Syntactic Dependencies (I)

CS 690N, Spring 2017

Advanced Natural Language Processing

http://people.cs.umass.edu/~brenocon/anlp2017/

Brendan O'Connor

College of Information and Computer Sciences

University of Massachusetts Amherst

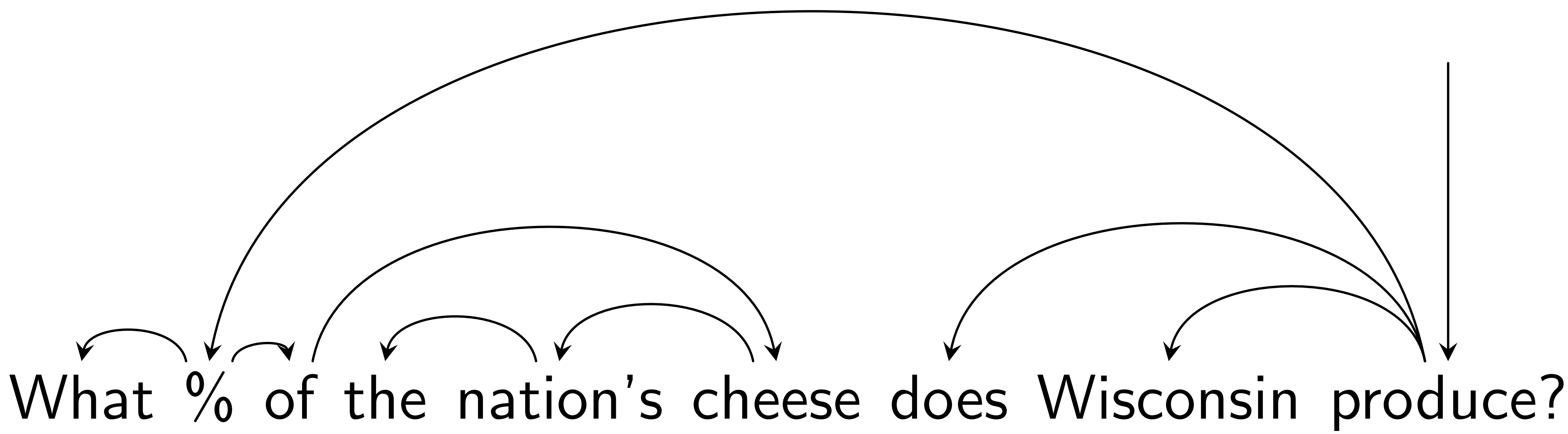
Dependency parsing in action

Dependency parsing is used in many real-world applications, like question answering (Cui et al, 2005):

[Example: Jacob Eisenstein]

Dependency parsing in action

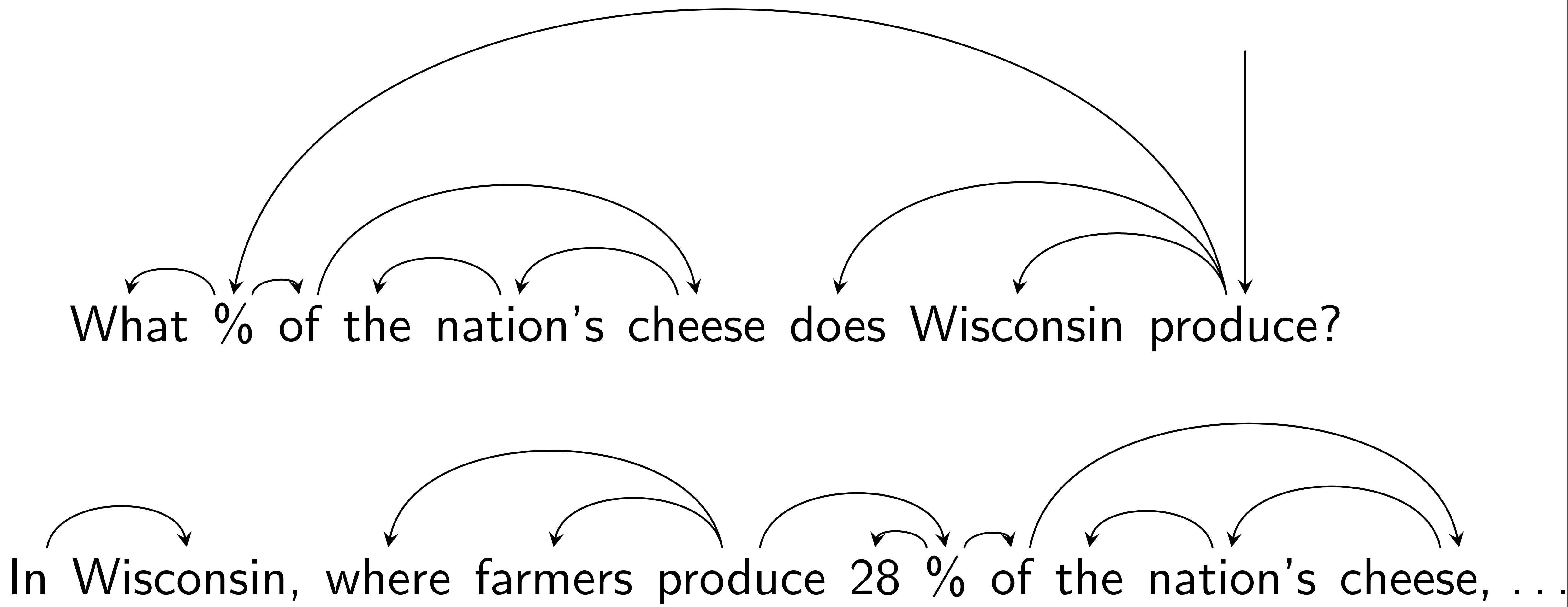
Dependency parsing is used in many real-world applications, like question answering (Cui et al, 2005):



[Example: Jacob Eisenstein]

Dependency parsing in action

Dependency parsing is used in many real-world applications, like question answering (Cui et al, 2005):

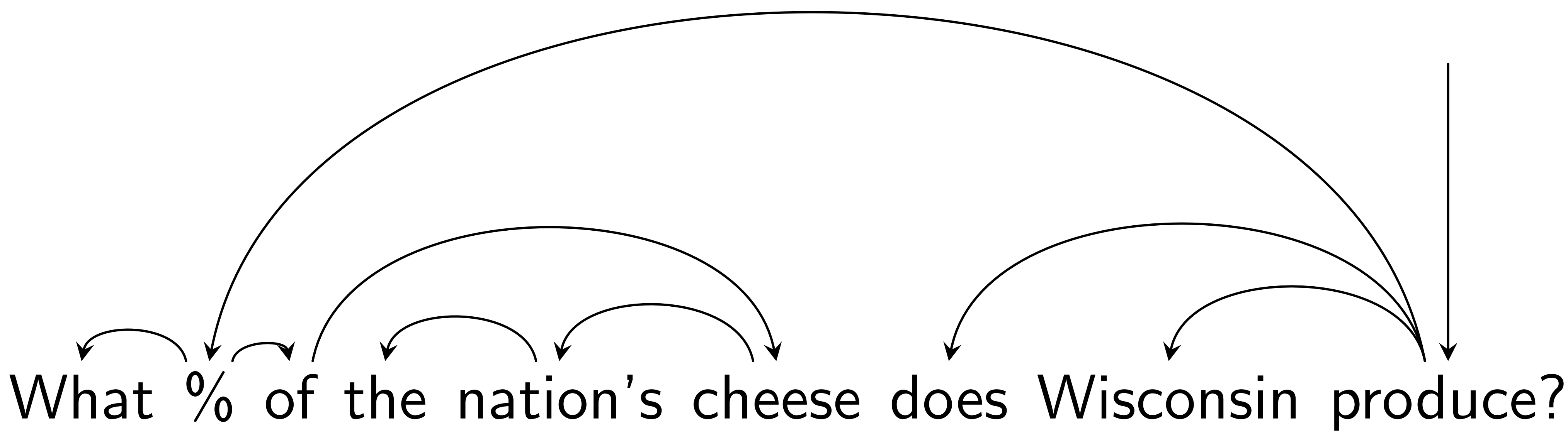


[Example: Jacob Eisenstein]

Dependency parsing in action

Question answering works by searching for statements which match well against the query.

- In the surface form of the question, *produce* and *%* are six words apart.
- But in the dependency parse, they're adjacent.

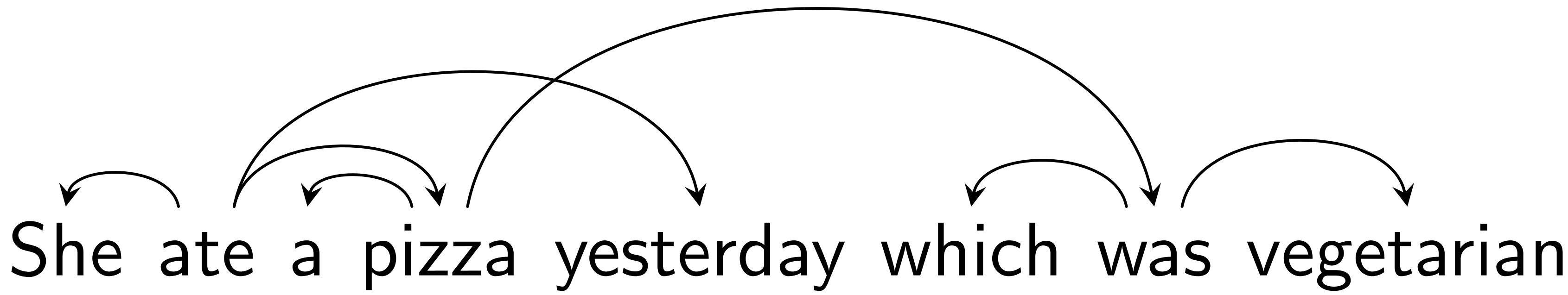


[Example: Jacob Eisenstein]

Projectivity

In **projective** dependency parsing, there are no crossing edges.

- Crossing edges are rare in English:



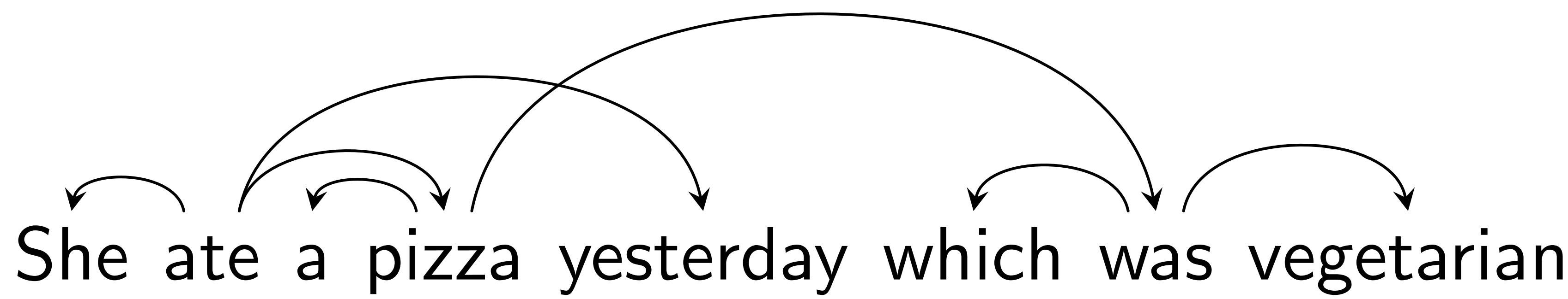
[2] figure from (Nivre 2007)

[Example: Jacob Eisenstein]

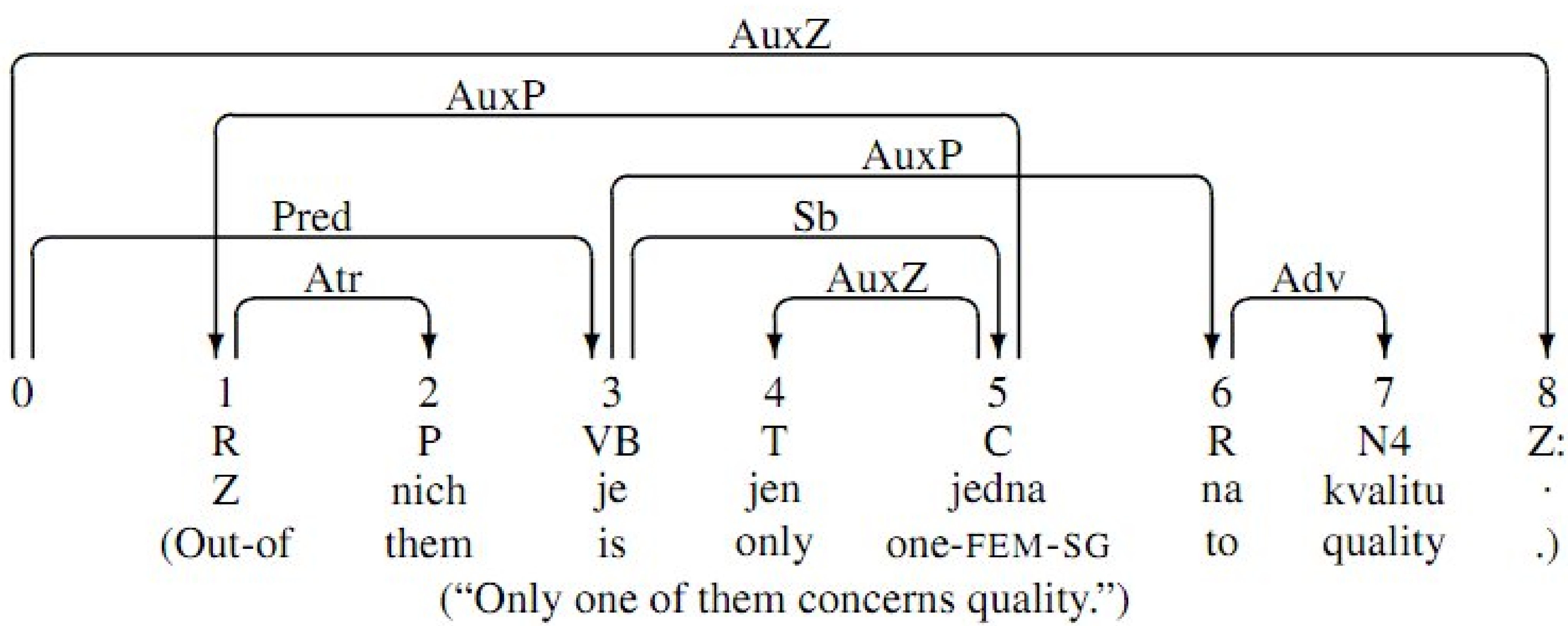
Projectivity

In **projective** dependency parsing, there are no crossing edges.

- Crossing edges are rare in English:



- They are more common in other languages, like Czech:[2]



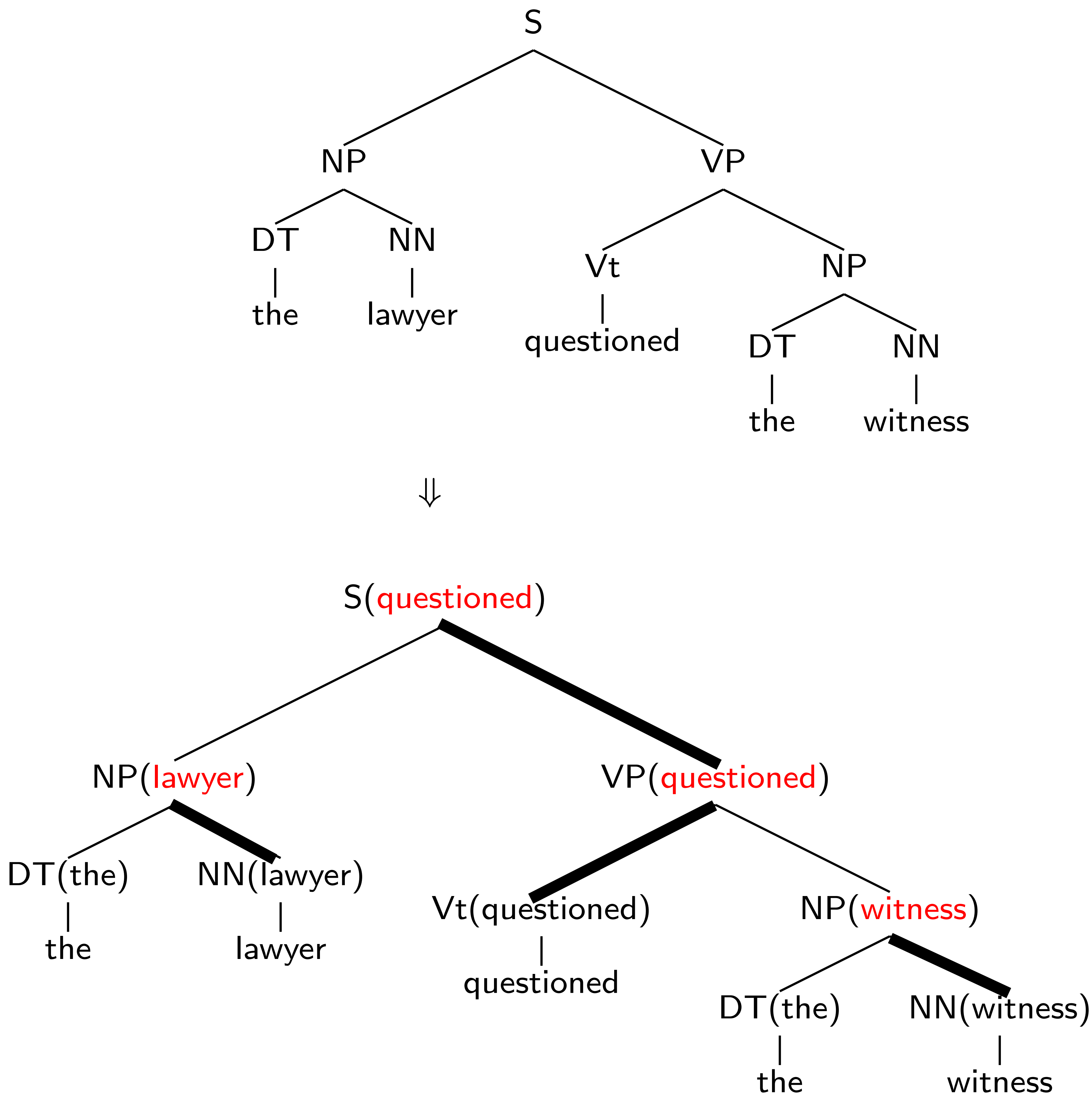
[2] figure from (Nivre 2007)

[Example: Jacob Eisenstein]

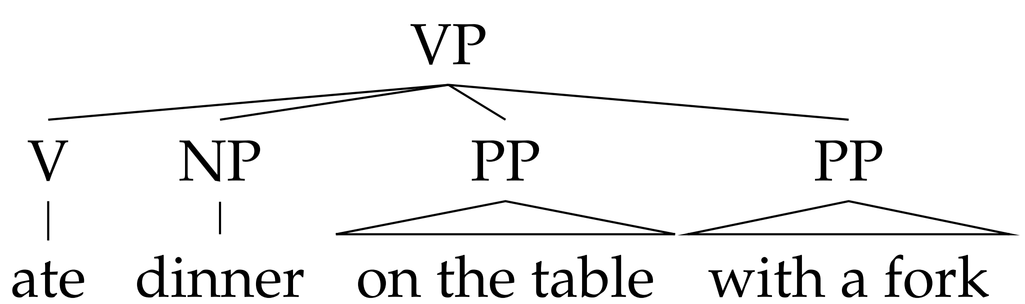
Constits -> Deps

- Every phrase has a head word. It dominates all other words of that phrase in the dep. graph.
- Head rules: for every nonterminal in tree, choose one of its children to be its "head". This will define head words.
- Every nonterminal type has a different head rule; e.g. from Collins (1997):

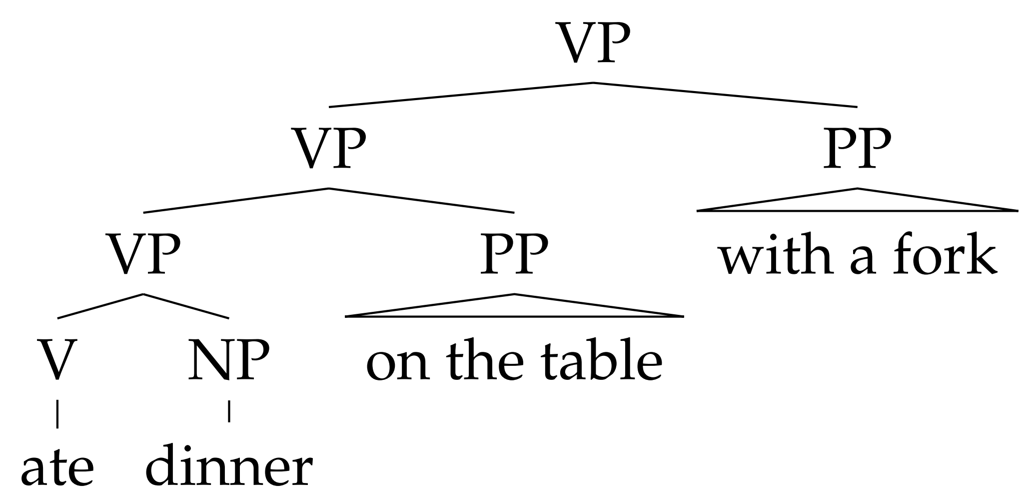
- If parent is NP,
  - Search from right-to-left for first child that's NN, NNP, NNPS, NNS, NX, JJR
  - Else: search left-to-right for first child which is NP


S
NP
VP
DT
NN
the
lawyer
Vt
questioned
NP
DT
NN
the
witness
⇓
S(questioned)
NP(lawyer)
VP(questioned)
DT(the)
NN(lawyer)
the
lawyer
Vt(questioned)
questioned
NP(witness)
DT(the)
NN(witness)
the
witness

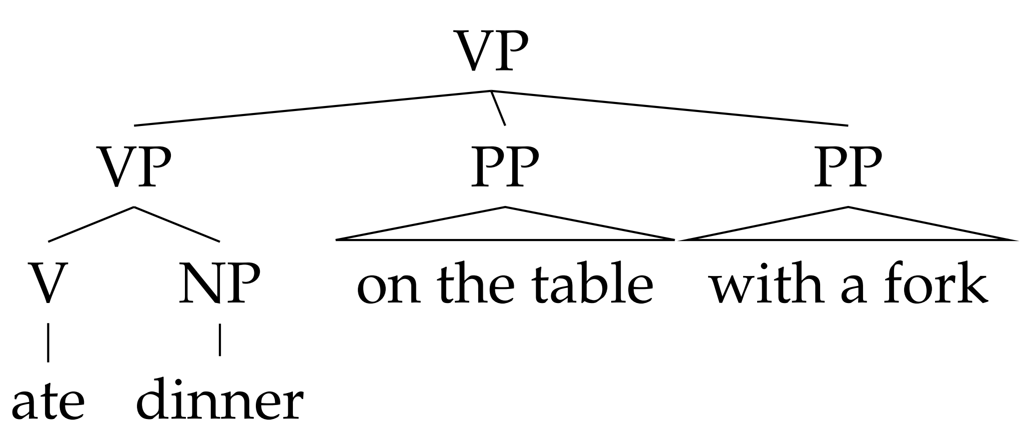
- Dependencies tend to be less specific than constituent structure



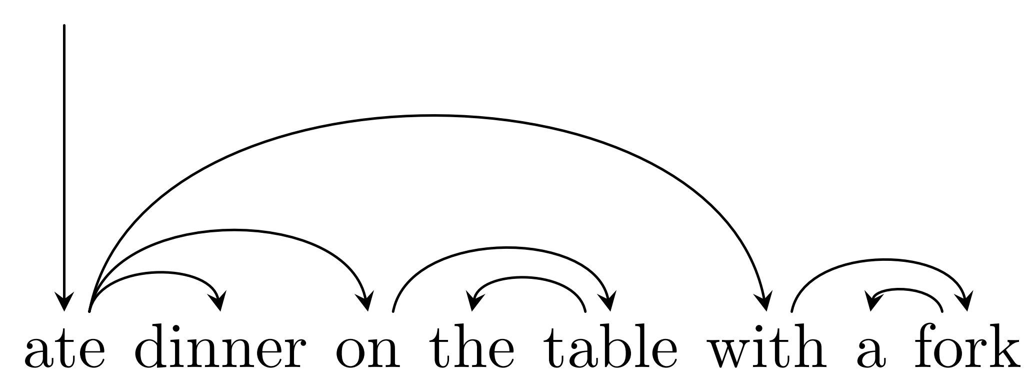
(a) Flat



(b) Two-level (PTB-style)



(c) Chomsky adjunction

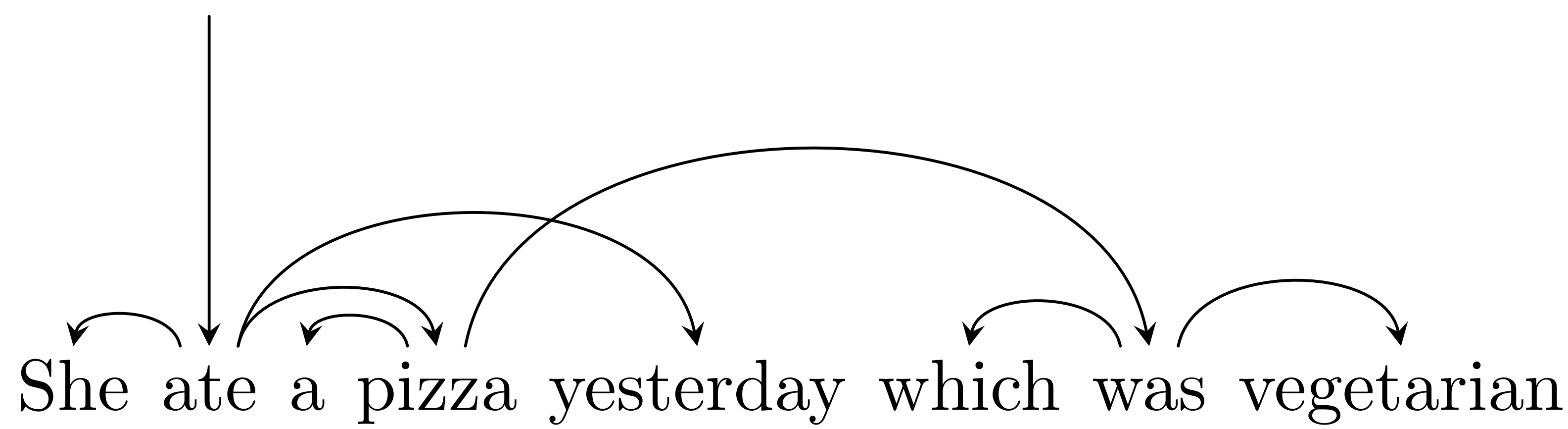


(d) Dependency representation

[Example: Jacob Eisenstein]

Projectivity

- Projectivity: no crossing arcs.
  Corresponds to neatly nested constituencies
- Non-projective example:



	% non-projective edges	% non-projective sentences
Czech	1.86%	22.42%
English	0.39%	7.63%
German	2.33%	28.19%

Table 12.1: Frequency of non-projective dependencies in three languages (Kuhlmann and Nivre, 2010)

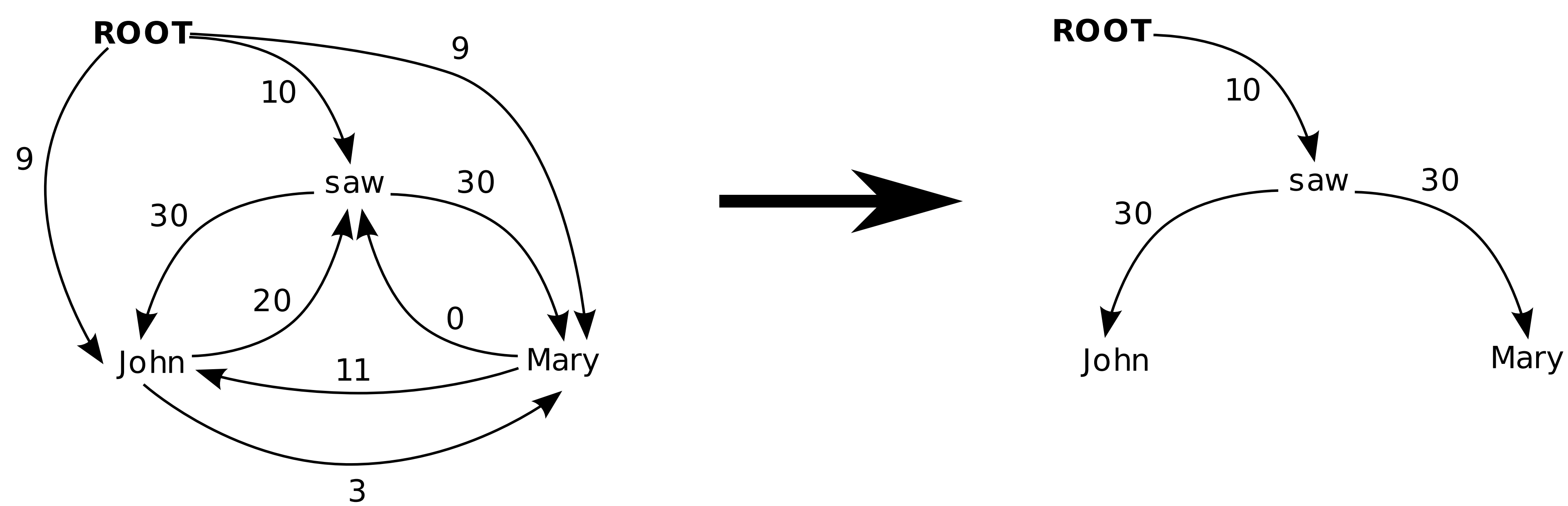
[Example: Jacob Eisenstein]

Parsing to dependencies

- Constituents -> Dependency conversion is one approach
- Direct dependency parsing more common
  - Annotating dependencies is easier
- Algorithmic approaches
  - Graph-based: global CRF-style models
  - History-based: shift-reduce (Nivre)

Graph-based parsing

Edge scoring models



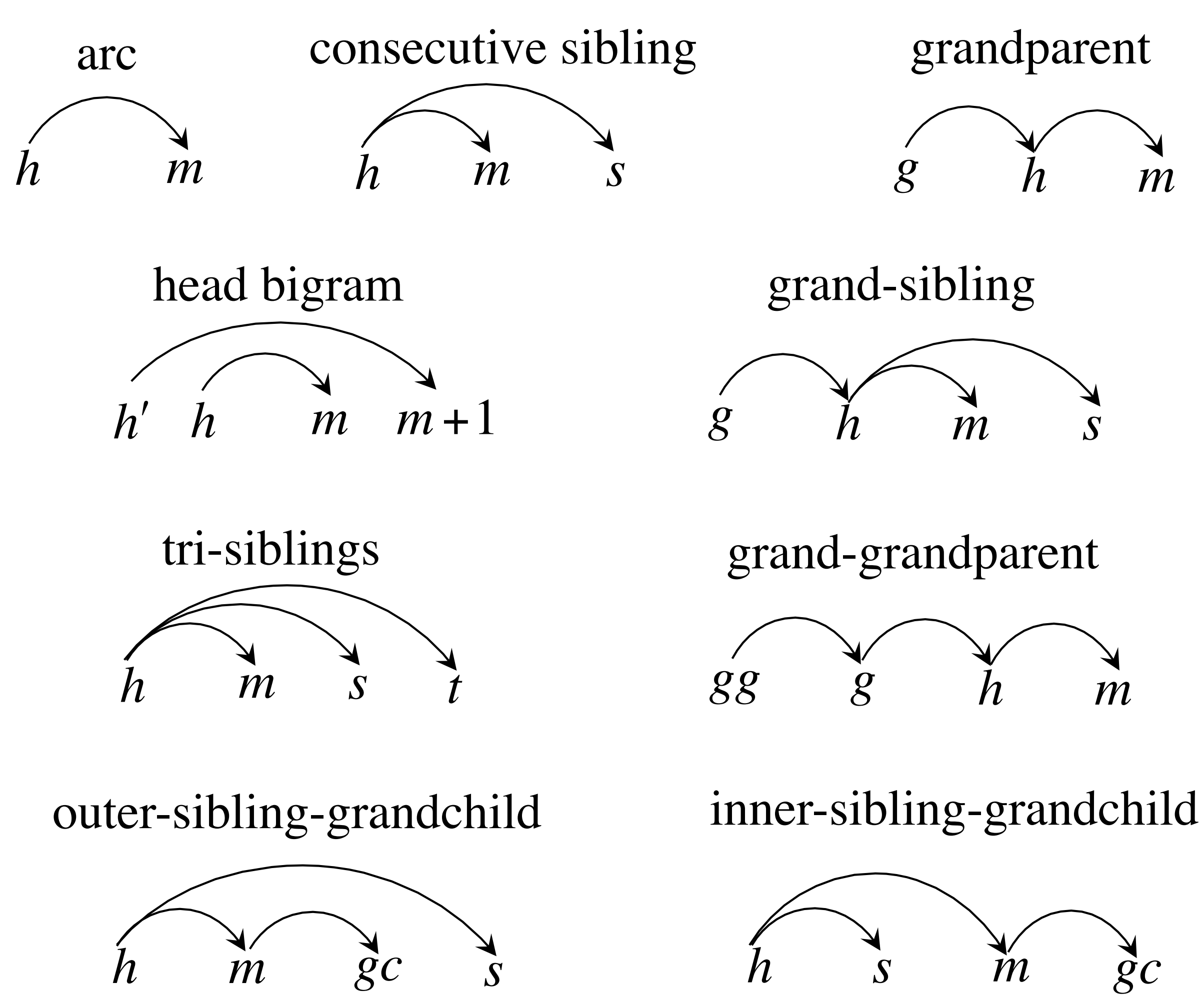
<u>Inference</u>: dynamic programming, (argmax) minimum spanning trees, (expectations) matrix tree theorem

<u>Learning</u>: structured perceptron/svm or crf loglik

[Slides: <u>McDonald and Nivre, EACL 2014 tutorial</u>]

Graph-based parsing

Higher order features: learn e.g. selectional restrictions
Decoding is more difficult



<u>Inference</u>: integer linear programs, gibbs sampling, easy-first ...
<u>Learning</u>: structured perceptron/svm or crf loglik

[Slides: <u>McDonald and Nivre, EACL 2014 tutorial</u>]

Arc-Eager Transition System [Nivre 2003]

Configuration: (S, B, A) $[S = \text{Stack},\ B = \text{Buffer},\ A = \text{Arcs}]$

Initial: $([\ ], [0, 1, \ldots, n], \{\ \})$

Terminal: $(S, [\ ], A)$

Shift: $(S, i|B, A) \Rightarrow (S|i, B, A)$

Reduce: $(S|i, B, A) \Rightarrow (S, B, A)$ $\quad h(i, A)$

Right-Arc(k): $(S|i, j|B, A) \Rightarrow (S|i|j, B, A \cup \{(i, j, k)\})$

Left-Arc(k): $(S|i, j|B, A) \Rightarrow (S, j|B, A \cup \{(j, i, k)\})$ $\quad \neg h(i, A) \wedge i \neq 0$

Notation: $S|i$ = stack with top i and remainder S
$j|B$ = buffer with head j and remainder B
$h(i, A)$ = i has a head in A

Example Transition Sequence

$[\text{ROOT}]_S$ $[\text{Economic, news, had, little, effect, on, financial, markets, .}]_B$

ROOT	Economic	news	had	little	effect	on	financial	markets	.
	adj	noun	verb	adj	noun	prep	adj	noun	.

Example Transition Sequence

$[\text{ROOT, Economic}]_S \quad [\text{news, had, little, effect, on, financial, markets, .}]_B$

ROOT	Economic	news	had	little	effect	on	financial	markets	.
	adj	noun	verb	adj	noun	prep	adj	noun	.

Example Transition Sequence

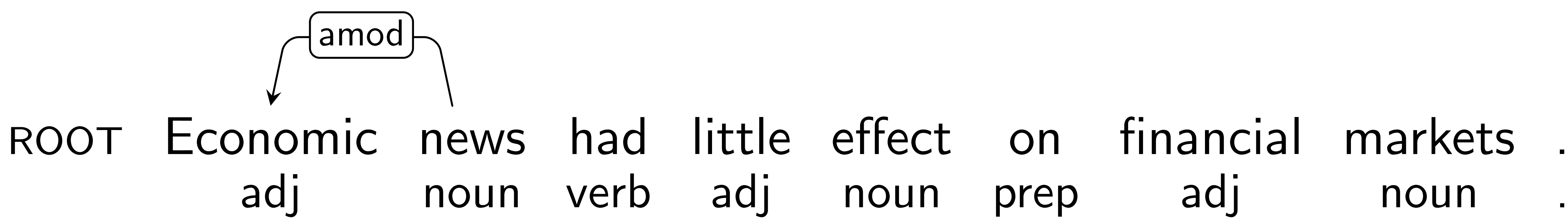
$[\text{ROOT}]_S$ $[\text{news, had, little, effect, on, financial, markets, .}]_B$

amod

ROOT	Economic	news	had	little	effect	on	financial	markets	.
	adj	noun	verb	adj	noun	prep	adj	noun	.

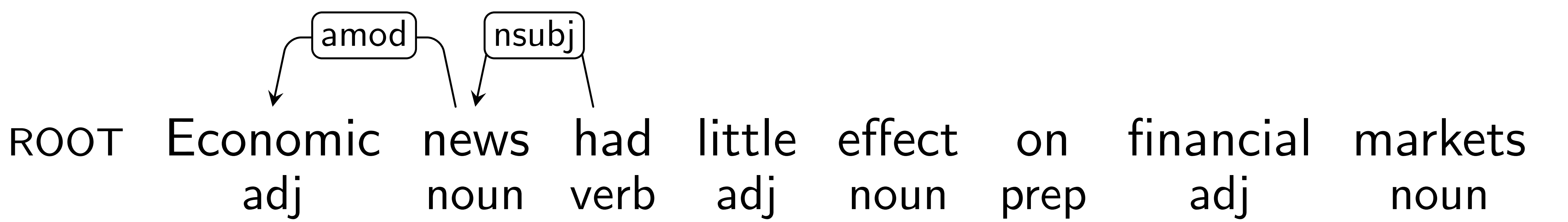
Example Transition Sequence

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



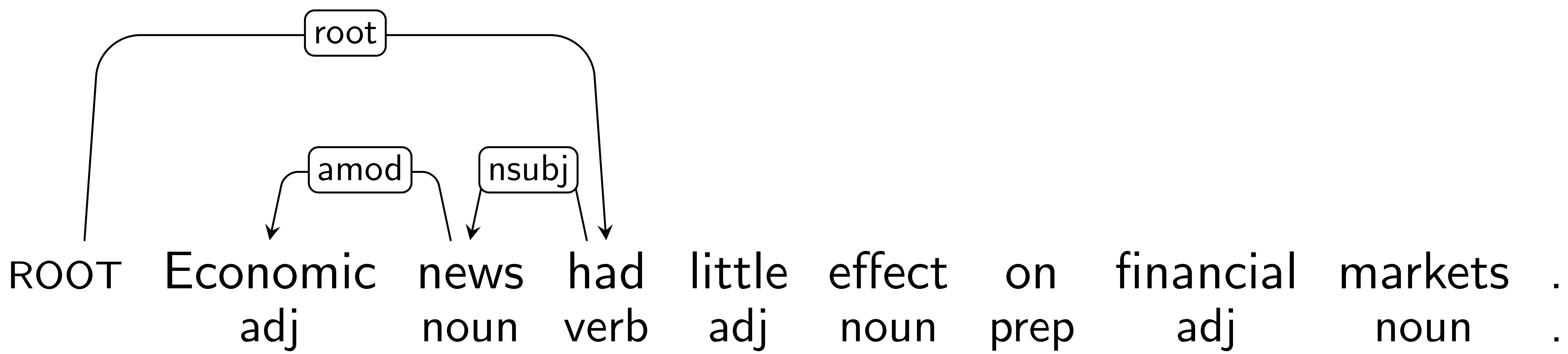
Example Transition Sequence

$[\text{ROOT}]_S \;\; [\text{had, little, effect, on, financial, markets, .}]_B$



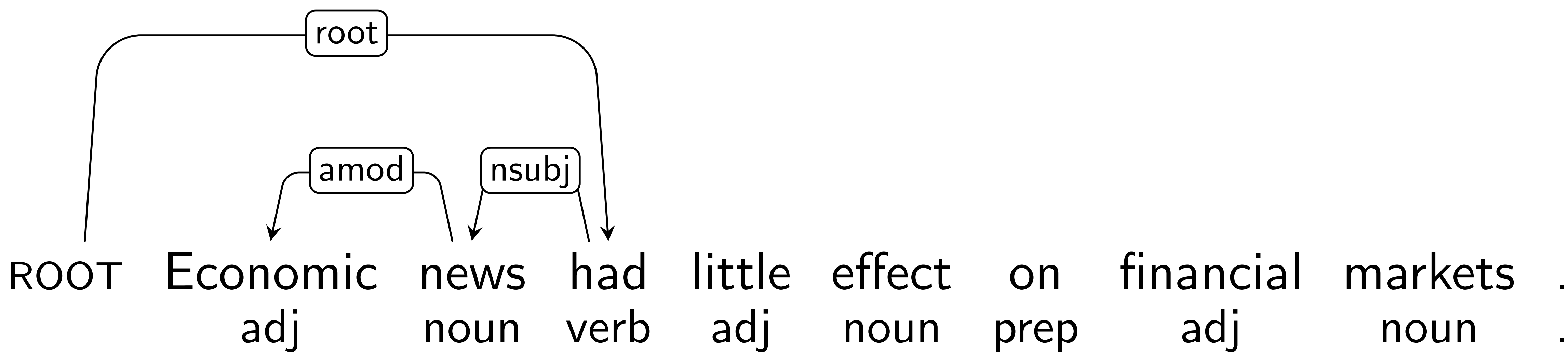
Example Transition Sequence

$[\text{ROOT, had}]_S \ [\text{little, effect, on, financial, markets, .}]_B$



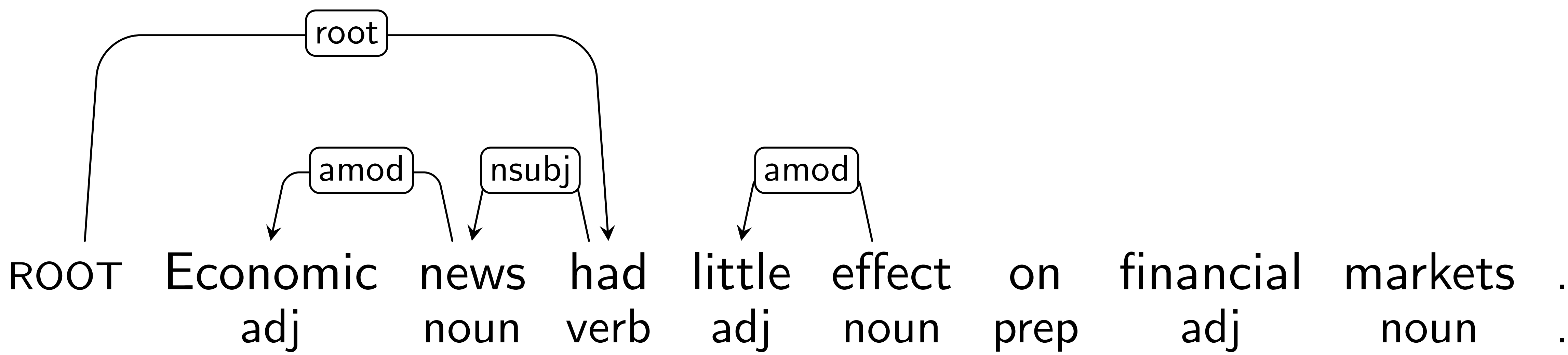
Example Transition Sequence

$[\text{ROOT, had, little}]_S \quad [\text{effect, on, financial, markets, .}]_B$



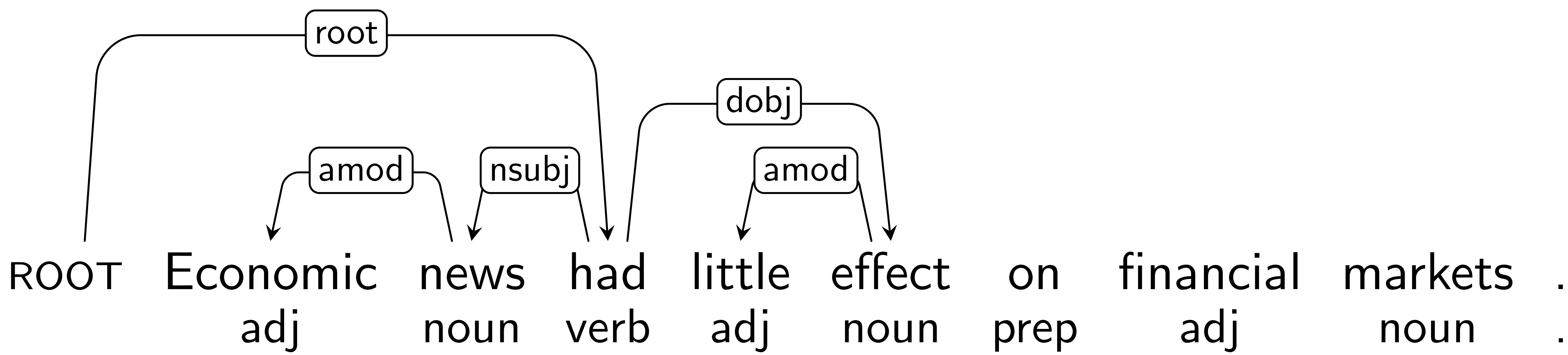
Example Transition Sequence

$[\text{ROOT, had}]_S$ $[\text{effect, on, financial, markets, .}]_B$



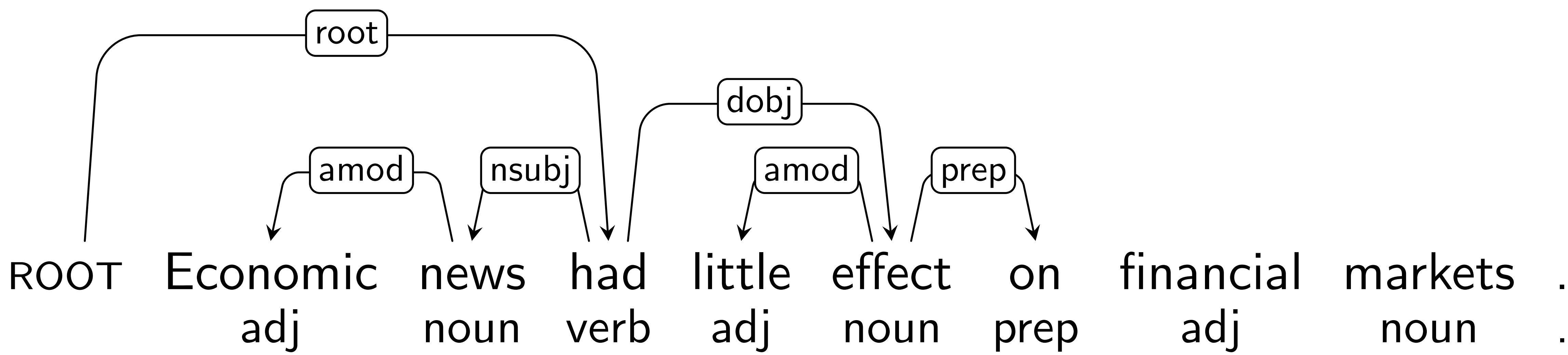
Example Transition Sequence

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



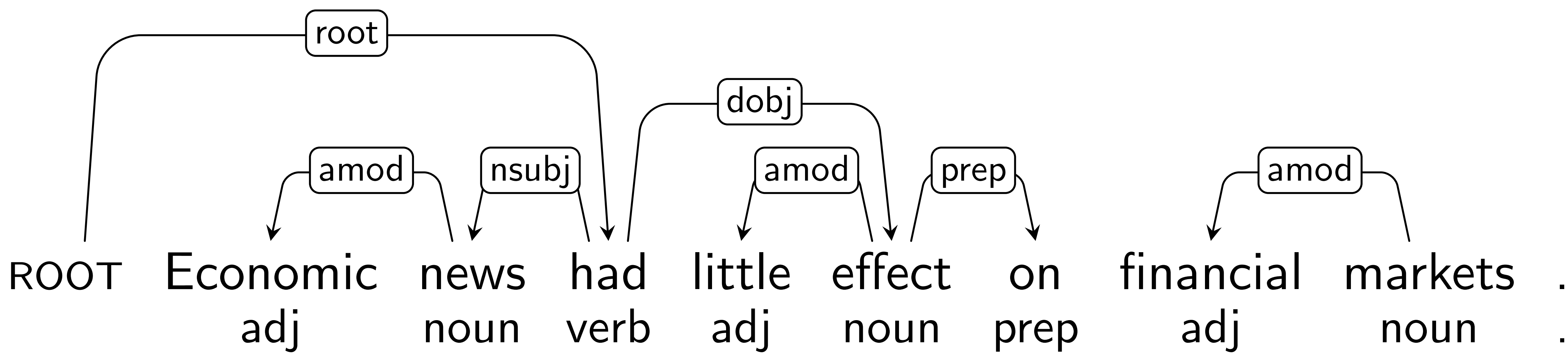
Example Transition Sequence

$[\text{ROOT, had, effect, on, financial}]_S \; [\text{markets, .}]_B$



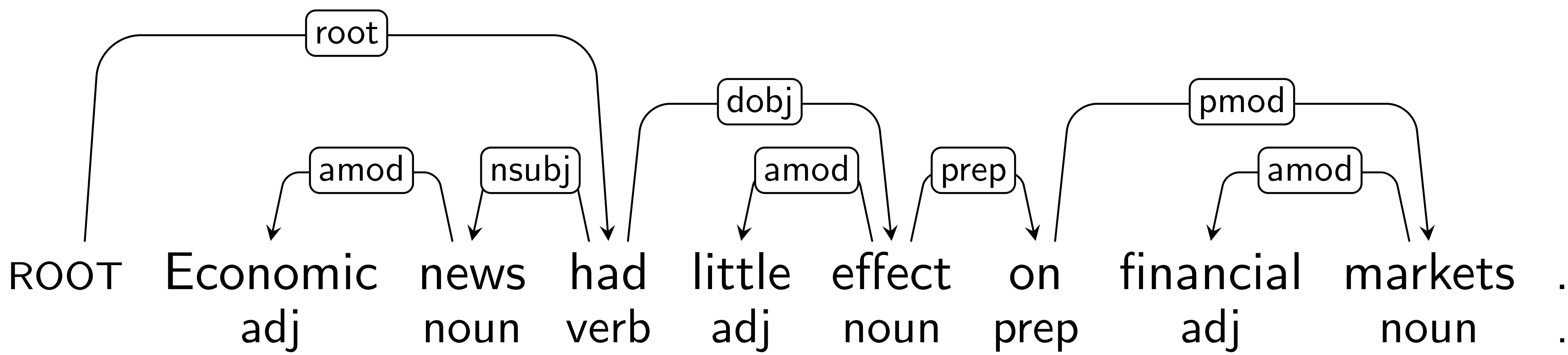
Example Transition Sequence

[ROOT, had, effect, on]$_S$ [markets, .]$_B$



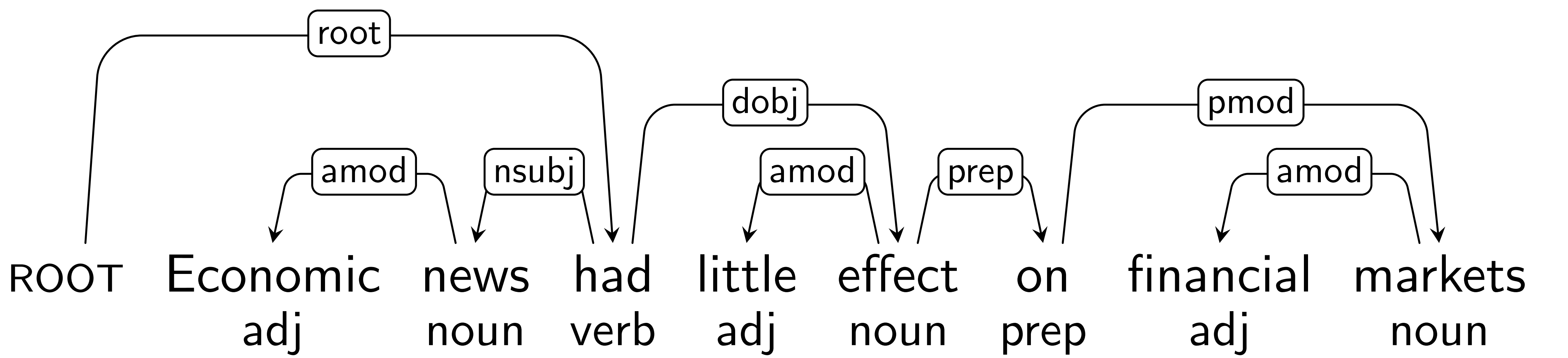
Example Transition Sequence

$[\text{ROOT, had, effect, on, markets}]_S \quad [\text{.}]_B$



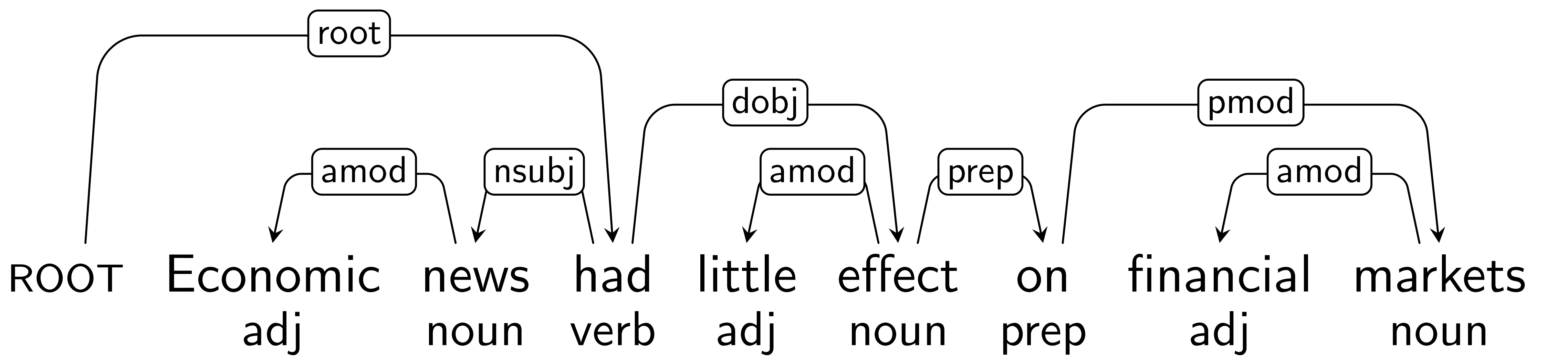
Example Transition Sequence

$[\text{ROOT, had, effect, on}]_S \quad [.]_B$



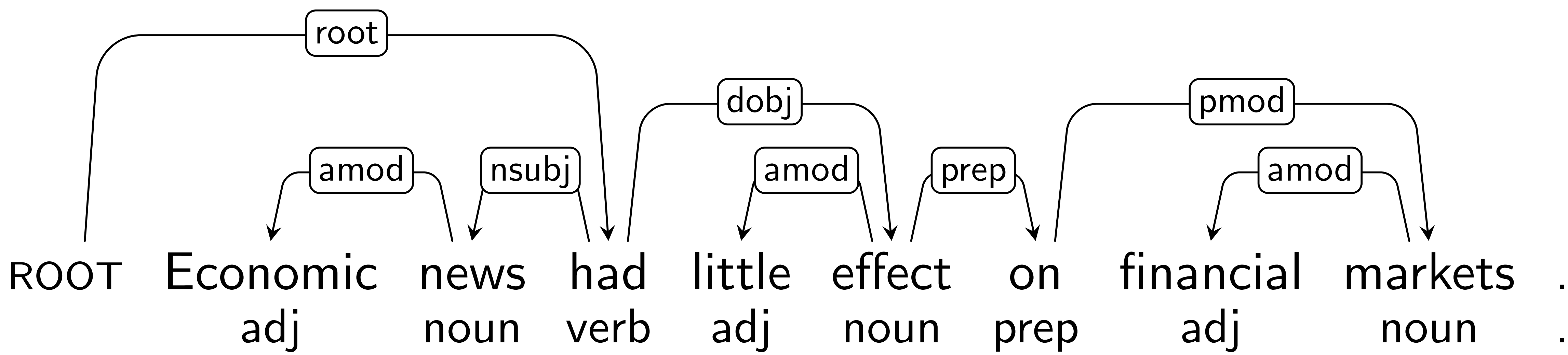
Example Transition Sequence

$[\text{ROOT, had, effect}]_S \quad [.]_B$



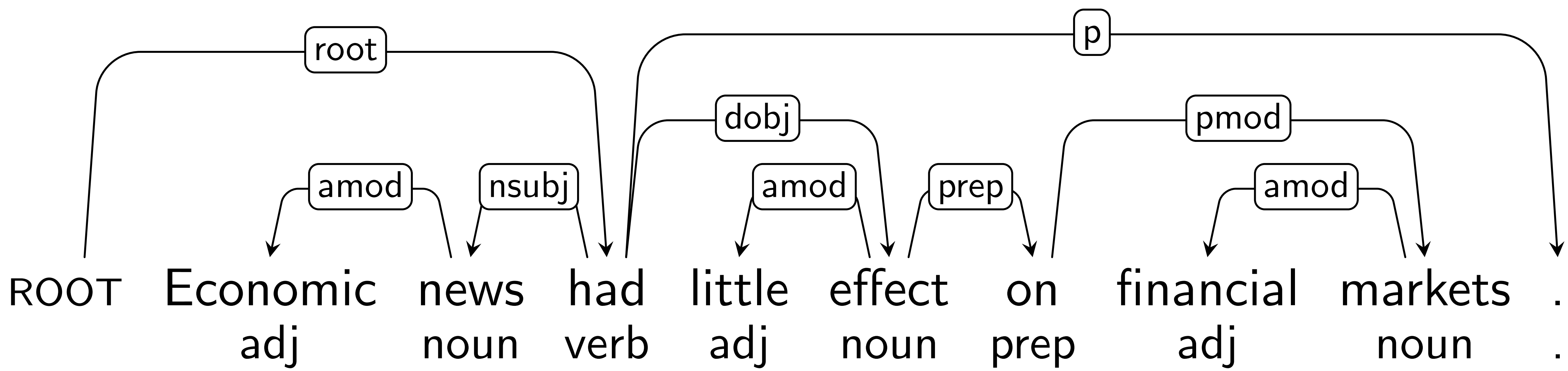
Example Transition Sequence

$[\text{ROOT, had}]_S \quad [.]_B$



Example Transition Sequence

$[\text{ROOT}, \text{had}, .]_S$ $[\ ]_B$



Greedy Inference

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

Parse$(w_1, \ldots, w_n)$

$c \leftarrow ([\,]_S, [0, 1, \ldots, n]_B, \{\,\})$

while $B_c \neq [\,]$

$\quad t \leftarrow o(c)$

$\quad c \leftarrow t(c)$

return $G = (\{0, 1, \ldots, n\}, A_c)$

- Complexity given by upper bound on number of transitions
- Parsing in $O(n)$ time for the arc-eager transition system

From Oracles to Classifiers

- An oracle can be approximated by a (linear) classifier:

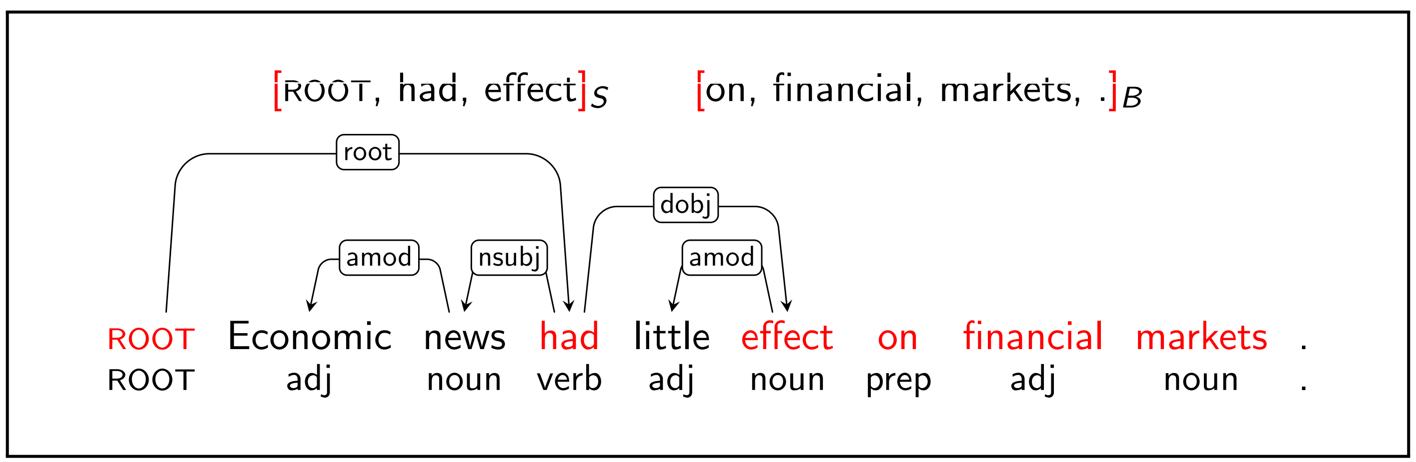
$$o(c) = \operatorname*{argmax}_{t} \mathbf{w} \cdot \mathbf{f}(c, t)$$

- History-based feature representation $\mathbf{f}(c, t)$
- Weight vector $\mathbf{w}$ learned from treebank data

Feature Representation

- Features over input tokens relative to S and B

Configuration



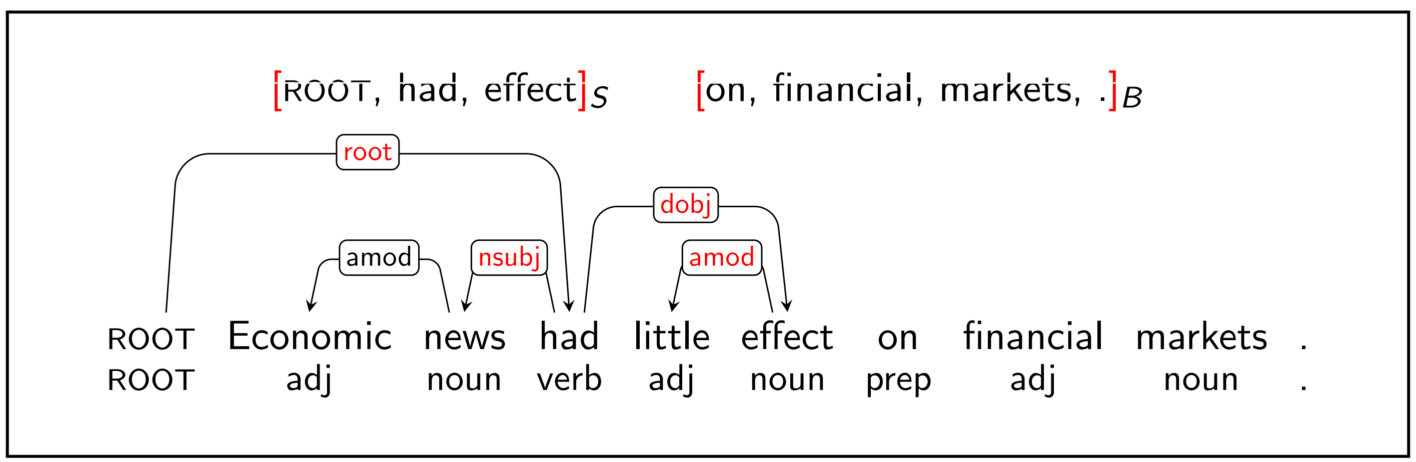
Features

$$
\begin{aligned}
\text{word}(S_2) &= \text{ROOT} \\
\text{word}(S_1) &= \text{had} \\
\text{word}(S_0) &= \text{effect} \\
\text{word}(B_0) &= \text{on} \\
\text{word}(B_1) &= \text{financial} \\
\text{word}(B_2) &= \text{markets}
\end{aligned}
$$

Feature Representation

- Features over input tokens relative to S and B
- Features over the (partial) dependency graph defined by A

Configuration



Features

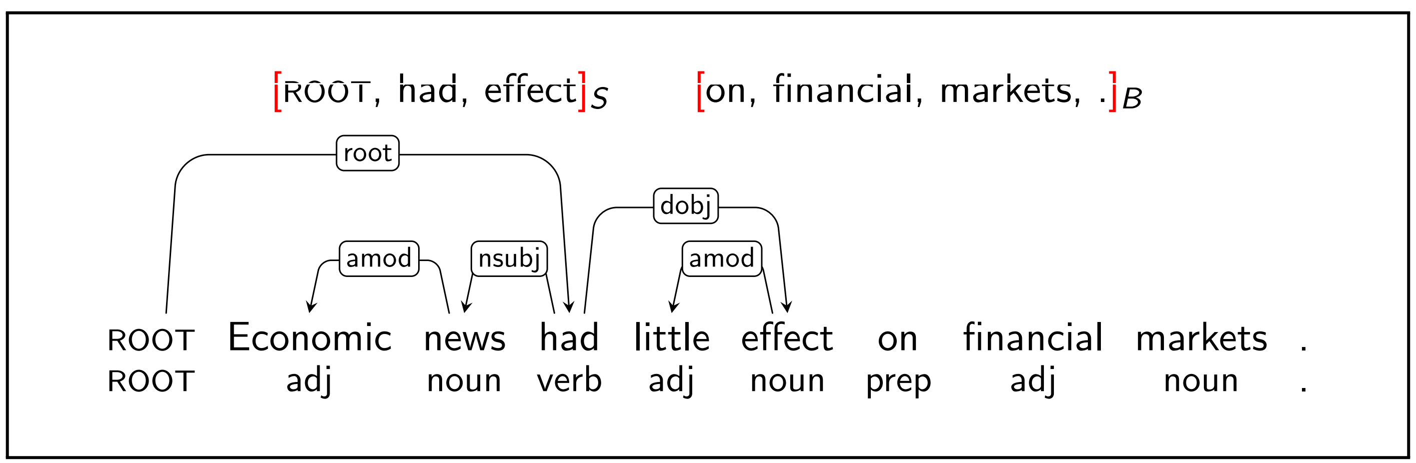
$$
\begin{aligned}
\mathrm{dep}(S_1) &= \mathrm{root} \\
\mathrm{dep}(\mathrm{lc}(S_1)) &= \mathrm{nsubj} \\
\mathrm{dep}(\mathrm{rc}(S_1)) &= \mathrm{dobj} \\
\mathrm{dep}(S_0) &= \mathrm{dobj} \\
\mathrm{dep}(\mathrm{lc}(S_0) &= \mathrm{amod} \\
\mathrm{dep}(\mathrm{rc}(S_0) &= \mathrm{NIL}
\end{aligned}
$$

Tuesday, March 21, 17

Feature Representation

- Features over input tokens relative to S and B
- Features over the (partial) dependency graph defined by A
- Features over the (partial) transition sequence

Configuration



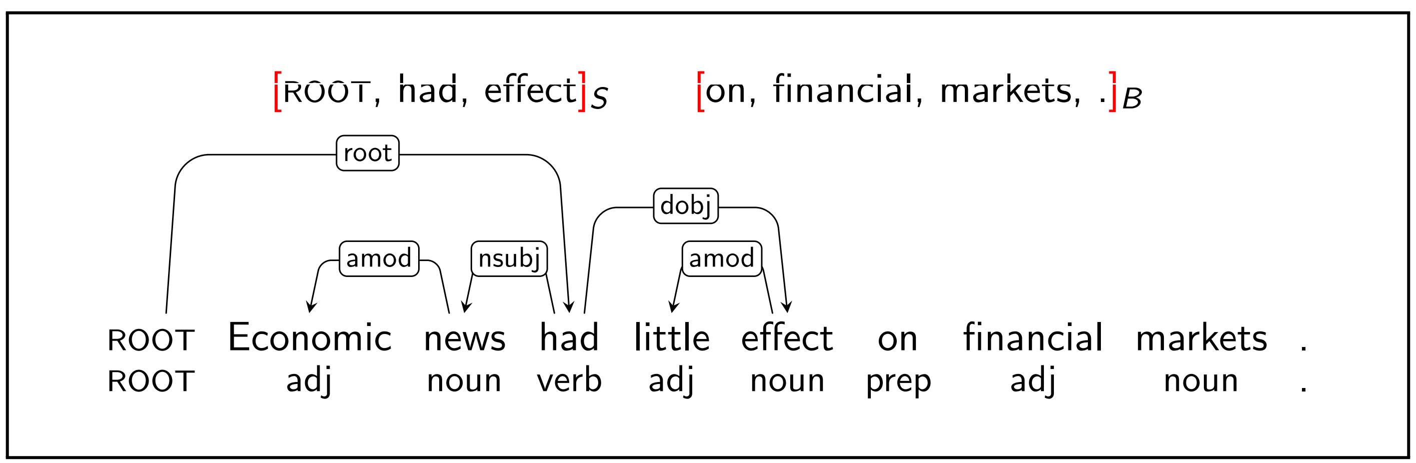
Features

$$
\begin{aligned}
t_{i-1} &= \text{Right-Arc(dobj)} \\
t_{i-2} &= \text{Left-Arc(amod)} \\
t_{i-3} &= \text{Shift} \\
t_{i-4} &= \text{Right-Arc(root)} \\
t_{i-5} &= \text{Left-Arc(nsubj)} \\
t_{i-6} &= \text{Shift}
\end{aligned}
$$

Feature Representation

- Features over input tokens relative to S and B
- Features over the (partial) dependency graph defined by A
- Features over the (partial) transition sequence

Configuration



Features

$$
\begin{aligned}
t_{i-1} &= \text{Right-Arc(dobj)}\\
t_{i-2} &= \text{Left-Arc(amod)}\\
t_{i-3} &= \text{Shift}\\
t_{i-4} &= \text{Right-Arc(root)}\\
t_{i-5} &= \text{Left-Arc(nsubj)}\\
t_{i-6} &= \text{Shift}
\end{aligned}
$$

- Feature representation unconstrained by parsing algorithm

Local Learning

- Given a treebank:
  - Reconstruct oracle transition sequence for each sentence
  - Construct training data set $D = \{(c, t) \mid o(c) = t\}$
  - Maximize accuracy of local predictions $o(c) = t$
- Any (unstructured) classifier will do (SVMs are popular)
- Training is local and restricted to oracle configurations

Linear vs neural features

- Non-stateful approaches
  - Nivre (~2003 & others), "MALT": linear SVM to make shift-reduce decisions, trained on oracle decisions
  - Chen and Manning (2014): neural softmax, trained on oracle decisions
  - Andors et al. (2016), "SyntaxNet": similar but with global normalization (CRF-ish)
- Stateful: Stack LSTM over state transitions (Dyer et al., like last week)

Greedy, Local, Transition-Based Parsing

- Advantages:
  - Highly efficient parsing – linear time complexity with constant time oracles and transitions
  - Rich history-based feature representations – no rigid constraints from inference algorithm
- Drawback:
  - Sensitive to search errors and error propagation due to greedy inference and local learning
- The major question in transition-based parsing has been how to improve learning and inference, while maintaining high efficiency and rich feature models

Beam Search

- Maintain the k best hypotheses [Johansson and Nugues 2006]:

  Parse($w_1, \ldots, w_n$)

  Beam $\leftarrow \{([\,]_S, [0, 1, \ldots, n]_B, \{\,\})\}$

  while $\exists c \in$ Beam $[B_c \neq [\,]]$

      foreach $c \in$ Beam

          foreach t

              Add($t(c)$, NewBeam)

      Beam $\leftarrow$ Top(k, NewBeam)

  return $G = (\{0, 1, \ldots, n\}, A_{\mathrm{Top}(1,\,\mathrm{Beam})})$

- Note:
  - $\mathrm{Score}(c_0, \ldots, c_m) = \sum_{i=1}^{m} \mathbf{w} \cdot \mathbf{f}(c_{i-1}, t_i)$
  - Simple combination of locally normalized classifier scores
  - Marginal gains in accuracy

(Beam search can even hurt, since training only on oracle paths)

State of the art

- Unlabeled attachment scores: Accuracy of choose-the-parent
- As of 2014, on old CoNLL 2006 data (variable quality)

	Best Published
Arabic	81.12 (MS11)
Bulgarian	94.02 (ZH13)
Chinese	92.68 (LX14)
Czech	91.04 (ZL14)
Danish	92.00 (ZH13)
Dutch	86.47 (ZL14)
English	93.22 (MA13)
German	92.41 (MA13)
Japanese	93.74 (LX14)
Portuguese	93.03 (KR10)
Slovene	86.95 (MS11)
Spanish	88.24 (ZL14)
Swedish	91.62 (ZH13)
Turkish	77.55 (KR10)
Average	89.58

Results table from Zhang et al.
http://people.csail.mit.edu/regina/my_papers/rand14.pdf

- Labeled attachment scores
- On newer "Universal Dependencies" data (higher quality??) with stack LSTM shift-reduce model

de	en	es	fr	it	pt	sv
79.3	**85.9**	83.7	81.7	88.7	85.7	83.5

Results from Ammar et al.
https://arxiv.org/pdf/1602.01595.pdf

- CoNLL 2017 shared task right now... http://universaldependencies.org/conll17/