arc(l)

$$\frac{([\ldots, w_i, w_j]_S, \quad Q, \quad A)}{([\ldots, w_i]_S, \quad Q, \quad A \cup \{(w_i, l, w_j)\})}$$

- Shift

$$\frac{([\ldots]_S, \quad [w_i, \ldots]_Q, \quad A)}{([\ldots, w_i]_S, \quad [\ldots]_Q, \quad A)}$$

Oracle Parsing

- Given an oracle o that correctly predicts the next transition $o(c)$, parsing is deterministic:

PARSE$(w_1, \ldots, w_n)$
1 $c \leftarrow ([w_0]_S, [w_1, \ldots, w_n]_Q, \{\})$
2 **while** $Q_c \neq [\,]$ **or** $|S_c| > 1$
3 $\quad t \leftarrow o(c)$
4 $\quad c \leftarrow t(c)$
5 **return** $T = (\{w_0, w_1, \ldots, w_n\}, A_c)$

Parsing Example

[**ROOT**]$_S$ [Economic, news, had, little, effect, on, financial, markets, .]$_Q$

ROOT Economic news had little effect on financial markets .

Parsing Example

[**ROOT**, Economic]$_S$ [news, had, little, effect, on, financial, markets, .]$_Q$

ROOT Economic news had little effect on financial markets .

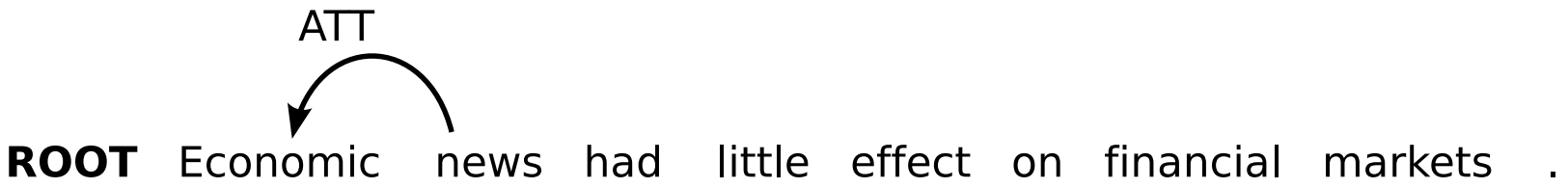
Parsing Example

[**ROOT**, Economic, news]$_S$ [had, little, effect, on, financial, markets, .]$_Q$

ROOT Economic news had little effect on financial markets .

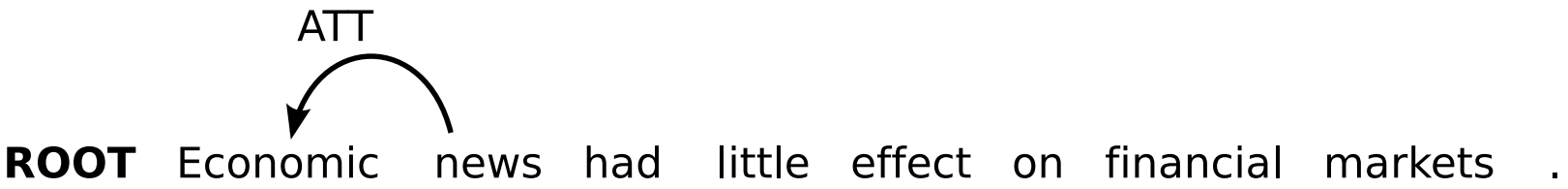
Parsing Example

[**ROOT**, news]$_S$ [had, little, effect, on, financial, markets, .]$_Q$



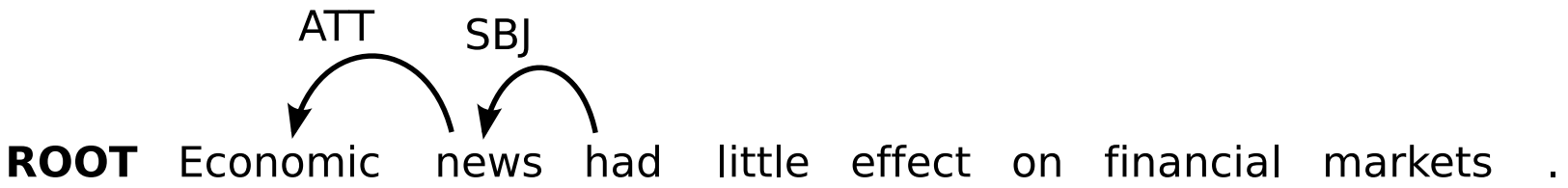
Parsing Example

[**ROOT**, news, had]$_S$ [little, effect, on, financial, markets, .]$_Q$



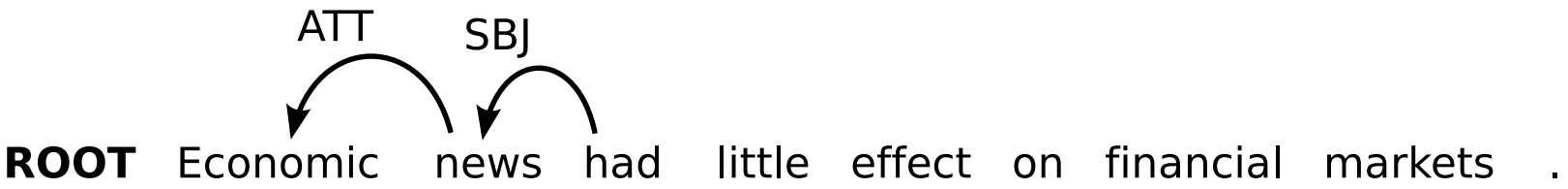
Parsing Example

[**ROOT**, had]$_S$ [little, effect, on, financial, markets, .]$_Q$



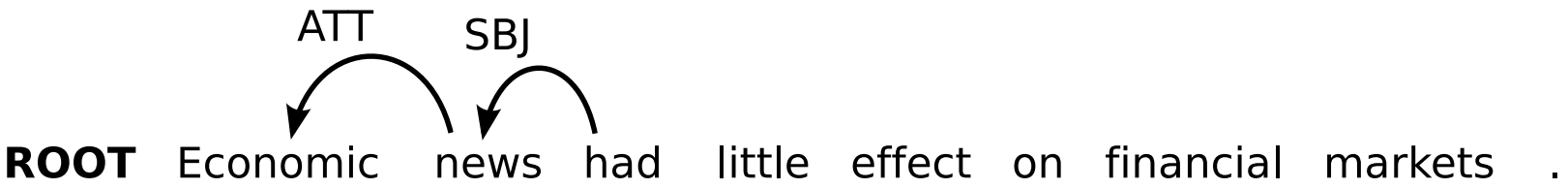
Parsing Example

[**ROOT**, had, little]$_S$ [effect, on, financial, markets, .]$_Q$



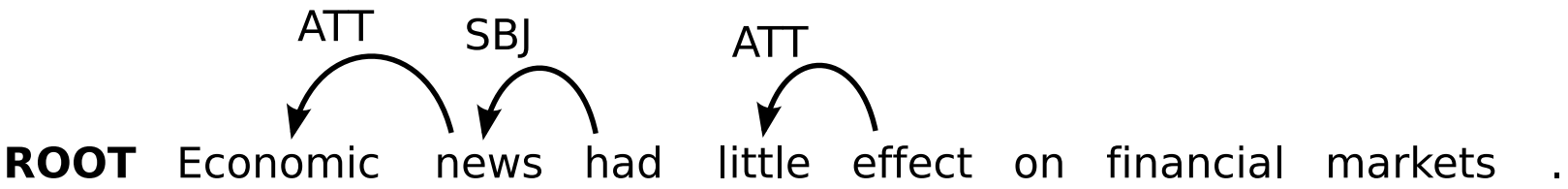
Parsing Example

[**ROOT**, had, little, effect]$_S$ [on, financial, markets, .]$_Q$



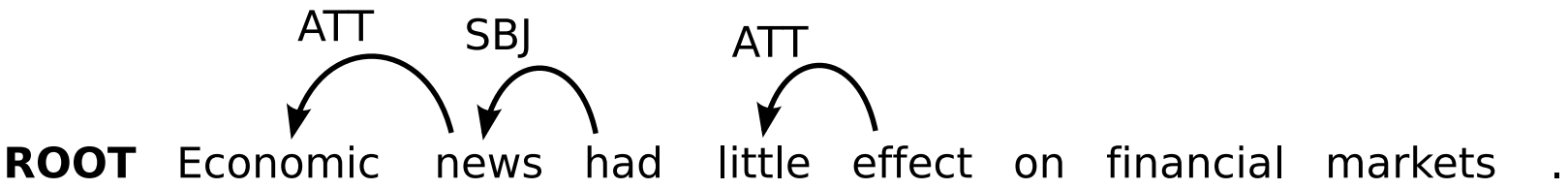
Parsing Example

[**ROOT**, had, effect]$_S$ [on, financial, markets, .]$_Q$



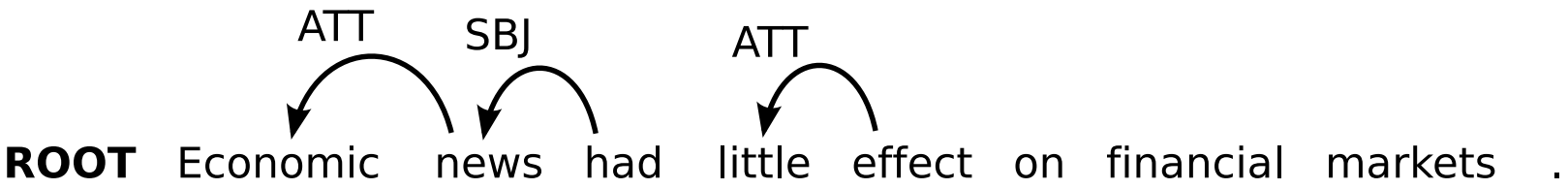
Parsing Example

[**ROOT**, had, effect, on]$_S$ [financial, markets, .]$_Q$



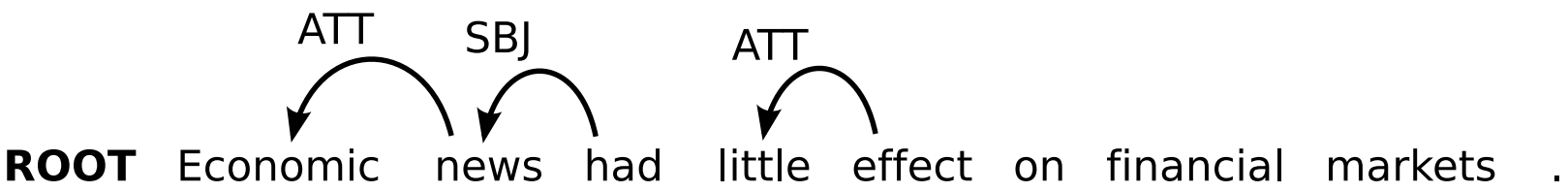
Parsing Example

[**ROOT**, had, effect, on, financial]$_S$ [markets, .]$_Q$



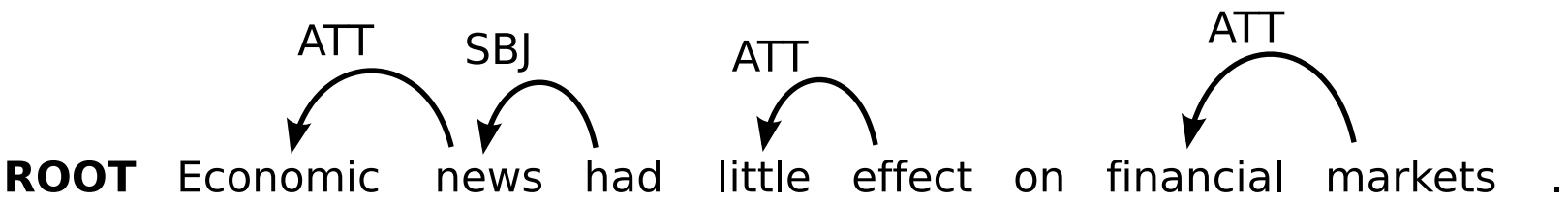
Parsing Example

[**ROOT**, had, effect, on, financial, markets]$_S$ [.]$_Q$



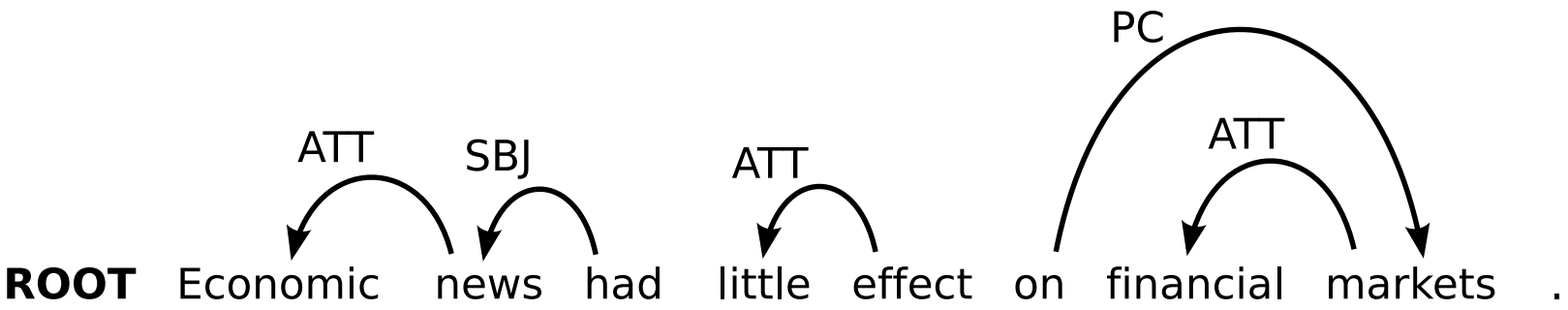
Parsing Example

[**ROOT**, had, effect, on, markets]$_S$ [.]$_Q$



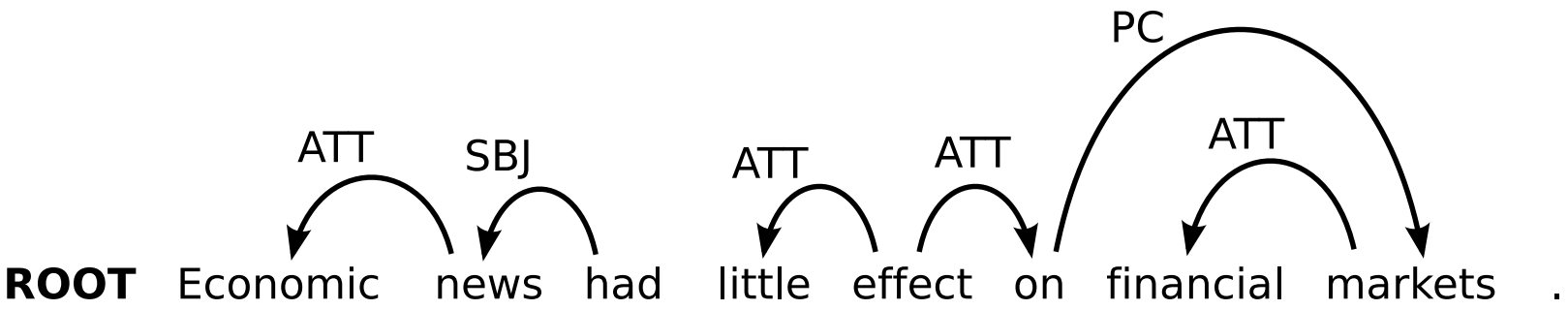
Parsing Example

[**ROOT**, had, effect, on]$_S$ [.]$_Q$



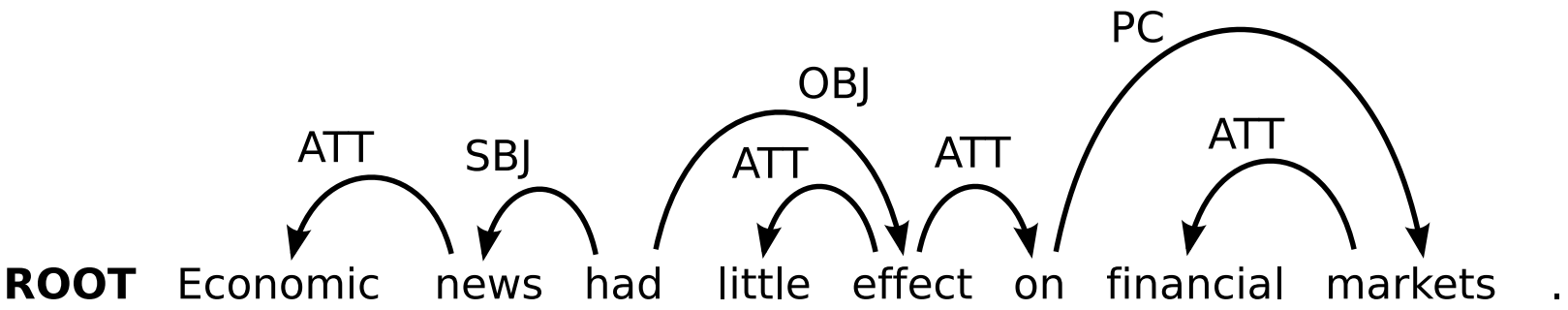
Parsing Example

[**ROOT**, had, effect]$_S$ [.]$_Q$



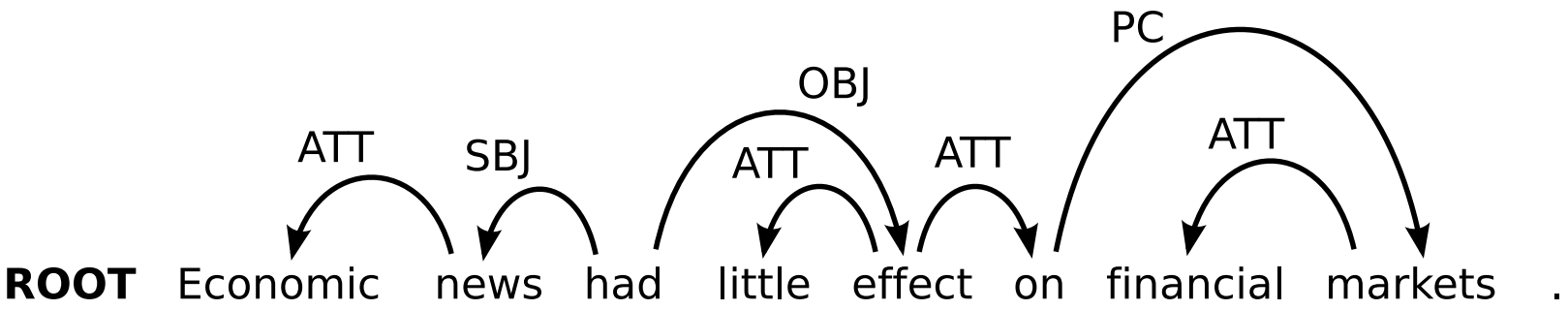
Parsing Example

[**ROOT**, had]$_S$ [.]$_Q$



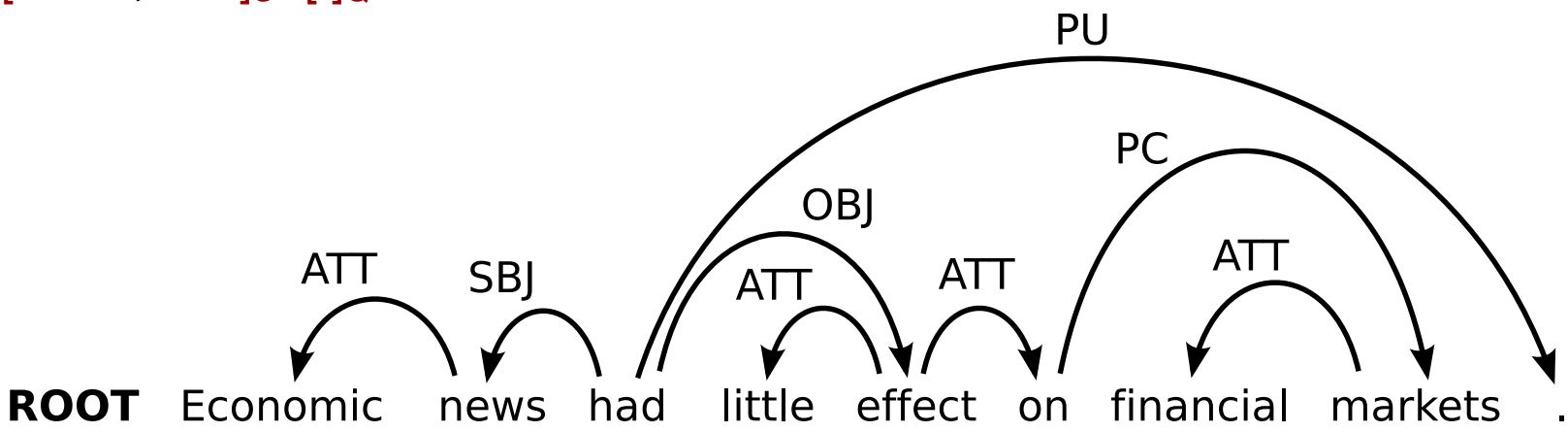
Parsing Example

[**ROOT**, had, .]$_S$ [ ]$_Q$

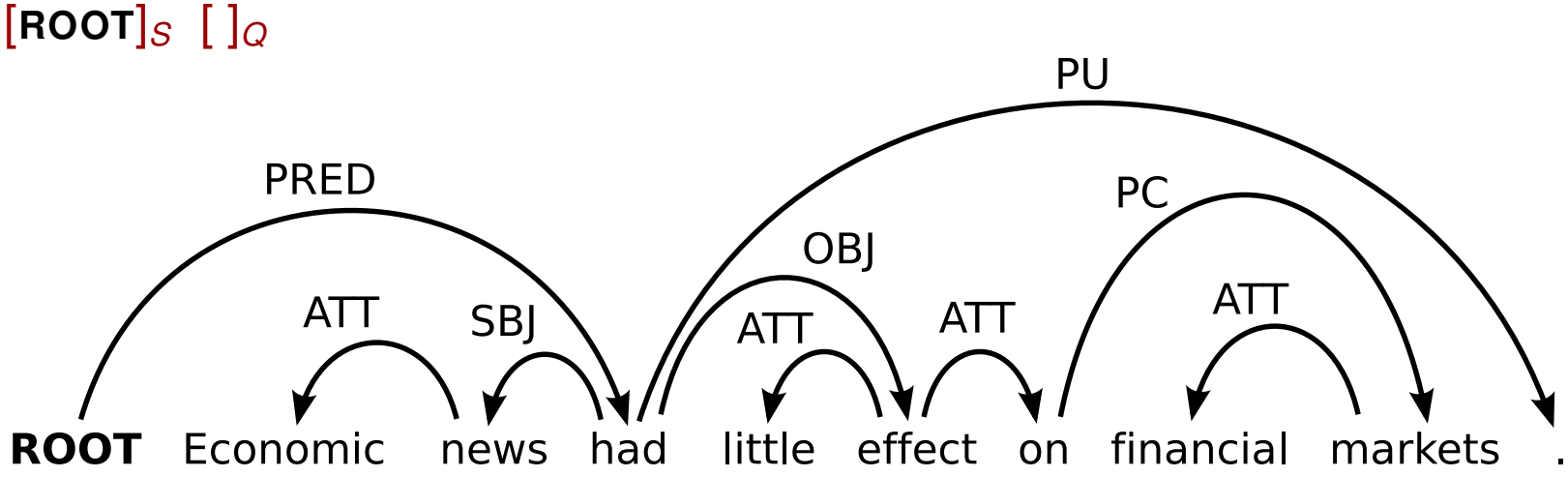


Parsing Example

$[\textbf{ROOT}, \text{had}]_S \; [\,]_Q$



Parsing Example



Algorithm Analysis

- Given an input sentence of length n, the parser terminates after exactly $2n$ transitions.
- The algorithm is sound and complete for projective dependency trees.
- The algorithm is arguably optimal with respect to
  - robustness (at least one analysis),
  - disambiguation (at most one analysis),
  - efficiency (linear time).
- Accuracy depends on how well we can approximate oracles using machine learning.

Approximating Oracles

- Feature-based scoring function:

$$\mathcal{S}(c,t) = \sum_{i=1}^{k} \mathbf{w}_i \cdot \mathbf{f}_i(c,t)$$

  where $\mathbf{w}_i$ is the (learned) weight for feature $\mathbf{f}_i(c,t)$

- Rich history-based models:
  - Features over input tokens relative to S and Q
  - Features over the (partial) dependency tree defined by A
  - Features over the (partial) transition sequence $t_1, \ldots, t_i$

  NB: No constraints on scope or combination of features

Typical Features

- Basic features:
  - PoS, Word, PoS+Word for $S_1, \ldots, S_k, Q_1, \ldots, Q_k$
  - PoS, Word, DepLabel for dependents of S_1 and S_2
  - Distance S_1–S_2
  - Left and right valency of S_1 and S_2
  - Left and right dependency label set of S_1 and S_2
- Conjoined features:
  - All pairs of S_1 and S_2 features
  - Triples of PoS features:
    - Trigrams over S and Q
    - Dependents of S_1 and S_2 conjoined with S_1+Q_1
    - Adjacent siblings of S_1 and S_2 conjoined with their head
  - Distance, valency and label sets conjoined with (pairs of) S_1 and S_2 features

Learning and Inference

- Local optimization [Yamada and Matsumoto 2003, Nivre et al. 2004]:

$$o(c) = \underset{t}{\text{argmax}}\ \mathcal{S}(c, t)$$

  - Learning to maximize accuracy of oracle prediction
  - Inference to 1-best configuration at each decision point

- Global optimization [Titov and Henderson 2007, Zhang and Clark 2008]:

$$\textsc{Parse}(w_1, \ldots, w_n) = \underset{T=(V,A_{c_m})}{\text{argmax}} \sum_{i=1}^{m} \mathcal{S}(c_{i-1}, t_i)$$

  - Learning to minimize loss over entire transition sequence
  - Inference to k-best configurations (beam search)

Summing Up

- Transition-based dependency parsing
  - Efficient thanks to low parsing complexity
  - Accurate thanks to rich non-local features
  - State of the art for projective parsing [Zhang and Nivre 2011]
- However:
  - Limited to projective dependency trees
  - What to do with discontinuous constructions?

Dealing with Non-Projective Trees

A Hard Problem

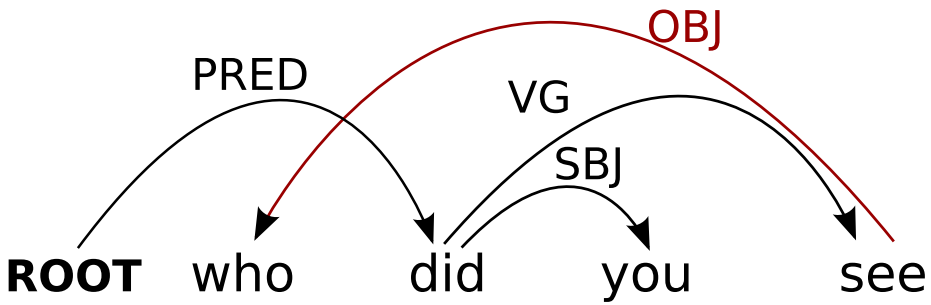
- Intractability results for parsing with non-projective trees:
  - Grammars with ID/LP separation [Neuhaus and Bröker 1997]
  - Weighted constraint grammars [Buch-Kromann 2006]
  - Spanning tree parsing with markovization or valency [McDonald and Pereira 2006, McDonald and Satta 2007]
- Complexity of deterministic parsing [Nivre 2008]:
  - $O(n)$ with projective trees, $O(n^2)$ with arbitrary trees
- Source of difficulty:
  - Word order unconstrained by tree structure (permutation)
  - Trees composed of non-adjacent subtrees (discontinuity)

Approaches

- Pseudo-projective parsing [Nivre and Nilsson 2005]
  - Preprocess training data, post-process parser output
- Extended arc transitions [Attardi 2006]
  - Transitions that add arcs between non-adjacent subtrees
- Online reordering [Nivre 2009]
  - Transitions that perform word order permutations

Pseudo-Projective Parsing

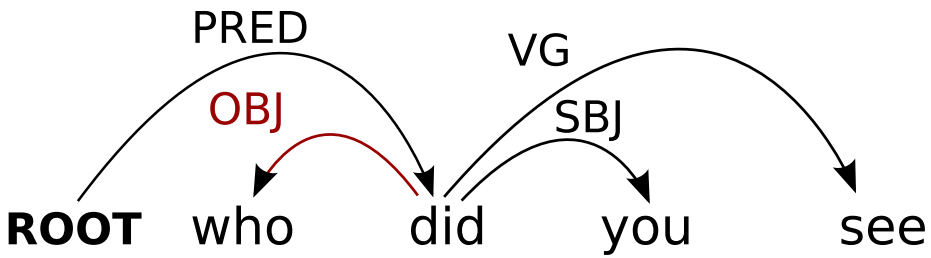
- Pseudo-projective transform [Nivre and Nilsson 2005]:
  - Attach non-projective arcs higher in the tree
  - Encode transformation in arc labels
  - Projective tree with augmented arc labels



- Data-driven parsing:
  - Transform training data to pseudo-projective trees
  - Apply (approximate) inverse transform to parser output

Pseudo-Projective Parsing

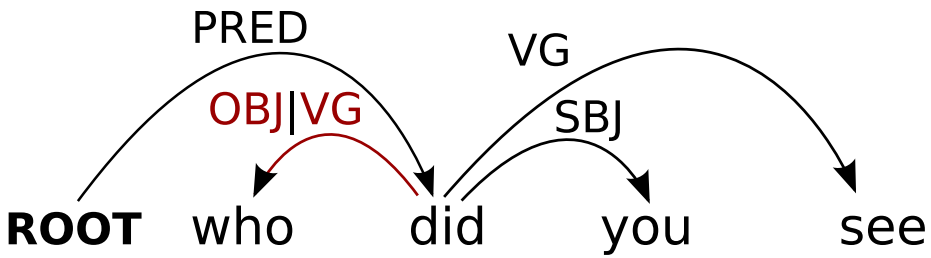
- Pseudo-projective transform [Nivre and Nilsson 2005]:
  - Attach non-projective arcs higher in the tree
  - Encode transformation in arc labels
  - Projective tree with augmented arc labels



- Data-driven parsing:
  - Transform training data to pseudo-projective trees
  - Apply (approximate) inverse transform to parser output

Pseudo-Projective Parsing

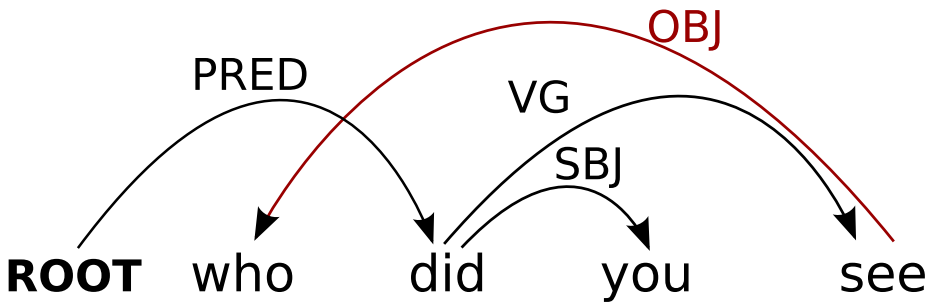
- Pseudo-projective transform [Nivre and Nilsson 2005]:
  - Attach non-projective arcs higher in the tree
  - Encode transformation in arc labels
  - Projective tree with augmented arc labels



- Data-driven parsing:
  - Transform training data to pseudo-projective trees
  - Apply (approximate) inverse transform to parser output

Pseudo-Projective Parsing

- Pseudo-projective transform [Nivre and Nilsson 2005]:
  - Attach non-projective arcs higher in the tree
  - Encode transformation in arc labels
  - Projective tree with augmented arc labels



- Data-driven parsing:
  - Transform training data to pseudo-projective trees
  - Apply (approximate) inverse transform to parser output

Extended Arc Transitions

- Add transitions for non-projective arcs [Attardi 2006]:
  - Left-Arc2(l) $\dfrac{([\ldots, w_i, w_j, w_k]_S, \quad Q, \quad A)}{([\ldots, w_j, w_k]_S, \quad Q, \quad A \cup \{(w_k, l, w_i)\})}$ $[i \neq 0]$
  - Right-Arc2(l) $\dfrac{([\ldots, w_i, w_j, w_k]_S, \quad Q, \quad A)}{([\ldots, w_i, w_j]_S, \quad Q, \quad A \cup \{(w_i, l, w_k)\})}$

$[\textbf{ROOT}]_S$ $[\text{who, did, you, see}]_Q$

ROOT who did you see

Extended Arc Transitions

- Add transitions for non-projective arcs [Attardi 2006]:
  - Left-Arc2(l) $\dfrac{([\ldots, w_i, w_j, w_k]_S, \quad Q, \quad A)}{([\ldots, w_j, w_k]_S, \quad Q, \quad A \cup \{(w_k, l, w_i)\})}$ $[i \neq 0]$
  - Right-Arc2(l) $\dfrac{([\ldots, w_i, w_j, w_k]_S, \quad Q, \quad A)}{([\ldots, w_i, w_j]_S, \quad Q, \quad A \cup \{(w_i, l, w_k)\})}$

$[\textbf{ROOT}, \text{who}]_S \;\; [\text{did}, \text{you}, \text{see}]_Q$

ROOT who did you see

Extended Arc Transitions

- Add transitions for non-projective arcs [Attardi 2006]:
  - Left-Arc2(l) $\dfrac{([\ldots, w_i, w_j, w_k]_S, \quad Q, \quad A)}{([\ldots, w_j, w_k]_S, \quad Q, \quad A \cup \{(w_k, l, w_i)\})}$ $[i \neq 0]$
  - Right-Arc2(l) $\dfrac{([\ldots, w_i, w_j, w_k]_S, \quad Q, \quad A)}{([\ldots, w_i, w_j]_S, \quad Q, \quad A \cup \{(w_i, l, w_k)\})}$

$[\textbf{ROOT}, \text{who}, \text{did}]_S \quad [\text{you}, \text{see}]_Q$

ROOT who did you see

Extended Arc Transitions

- Add transitions for non-projective arcs [Attardi 2006]:
  - Left-Arc2(l) $\dfrac{([\ldots, w_i, w_j, w_k]_S, \quad Q, \quad A)}{([\ldots, w_j, w_k]_S, \quad Q, \quad A \cup \{(w_k, l, w_i)\})}$ $[i \neq 0]$
  - Right-Arc2(l) $\dfrac{([\ldots, w_i, w_j, w_k]_S, \quad Q, \quad A)}{([\ldots, w_i, w_j]_S, \quad Q, \quad A \cup \{(w_i, l, w_k)\})}$

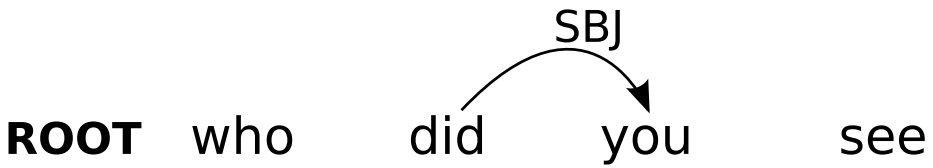
$[\textbf{ROOT}, \text{who}, \text{did}, \text{you}]_S \quad [\text{see}]_Q$

ROOT who did you see

Extended Arc Transitions

- Add transitions for non-projective arcs [Attardi 2006]:
  - Left-Arc2(l) $\dfrac{([\ldots, w_i, w_j, w_k]_S, \quad Q, \quad A)}{([\ldots, w_j, w_k]_S, \quad Q, \quad A \cup \{(w_k, l, w_i)\})}$ $[i \neq 0]$
  - Right-Arc2(l) $\dfrac{([\ldots, w_i, w_j, w_k]_S, \quad Q, \quad A)}{([\ldots, w_i, w_j]_S, \quad Q, \quad A \cup \{(w_i, l, w_k)\})}$

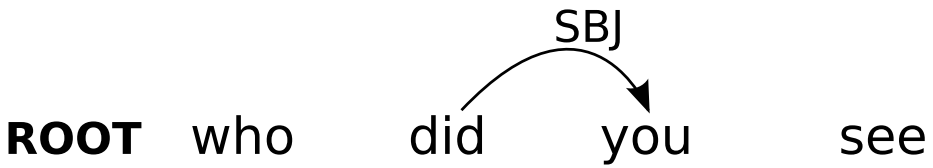
[**ROOT**, who, did]$_S$ [see]$_Q$



Extended Arc Transitions

- Add transitions for non-projective arcs [Attardi 2006]:
  - Left-Arc2(l) $\dfrac{([\ldots, w_i, w_j, w_k]_S, \quad Q, \quad A)}{([\ldots, w_j, w_k]_S, \quad Q, \quad A \cup \{(w_k, l, w_i)\})}$ $[i \neq 0]$
  - Right-Arc2(l) $\dfrac{([\ldots, w_i, w_j, w_k]_S, \quad Q, \quad A)}{([\ldots, w_i, w_j]_S, \quad Q, \quad A \cup \{(w_i, l, w_k)\})}$

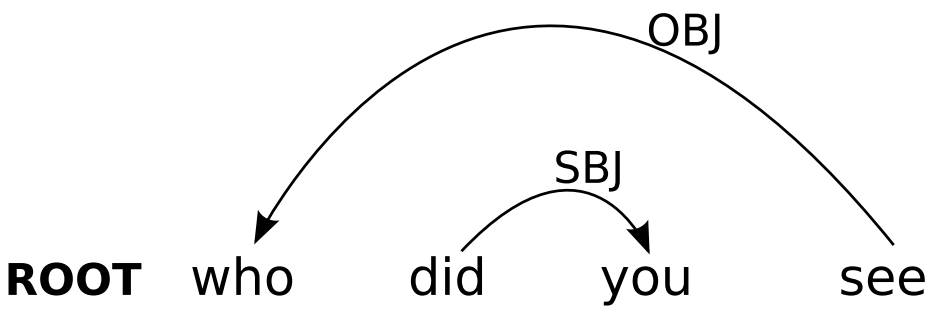
$[\textbf{ROOT}, \text{who}, \text{did}, \text{see}]_S \ [\ ]_Q$



Extended Arc Transitions

- Add transitions for non-projective arcs [Attardi 2006]:
  - Left-Arc2(l) $\dfrac{([\ldots, w_i, w_j, w_k]_S, \quad Q, \quad A)}{([\ldots, w_j, w_k]_S, \quad Q, \quad A \cup \{(w_k, l, w_i)\})}$ $[i \neq 0]$
  - Right-Arc2(l) $\dfrac{([\ldots, w_i, w_j, w_k]_S, \quad Q, \quad A)}{([\ldots, w_i, w_j]_S, \quad Q, \quad A \cup \{(w_i, l, w_k)\})}$

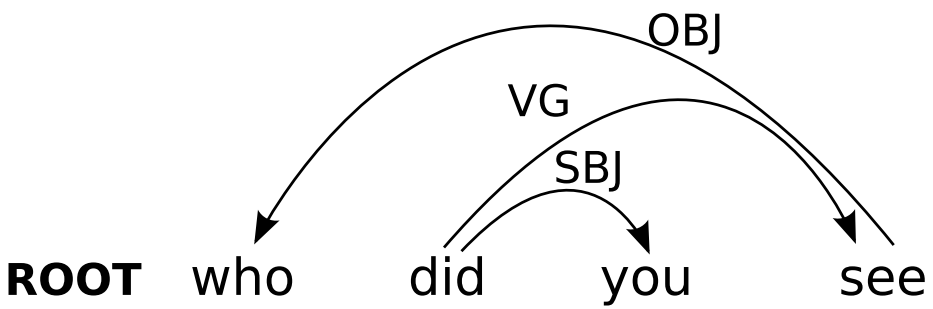
$[\textbf{ROOT}, \text{did}, \text{see}]_S \ [\,]_Q$



Extended Arc Transitions

- Add transitions for non-projective arcs [Attardi 2006]:
  - Left-Arc2(l) $\dfrac{([\ldots, w_i, w_j, w_k]_S, \quad Q, \quad A)}{([\ldots, w_j, w_k]_S, \quad Q, \quad A \cup \{(w_k, l, w_i)\})}$ $[i \neq 0]$
  - Right-Arc2(l) $\dfrac{([\ldots, w_i, w_j, w_k]_S, \quad Q, \quad A)}{([\ldots, w_i, w_j]_S, \quad Q, \quad A \cup \{(w_i, l, w_k)\})}$

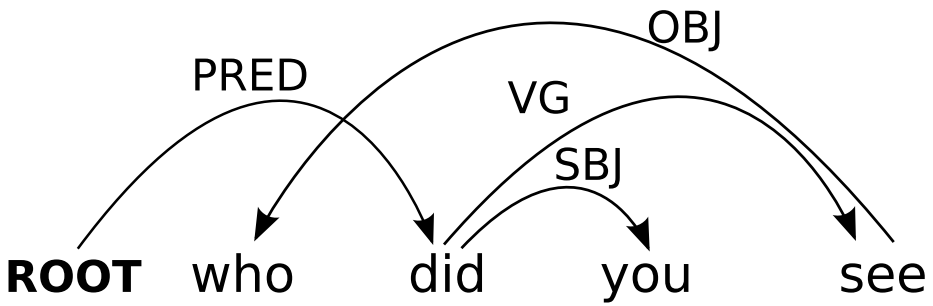
$[\textbf{ROOT}, \text{did}]_S \ [\ ]_Q$



Extended Arc Transitions

- Add transitions for non-projective arcs [Attardi 2006]:
  - Left-Arc2(l) $\dfrac{([\ldots, w_i, w_j, w_k]_S, \quad Q, \quad A)}{([\ldots, w_j, w_k]_S, \quad Q, \quad A \cup \{(w_k, l, w_i)\})}$ $[i \neq 0]$
  - Right-Arc2(l) $\dfrac{([\ldots, w_i, w_j, w_k]_S, \quad Q, \quad A)}{([\ldots, w_i, w_j]_S, \quad Q, \quad A \cup \{(w_i, l, w_k)\})}$

$[\textbf{ROOT}]_S \ [\,]_Q$



Online Reordering

- Add transition for reordering words [Nivre 2009]:
  - Swap $\dfrac{([\ldots, w_i, w_j]_S, \quad [\ldots]_Q, \quad A)}{([\ldots, w_j]_S, \quad [w_i, \ldots]_Q, \quad A)}$ $[i \neq 0]$

$[\textbf{ROOT}]_S$ [who, did, you, see]$_Q$

ROOT who did you see

Online Reordering

- Add transition for reordering words [Nivre 2009]:
  - Swap $\dfrac{([\ldots, w_i, w_j]_S, \quad [\ldots]_Q, \quad A)}{([\ldots, w_j]_S, \quad [w_i, \ldots]_Q, \quad A)}$ $[i \neq 0]$

$[\textbf{ROOT}, \text{who}]_S \quad [\text{did, you, see}]_Q$

ROOT who did you see

Online Reordering

- Add transition for reordering words [Nivre 2009]:
  - Swap $\dfrac{([\ldots, w_i, w_j]_S, \quad [\ldots]_Q, \quad A)}{([\ldots, w_j]_S, \quad [w_i, \ldots]_Q, \quad A)}$ $[i \neq 0]$

$[\textbf{ROOT}, \text{who}, \text{did}]_S \quad [\text{you}, \text{see}]_Q$

ROOT who did you see

Online Reordering

- Add transition for reordering words [Nivre 2009]:
  - Swap $\dfrac{([\ldots, w_i, w_j]_S, \quad [\ldots]_Q, \quad A)}{([\ldots, w_j]_S, \quad [w_i, \ldots]_Q, \quad A)}$ $[i \neq 0]$

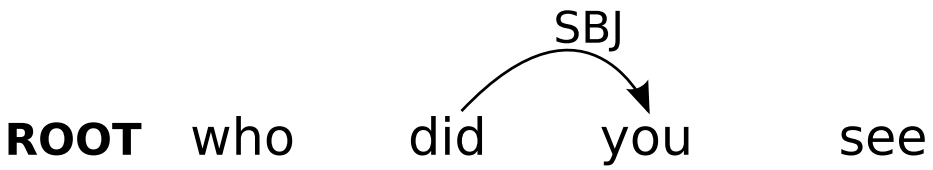
$[\textbf{ROOT}, \text{who}, \text{did}, \text{you}]_S \quad [\text{see}]_Q$

ROOT who did you see

Online Reordering

- Add transition for reordering words [Nivre 2009]:
  - Swap $\dfrac{([\ldots, w_i, w_j]_S, \quad [\ldots]_Q, \quad A)}{([\ldots, w_j]_S, \quad [w_i, \ldots]_Q, \quad A)}$ $[i \neq 0]$

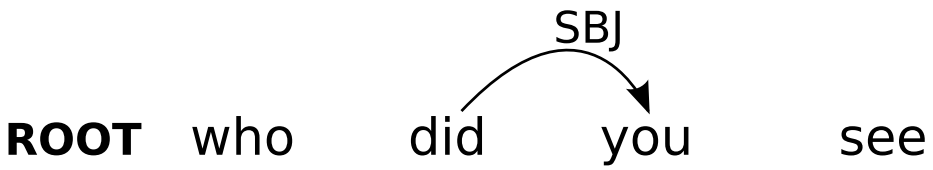
$[\textbf{ROOT}, \text{who}, \text{did}]_S \quad [\text{see}]_Q$



Online Reordering

- Add transition for reordering words [Nivre 2009]:
  - Swap $\dfrac{([\ldots, w_i, w_j]_S, \quad [\ldots]_Q, \quad A)}{([\ldots, w_j]_S, \quad [w_i, \ldots]_Q, \quad A)}$ $[i \neq 0]$

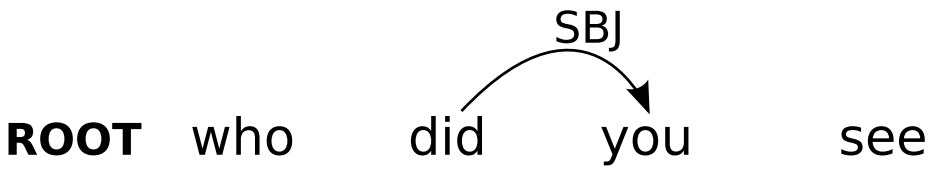
$[\textbf{ROOT}, \text{did}]_S \quad [\text{who}, \text{see}]_Q$



Online Reordering

- Add transition for reordering words [Nivre 2009]:
  - Swap $\dfrac{([\ldots, w_i, w_j]_S, \quad [\ldots]_Q, \quad A)}{([\ldots, w_j]_S, \quad [w_i, \ldots]_Q, \quad A)}$ $[i \neq 0]$

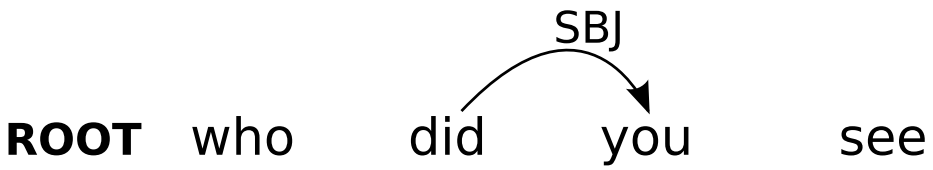
$[\textbf{ROOT}, \text{did}, \text{who}]_S \quad [\text{see}]_Q$



Online Reordering

- Add transition for reordering words [Nivre 2009]:
  - Swap $\dfrac{([\ldots, w_i, w_j]_S, \quad [\ldots]_Q, \quad A)}{([\ldots, w_j]_S, \quad [w_i, \ldots]_Q, \quad A)}$ $[i \neq 0]$

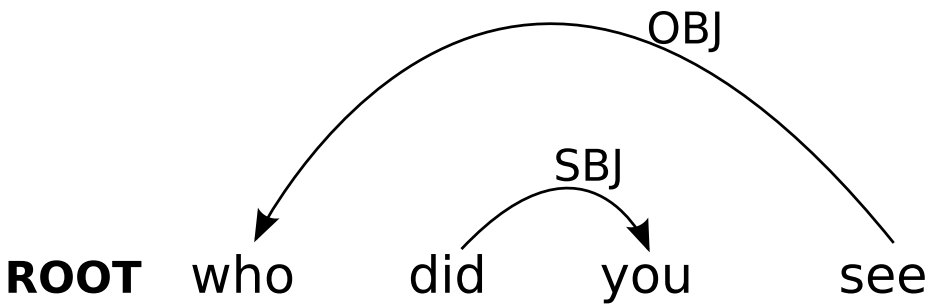
$[\textbf{ROOT}, \text{did}, \text{who}, \text{see}]_S \;\; [\,]_Q$



Online Reordering

- Add transition for reordering words [Nivre 2009]:
  - Swap $\dfrac{([\ldots, w_i, w_j]_S, \quad [\ldots]_Q, \quad A)}{([\ldots, w_j]_S, \quad [w_i, \ldots]_Q, \quad A)}$ $[i \neq 0]$

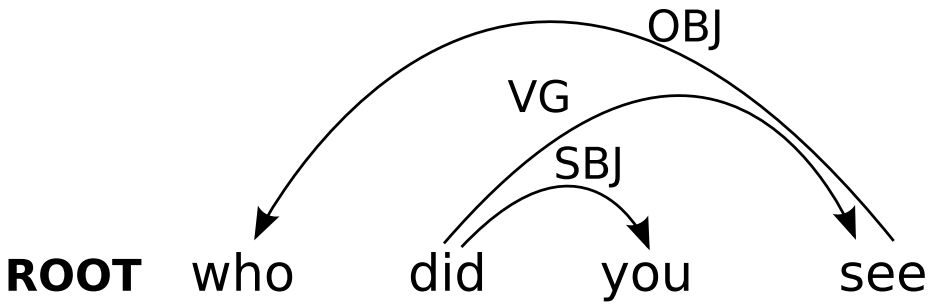
$[\textbf{ROOT}, \text{did}, \text{see}]_S \ [\,]_Q$



Online Reordering

- Add transition for reordering words [Nivre 2009]:
  - Swap $\dfrac{([\ldots, w_i, w_j]_S, \quad [\ldots]_Q, \quad A)}{([\ldots, w_j]_S, \quad [w_i, \ldots]_Q, \quad A)}$ $[i \neq 0]$

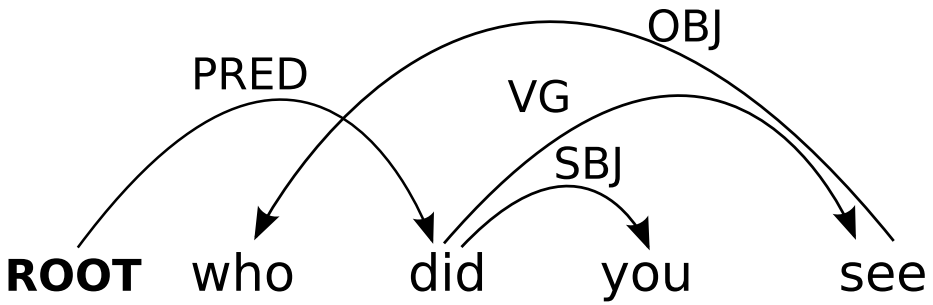
$[\textbf{ROOT}, \text{did}]_S \ [\ ]_Q$



Online Reordering

- Add transition for reordering words [Nivre 2009]:
  - Swap $\dfrac{([\ldots, w_i, w_j]_S, \quad [\ldots]_Q, \quad A)}{([\ldots, w_j]_S, \quad [w_i, \ldots]_Q, \quad A)}$ $[i \neq 0]$

$[\textbf{ROOT}]_S \ [\,]_Q$



Comparison

	PProj	**EArc**	**Swap**
Completeness	no	no	yes
Parsing complexity	$O(n)^*$	$O(n)$	$O(n^2)$
Post-processing	yes	no	no

Empirical Evaluation

- Comparative evaluation [Kuhlmann and Nivre 2010]:
  - Theoretical coverage (oracle parsing)
  - Empirical performance (trained parser models)
- Data sets from CoNLL 2009 [Hajič et al. 2009]:
  - Czech: Prague Dependency Treebank [Hajič et al. 2001]
  - English: Penn Treebank [Marcus et al. 1993]
  - German: Tiger [Brants et al. 2002]
- Implementation: MaltParser [Nivre et al. 2006]

Training Set Statistics

	NPSent	NPArcs	NPLen	PLen
Czech	22.4	1.9	4.5	3.6
English	7.6	0.4	8.2	3.4
German	28.1	2.3	10.6	3.9

NPSent = Percentage of sentences with a non-projective tree
NPArcs = Percentage of arcs that are non-projective
NPLen = Average length of non-projective arcs
PLen = Average length of projective arcs

Theoretical Coverage

	Czech		English		German	
	LAS	**LEM**	**LAS**	**LEM**	**LAS**	**LEM**
Proj	98.0	76.9	99.7	93.7	97.7	73.1
PProj	99.8	97.9	*100.0	99.6	99.8	97.9
EArc	99.6	98.9	99.6	98.3	98.5	96.0
Swap	100.0	100.0	100.0	100.0	100.0	100.0

LAS = Labeled attachment score
LEM = Labeled exact match

Theoretical Coverage

	Czech		English		German	
	LAS	**LEM**	**LAS**	**LEM**	**LAS**	**LEM**
Proj	98.0	76.9	99.7	93.7	97.7	73.1
PProj	99.8	97.9	*100.0	99.6	99.8	97.9
EArc	99.6	98.9	99.6	98.3	98.5	96.0
Swap	100.0	100.0	100.0	100.0	100.0	100.0

LAS = Labeled attachment score
LEM = Labeled exact match

Theoretical Coverage

	Czech		English		German	
	LAS	**LEM**	**LAS**	**LEM**	**LAS**	**LEM**
Proj	98.0	76.9	99.7	93.7	97.7	73.1
PProj	99.8	97.9	*100.0	99.6	99.8	97.9
EArc	99.6	98.9	99.6	98.3	98.5	96.0
Swap	100.0	100.0	100.0	100.0	100.0	100.0

LAS = Labeled attachment score
LEM = Labeled exact match

Empirical Performance

	Czech		English		German	
	LAS	**LEM**	**LAS**	**LEM**	**LAS**	**LEM**
Proj	79.7	27.9	84.9	17.8	83.3	31.9
PProj	80.6	30.7	85.0	18.5	84.1	34.7
EArc	80.6	31.9	84.6	16.6	82.8	32.5
Swap	80.7	31.3	85.0	18.6	83.7	34.6

LAS = Labeled attachment score
LEM = Labeled exact match

Empirical Performance

	Czech		English		German	
	LAS	**LEM**	**LAS**	**LEM**	**LAS**	**LEM**
Proj	79.7	27.9	84.9	17.8	83.3	31.9
PProj	80.6	30.7	85.0	18.5	84.1	34.7
EArc	80.6	31.9	84.6	16.6	82.8	32.5
Swap	80.7	31.3	85.0	18.6	83.7	34.6

LAS = Labeled attachment score
LEM = Labeled exact match

Empirical Performance: Non-Projective Arcs

	Czech		English		German	
	LP	**LR**	**LP**	**LR**	**LP**	**LR**
Proj	–	5.0	–	8.4	–	4.2
PProj	76.1	53.5	63.2	47.4	71.1	43.6
EArc	79.7	65.0	62.5	46.3	69.1	49.7
Swap	77.9	65.5	55.8	49.5	61.2	48.9

LP = Labeled precision on non-projective arcs output by the parser
LR = Labeled recall on non-projective arcs in gold standard

Empirical Performance: Non-Projective Arcs

	Czech		English		German	
	LP	**LR**	**LP**	**LR**	**LP**	**LR**
Proj	–	5.0	–	8.4	–	4.2
PProj	76.1	53.5	63.2	47.4	71.1	43.6
EArc	79.7	65.0	62.5	46.3	69.1	49.7
Swap	77.9	65.5	55.8	49.5	61.2	48.9

LP = Labeled precision on non-projective arcs output by the parser
LR = Labeled recall on non-projective arcs in gold standard

Empirical Performance: Non-Projective Arcs

	Czech		English		German	
	LP	**LR**	**LP**	**LR**	**LP**	**LR**
Proj	–	5.0	–	8.4	–	4.2
PProj	76.1	53.5	63.2	47.4	71.1	43.6
EArc	79.7	65.0	62.5	46.3	69.1	49.7
Swap	77.9	65.5	55.8	49.5	61.2	48.9

LP = Labeled precision on non-projective arcs output by the parser
LR = Labeled recall on non-projective arcs in gold standard

Discussion

- Overall parsing accuracy (LAS, LEM):
  - Significant improvement if non-projective dependencies are frequent enough (Czech and German but not English)
- Non-projective dependencies (LP, LR):
  - Precision relatively high (60–80), sometimes almost equivalent to projective dependencies modulo length
  - Recall generally lower (45–65) and language-dependent (better for Czech than for English and German)
- Comparing methods:
  - Very similar results for all three methods
  - EArc harmful if non-projective dependencies are longer (English and German but not Czech)

Conclusion

Conclusion

- Transition-based parsing:
  - Efficient parsing using greedy inference or beam search
  - Unconstrained history-based feature models
- Non-projective dependencies:
  - Three transition-based techniques:
    - Pseudo-projective parsing
    - Extended arc transitions
    - Online reordering (swap)
  - High precision, moderate recall
- Future work on non-projective dependencies:
  - Study the effect of global optimization
  - Compare with other kinds of dependency parsers

References

- I. Aduriz, M. J. Aranzabe, J. M. Arriola, A. Atutxa, A. Díaz de Ilarraza, A. Garmendia, and M. Oronoz. 2003. Construction of a Basque dependency treebank. In *Proceedings of the 2nd Workshop on Treebanks and Linguistic Theories (TLT)*, pages 201–204.
- Giuseppe Attardi. 2006. Experiments with a multilanguage non-projective dependency parser. In *Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL)*, pages 166–170.
- Igor Boguslavsky, Svetlana Grigorieva, Nikolai Grigoriev, Leonid Kreidlin, and Nadezhda Frid. 2000. Dependency treebank for Russian: Concept, tools, types of information. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING)*, pages 987–991.
- Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. TIGER treebank. In *Proceedings of the 1st Workshop on Treebanks and Linguistic Theories (TLT)*, pages 24–42.
- Matthias Buch-Kromann. 2006. *Discontinuous Grammar: A Model of Human Parsing and Language Acquisition*. Ph.D. thesis, Copenhagen Business School.
- Sašo Džeroski, Tomaž Erjavec, Nina Ledinek, Petr Pajas, Zdenek Žabokrtsky, and Andreja Žele. 2006. Towards a Slovene dependency treebank. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC)*.
- Jan Hajič, Barbora Vidova Hladka, Jarmila Panevová, Eva Hajičová, Petr Sgall, and Petr Pajas. 2001. Prague Dependency Treebank 1.0. LDC, 2001T10.
- Jan Hajič, Otakar Smrž, Petr Zemánek, Jan Šnaidauf, and Emanuel Beška. 2004. Prague Arabic Dependency Treebank: Development in data and tools. In *Proceedings of the NEMLAR International Conference on Arabic Language Resources and Tools*.
- Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL): Shared Task*, pages 1–18.

References

- Matthias Trautner Kromann. 2003. The Danish Dependency Treebank and the DTAG treebank tool. In *Proceedings of the 2nd Workshop on Treebanks and Linguistic Theories (TLT)*, pages 217–220.
- Marco Kuhlmann and Joakim Nivre. 2010. Transition-based techniques for non-projective dependency parsing. *Norther European Journal of Language Technology*, 1:1–19.
- Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19:313–330.
- 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 and Giorgio Satta. 2007. On the complexity of non-projective data-driven dependency parsing. In *Proceedings of the 10th International Conference on Parsing Technologies (IWPT)*, pages 122–131.
- 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.
- 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, Johan Hall, and Jens Nilsson. 2004. Memory-based dependency parsing. In *Proceedings of the 8th Conference on Computational Natural Language Learning (CoNLL)*, pages 49–56.
- Joakim Nivre, Johan Hall, and Jens Nilsson. 2006. Maltparser: A data-driven parser-generator for dependency parsing. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC)*, pages 2216–2219.
- Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34:513–553.

References

- Joakim Nivre. 2009. Non-projective dependency parsing in expected linear time. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP (ACL-IJCNLP)*, pages 351–359.
- Kemal Oflazer, Bilge Say, Dilek Zeynep Hakkani-Tür, and Gökhan Tür. 2003. Building a Turkish treebank. In Anne Abeillé, editor, *Treebanks: Building and Using Parsed Corpora*, pages 261–277. Kluwer.
- P. Prokopidis, E. Desypri, M. Koutsombogera, H. Papageorgiou, and S. Piperidis. 2005. Theoretical and practical issues in the construction of a Greek dependency treebank. In *Proceedings of the 3rd Workshop on Treebanks and Linguistic Theories (TLT)*, pages 149–160.
- Ivan Titov and James Henderson. 2007. A latent variable model for generative dependency parsing. In *Proceedings of the 10th International Conference on Parsing Technologies (IWPT)*, pages 144–155.
- Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*, pages 195–206.
- Yue Zhang and Stephen Clark. 2008. A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 562–571.
- Yue Zhang and Joakim Nivre. 2011. Transition-based parsing with rich non-local features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL)*.