Lecture 11: Parsing II

Xu Ruifeng

Harbin Institute of Technology, Shenzhen

Last Time

- Chomsky Hierarchy and Grammar
- Parsing Approaches for Context-Free Grammar (CFG)
  - Top-down Approach
  - Bottom-up Approach
- Parsing Algorithms
  - Shift-reduce Parsing
  - Chart Parsing

Today's Class

- Probabilistic Context-Free Grammar (PCFG)
- Lexicalized PCFG
- Dependency Syntax
- Dependency Parsing
- Parsing Resources

Statistical Parsing

- Statistical parsing uses a probabilistic model of syntax in order to assign probabilities to each parse tree.
- Provides principled approach to resolving syntactic ambiguity.
- Allows supervised learning of parsers from tree-banks of parse trees provided by human linguists.
- Also allows unsupervised learning of parsers from raw text, but the accuracy of such parsers has been limited.

Probabilistic Context Free Grammar (PCFG)

- A PCFG is a probabilistic version of a CFG where each production has a probability.
- Probabilities of all productions rewriting a given non-terminal must add to 1, defining a distribution for each non-terminal.

$$\sum_{a} P(A -> \alpha) = 1$$

- String generation is now probabilistic where production probabilities are used to non-deterministically select a production for rewriting a given non-terminal.

PCFG

- Assumption 1: Place invariance
- Assumption 2: Context-free
- Assumption 3: Ancestor-free

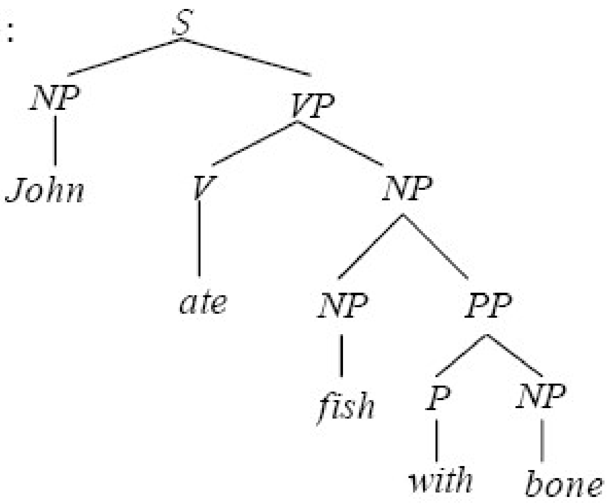
- The parse tree probability is the production of the probability of all applied parsing rules

Simple PCFG Example

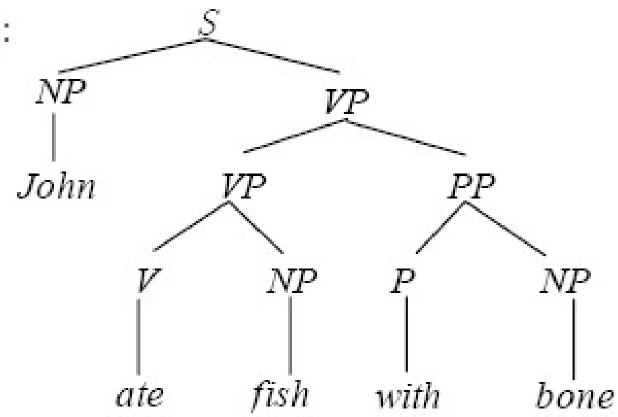
- Assume productions for each node are chosen independently.
- Probability of derivation is the product of the probabilities of its productions.
- Resolve ambiguity by picking most probable parse tree.

$sentence$ = "*John ate fish with bone*"

t_1:



t_2:



$S \rightarrow NP\ VP$	1.0	$NP \rightarrow NP\ PP$	0.4
$PP \rightarrow P\ NP$	1.0	$NP \rightarrow John$	0.1
$VP \rightarrow V\ NP$	0.7	$NP \rightarrow bone$	0.18
$VP \rightarrow VP\ PP$	0.3	$NP \rightarrow star$	0.04
$P \rightarrow with$	1.0	$NP \rightarrow fish$	0.18
$V \rightarrow ate$	1.0	$NP \rightarrow telescope$	0.1

$$P(t_1) = 1.0 \times 0.1 \times 0.7 \times 1.0 \times 0.4 \times 0.18 \times 1.0 \times 1.0 \times 0.18$$
$$= 0.0009072$$

$$P(t_2) = 1.0 \times 0.1 \times 0.3 \times 0.7 \times 1.0 \times 0.18 \times 1.0 \times 1.0 \times 0.18$$
$$= 0.0006804$$

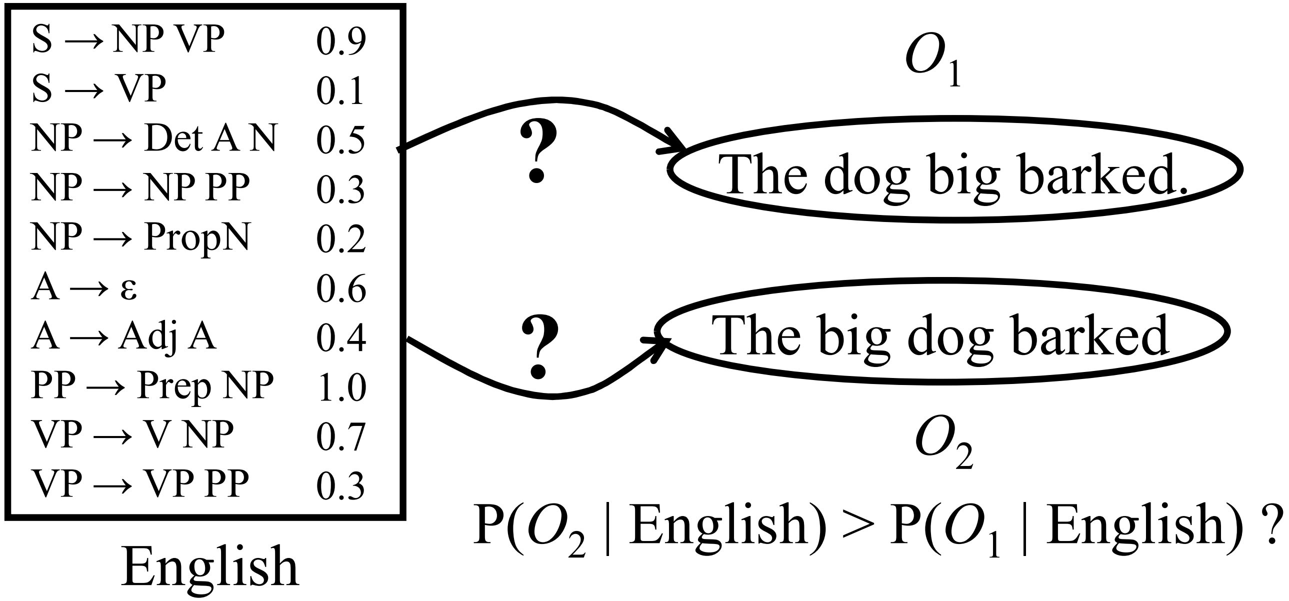
$$P(sentence) = P(t_1) + P(t_2) = 0.0015876$$

Three PCFG Tasks

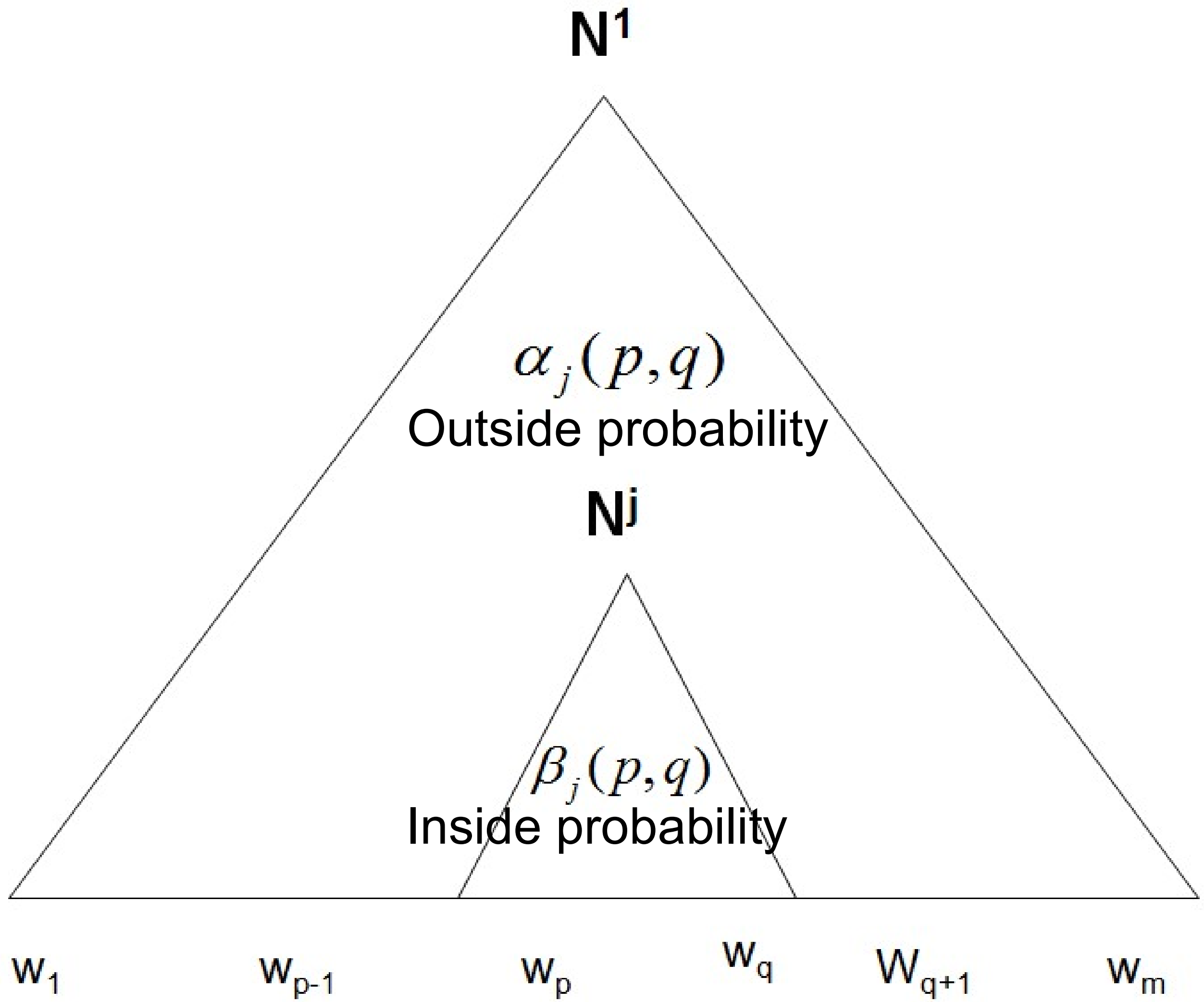
- Observation likelihood: To classify and order sentences. P(W|G) for sentence $W=w_1w_2…w_n$
  - Inside Algorithm
- Most likely derivation: To determine the most likely parse tree for a sentence.
  - Viterbi Algorithm
- Maximum likelihood training: To train a PCFG to fit empirical training data, i.e. maximize P(W|G)

PCFG: Observation Likelihood

- There is an analog to Forward algorithm for HMMs called the Inside algorithm for efficiently determining how likely a string is to be produced by a PCFG.



Notation



N^1
$\alpha_j(p,q)$
Outside probability
N^j
$\beta_j(p,q)$
Inside probability
w_1
w_{p-1}
w_p
w_q
W_{q+1}
w_m

Inside and Outside Probability

- Inside probability: total probability of generating words $w_p \ldots w_q$ from non-terminal N^j.

$$\beta_j(p,q) = P(w_{pq} \mid N^j_{pq})$$

- Outside probability: total probability of beginning with the start symbol N^1 and generating N^j_{pq} and all the words outside $w_p \ldots w_q$

$$\alpha_j(p,q) = P(w_{1(p-1)}, N^j_{pq}, w_{(q+1)m})$$

- When p>q,

$$\alpha_j(p,q) = \beta_j(p,q) = 0$$

Probabilistic CKY Algorithm

- Revised CKY with probabilistic information
- Set and maintain production probability during transfer grammar rule to CNF
- Each cell provide a probability for each non-terminal
- Cell[i,j] ： the most likely parsing contains all of Non-terminals which covers Word_{i+1} to Word_{j} and its probability

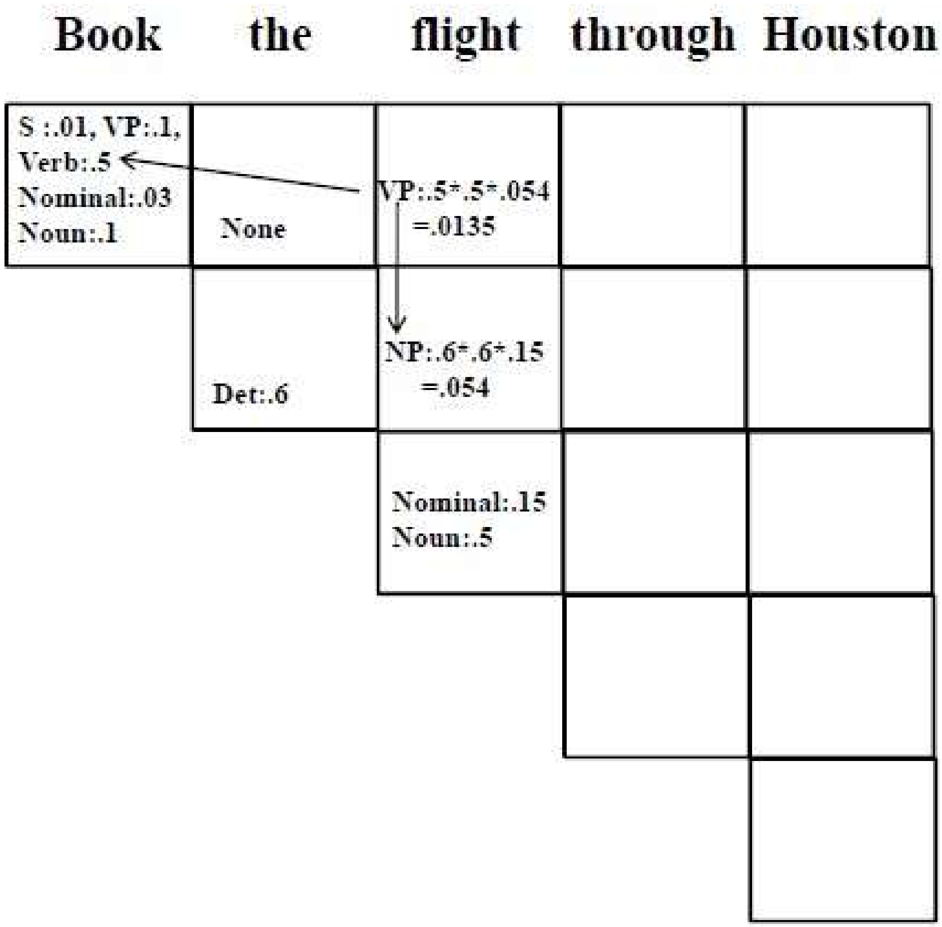
Probabilistic CKY Algorithm

Original Grammar		Chomsky Normal Form	
S → NP VP	0.8	S → NP VP	0.8
S → Aux NP VP	0.1	S → X1 VP	0.1
		X1 → Aux NP	1.0
S → VP	0.1	S → book | include | prefer 0.01 0.004 0.006	
		S → Verb NP	0.05
		S → VP PP	0.03
NP → Pronoun	0.2	NP → I | he | she | me 0.1 0.02 0.02 0.06	
NP → Proper-Noun	0.2	NP → Houston | NWA 0.16 .04	
NP → Det Nominal	0.6	NP → Det Nominal	0.6
Nominal → Noun	0.3	Nominal → book | flight | meal | money 0.03 0.15 0.06 0.06	
Nominal → Nominal Noun	0.2	Nominal → Nominal Noun	0.2
Nominal → Nominal PP	0.5	Nominal → Nominal PP	0.5
VP → Verb	0.2	VP → book | include | prefer 0.1 0.04 0.06	
VP → Verb NP	0.5	VP → Verb NP	0.5
VP → VP PP	0.3	VP → VP PP	0.3
PP → Prep NP	1.0	PP → Prep NP	1.0

Probabilistic CKY Algorithm

Book	the	flight	through	Houston
S :.01, VP:.1, Verb:.5 Nominal:.03 Noun:.1	None			
	Det:.6	NP:.6*.6*.15 =.054		
		Nominal:.15 Noun:.5		

Probabilistic CKY Algorithm



Probabilistic CKY Algorithm

Book	the	flight	through	Houston
S :.01, VP:.1, Verb:.5 Nominal:.03 Noun:.1	None	S:.05*.5*.054 =.00135 VP:.5*.5*.054 =.0135		
	Det:.6	NP:.6*.6*.15 =.054		
		Nominal:.15 Noun:.5		

Probabilistic CKY Algorithm

Book	the	flight	through	Houston
S :.01, VP:.1, Verb:.5 Nominal:.03 Noun:.1	None	S:.05*.5*.054 =.00135 VP:.5*.5*.054 =.0135	None	
	Det:.6	NP:.6*.6*.15 =.054	None	
		Nominal:.15 Noun:.5	None	
			Prep:.2	

Probabilistic CKY Algorithm

Book	the	flight	through	Houston
S :.01, VP:.1, Verb:.5 Nominal:.03 Noun:.1	None	S:.05*.5*.054 =.00135 VP:.5*.5*.054 =.0135	None	
	Det:.6	NP:.6*.6*.15 =.054	None	
		Nominal:.15 Noun:.5	None	
			Prep:.2	PP:1.0*.2*.16 =.032
				NP:.16 PropNoun:.8

Probabilistic CKY Algorithm

Book	the	flight	through	Houston
S :.01, VP:.1, Verb:.5 Nominal:.03 Noun:.1	None	S:.05*.5*.054 =.00135 VP:.5*.5*.054 =.0135	None	
	Det:.6	NP:.6*.6*.15 =.054	None	
		Nominal:.15 Noun:.5	None	Nominal: .5*.15*.032 =.0024
			Prep:.2	PP:1.0*.2*.16 =.032
				NP:.16 PropNoun:.8

Probabilistic CKY Algorithm

Book	the	flight	through	Houston
S :.01, VP:.1, Verb:.5 Nominal:.03 Noun:.1	None	S:.05*.5*.054 =.00135 VP:.5*.5*.054 =.0135	None	
	Det:.6	NP:.6*.6*.15 =.054	None	NP:.6*.6* .0024 =.000864
		Nominal:.15 Noun:.5	None	Nominal: .5*.15*.032 =.0024
			Prep:.2	PP:1.0*.2*.16 =.032
				NP:.16 PropNoun:.8

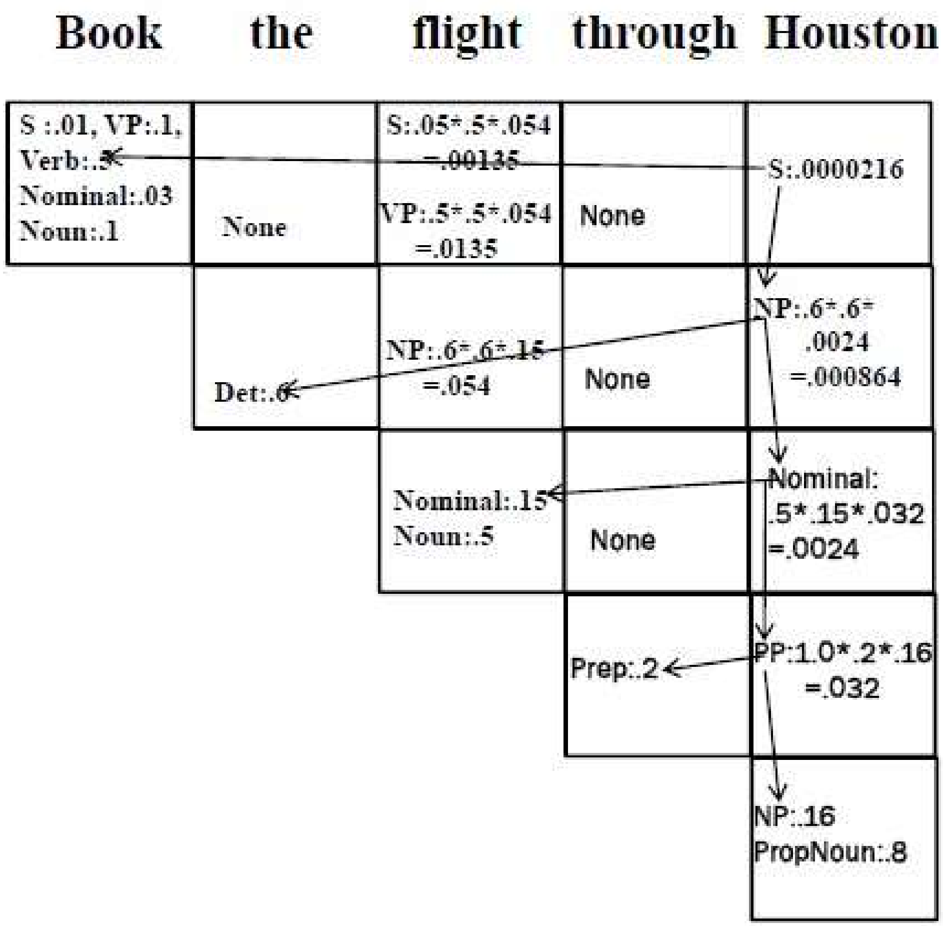
Probabilistic CKY Algorithm

Book	the	flight	through	Houston
S :.01, VP:.1, Verb:.5 Nominal:.03 Noun:.1	None	S:.05*.5*.054 =.00135 VP:.5*.5*.054 =.0135	None	S:.05*.5* .000864 =.0000216
	Det:.6	NP:.6*.6*.15 =.054	None	NP:.6*.6* .0024 =.000864
		Nominal:.15 Noun:.5	None	Nominal: .5*.15*.032 =.0024
			Prep:.2	PP:1.0*.2*.16 =.032
				NP:.16 PropNoun:.8

Probabilistic CKY Algorithm

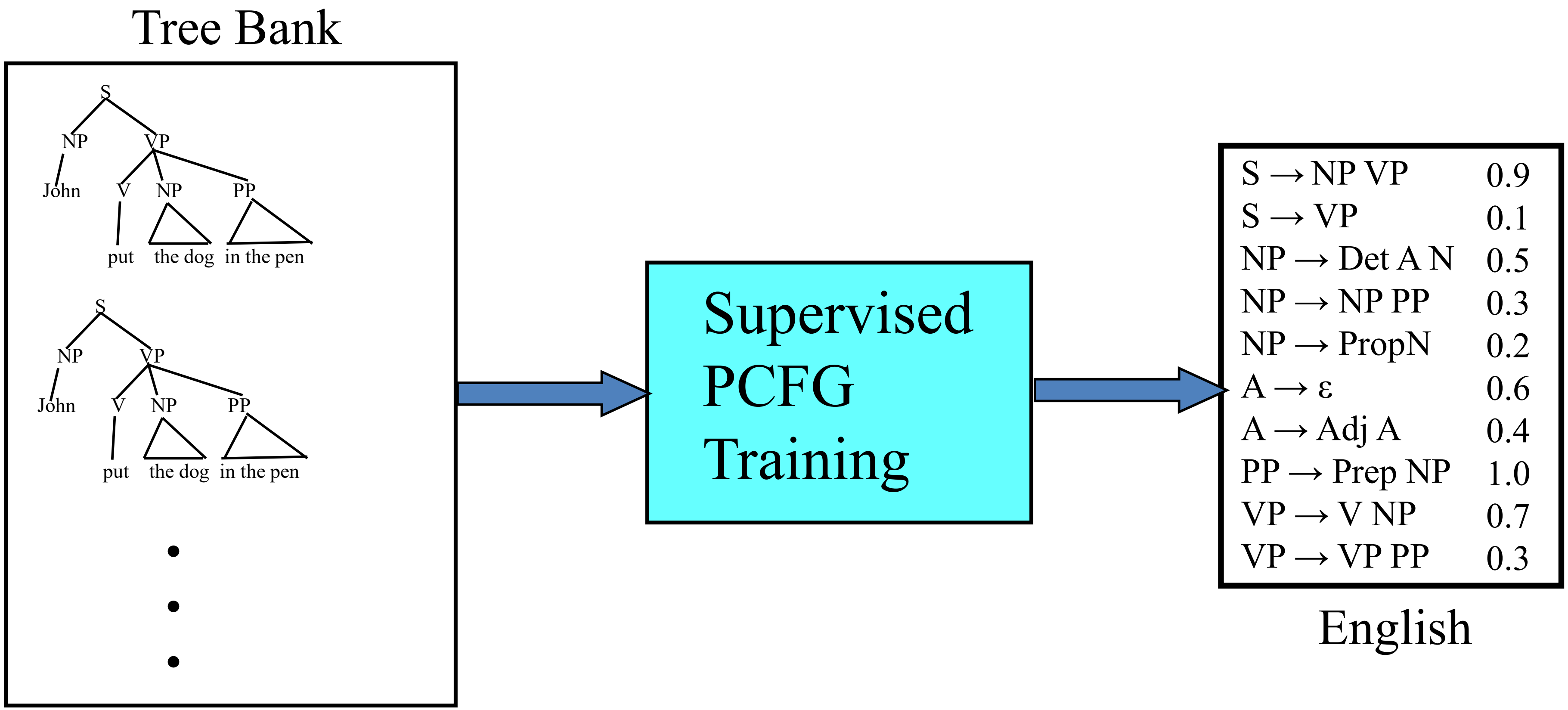
Book	the	flight	through	Houston
S :.01, VP:.1, Verb:.5 Nominal:.03 Noun:.1	None	S:.05*.5*.054 =.00135 VP:.5*.5*.054 =.0135	None	S:.03*.0135* .032 =.00001296 S:.0000216
	Det:.6	NP:.6*.6*.15 =.054	None	NP:.6*.6* .0024 =.000864
		Nominal:.15 Noun:.5	None	Nominal: .5*.15*.032 =.0024
			Prep:.2	PP:1.0*.2*.16 =.032
				NP:.16 PropNoun:.8

Probabilistic CKY Algorithm



PCFG: Supervised Training

- If parse trees are provided for training sentences, a grammar and its parameters can be estimated directly from counts accumulated from the tree-bank (with appropriate smoothing).



Estimating Production Probabilities

- Set of production rules can be taken directly from the set of rewrites in the treebank.
- Parameters can be directly estimated from frequency counts in the treebank.

$$P(\alpha \rightarrow \beta \mid \alpha) = \frac{\text{count}(\alpha \rightarrow \beta)}{\sum_{\gamma} \text{count}(\alpha \rightarrow \gamma)} = \frac{\text{count}(\alpha \rightarrow \beta)}{\text{count}(\alpha)}$$

PCFG: Maximum Likelihood Training

- Given a set of sentences, induce a grammar that maximizes the probability that this data was generated from this grammar.
- Assume the number of non-terminals in the grammar is specified.
- Only need to have an unannotated set of sequences generated from the model. Does not need correct parse trees for these sentences. In this sense, it is unsupervised.

PCFG: Maximum Likelihood Training

Training Sentences

John ate the apple
A dog bit Mary
Mary hit the dog
John gave Mary the cat.
•
•
•

PCFG Training

S → NP VP	0.9
S → VP	0.1
NP → Det A N	0.5
NP → NP PP	0.3
NP → PropN	0.2
A → ε	0.6
A → Adj A	0.4
PP → Prep NP	1.0
VP → V NP	0.7
VP → VP PP	0.3

English

Inside-Outside Algorithm

- The **Inside-Outside algorithm** is a version of EM for unsupervised learning of a PCFG.
  - Analogous to Baum-Welch (forward-backward) for HMMs
- Given the number of non-terminals, construct all possible CNF productions with these non-terminals and observed terminal symbols.
- Use EM to iteratively train the probabilities of these productions to locally maximize the likelihood of the data.
- Experimental results are not impressive, but recent work imposes additional constraints to improve unsupervised grammar learning.

Summary of PCFG

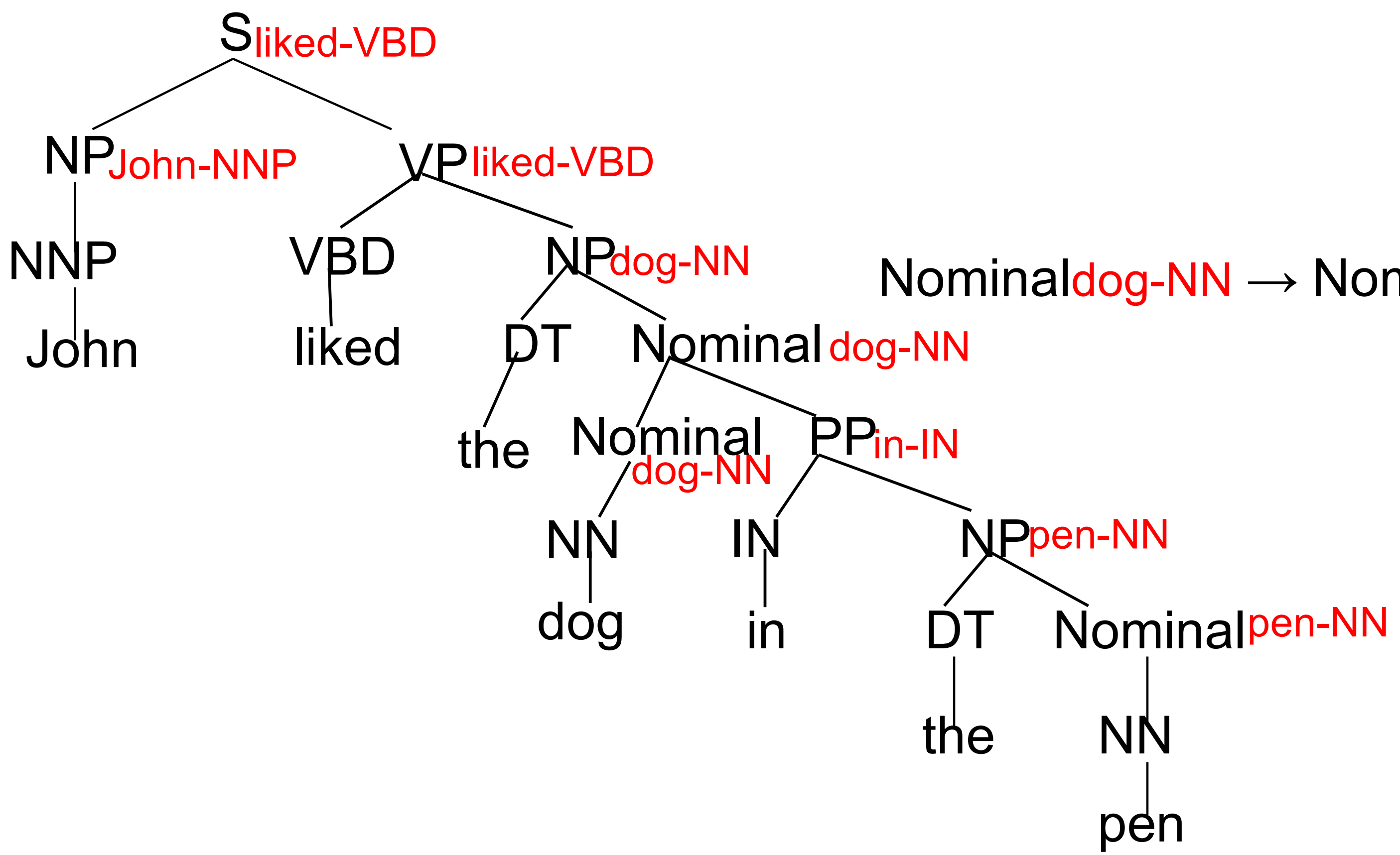
- Pros
  - Disambiguation of structurally different parses
  - Speed syntactic analysis by eliminate low probability structures
  - Improve the robust of analyzer by use of low probabilities
  - Helpful to grammar induction
- Cons
  - Since probabilities of productions do not rely on specific words or concepts, only general structural disambiguation is possible
  - cannot resolve syntactic ambiguities that require semantics to resolve, e.g. ate with fork vs. meatballs.

Lexicalized PCFG

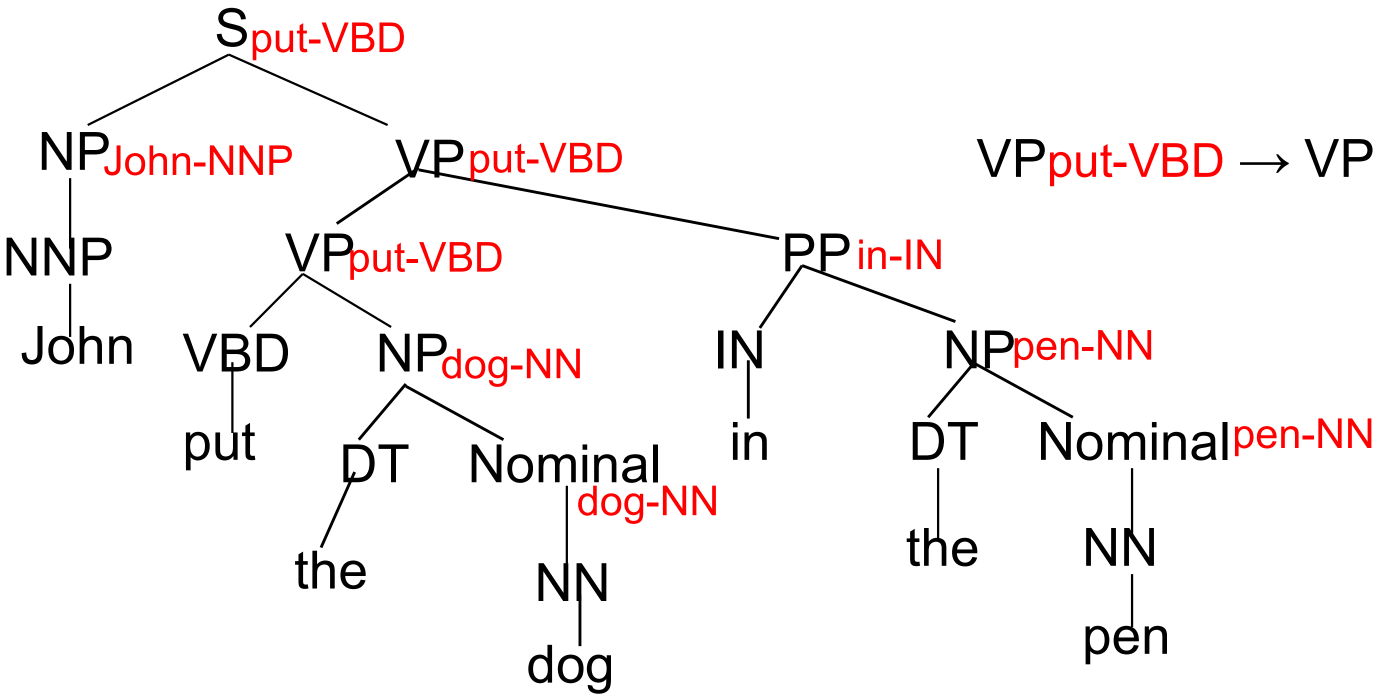
- PCFGs must be lexicalized, i.e. productions must be specialized to specific words by including their head-word in their LHS non-terminals
- Each non-terminal is associated to headword *w*
- Syntactic phrases usually have a word in them that is most "central" to the phrase.
- Process Rule
  - Head of a VP is the main verb
  - Head of an NP is the main noun
  - Head of a PP is the preposition
  - Head of a sentence is the head of its VP

Lexicalized Productions

- Specialized productions can be generated by including the head word and its POS of each non-terminal as part of that non-terminal's symbol



$\text{Nominal}_{\text{dog-NN}} \rightarrow \text{Nominal}_{\text{dog-NN}}\ \text{PP}_{\text{in-IN}}$



$$VP_{put\text{-}VBD} \rightarrow VP_{put\text{-}VBD}\ PP_{in\text{-}IN}$$

Parameterizing Lexicalized Productions

- Accurately estimating parameters on such a large number of very specialized productions requires enormous amounts of treebank data.
- Need some way of estimating parameters for lexicalized productions that makes reasonable independence assumptions so that accurate probabilities for very specific rules can be learned.

Collins' Parser

- Collins' (1999) parser assumes a simple generative model of lexicalized productions.
- Models productions based on context to the left and the right of the head daughter.
  - LHS → $L_n L_{n-1} \ldots L_1 H\ R_1 \ldots R_{m-1} R_m$
- First generate the head (H) and then repeatedly generate left (L_i) and right (R_i) context symbols until the symbol STOP is generated.

Sample Production

VPput-VBD → VBDput-VBD NPdog-NN PPin-IN

VPput-VBD → STOP VBDput-VBD NPdog-NN PPin-IN STOP

L_1 H R_1 R_2 R_3

P_L(STOP | VPput-VBD) * P_H(VBD | VPput-VBD)*
P_R(NPdog-NN | VPput-VBD)*
P_R(PPin-IN | VPput-VBD) * P_R(STOP | VPput-VBD)

Estimating Production Generation Parameters

- Estimate P_H, P_L, and P_R parameters from treebank data.

$$P_R(\text{PPin-IN} \mid \text{VPput-VBD}) = \frac{\text{Count(PPin-IN right of head in a VPput-VBD production)}}{\text{Count(symbol right of head in a VPput-VBD)}}$$

$$P_R(\text{NPdog-NN} \mid \text{VPput-VBD}) = \frac{\text{Count(NPdog-NN right of head in a VPput-VBD production)}}{\text{Count(symbol right of head in a VPput-VBD)}}$$

- Smooth estimates by linearly interpolating with simpler models conditioned on just POS tag or no lexical info.

$$\begin{aligned} smP_R(\text{PPin-IN} \mid \text{VPput-VBD}) = {} & \lambda_1 P_R(\text{PPin-IN} \mid \text{VPput-VBD}) \\ & + (1-\lambda_1)\,(\lambda_2 P_R(\text{PPin-IN} \mid \text{VPVBD}) + \\ & (1-\lambda_2)\, P_R(\text{PPin-IN} \mid \text{VP})) \end{aligned}$$

Parsing Evaluations Metrics

- PARSEVAL metrics measure the fraction of the constituents that match between the computed and human parse trees. If *P* is the system's parse tree and *T* is the human parse tree (the "gold standard"):
  - ***Recall*** = (# correct constituents in *P*) / (# constituents in *T*)
  - ***Precision*** = (# correct constituents in *P*) / (# constituents in *P*)
- ***Labeled Precision*** and ***labeled recall*** require getting the non-terminal label on the constituent node correct to count as correct.
- F_1 is the harmonic mean of precision and recall.

Summary of Probabilistic Parsing

- Statistical models such as PCFGs allow for probabilistic resolution of ambiguities.
- PCFGs can be easily learned from treebanks.
- Lexicalization and non-terminal splitting are required to effectively resolve many ambiguities.
- Current statistical parsers are quite accurate but not yet at the level of human-expert agreement.

Dependency

- Syntactic structure consists of lexical items, linked by binary asymmetric relations called dependencies.
- [Tesniere 1959]

> The sentence is an *organized whole*, the constituent elements of which are *words*. 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. The structural connections establish *dependency* relations between the words. Each connection in principle unites a *superior* term and an *inferior* term. The superior term receives the name *governor*. The inferior term receives the name *subordinate*.

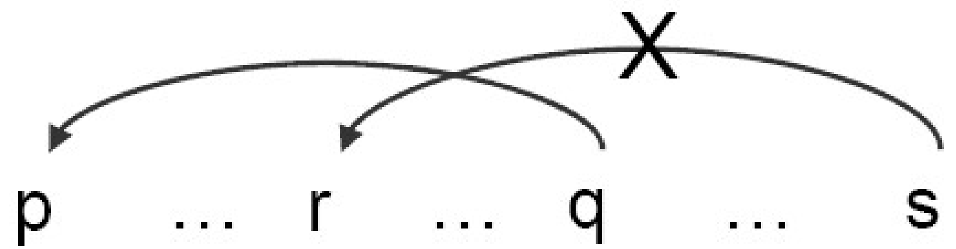
Robinson’s axiom

(a) one and only one element is independent;

(b) all others depend directly on some element;

(c) no element depends directly on more than one other; and

(d) (non-crossing constraint): if ***A*** depends directly to ***B*** and some element ***C*** intervenes between them (in linear order of string), then ***C*** depends directly on ***A*** or on ***B*** or to some other intervening element.



(e) An element cannot have dependents lying on the other side of its own governor.(Huang, Changning)

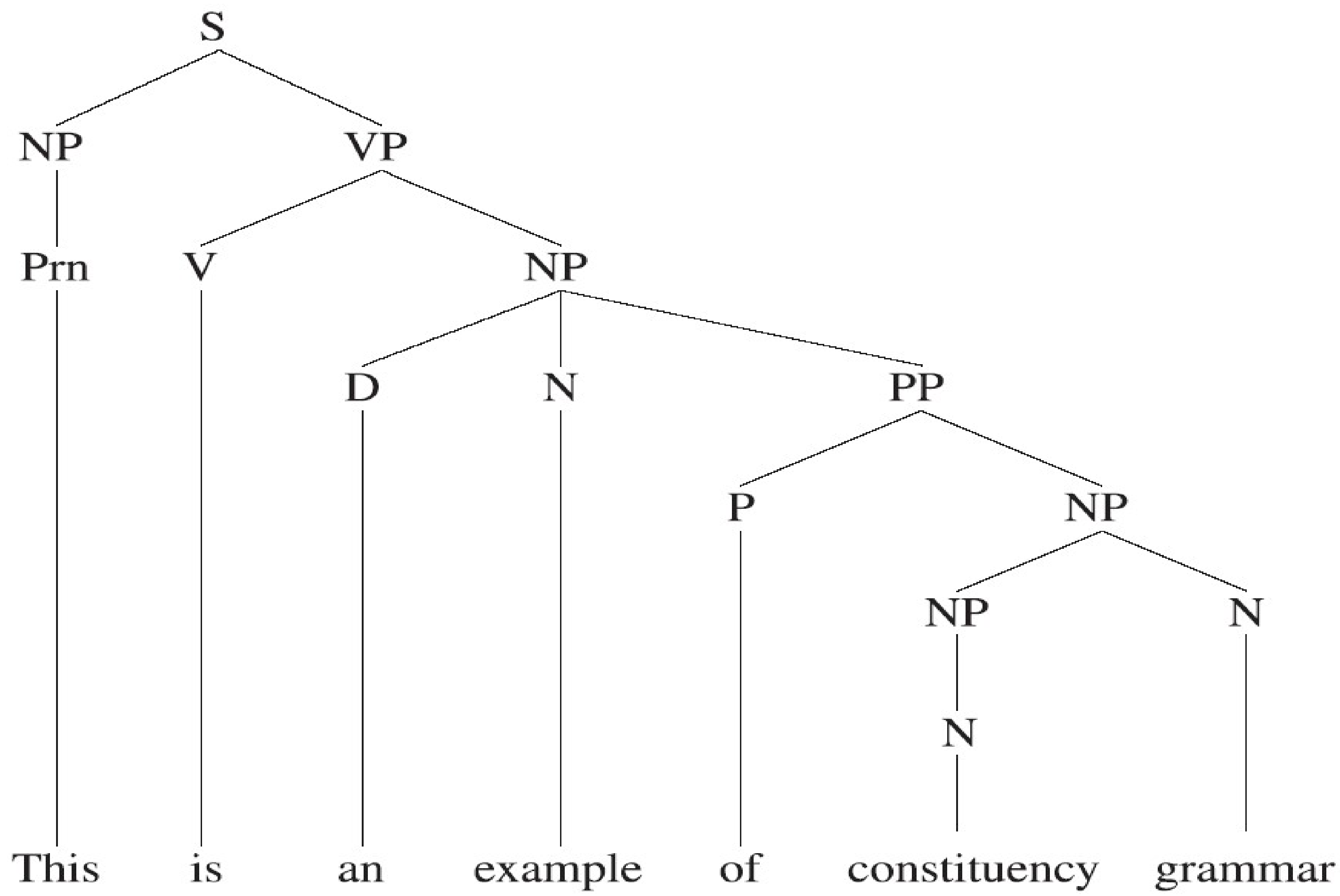
Element:

- Head (Superior, Head, Governor, Regent, 核心词)
  - Plays the role in determining the behavior of the pair.
- Dependent (Inferior, Modifier, Subordinate,从属词)
  - Modifier, object or complement
  - Pre-dependent, post-dependent
- Link – the relation (Arc)
  - Close to semantic relation
  - With Dependency Type

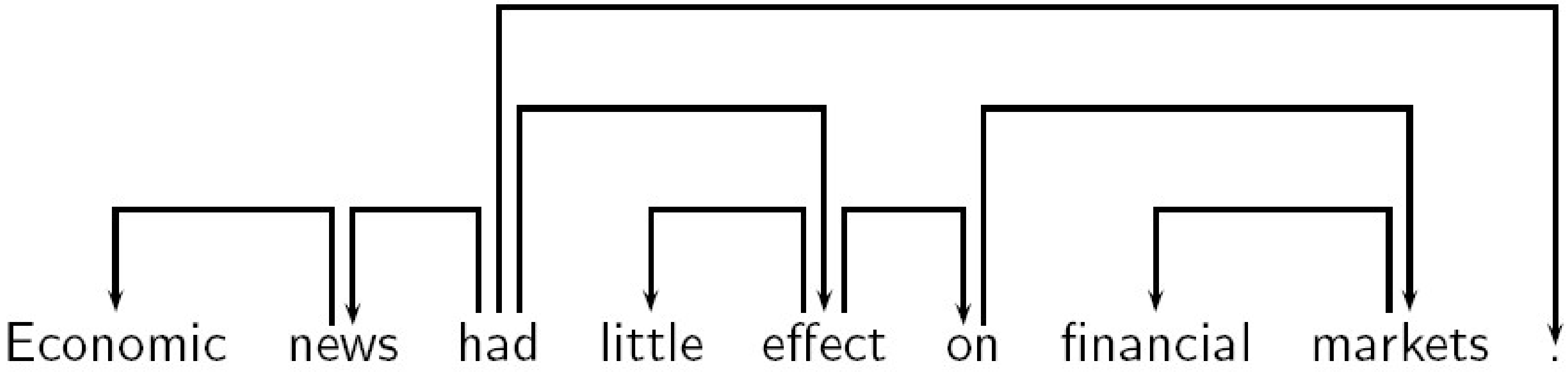
Typical Dependency Types

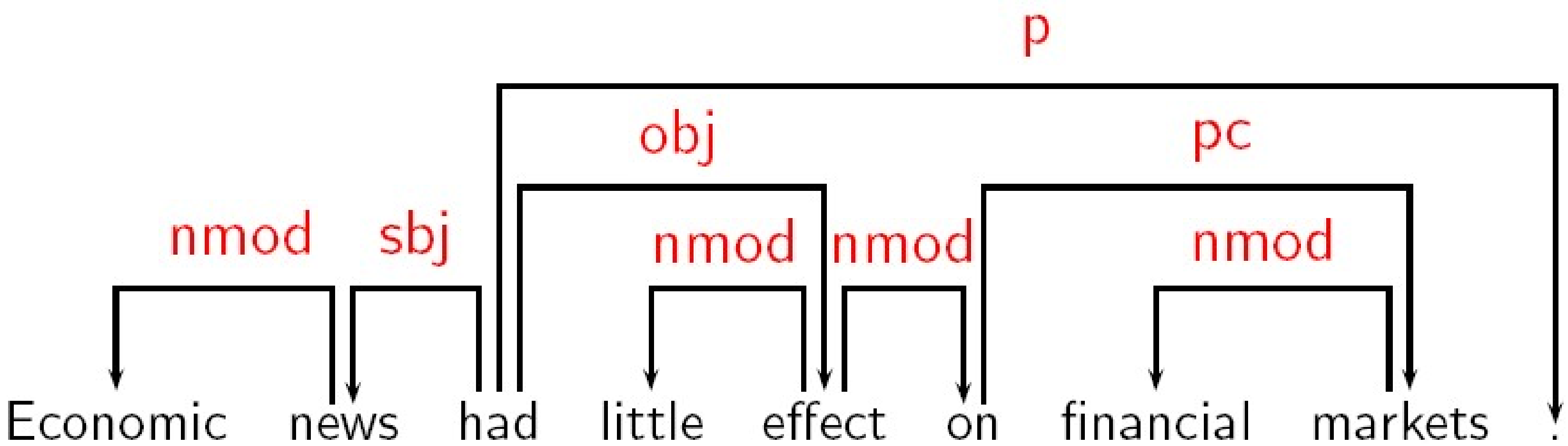
	Dependency Description	Example
	Adjective and its adverbial modifier	极其/d 惨痛/a *greatly painful*
ADV	Predicate and its adverbial modifier in which the predicate serves as head	沉重/ad 打击/v *heavily strike*
AN	Noun and its adjective modifier	合法/a 收入/n *lawful incoming*
CMP	Predicate and its complement in which the predicate serves as head	医治/v 无效/v *ineffectively treat*
NJX	Juxtaposition structure	公正/a 合理/a *fair and reasonable*
NN	Noun and its nominal modifier	人身/n 安全/n *personal safety*
SBV	Predicate and its subject	财产/n 转移/v *property transfer*
VO	Predicate and its object in which the predicate serves as head	转换/v 机制/n *change mechanism*
VV	Serial verb constructions which indicates that there are serial actions	跟踪/v 报导/v *trace and report*
OT	Others	

Phrase Structure



Dependency Structure





The Comparison

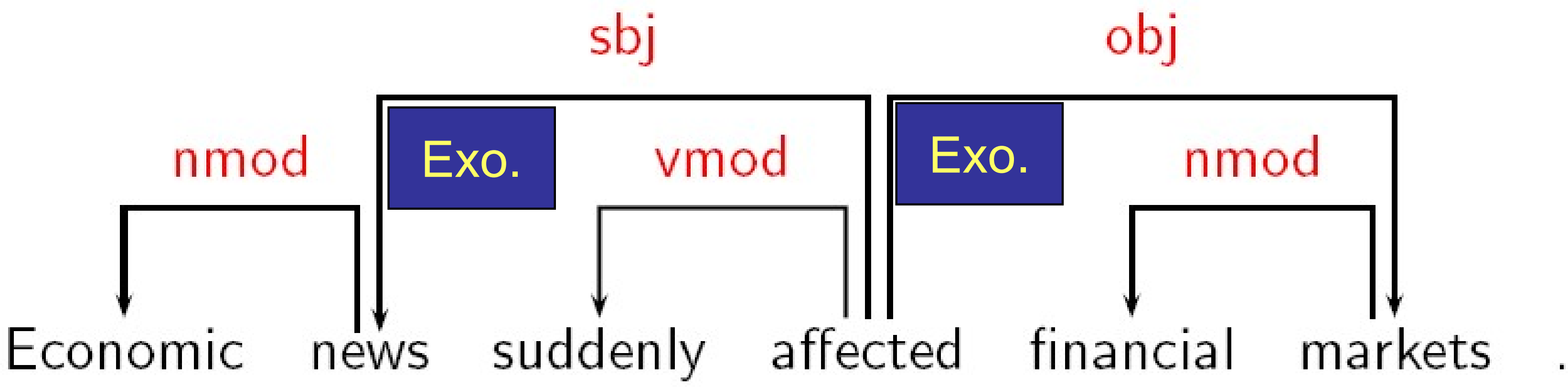
	Phrase structure	Dependency structure
word relation	phrasal constitution	head-dependent
categories	syntactic functional	syntactic structural
new node (Y/N)	multiple nodes per word	one node per word
operation	waiting for complete phrase	word-at-a-time

Some Theoretical Frameworks

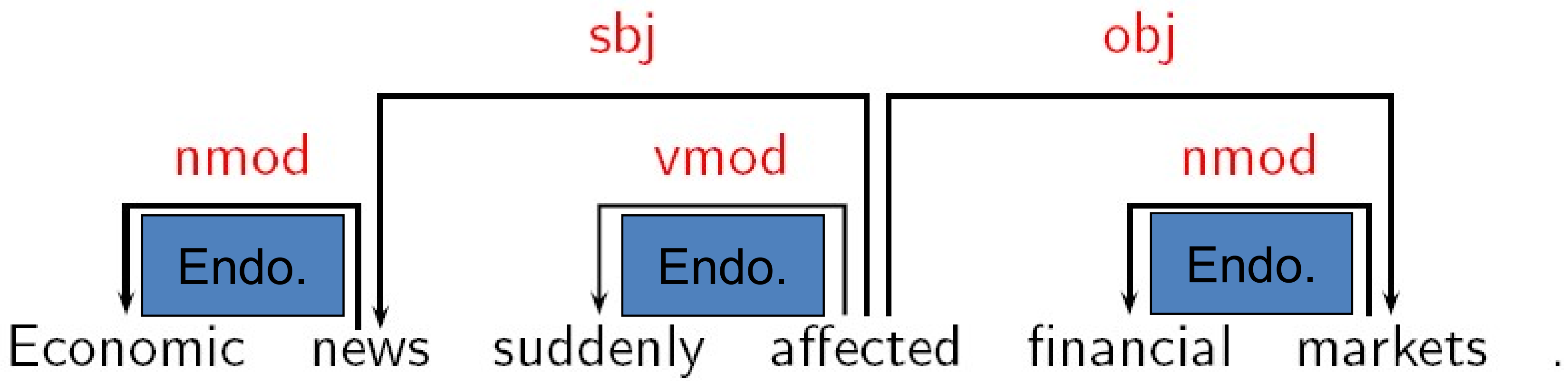
- Word Grammar (WG) [Hudson 1984, Hudson 1990]
- Functional Generative Description (FGD) [Sgall et al. 1986]
- Dependency Unification Grammar (DUG)
  - [Hellwig 1986, Hellwig 2003]
- Meaning-Text Theory (MTT) [Melcuk 1988]
- (Weighted) Constraint Dependency Grammar ([W]CDG)
  - [Maruyama 1990, Harper and Helzerman 1995,
  - Menzel and Schr ¨ oder 1998, Schr ¨ oder 2002]
- Functional Dependency Grammar (FDG)
  - [Tapanainen and Jarvinen 1997, Jarvinen and Tapanainen 1998]
- Topological/Extensible Dependency Grammar ([T/X]DG)
  - [Duchier and Debusmann 2001, Debusmann et al. 2004]

Examples

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

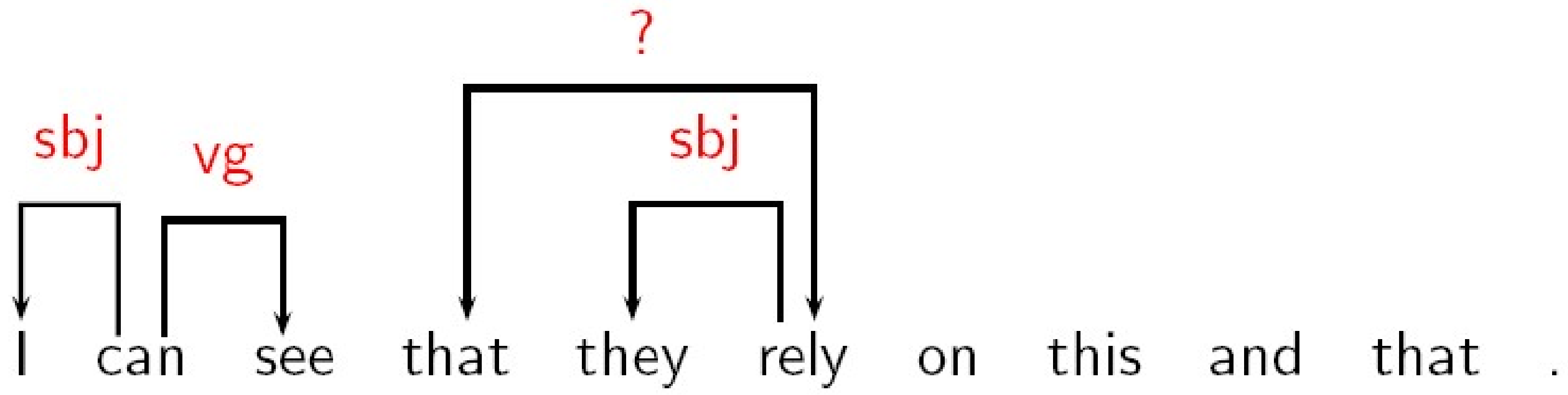


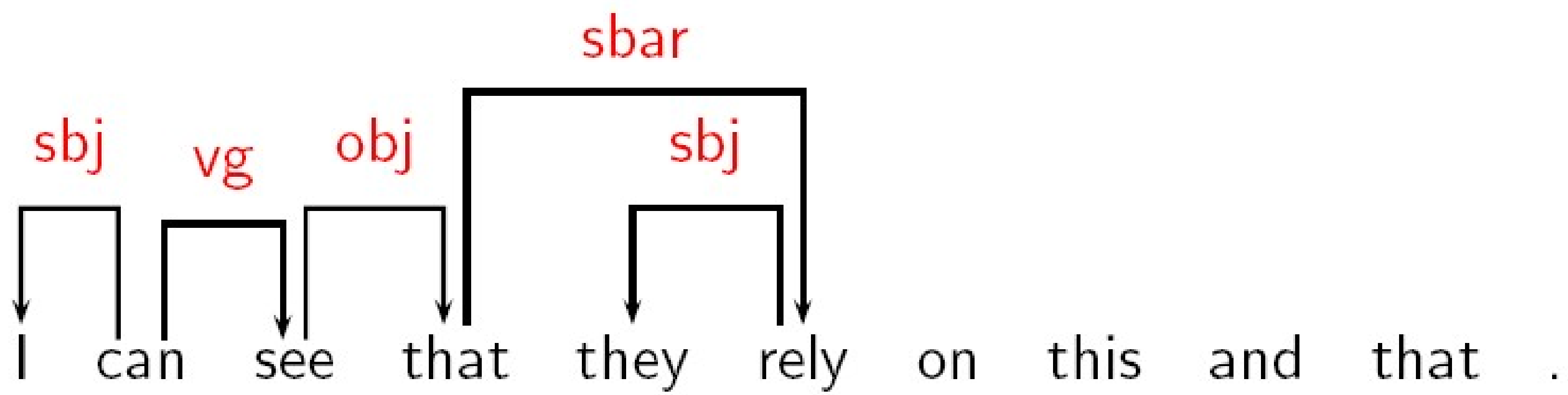
	Construction	Head	Dependent
outwards	Exocentric	Verb	Subject (sbj)
		Verb	Object (obj)
inwards	Endocentric	Verb	Adverbial (vmod)
		Noun	Attribute (nmod)



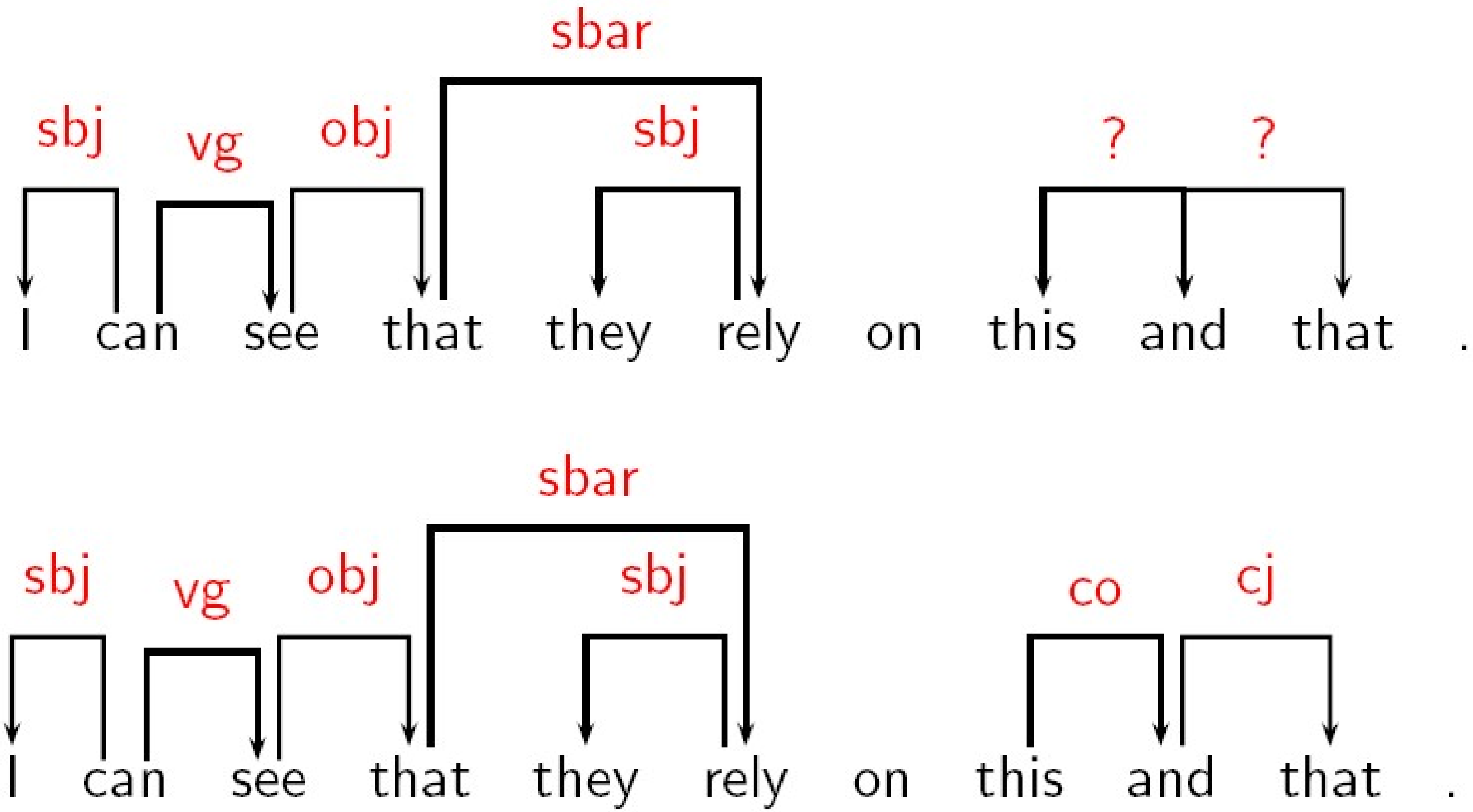
Tricky Cases

- Subordinate clauses (complementizer ↔ verb)

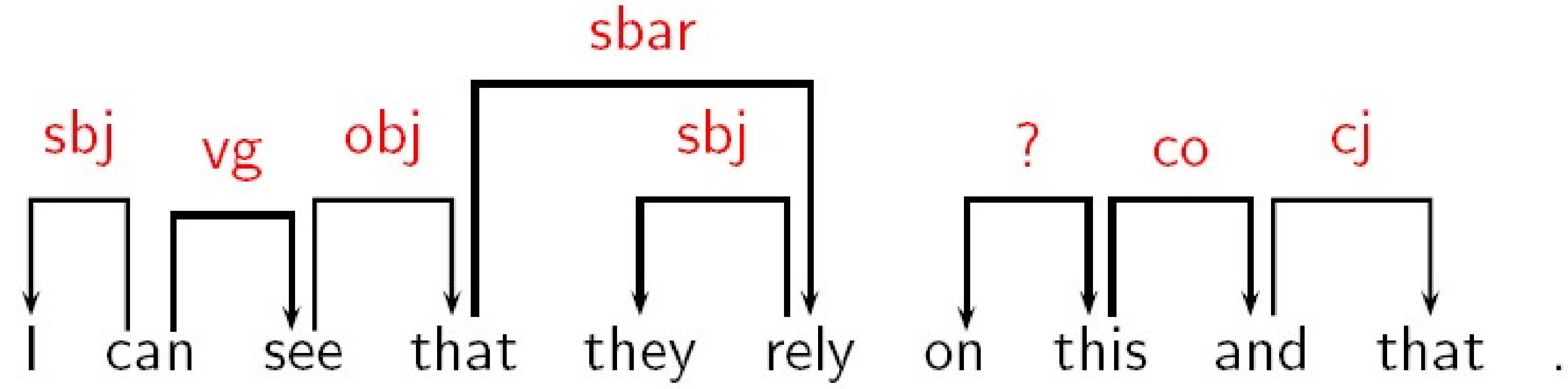


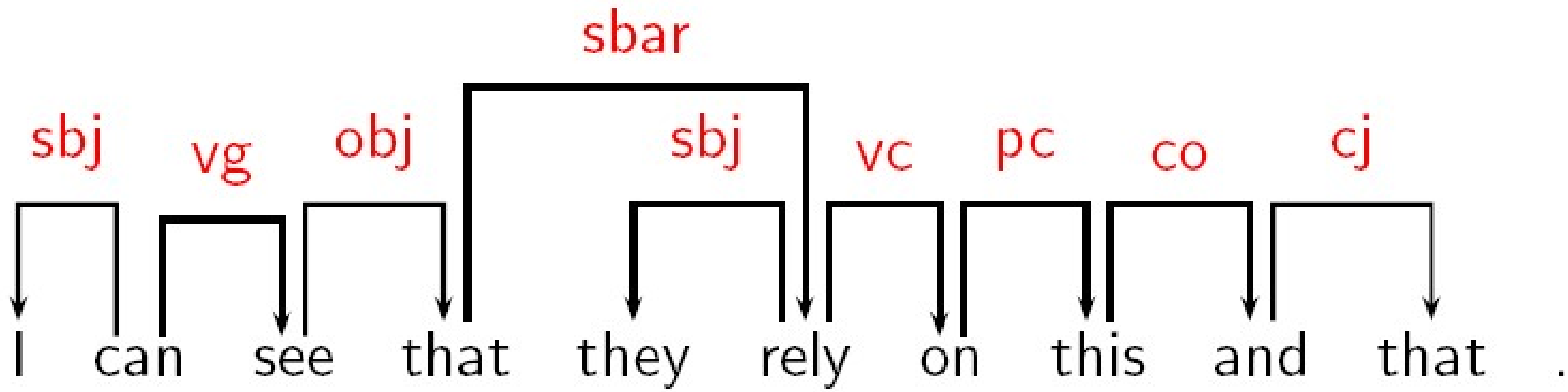


- Coordination (coordinator ↔ conjuncts)

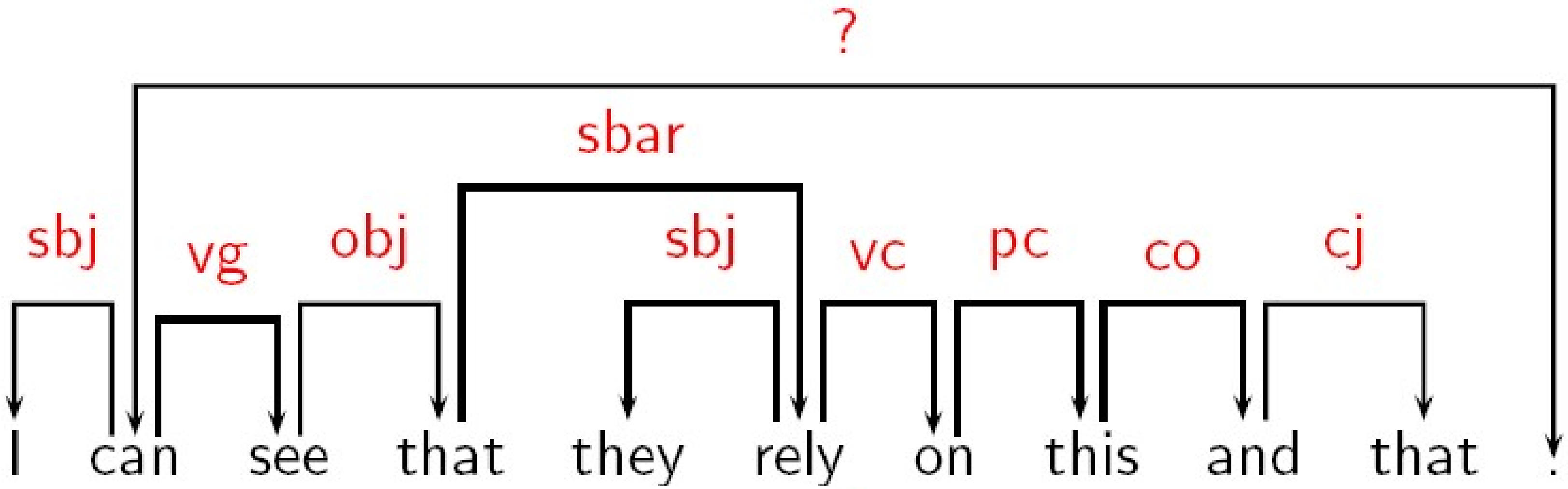


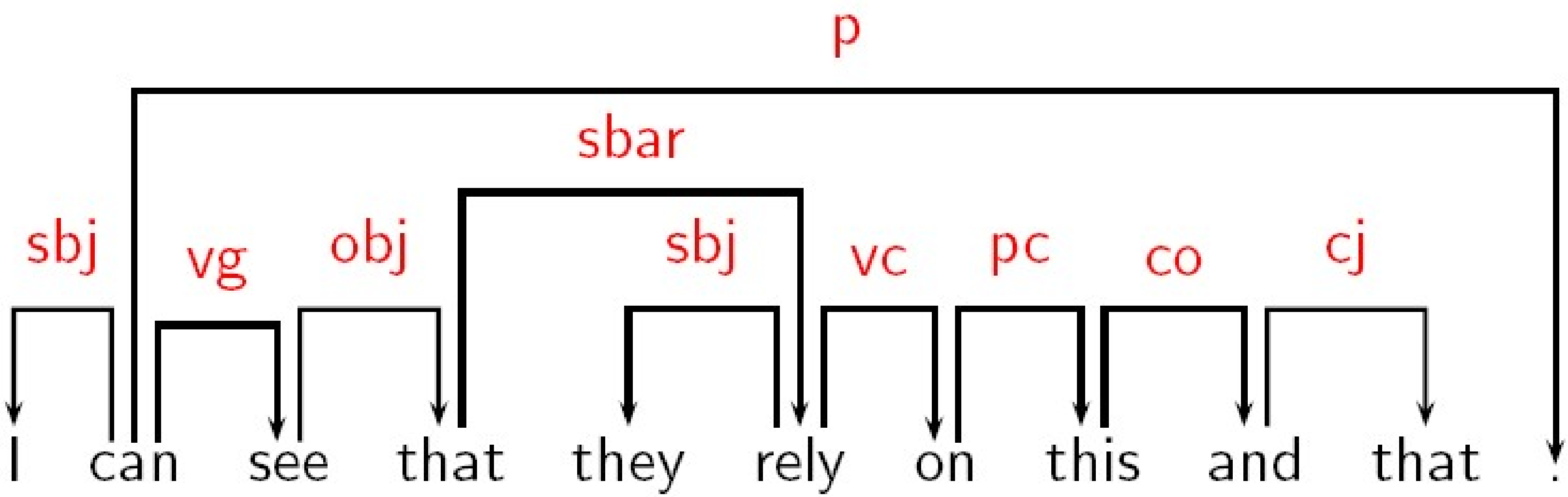
- Prepositional phrases (preposition ↔ nominal)





- Punctuation





Dependency parsing

- **Input**:

  Sentence $W = w_0, w_1, \ldots, w_n$ with w_0=*root*

- **Output**:

  Dependency graph $G = (V, A)$ for W where:

  – $V = \{0, 1, \ldots, n\}$ is the *vertex* set,

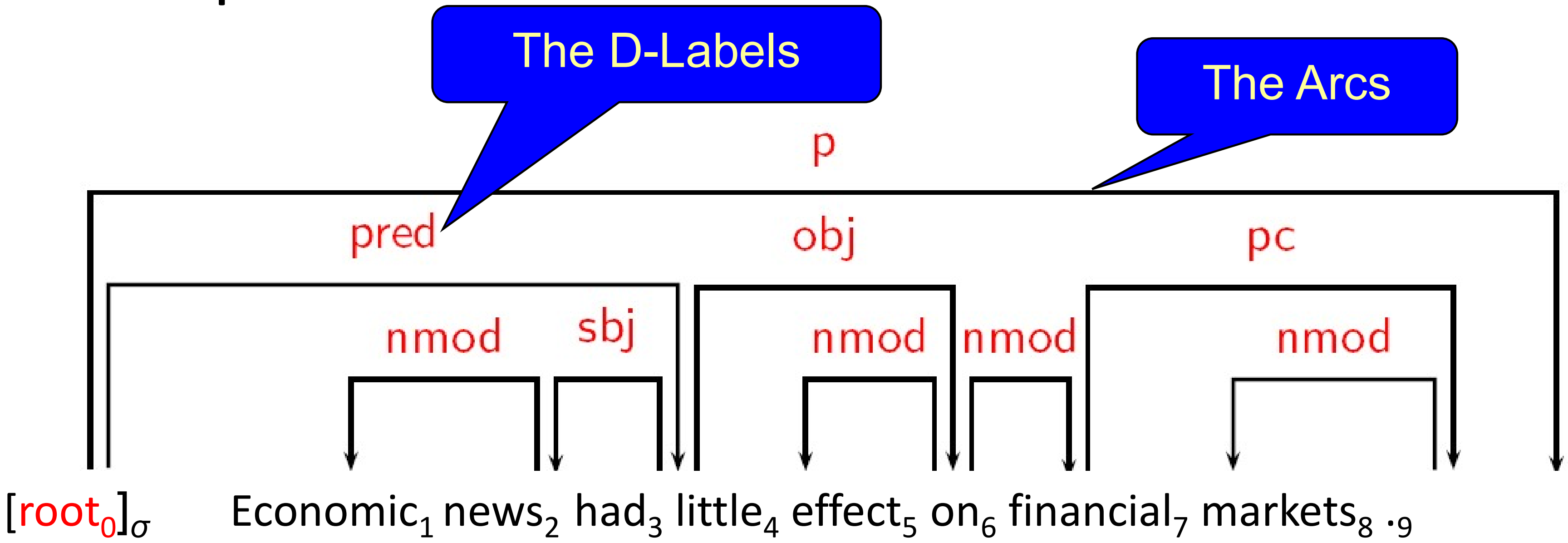
  – A is the *arc* set, i.e., $(i, j, k) \in A$ represents a dependency from w_i to w_j with label $l_k \in L$

Example

- Input:

[$\color{red}{\text{root}_0}$]$_\sigma$ Economic$_1$ news$_2$ had$_3$ little$_4$ effect$_5$ on$_6$ financial$_7$ markets$_8$.$_9$

- Output:



Approaches

- Grammar-based parsing
  - Context-free dependency grammar
  - Constraint dependency grammar
- Data-driven parsing
  - Transition-based models
  - Graph-based models

Context-free dependency grammar

- Basic idea
  - Dependency grammar as lexicalized context-free grammar
  - Standard context-free parsing algorithms (CKY, Earley, etc.)

- Recent developments:
  - Link Grammar [Sleator and Temperley 1991]
  - Earley-style parser with left-corner filtering
  - [Lombardo and Lesmo 1996]
  - Bilexical grammars [Eisner 1996, Eisner 2000]

Constraint Dependency Grammar

- Parsing as constraint satisfaction [Maruyama 1990]:
  - Grammar consists of a set of boolean constraints, i.e. logical formulas that describe well-formed dependency graphs.
  - Constraint propagation removes candidate graphs that contradict constraints (eliminative parsing).
- Recent developments:
  - Weighted Constraint Dependency Grammar [Menzel and Schr ¨ oder 1998, Foth et al. 2004]
  - Probabilistic Constraint Dependency Grammar [Harper and Helzerman 1995, Wang and Harper 2004]
  - Topological/Extensible Dependency Grammar [Duchier and Debusmann 2001, Debusmann et al. 2004]

Transition-based models

- Basic idea:
  - Define a transition system (state machine, MM) for mapping a sentence to its dependency graph.
  - Learning: Induce a model for predicting the next state transition, given the transition history.
  - Parsing: Construct the optimal transition sequence, given the induced model.

- Characteristics:
- Local training of a model for optimal transitions
- Greedy search/inference

Lexicalized Dependency Probabilistic Model

- Collins 1996
- Training: Retrieve the probability of any words with specific dependency through Maximal Likelihood Estimation.
- Parsing (Decoding): Search a dependency tree with the highest product of production probability of all dependent pairs

Probabilistic Dependency Model

- Eisner, 1996
- For each candidate dependency tree, *T*, its production probability is defined as the probability product of the production probability of each node in T

$$Gen(T) = \prod_{x \in T} Gen(x)$$

- Task of Parsing: Looking for the dependency tree with the highest production probability

Shift-Reduce Dependency Parsing

- Transitions:
  - Left-Arc$_k$:
    $$(\sigma|i,j|\beta, A) \Rightarrow (\sigma, j|\beta, A \cup \{(j, i, k)\})$$
  - Right-Arc$_k$:
    $$(\sigma|i,j|\beta, A) \Rightarrow (\sigma, i|\beta, A \cup \{(i, j, k)\})$$
  - Shift:
    $$(\sigma, i|\beta, A) \Rightarrow (\sigma|i, \beta, A)$$
- Preconditions:
  - Left-Arc$_k$:
    $$\neg[i = 0]$$
    $$\neg\exists i' \exists k' [(i', i, k') \in A]$$
  - Right-Arc$_k$:
    $$\neg\exists i' \exists k' [(i', j, k') \in A]$$

Example: Shift-Reduce Parsing - 1

$[\text{root}_0]_\sigma$ $[\text{Economic}_1\ \text{news}_2\ \text{had}_3\ \text{little}_4\ \text{effect}_5\ \text{on}_6\ \text{financial}_7\ \text{markets}_8\ ._9]_\beta$

We have nothing in the configuration.
Just shift to first word

2

[root$_0$ Economic$_1$]$_\sigma$ [news$_2$ had$_3$ little$_4$ effect$_5$ on$_6$ financial$_7$ markets$_8$.$_9$]$_\beta$

Shift

We have one word in the configuration. Now to search the in/out arcs …

3

nmod

$[\text{root}_0]_\sigma$ Economic_1 $[\text{news}_2$ had_3 little_4 effect_5 on_6 financial_7 markets_8 $._9]_\beta$

$\text{Left-Arc}_{\text{nmod}}$

A left-arc is found. Include the dependent in current configuration …

4

nmod

$[\text{root}_0\ \text{Economic}_1\ \text{news}_2]_\sigma\ [\text{had}_3\ \text{little}_4\ \text{effect}_5\ \text{on}_6\ \text{financial}_7\ \text{markets}_8\ ._9]_\beta$

Shift

Current configuration is self-contained. Now shift is required to include the next word …

5

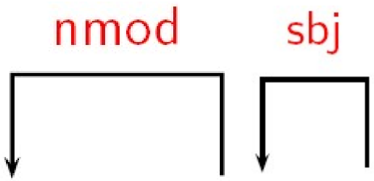
nmod sbj

[root$_0$]$_\sigma$ Economic$_1$ news$_2$ [had$_3$ little$_4$ effect$_5$ on$_6$ financial$_7$ markets$_8$.$_9$]$_\beta$

Left-Arc$_{sbj}$

But the next word holds a *SBJ* left arc to word *news*.
The configuration is reduced to empty again.

6



[root$_0$ Economic$_1$ news$_2$ had$_3$]$_\sigma$ [little$_4$ effect$_5$ on$_6$ financial$_7$ markets$_8$.$_9$]$_\beta$

Shift

The *SBJ* left arc creates a new configuration. We now shift to the next word.

7

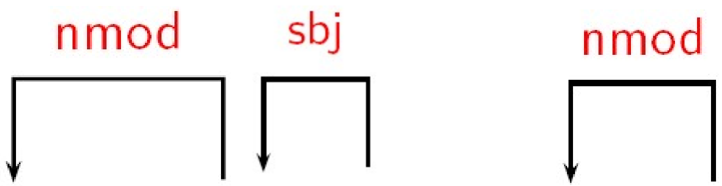
nmod sbj

$[\text{root}_0\ \text{Economic}_1\ \text{news}_2\ \text{had}_3\ \text{little}_4]_\sigma\ \text{effect}_5\ \text{on}_6\ \text{financial}_7\ \text{markets}_8\ ._9]_\beta$

Shift

The next word holds no valid arc with current configuration. Then we continue to shift.

8

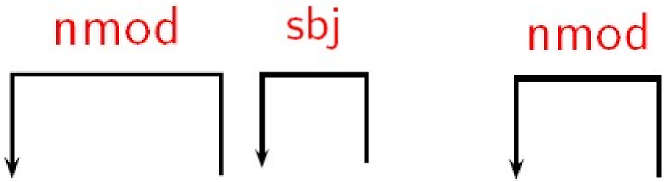


[root$_0$ Economic$_1$ news$_2$ had$_3$]$_\sigma$ little$_4$ [effect$_5$ on$_6$ financial$_7$ markets$_8$.$_9$]$_\beta$

Left-Arc$_{nmod}$

Now a NMOD left arc is found. The current configuration is reduced to previous one.

9

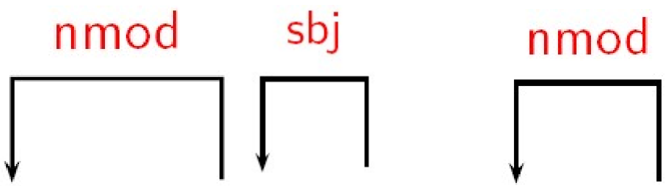


[root$_0$ Economic$_1$ news$_2$ had$_3$ little$_4$ effect$_5$]$_\sigma$ [on$_6$ financial$_7$ markets$_8$.$_9$]$_\beta$

Shift

The valid NMOD left arc is included in the current configuration. Shift to next word.

10



[root$_0$ Economic$_1$ news$_2$ had$_3$ little$_4$ effect$_5$ on$_6$]$_\sigma$ [financial$_7$ markets$_8$.$_9$]$_\beta$

Shift

The word on should hold an arc to its neighbors. It is not word effect. So we continue to shift.

11

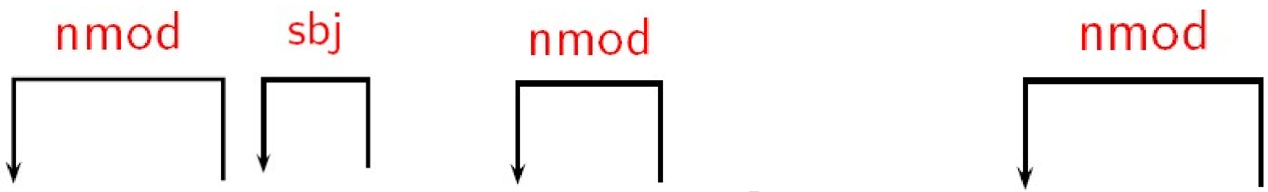
nmod sbj nmod

[root$_0$ Economic$_1$ news$_2$ had$_3$ little$_4$ effect$_5$ on$_6$ financial$_7$]$_\sigma$ [markets$_8$.$_9$]$_\beta$

Shift

We include the new word in the configuration. However, the arc from *on* to *financial* is not valid. Shift but reduce to previous configuration.

12

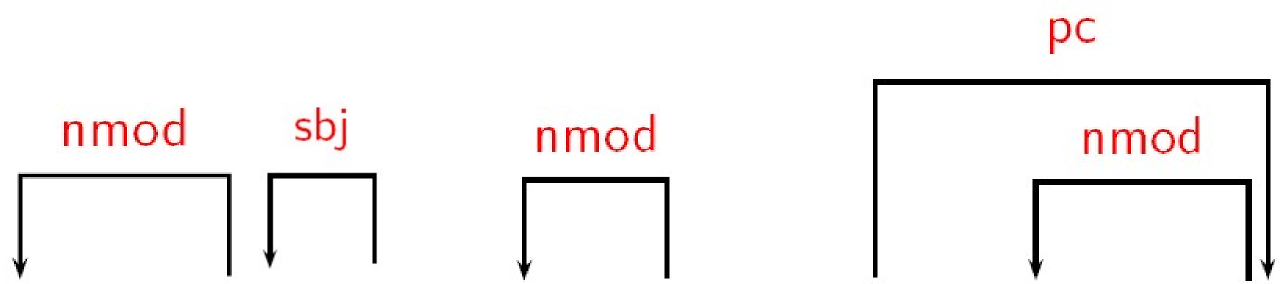


[root$_0$ Economic$_1$ news$_2$ had$_3$ little$_4$ effect$_5$ on$_6$]$_\sigma$ financial$_7$ [markets$_8$.$_9$]$_\beta$

New NMOD left arc is found. The remaining word is punctuation. Stop shifting. Search right arc.

Left-Arc$_{nmod}$

13

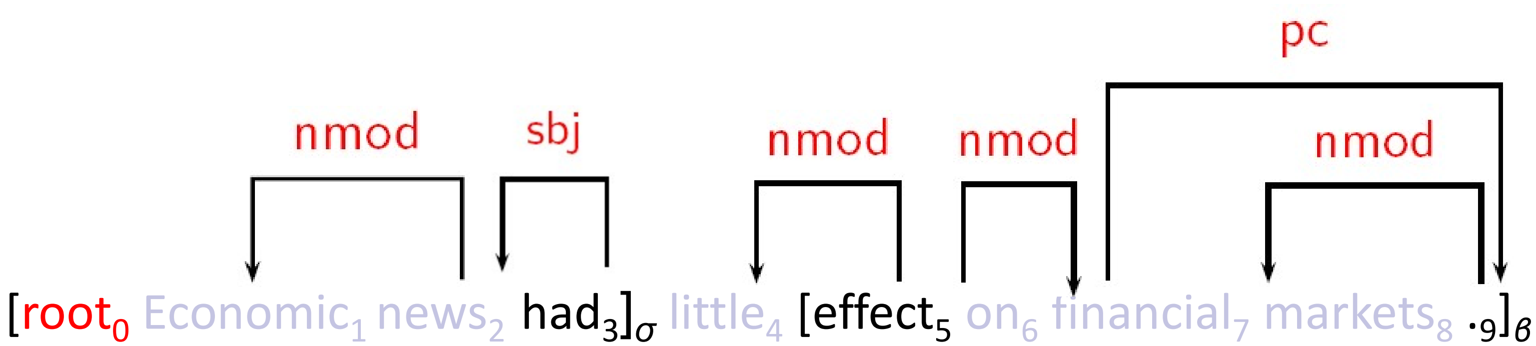


[root_0 Economic_1 news_2 had_3 little_4 $\text{effect}_5]_\sigma$ [on_6 financial_7 markets_8 $._9]_\beta$

Right-Arc_{pc}

A PC right arc is found from *on* to *markets*. Reduce configuration. Continue searching right arc.

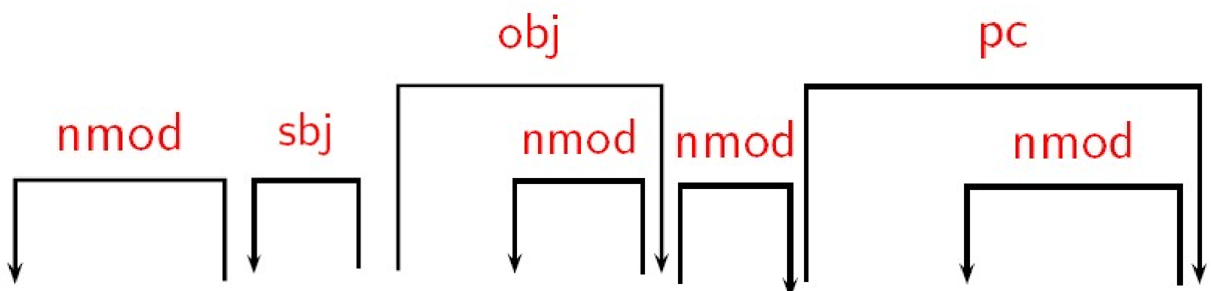
14



Right-Arc$_{nmod}$

A PC right arc is found from *effect* to *on*. Reduce configuration. Continue searching right arc.

15

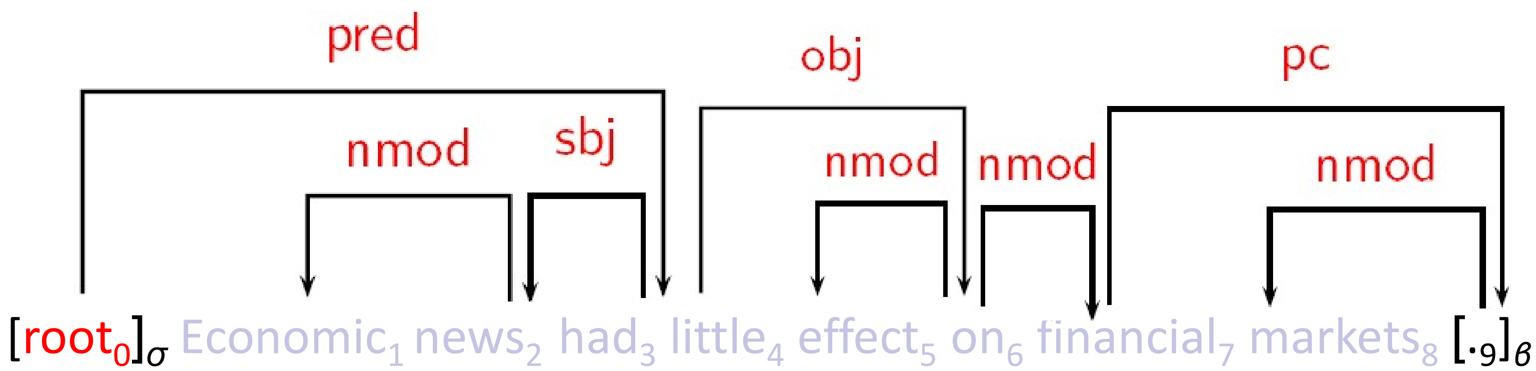


[root_0]$_\sigma$ Economic_1 news_2 [had_3 little_4 effect_5 on_6 financial_7 markets_8 $._9$]$_\beta$

A PC right arc is found from *had* to *effect*. Reduce configuration. Continue searching right arc.

Right-Arc$_{obj}$

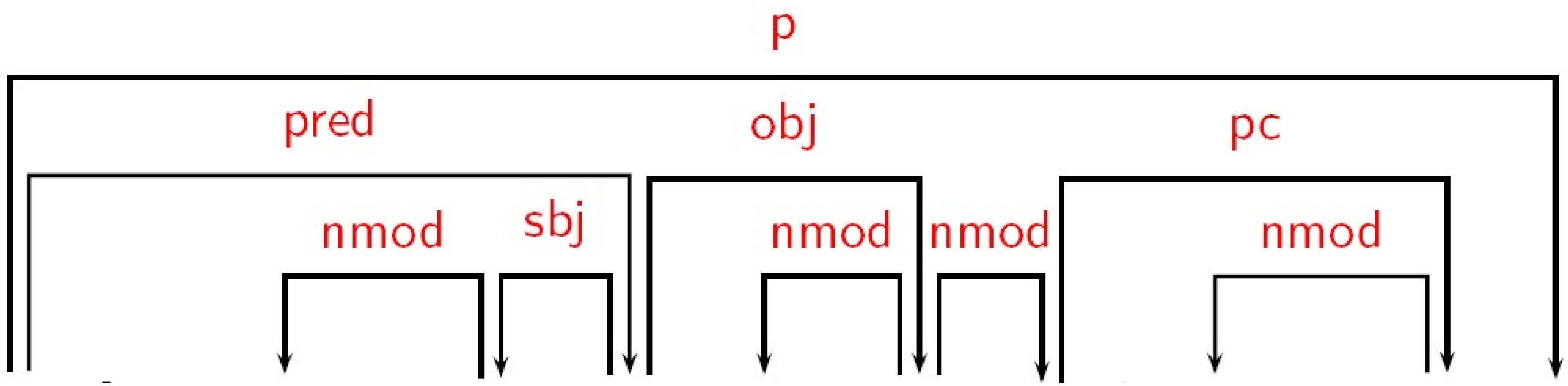
16



A PC right arc is found from *root* to *had*. Reduce configuration. Continue searching right arc.

Right-Arc$_{pred}$

17

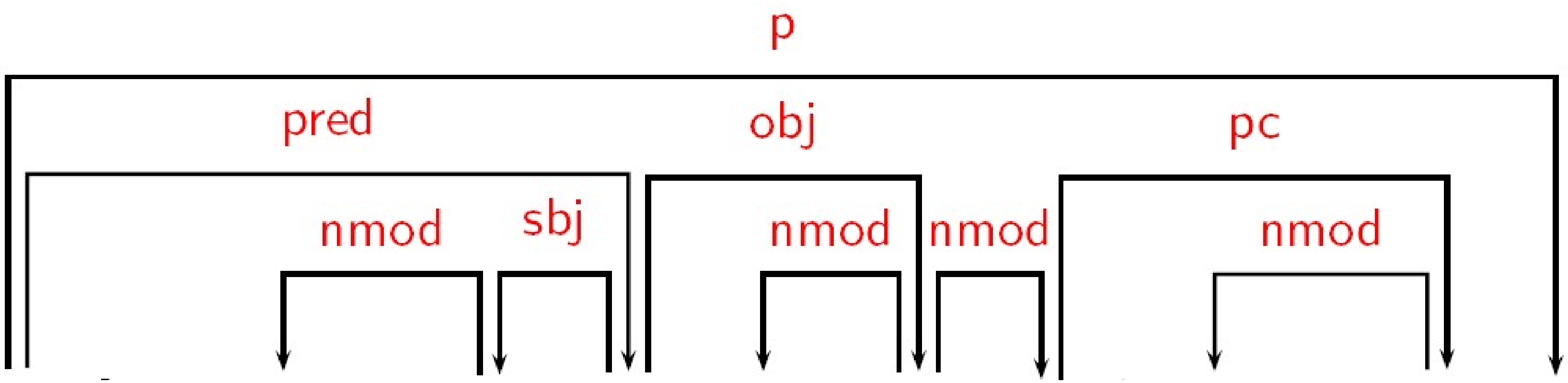


$[]_{\sigma}$ $[\text{root}_0]$ $_{\beta}$ Economic$_1$ news$_2$ had$_3$ little$_4$ effect$_5$ on$_6$ financial$_7$ markets$_8$.$_9$

Right-Arc$_p$

A PC right arc is found from *root* to *punctuation*. Reduce configuration. Continue searching right arc.

18

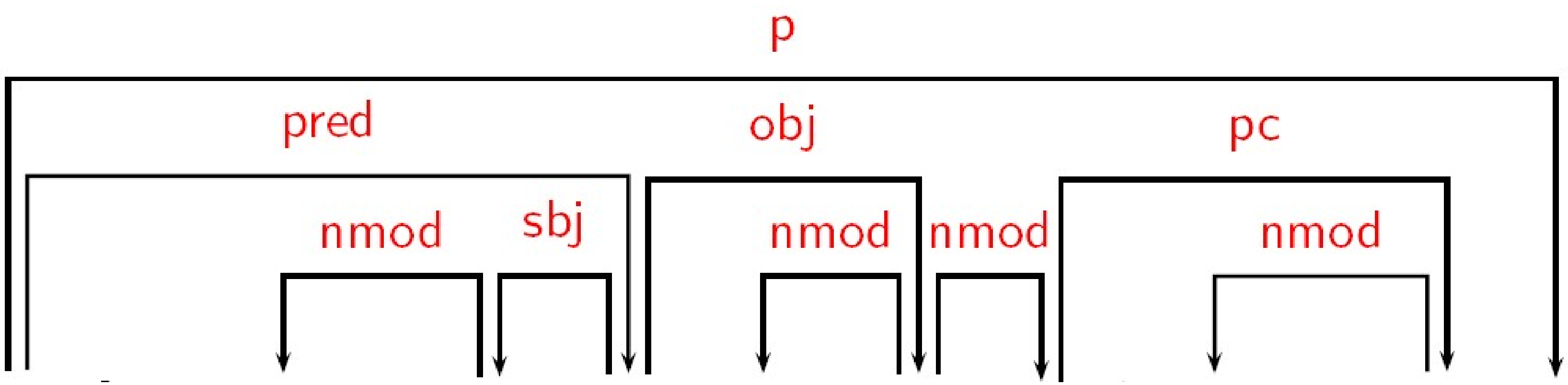


$[\text{root}_0]_\sigma$ Economic_1 news_2 had_3 little_4 effect_5 on_6 financial_7 markets_8 $._9$

Shift

No right arc is found. Remove the empty configuration.
Parsing over.

19



[root_0]$_\sigma$ Economic_1 news_2 had_3 little_4 effect_5 on_6 financial_7 markets_8 $._9$

Shift

Graph-based models

- Basic idea:
  - Define a space of candidate dependency graphs for a sentence.
  - Learning: Induce a model for *scoring* an entire dependency graph for a sentence.
  - Parsing: Find the highest-scoring dependency graph, given the induced model.

- Characteristics:
  - Global training of a model for optimal dependency graphs
  - Exhaustive search/inference

Treebanks

- English Penn Treebank: Standard corpus for testing syntactic parsing consists of 1.2 M words of text from the Wall Street Journal (WSJ).
- Typical to train on about 40,000 parsed sentences and test on an additional standard disjoint test set of 2,416 sentences.
- Chinese Penn Treebank: 100K words from the Xinhua news service.
- Other corpora existing in many languages, see the Wikipedia article “Treebank”

Parsing Resources

- Michael Collins 's Parser : English
  - http://people.csail.mit.edu/mcollins/code.html
- Dan Bikel 's Parser: English / Chinese / Arabic
  - http://www.cis.upenn.edu/~dbikel/software.html#stat-parser
- Stanford Parser: English / Chinese / German
  - http://www-nlp.stanford.edu/software/lex-parser.shtml
  - English / Chinese / German
- David Chiang's Parser: English / Chinese
  - http://www.isi.edu/~chiang/
- HIT IR Dependency Parser
  - http://ir.hit.edu.cn