



ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

Probabilistic parsing

Lê Thanh Hương

School of Information and Communication
Technology

Email: huonglt@soict.hust.edu.vn

Motivation: how to choose a parse structure?

- Choice among parses, e.g.

I saw a man with a telescope.

- As #rules increases, possibility of ambiguity goes *up*
- Large NYU grammars: Apple pie parser: 20,000-30,000 CF rules for English
- Choice between two rules: V DT NN PP

(1) $VP \rightarrow V NP PP$

$NP \rightarrow DT NN$

(2) $VP \rightarrow V NP$

$NP \rightarrow DT NN PP$

Word associations (bigrams pr)

Example:

Eat ice-cream (high freq)

Eat John (low, except on Survivor)

Some disadvantages:

- P(John decided to bake a) has a high probability

- Consider:

$$P(w_3) = P(w_3|w_2w_1) = P(w_3|w_2)P(w_2|w_1)P(w_1)$$

The assumption is too strong, e.g., the subject of a sentence can 'select' the object:

Clinton admires honesty

- use syntactic structure to stop selection from propagating
- Consider Fred watered his mother's small garden. What is pr contributed from *garden*?
 - $\Pr(\text{garden} | \text{mother's small})$ is not high \Rightarrow trigram model would *not* do well
 - $\Pr(\text{garden} | \text{X is head of object NP to water})$ is higher
- use bigram + syntactic relation

Syntactic associations (Pr cfg)

- V takes a certain kind of argument
⇒ Verb-with-obj, verb-without-obj
- The correspondance between sbj and obj:

John admires honesty

Honesty admires John ???

Disadvantages:

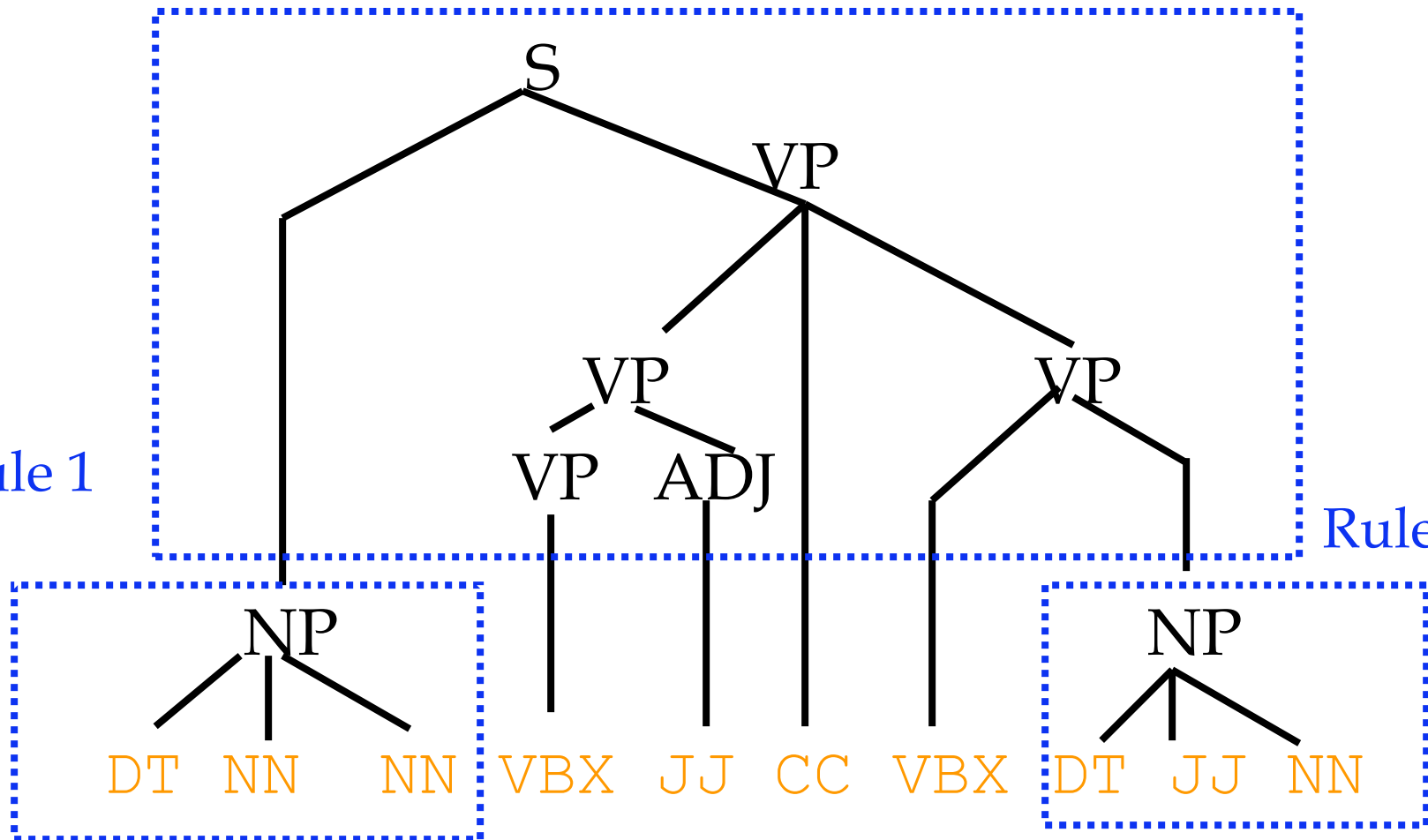
- Grammar size increases
 - 1 year of Wall Street Journal (WSJ corpus): 47,219 sentences, avg. length 23 words; bracketed by hand: only 4.7% or 2,232 have exactly *same* structure as any other in the corpus
- Can't do it all by table lookup. Instead, build up set of particular little pieces

Example

Rule 3

Rule 1

Rule 2



This apple pie looks good and is a real treat

Rules

1. $NP \rightarrow DT\ NN\ NN$
2. $NP \rightarrow DT\ JJ\ NN$
3. $S \rightarrow NP\ VBX\ JJ\ CC\ VBX\ NP$
 - Collapse (NNS, NN) to NX; (NNP, NNPs)=NPX;
(VBP, VBZ, VBD)=VBX;
 - Choose rules by their frequencies

Calculating frequencies

$$\Pr(X \rightarrow Y) = \frac{\text{Frequency of } X \rightarrow Y}{\text{Frequency of } X} = \frac{1470}{9711} = 0.1532$$

Diagram illustrating the calculation of the probability $\Pr(X \rightarrow Y)$ using frequency counts:

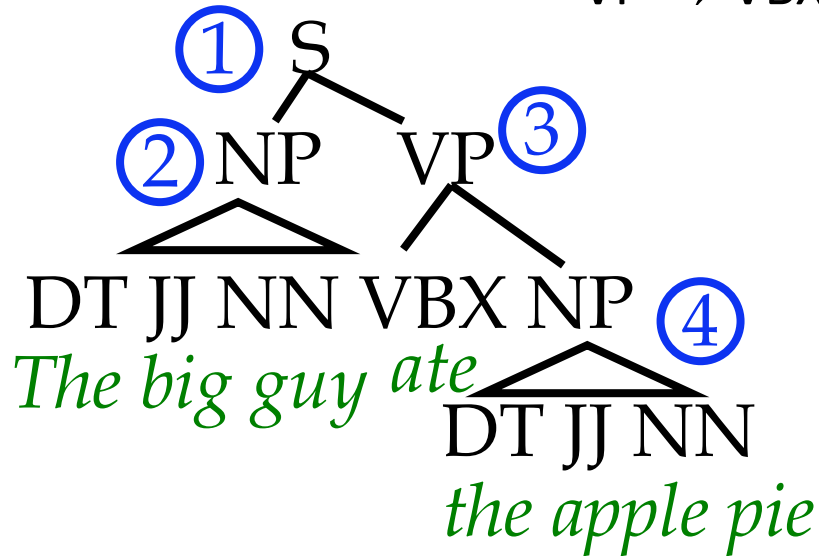
- The numerator represents the frequency of the transition $X \rightarrow Y$, which is 1470.
- The denominator represents the frequency of the source state X , which is 9711.

Pr calculation

$S \rightarrow NP VP$; 0.35

$NP \rightarrow DT JJ NN$; 0.1532

$VP \rightarrow VBX NP$; 0.302



Rule applied

Pr chain

1 $S \rightarrow NP VP$

0.35

2 $NP \rightarrow DT JJ NN$

0.1532 x 0.35 = 0.0536

3 $VP \rightarrow VBX NP$

0.302 x 0.0536 = 0.0162

4 $NP \rightarrow DT JJ NN$

0.1532 x 0.0162 = 0.0025

Pr = 0.0025

PCFGs

- A PCFG G consists of the usual parts of a CFG
- A set of terminals, $\{w^k\}$, $k = 1, \dots, V$
- A set of nonterminals, $\{N^i\}$, $i = 1, \dots, n$
- A designated start symbol, N^1
- A set of rules, $\{N^i \rightarrow \zeta^j\}$, (where ζ^j is a sequence of terminals and nonterminals)

and

- A corresponding set of **probabilities** on rules such that:

$$\forall i \sum_j P(N^i \rightarrow \zeta^j) = 1$$

- Probability of a derivation (i.e. parse) tree:

$$P(T) = \prod_{i=1..n} p(r(i))$$

Assumptions

- **Place Invariance**: The probability of a subtree does not depend on *where in the string* the words it dominates are.

$$\forall k, P(N_{jk}(k+c) \rightarrow \zeta) \text{ is the same}$$

- **Context Free**: The probability of a subtree does not depend on words *not dominated* by the subtree.

$$P(N_{jkl} \rightarrow \zeta \mid \text{anything outside } k \text{ through } l) = P(N_{jkl} \rightarrow \zeta)$$

- **Ancestor Free**: The probability of a subtree does not depend on nodes in the derivation *outside* the subtree

$$P(N_{jkl} \rightarrow \zeta \mid \text{anything ancestor nodes outside } N_{jkl}) = P(N_{jkl} \rightarrow \zeta)$$

Parsing algorithms

- CKY
- Beam search
- Agenda/chart-based search
- ...

CKY with probabilities

- Data structure:
 - Dynamic programming array $\pi[i,j,a]$ holds the **maximum probability** for a constituent with nonterminal a spanning words $i...j$.
 - **Backptrs** store links to constituents in tree
- Output: Maximum probability of parse

Compute Pr by induction

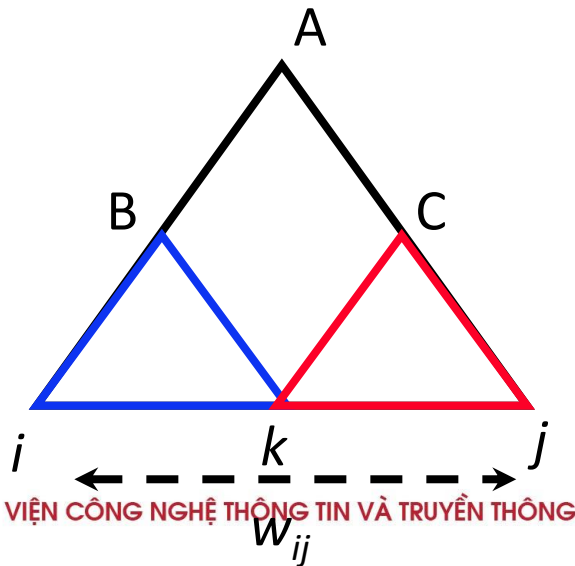
- Base case: input is a single word.

$$\text{Pr}(\text{tree}) = \text{pr}(A \rightarrow w_i)$$

- Recursive case. Input is a string of words.

$$A \Rightarrow^* w_{ij} \text{ if } \exists k: A \rightarrow BC, B \Rightarrow^* w_{ik}, C \Rightarrow^* w_{kj}, i \leq k \leq j.$$

$$p[i,j] = \max(p(A \rightarrow BC) \times p[i,k] \times p[k,j]).$$



function CYK(*words, grammar*) **returns** *best_parse*

Create and clear $p[num_words, num_words, num_nonterminals]$

base case

for $i = 1$ **to** num_words

for $A = 1$ **to** $num_nonterminals$

if $A \rightarrow w_i$ is in grammar **then**

$\pi[i, i, A] = P(A \rightarrow w_i)$

recursive case

for $j = 2$ **to** num_words

for $i = 1$ **to** $num_words - j + 1$

for $k = 1$ **to** $j - 1$

for $A = 1$ **to** $num_nonterminals$

for $B = 1$ **to** $num_nonterminals$

for $C = 1$ **to** $num_nonterminals$

$prob = \pi[i, k, B] \times p[i+k, j-k, C] \times P(A \rightarrow BC)$

if ($prob > \pi[i, j, A]$) **then**

$\pi[i, j, A] = prob$

$B[i, j, A] = \{k, A, B\}$

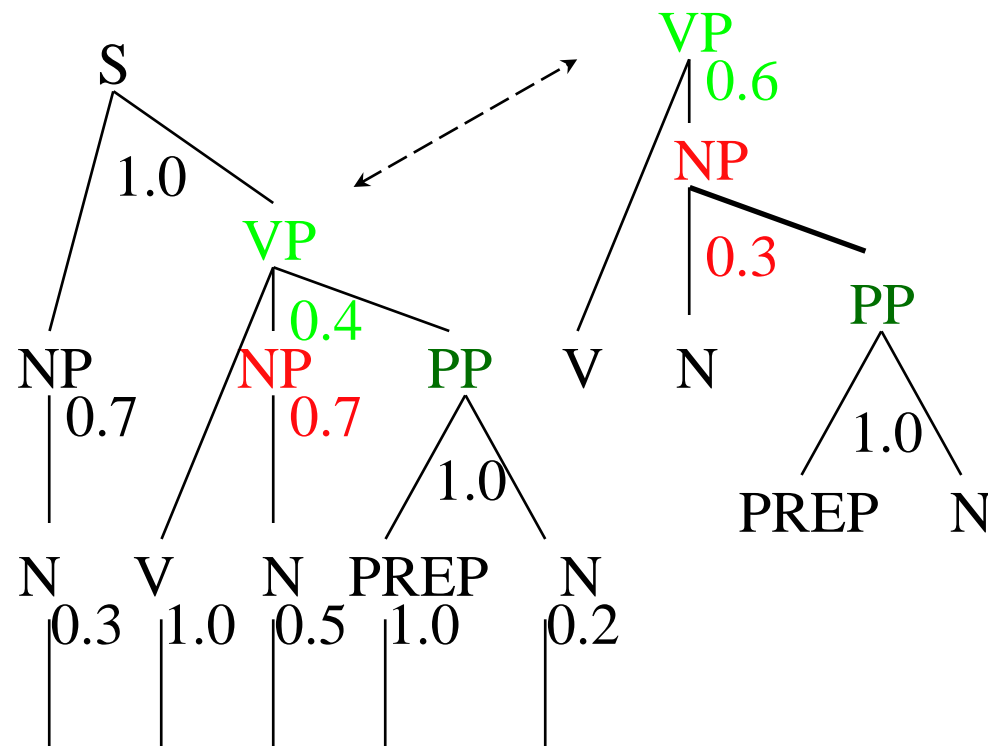
Calculation of Viterbi probabilities (CKY algorithm)

$S \rightarrow NP VP$	1.0	$NP \rightarrow NP PP$	0.4
$PP \rightarrow P NP$	1.0	$NP \rightarrow \textit{astronomers}$	0.1
$VP \rightarrow V NP$	0.7	$NP \rightarrow \textit{ears}$	0.18
$VP \rightarrow VP PP$	0.3	$NP \rightarrow \textit{saw}$	0.04
$P \rightarrow \textit{with}$	1.0	$NP \rightarrow \textit{stars}$	0.18
$V \rightarrow \textit{saw}$	1.0	$NP \rightarrow \textit{telescopes}$	0.1

	1	2	3	4	5
1	$\delta_{NP} = 0.1$		$\delta_S = 0.0126$		$\delta_S = 0.0009072$
2		$\delta_{NP} = 0.04$ $\delta_V = 1.0$	$\delta_{VP} = 0.126$		$\delta_{VP} = 0.009072$
3			$\delta_{NP} = 0.18$		$\delta_{NP} = 0.01296$
4				$\delta_P = 1.0$	$\delta_{PP} = 0.18$
5					$\delta_{NP} = 0.18$
	<i>astronomers</i>	<i>saw</i>	<i>stars</i>	<i>with</i>	<i>ears</i>

Pr calculation

1. $S \rightarrow NP VP$ 1.0
2. $VP \rightarrow V NP PP$ 0.4
3. $VP \rightarrow V NP$ 0.6
4. $NP \rightarrow N$ 0.7
5. $NP \rightarrow N PP$ 0.3
6. $PP \rightarrow PREP N$ 1.0
7. $N \rightarrow a_dog$ 0.3
8. $N \rightarrow a_cat$ 0.5
9. $N \rightarrow a_telescop$ 0.2
10. $V \rightarrow saw$ 1.0
11. $PREP \rightarrow with$ 1.0



a_dog saw a_cat with a_telescope

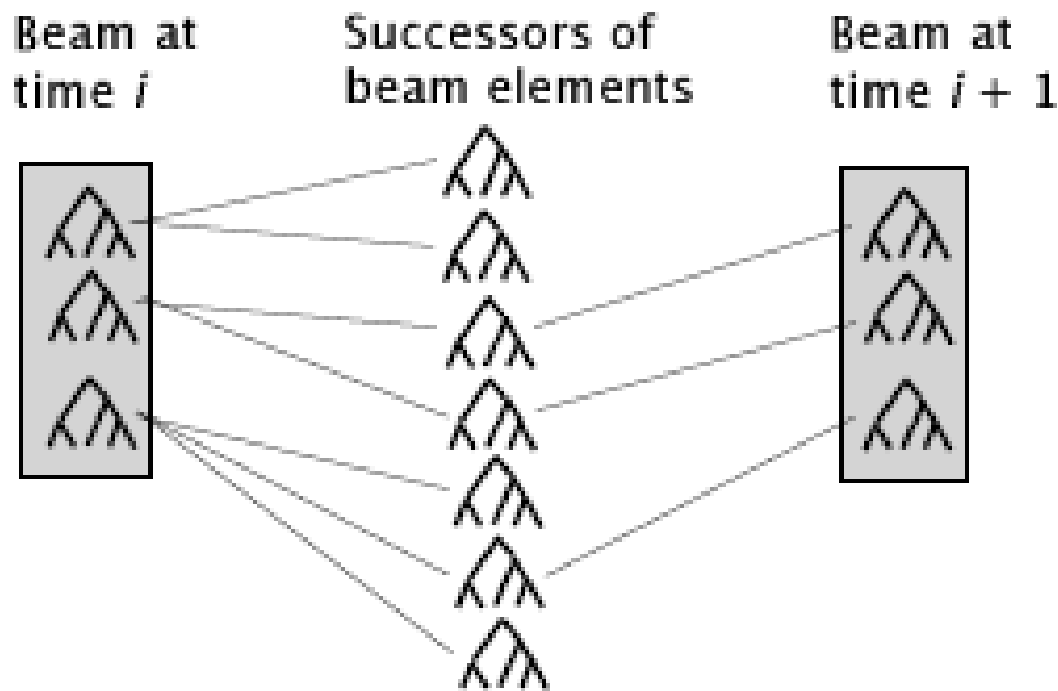
$$P_l = 1' \cdot 7' \cdot 4' \cdot 3' \cdot 7' \cdot 1' \cdot 5' \cdot 1' \cdot 1' \cdot 2 = .00588$$

$$P_r = 1' \cdot 7' \cdot 6' \cdot 3' \cdot 3' \cdot 1' \cdot 5' \cdot 1' \cdot 1' \cdot 2 = .00378$$

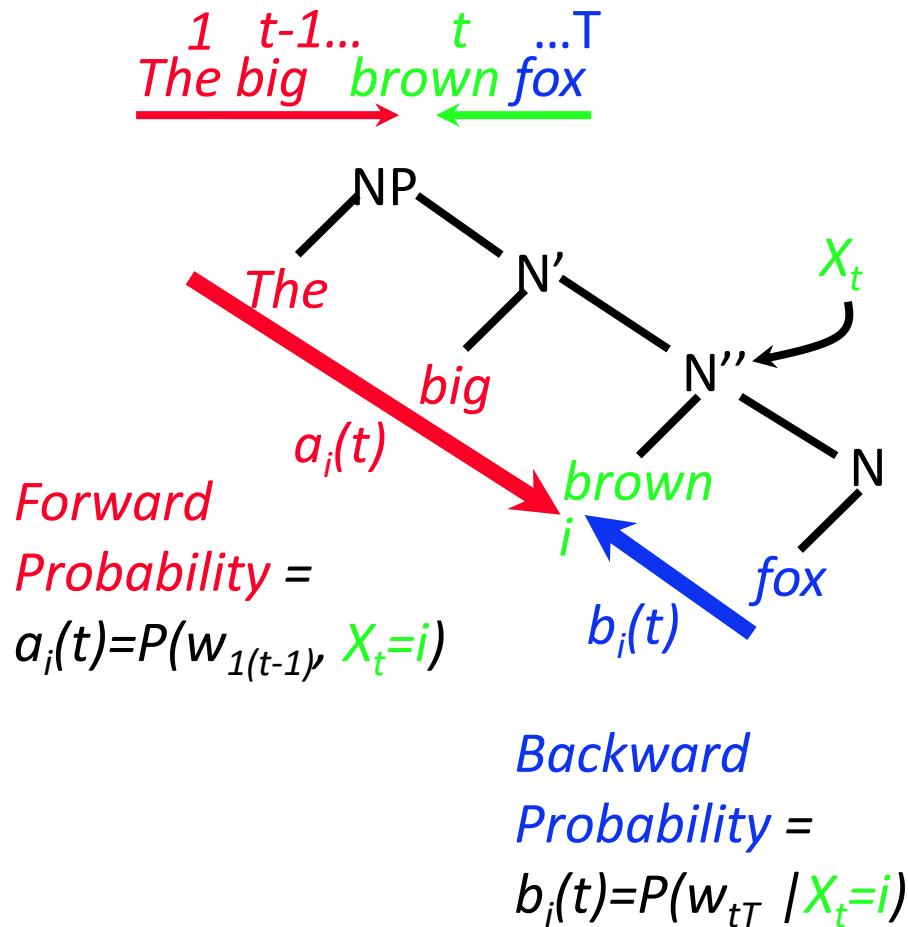
➤ P_l is chosen

Beam search

- State space search
- States are partial parses with an associated probability
 - Keep only the top scoring elements at each stage of the beam search
- All parses of a sentence have the same number N steps

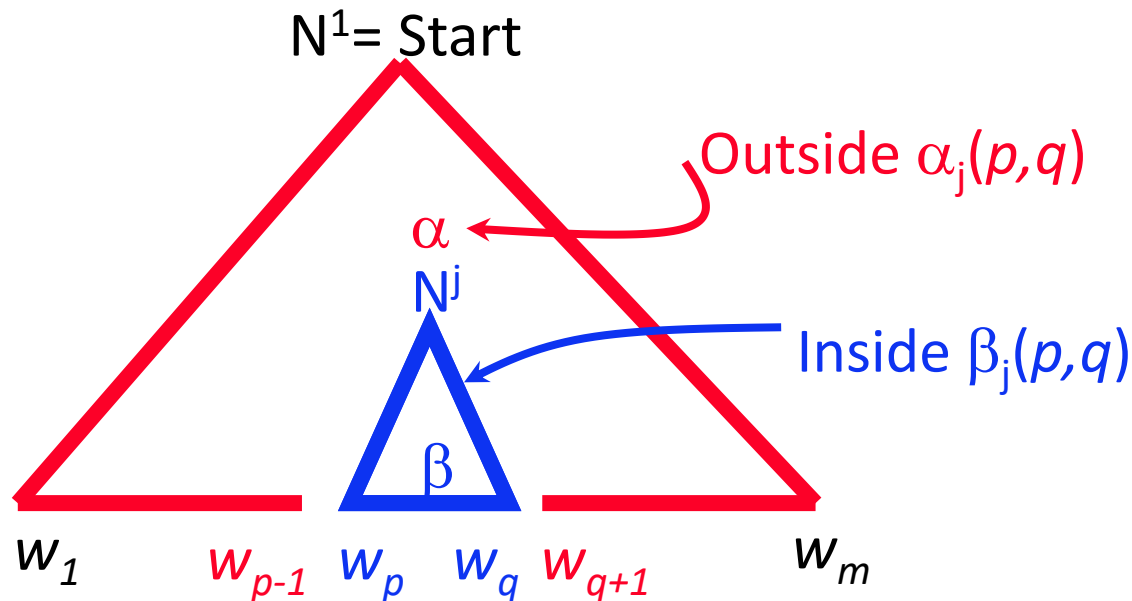


Forward and Backward Pr



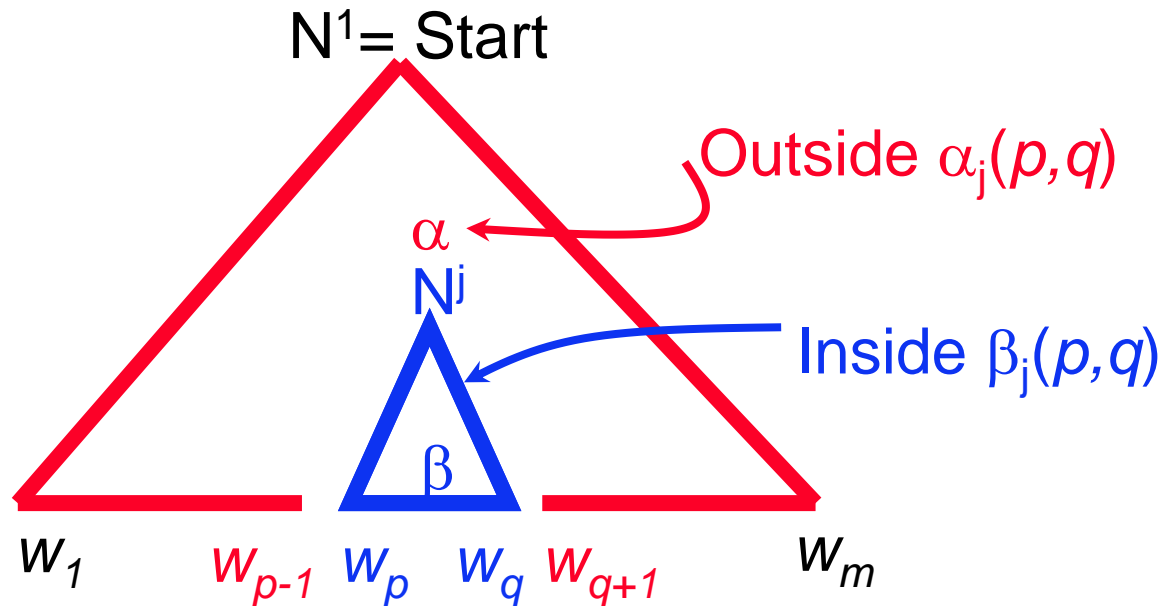
- *Forward* = probability of everything above & including a certain node
- *Backward* = probability of everything below the node, *given* the node

Inside and outside probabilities



- N_{pq} = Nonterminal N^j spans positions p through q in string (phrase N^j dominates words w_{pq})
- α_j = outside probabilities
- β_j = inside probabilities
- N^j dominates words $w_p \dots w_q$, iff $N^j \Rightarrow^* w_p \dots w_q$

Inside and outside probabilities



$$\alpha_j(p, q) = P(w_{1(p-1)}, N_{pq}^j, w_{(q+1)m} | G)$$

$$\beta_j(p, q) = P(w_{pq} | N_{pq}^j, G)$$

$$\begin{aligned} \alpha_j(p, q) \beta_j(p, q) &= P(N^1 \Rightarrow^* w_{1m}, N_j^i \Rightarrow^* w_{pq} | G) \\ &= P(N^1 \Rightarrow^* w_{1m} | G) \bullet P(N_j^i \Rightarrow^* w_{pq} | N^1 \Rightarrow^* w_{1m}, G) \end{aligned}$$

Compute Pr of a string

- We use the **Inside Algorithm**, a dynamic programming algorithm based on the inside probabilities:

$$P(w_{1m}|G) = P(N^1 \Rightarrow^* w_{1m}|G) = P(w_{1m}|N_{1m}^1, G) = \beta_1(1,m)$$

- Base Case:

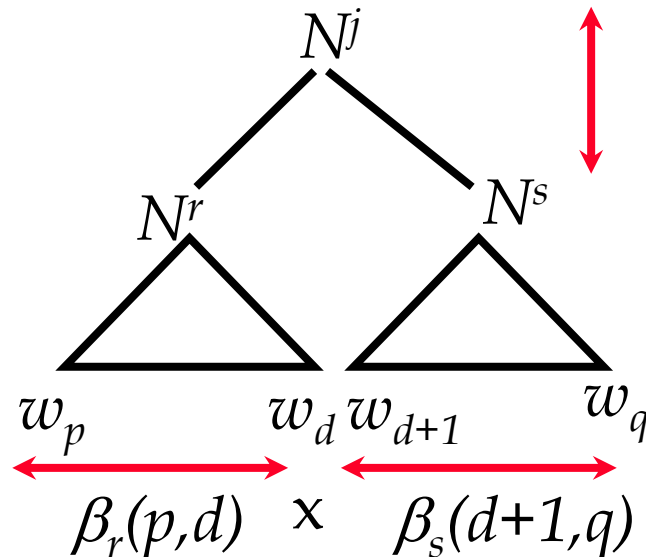
$$\beta_j(k,k) = P(w_k|N_{kk}^j, G) = P(N^j \rightarrow w_k|G)$$

- Induction:

$$\beta_j(p,q) = \sum_{r,s} \sum_{d \in (p,q-1)} P(N^j \rightarrow N^r N^s) \beta_r(p,d) \beta_s(d+1,q)$$

Induction

Find $\beta_j(p,q)$ for $p < q$ – calculate over all ‘splits’ j –
do this ‘bottom up’

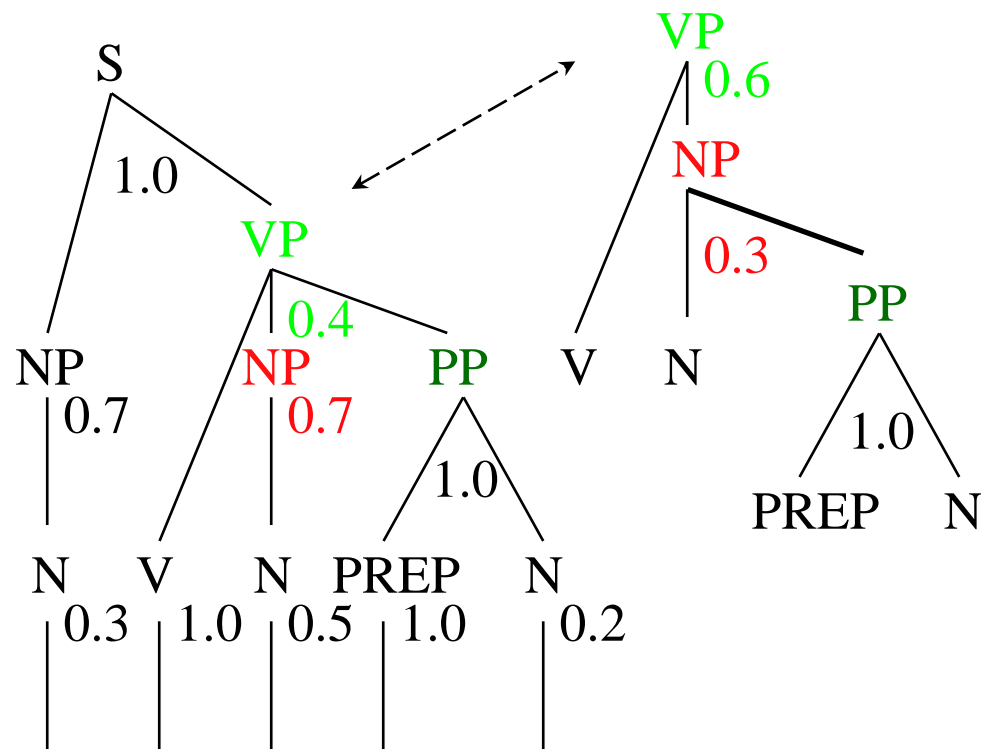


$$P(N^j \rightarrow N^r N^s)$$

-multiply these 3
factors; sum over
all j, r, s .

Example PCFG

1. $S \rightarrow NP VP$ 1.0
2. $VP \rightarrow V NP PP$ 0.4
3. $VP \rightarrow V NP$ 0.6
4. $NP \rightarrow N$ 0.7
5. $NP \rightarrow N PP$ 0.3
6. $PP \rightarrow PREP N$ 1.0
7. $N \rightarrow a_dog$ 0.3
8. $N \rightarrow a_cat$ 0.5
9. $N \rightarrow a_telescope$ 0.2
10. $V \rightarrow saw$ 1.0
11. $PREP \rightarrow with$ 1.0

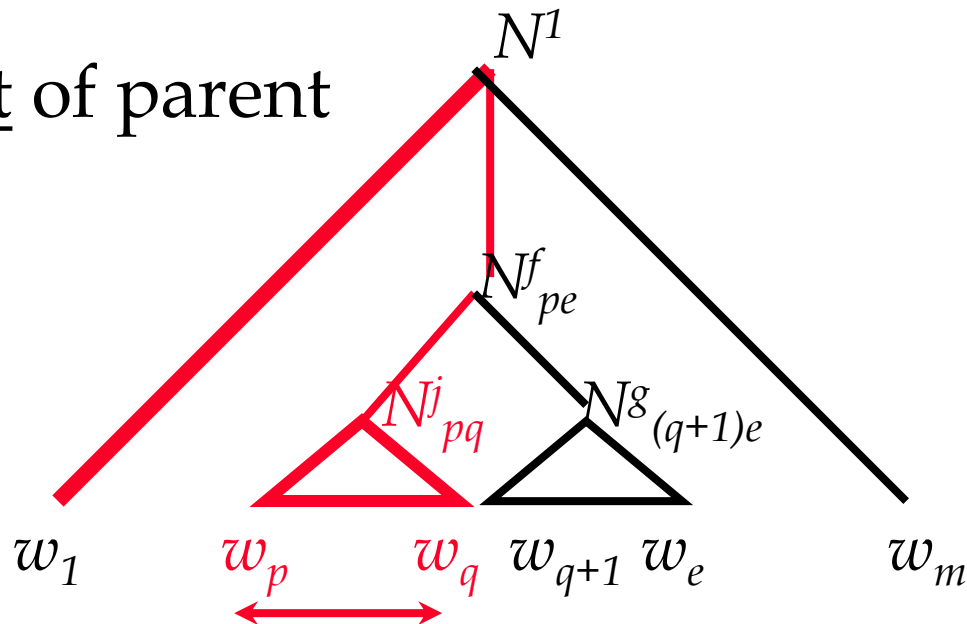


$P(a_dog \text{ saw } a_cat \text{ with } a_telescope) =$

$$1' \cdot 7' \cdot 4' \cdot 3' \cdot 7' \cdot 1' \cdot 5' \cdot 1' \cdot 1' \cdot 2 + \dots \cdot 6 \dots \cdot 3 \dots = .00588 + .00378 = .00966$$

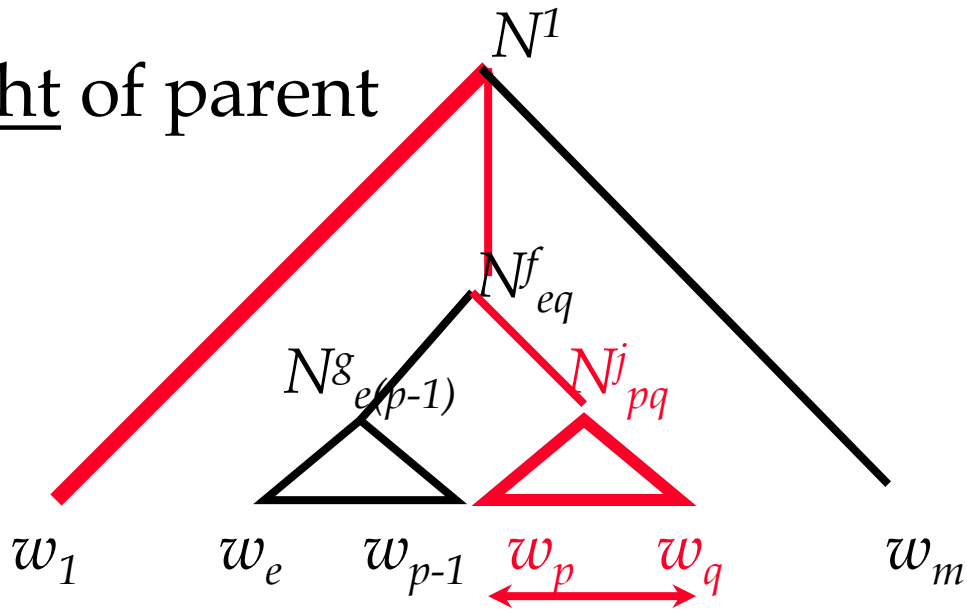
Compute outside Pr, $\alpha_j(p,q)$

Left of parent



Compute outside Pr, $\alpha_j(p,q)$

Right of parent

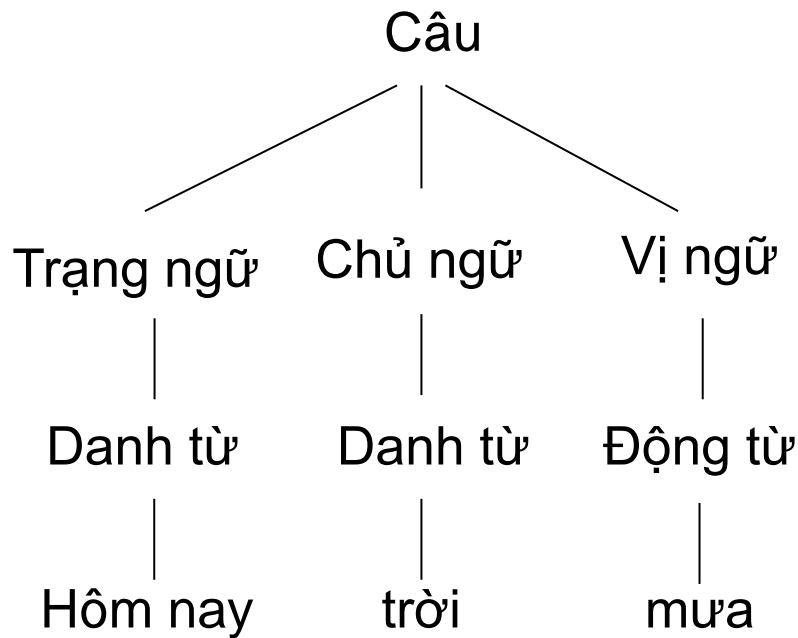


Sum over both; restrict $g \neq j$
to avoid double counting $N^j N^j$

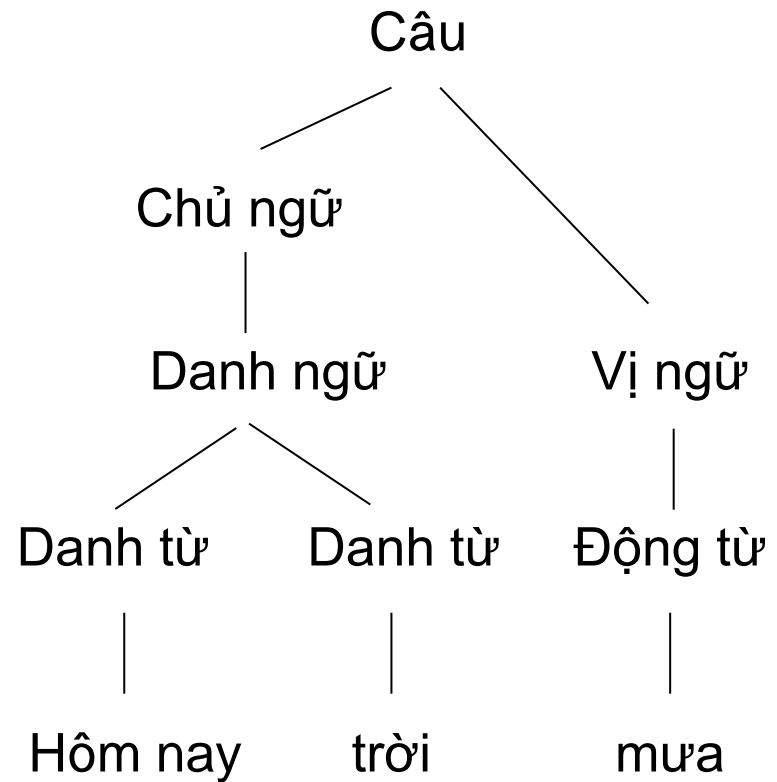
Ambiguity in Vietnamese syntactic parsing

- 2 types of ambiguities:
 - A sentence can be understood by different ways, resulting in different syntactic trees
 - Eg., “*Tôi nhìn thấy anh Hải ở tầng hai*”
 - A sentence with only one meaning but the syntactic parser generates more than one syntactic tree, in which only one tree is correct.
 - Eg., “*Hôm nay trời mưa*”

Ambiguity in Vietnamese syntactic parsing



(a)



(b)

Ambiguity in Vietnamese syntactic parsing

Solution:

Solution 1: Using more detailed syntactic labels Phân loại chi tiết hơn các nhãn từ loại/ngữ loại:

Instead of the rule

$\langle \text{Danh ngữ} \rangle \rightarrow \langle \text{Danh từ} \rangle \langle \text{Danh từ} \rangle$

Using a rule:

$\langle \text{Danh ngữ} \rangle \rightarrow \langle \text{Danh từ loại A} \rangle \langle \text{Danh từ loại B} \rangle.$

Disadvantages:

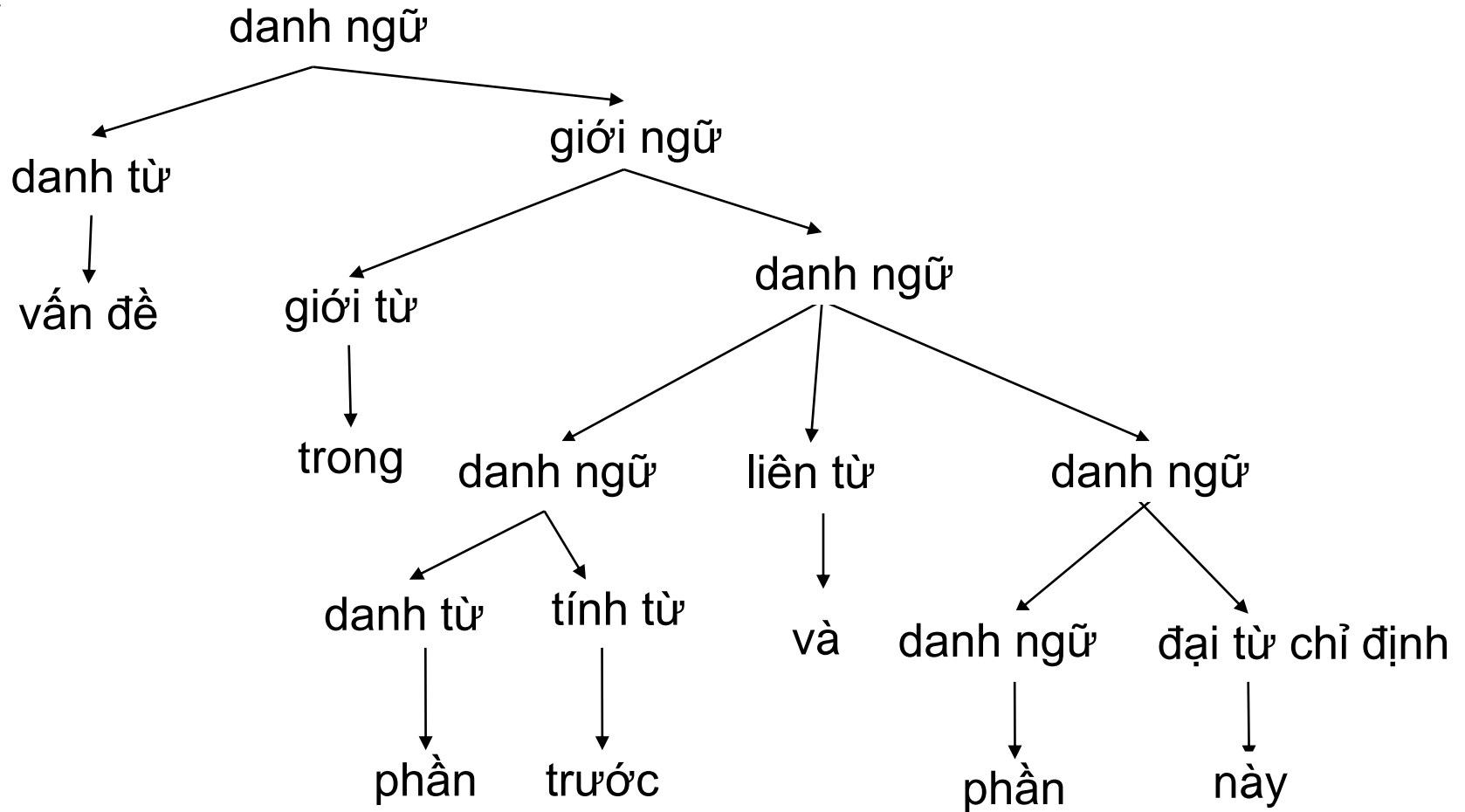
- The set of syntactic labels is not unique.
- The size of the rule set is increased remarkable
- The rule set needs to be created manually \rightarrow difficult to be done

Ambiguity in Vietnamese syntactic parsing

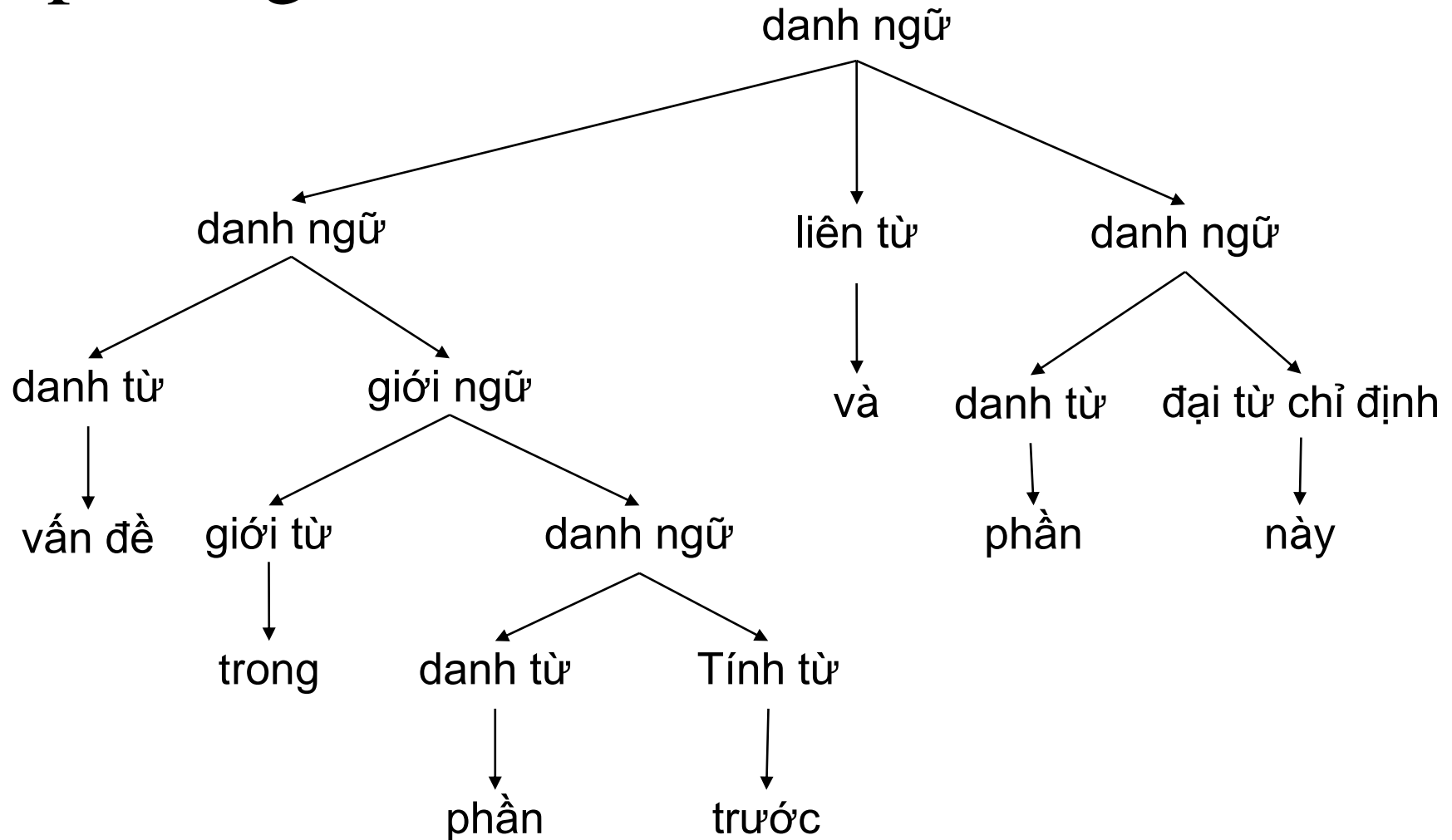
Solution 2: add probabilities into the rule set

- The ambiguity in the sentence “*Tôi nhìn thấy anh Hải ở tầng hai*” can be solved
- The ambiguity of word characteristics has not been solved.
- Eg., noun phrase “*vấn đề trong phần trước và phần này*”

Ambiguity in Vietnamese syntactic parsing



Ambiguity in Vietnamese syntactic parsing



Specific words may affect the result of syntactic parsing

For example:

- “*Tôi ăn*” rarely be accepted as a sentence because the information in that sentence is small.
 - “*Tôi đang ăn*” is more likely to be accepted.
- Has to consider the characteristic of the main word in a sentence
2. Ambiguity due to removing the conjunction word
- Can say: *bạn tôi, con tôi*;
 - Cannot say: *con chó tôi, con mèo tôi*.
- Word also play an important role in syntactic parsing
- Add word information to the grammar (enriching PCFG)

Enriching a PCFG

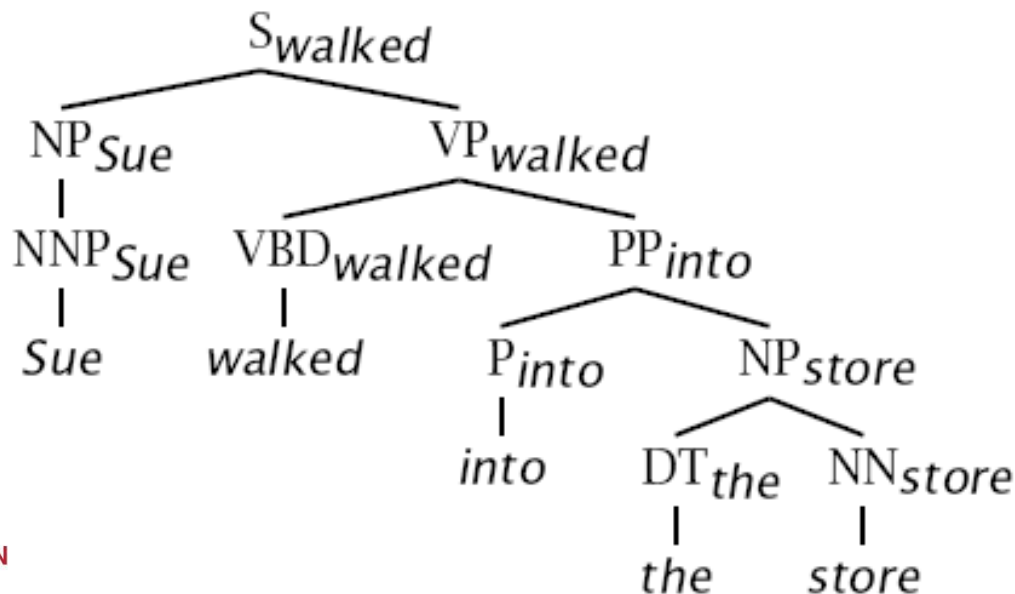
- A naive PCFG works quite poorly due to the independence assumptions
- Fix: encode more information into the nonterminal space
 - Structure sensitivity
 - Expansion of nodes depends a lot on their position in the tree (independent of lexical content)
 - E.g., enrich nodes by also recording their parents:
 ${}^S\text{NP}$ is different to ${}^V\text{P}\text{NP}$

Enriching a PCFG

- (Head) Lexicalization (Collins 1997; Charniak 1997)
 - The head word of a phrase gives a good representation of the phrase's structure and meaning
 - Puts the properties of words back into a PCFG

VP(dumped) \rightarrow VBD(dumped) NP(sacks) PP(into) 3×10^{-10}

VP(dumped) \rightarrow VBD(dumped) NP(cats) PP(into) 8×10^{-11}



Enriching a PCFG

- Lexicalized PCFG : PLCFG (Probabilistic Lexicalized CFG, Collins 1997; Charniak 1997)
- Puts the properties of words back into a PCFG
- **Head** structure
 - Each node in the parsed tree is attached with a *lexical head*
 - To define a *head* node, we have to find it among all of its children (define *head* in the RHS of a rule).

Enriching a PCFG

$\text{VP(dumped)} \rightarrow \text{VBD(dumped)} \text{ NP(sacks)} \text{ PP(into)} \quad 3 \cdot 10^{-10}$

$\text{VP(dumped)} \rightarrow \text{VBD(dumped)} \text{ NP(cats)} \text{ PP(into)} \quad 8 \cdot 10^{-11}$

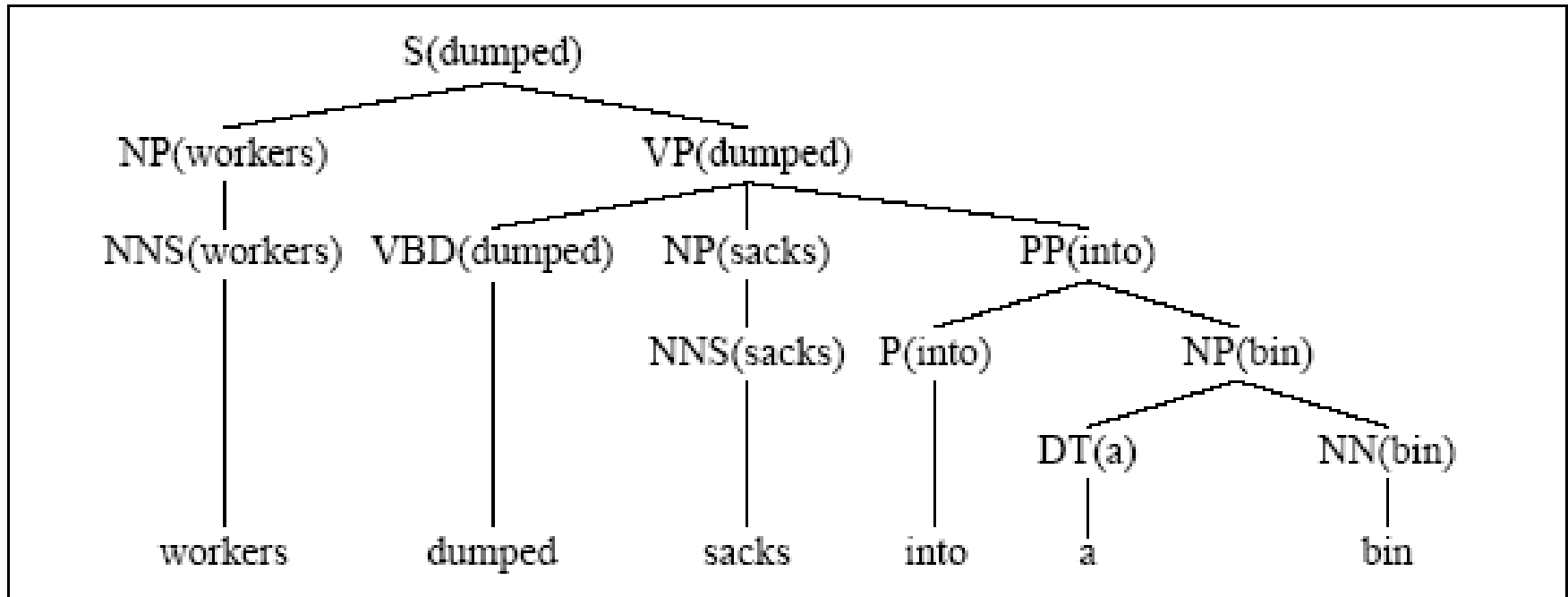


Figure 12.12 A lexicalized tree from Collins (1999).

Limitations of PLCFG

VP -> VBD NP PP

VP(*dumped*) -> VBD(*dumped*) NP(*sacks*)
PP(*into*)

- We don't have a large enough corpus!
 - To represent all syntactic cases for each word

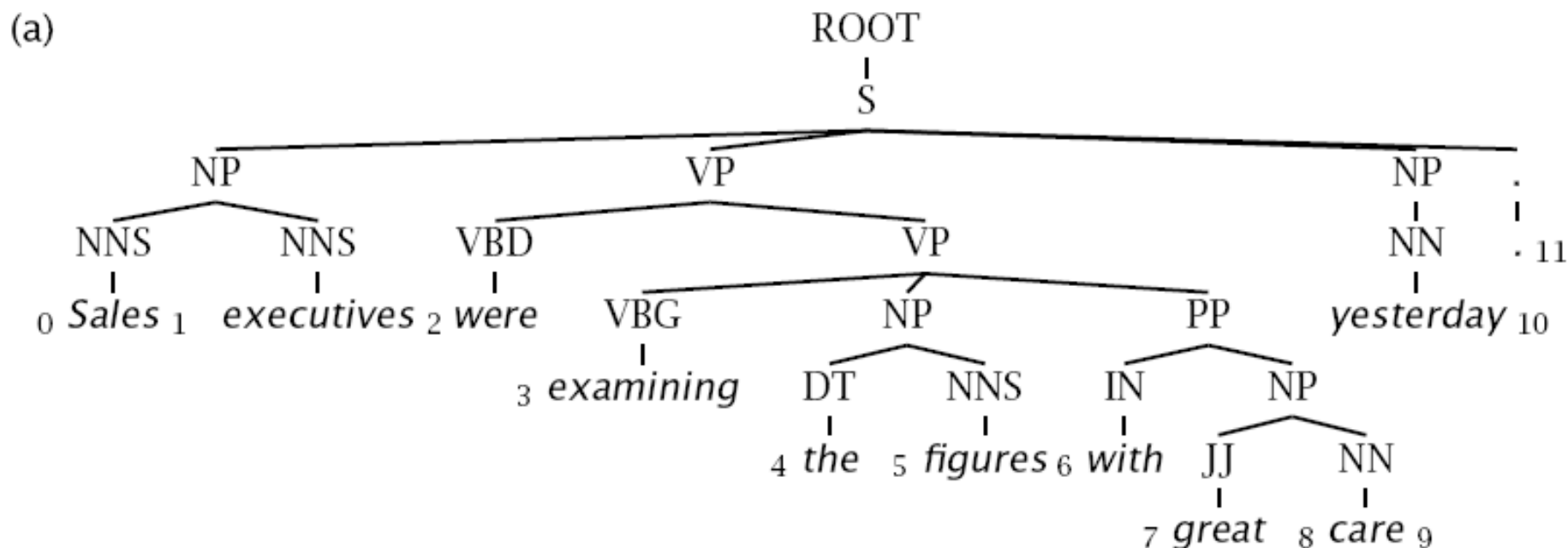
Penn Treebank

- The Penn Treebank – 1 million words of parsed English WSJ – has been a key resource
- Sparseness:
 - 965,000 constituents, but only 66 WHADJP, of which only 6 aren't *how much* or *how many*
- Most intelligent processing depends on **bilexical statistics**: likelihoods of relationships between pairs of words.

A Penn Treebank tree

```
( (S
  (NP-SBJ
    (NP (NNP Pierre) (NNP Vinken) )
    ( , , )
    (ADJP
      (NP (CD 61) (NNS years) )
      (JJ old) )
    ( , , ) )
  (VP (MD will)
    (VP (VB join)
      (NP (DT the) (NN board) )
      (PP-CLR (IN as)
        (NP (DT a) (JJ nonexecutive) (NN director) ))
      (NP-TMP (NNP Nov.) (CD 29) )))
  ( . . ) ))
```

Evaluation



(b) Brackets in gold standard tree (a.):

S-(0:11), **NP-(0:2)**, VP-(2:9), VP-(3:9), **NP-(4:6)**, PP-(6-9), NP-(7,9), *NP-(9:10)

(c) Brackets in candidate parse:

S-(0:11), **NP-(0:2)**, VP-(2:10), VP-(3:10), NP-(4:10), **NP-(4:6)**, PP-(6-10), NP-(7,10)

(d) Precision:	3/8 = 37.5%	Crossing Brackets:	0
Recall:	3/8 = 37.5%	Crossing Accuracy:	100%
Labeled Precision:	3/8 = 37.5%	Tagging Accuracy:	10/11 = 90.9%
Labeled Recall:	3/8 = 37.5%		

Performance's Measurements

		Human assignments		Total
		Yes	No	
System assignments	Yes	HSA	SA - HSA	SA
	No	HA - HSA		
Total		HA		

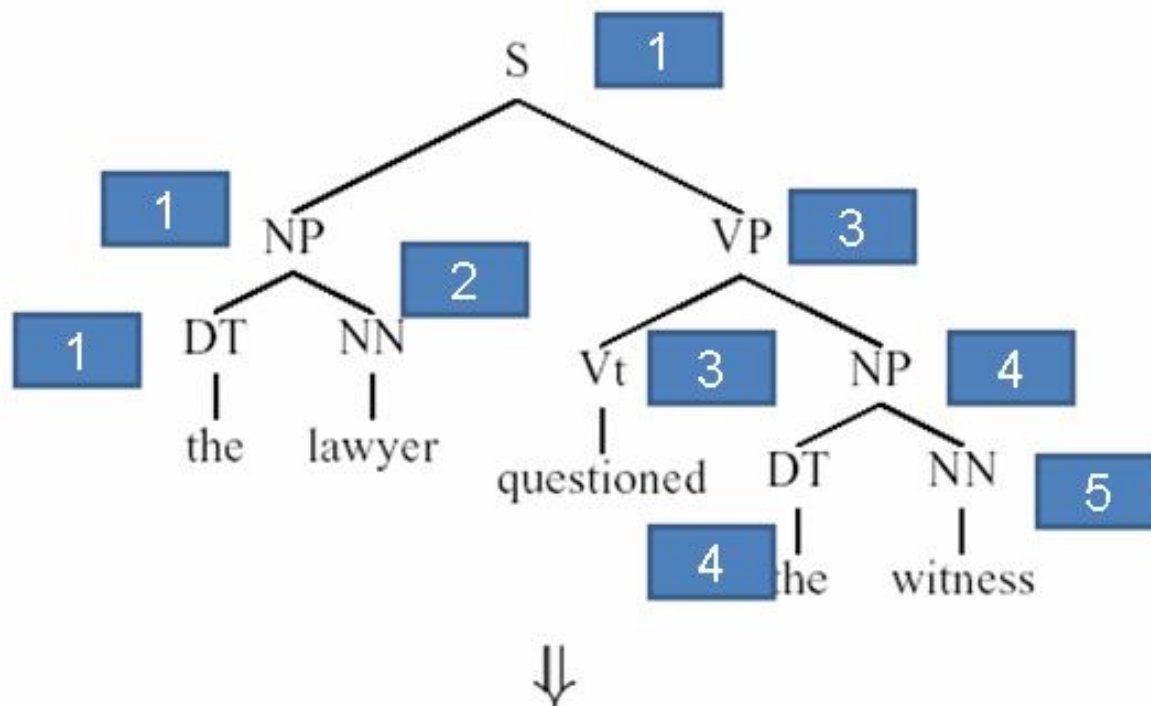
Precision: %assignments made that were correct (%THợp hệ tính đúng).

Recall: %possible assignments that were actually assigned (%THợp hệ tính đúng so với con người).

$$precision = \frac{HSA}{SA}$$

$$recall = \frac{HSA}{HA}$$

Represent a tree by its syntactic constituents



Label	Start Point	End Point
NP	1	2
NP	4	5
VP	3	5
S	1	5

Evaluate

Precision and Recall

Label	Start Point	End Point
NP	1	2
NP	4	5
NP	4	8
PP	6	8
NP	7	8
VP	3	8
S	1	8

Label	Start Point	End Point
NP	1	2
NP	4	5
PP	6	8
NP	7	8
VP	3	8
S	1	8

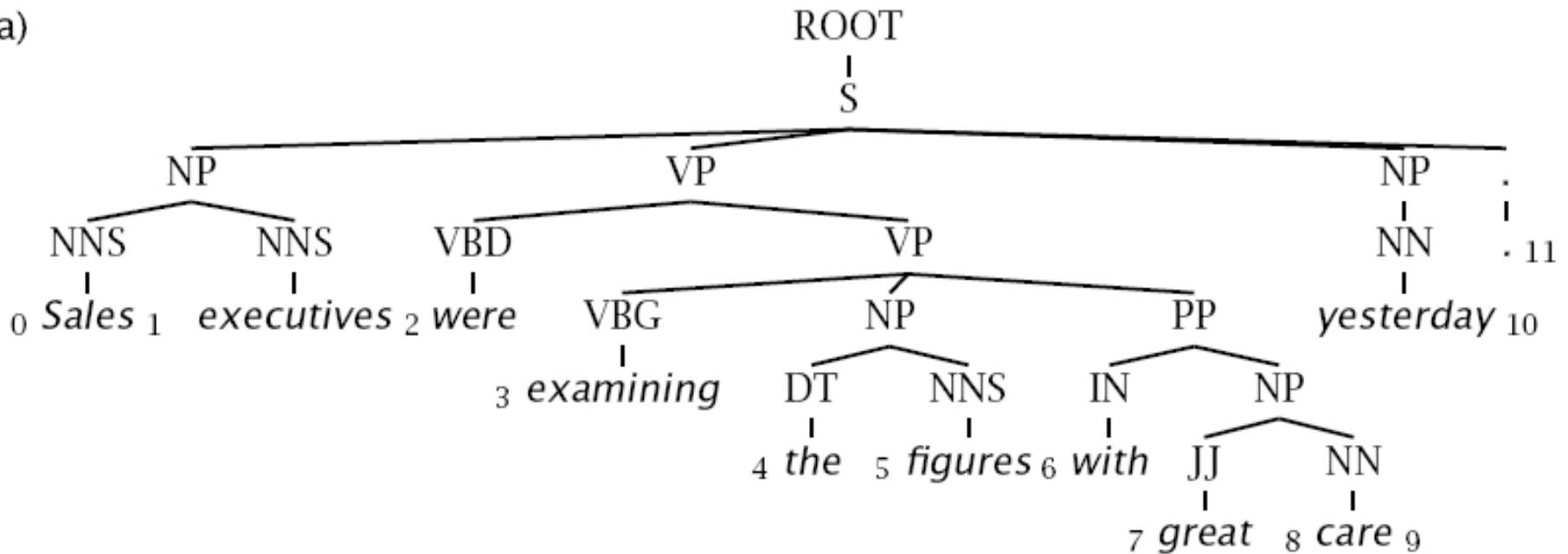
- G = number of constituents in **gold standard** = 7
- P = number in **parse output** = 6
- C = number correct = 6

$$\text{Recall} = 100\% \times \frac{C}{G} = 100\% \times \frac{6}{7}$$

$$\text{Precision} = 100\% \times \frac{C}{P} = 100\% \times \frac{6}{6}$$

Example 2

(a)



(b) Brackets in gold standard tree (a.):

S-(0:11), **NP-(0:2)**, **VP-(2:9)**, **VP-(3:9)**, **NP-(4:6)**, **PP-(6-9)**, **NP-(7,9)**, *NP-(9:10)

(c) Brackets in candidate parse:

S-(0:11), **NP-(0:2)**, **VP-(2:10)**, **VP-(3:10)**, **NP-(4:10)**, **NP-(4:6)**, **PP-(6-10)**, **NP-(7,10)**

(d) Precision:	$3/8 = 37.5\%$	Crossing Brackets:	0
Recall:	$3/8 = 37.5\%$	Crossing Accuracy:	100%
Labeled Precision:	$3/8 = 37.5\%$	Tagging Accuracy:	$10/11 = 90.9\%$
Labeled Recall:	$3/8 = 37.5\%$		

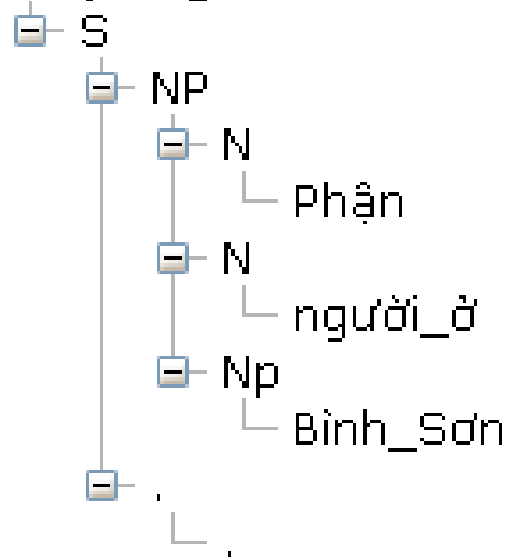
Exercise – compute P, R

Gold standard syntactic structure:

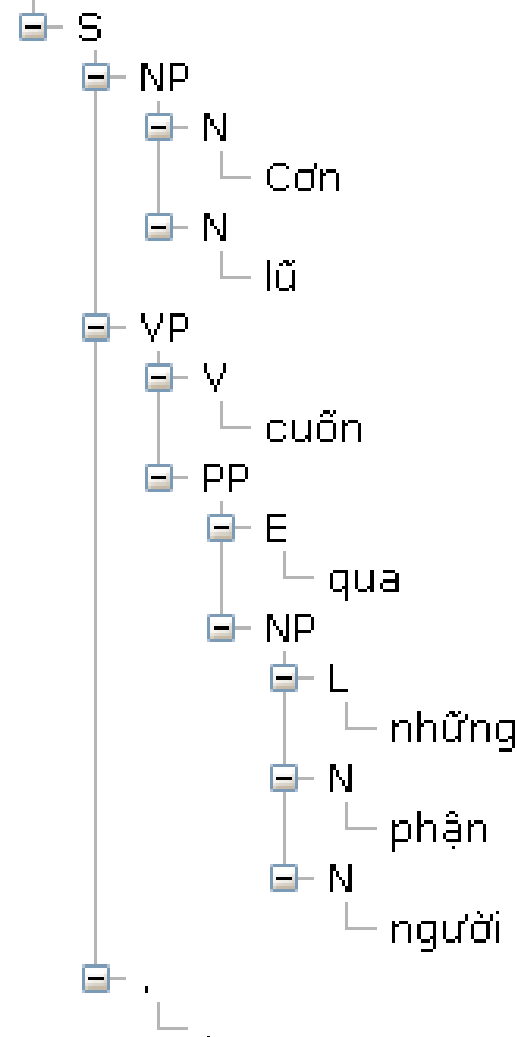
- (S (NP (N Cơn)(N lũ)) (VP(V cuốn)(V qua) (NP (L những)(N phận)(N người)))) (. .))
- (S(NP(N Phận)(N người) (PP(E ở) (NP(Np Bình Sơn)))))(. .))

Automatically generated syntactic structure:

Phận người ở Bình Sơn .



Cơn lũ cuốn qua những phận người .



Some syntactic parsers:

- CFG (context free grammar):
 - Berkeley : <http://nlp.cs.berkeley.edu/software.shtml>
 - Charniak: <http://bllip.cs.brown.edu/resources.shtml>
- HPSG (Head-driven Phrase Structure Grammar)
 - Enju, deepNLP: <https://mynlp.github.io/enju/>
- Dependency grammar
 - ClearNLP : <http://clearnlp.wikispaces.com/depParser>
 - Google SyntaxNet: open-source, using deep learning
 - <https://research.googleblog.com/2016/05/announcing-syntaxnet-worlds-most.html>
 - Netbase, for twitter sentences
 - <https://www.codeproject.com/Articles/43372/NetBase-A-Minimal-NET-Database-with-a-Small-SQL>
 - Stanford : <https://nlp.stanford.edu/software/lex-parser.shtml>