دانشگاه صنعتی امیرکبیر
دانشکده مهندسی کامپیوتر و فناوری اطلاعات

برقراری ارتباط با زبان طبیعی

«هوش مصنوعی: یک رهیافت نوین»، فصل ۲۳

ارائه‌دهنده: سیده فاطمه موسوی

نیم‌سال اول ۱۳۹۹–۱۳۹۸

Context Free Grammar

A context-free grammar (CFG) is a 4-tuple $G = (N, \Sigma, R, S)$ where:

- N is a finite set of non-terminal symbols.
- Σ is a finite set of terminal symbols.
- R is a finite set of rules of the form $X \rightarrow Y_1 Y_2 \ldots Y_n$, where $X \in N, n \geq 0$, and $Y_i \in (N \cup \Sigma)$ for $i = 1 \ldots n$.
- $S \in N$ is a distinguished start symbol.

A Simple CFG - Example

$N = \{\text{S}, \text{NP}, \text{VP}, \text{PP}, \text{DT}, \text{Vi}, \text{Vt}, \text{NN}, \text{IN}\}$
$S = \text{S}$
$\Sigma = \{\text{sleeps}, \text{saw}, \text{man}, \text{woman}, \text{dog}, \text{telescope}, \text{the}, \text{with}, \text{in}\}$

$R =$

S	→	NP	VP
VP	→	Vi	
VP	→	Vt	NP
VP	→	VP	PP
NP	→	DT	NN
NP	→	NP	PP
PP	→	IN	NP

Vi	→	sleeps
Vt	→	saw
NN	→	man
NN	→	woman
NN	→	telescope
NN	→	dog
DT	→	the
IN	→	with
IN	→	in

N contains a basic set of syntactic categories: S=sentence, VP=verb phrase, NP=noun phrase, PP=prepositional phrase, DT=determiner, Vi=intransitive verb, Vt=transitive verb, NN=noun, IN=preposition. The set Σ is the set of possible words in the language.

Left-Most Derivation (Parse Tree)

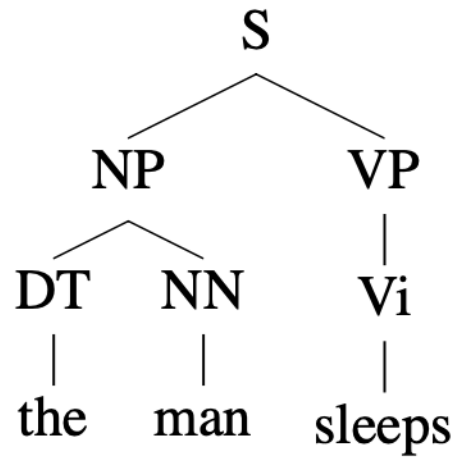
Given a context-free grammar G, a left-most derivation is a sequence of strings $s_1 \ldots s_n$ where

- $s_1 = \mathtt{S}$. i.e., s_1 consists of a single element, the start symbol.
- $s_n \in \Sigma^*$, i.e. s_n is made up of terminal symbols only (we write Σ^* to denote the set of all possible strings made up of sequences of words taken from Σ.)
- Each s_i for $i = 2 \ldots n$ is derived from s_{i-1} by picking the left-most non-terminal X in s_{i-1} and replacing it by some β where $X \rightarrow \beta$ is a rule in R.

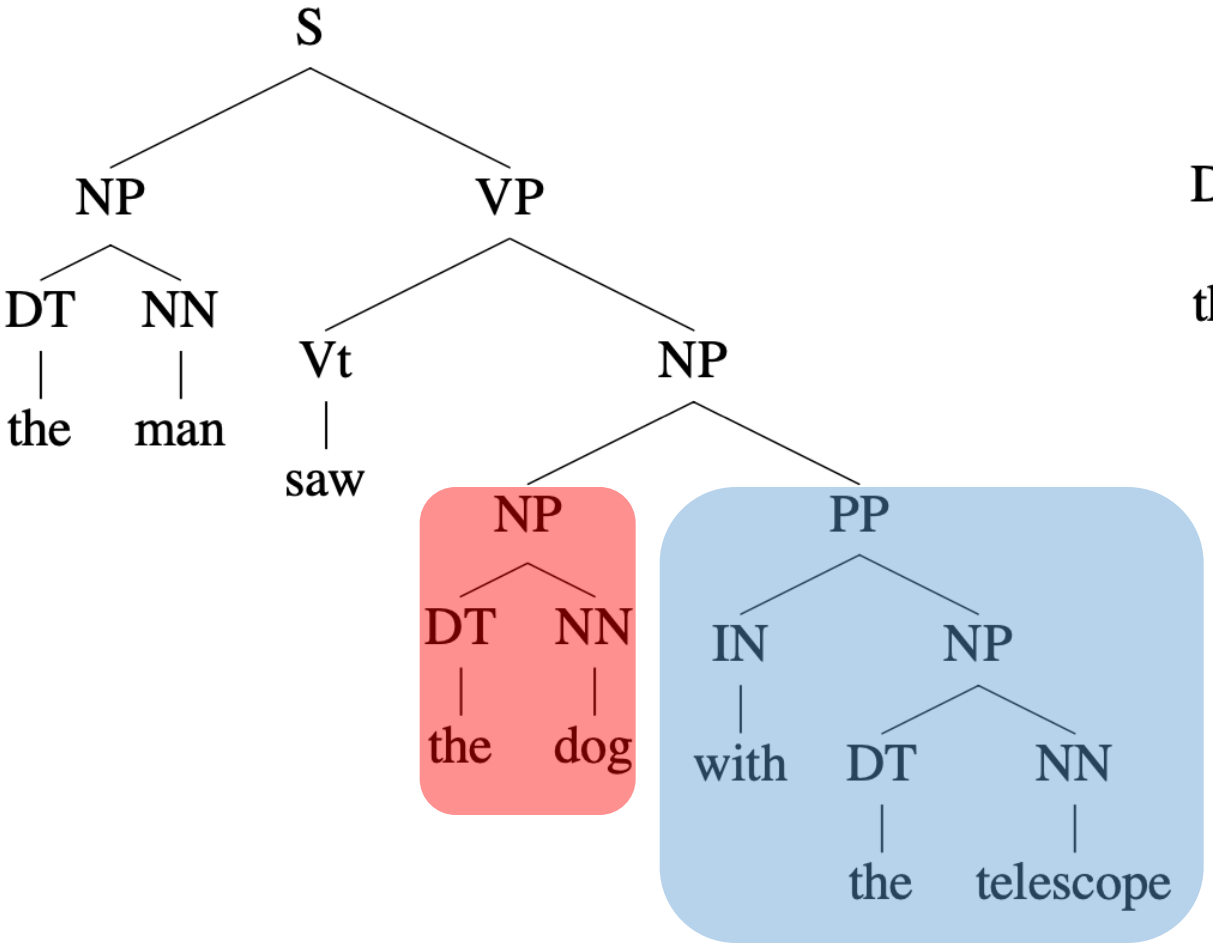
A string $s \in \Sigma^*$ is said to be in the *language* defined by the CFG, if there is at least one derivation whose yield is s.

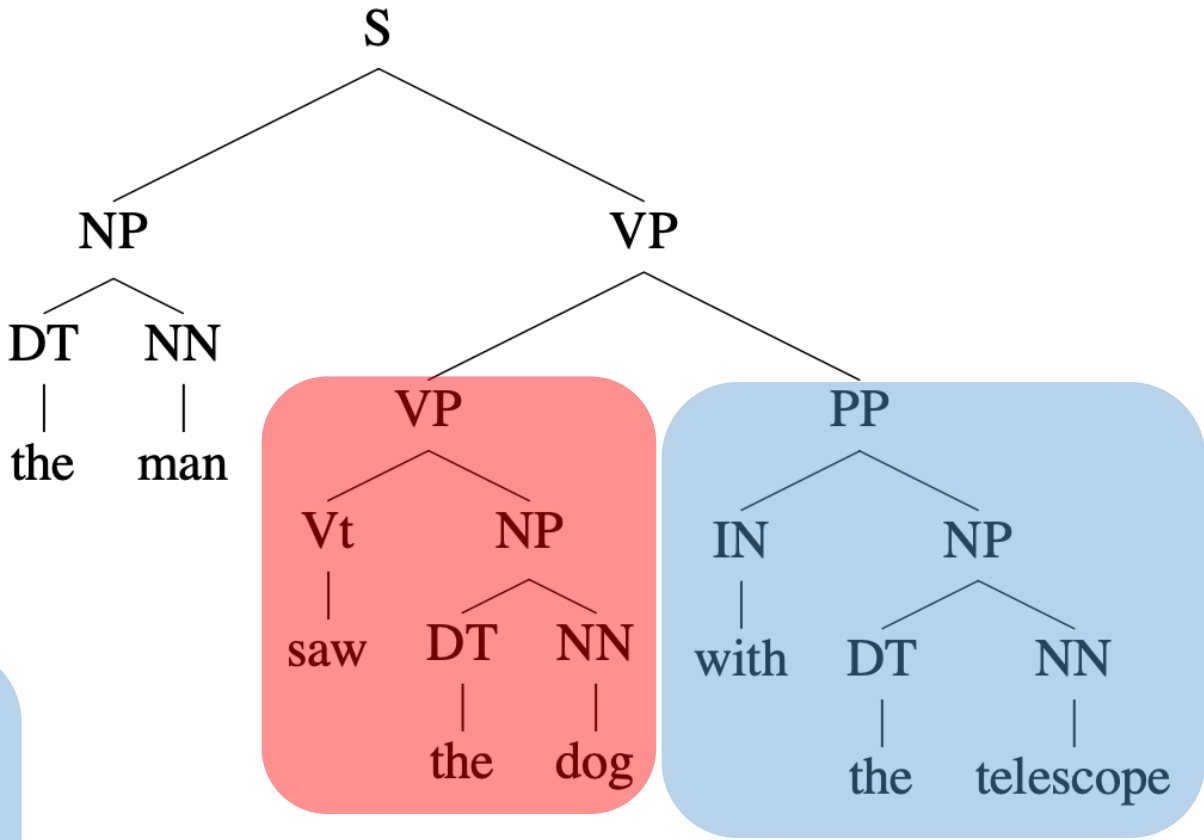
Parse Tree - Example

- $s_1 = \texttt{S}$.
- $s_2 = \texttt{NP VP}$. (We have taken the left-most non-terminal in s_1, namely S, and chosen the rule $\texttt{S} \rightarrow \texttt{NP VP}$, thereby replacing S by NP followed by VP.)
- $s_3 = \texttt{DT NN VP}$. (We have used the rule $\texttt{NP} \rightarrow \texttt{DT NN}$ to expand the left-most non-terminal, namely NP.)
- $s_4 = \texttt{the NN VP}$. (We have used the rule $\texttt{DT} \rightarrow \texttt{the}$.)
- $s_5 = \texttt{the man VP}$. (We have used the rule $\texttt{NN} \rightarrow \texttt{man}$.)
- $s_6 = \texttt{the man Vi}$. (We have used the rule $\texttt{VP} \rightarrow \texttt{Vi}$.)
- $s_7 = \texttt{the man sleeps}$. (We have used the rule $\texttt{Vi} \rightarrow \texttt{sleeps}$.)



Ambiguity


S
NP
VP
DT
NN
Vt
NP
the
man
saw
NP
PP
DT
NN
IN
NP
the
dog
with
DT
NN
the
telescope


S
NP
VP
DT
NN
VP
PP
the
man
Vt
NP
IN
NP
saw
DT
NN
with
DT
NN
the
dog
the
telescope

Probabilistic Context-Free Grammars

- $\mathcal{T}_G$ is the set of all possible left-most derivations (parse trees) under the grammar G. When the grammar G is clear from context we will often write this as simply $\mathcal{T}$.
- For any derivation $t \in \mathcal{T}_G$, we write $\textsf{yield}(t)$ to denote the string $s \in \Sigma^*$ that is the yield of t (i.e., $\textsf{yield}(t)$ is the sequence of words in t).
- For a given sentence $s \in \Sigma^*$, we write $\mathcal{T}_G(s)$ to refer to the set

$$\{t : t \in \mathcal{T}_G, \textsf{yield}(t) = s\}$$

  That is, $\mathcal{T}_G(s)$ is the set of possible parse trees for s.
- We say that a sentence s is *ambiguous* if it has more than one parse tree, i.e., $|\mathcal{T}_G(s)| > 1$.
- We say that a sentence s is *grammatical* if it has at least one parse tree, i.e., $|\mathcal{T}_G(s)| > 0$.

Probabilistic Context-Free Grammars

The key idea in probabilistic context-free grammars is to extend our definition to give a *probability distribution over possible derivations*. That is, we will find a way to define a distribution over parse trees, $p(t)$, such that for any $t \in \mathcal{T}_G$,

$$p(t) \geq 0$$

and in addition such that

$$\sum_{t \in \mathcal{T}_G} p(t) = 1$$

Probabilistic Context-Free Grammars

Why is this a useful problem? A crucial idea is that once we have a function $p(t)$, we have a ranking over possible parses for any sentence in order of probability. In particular, given a sentence s, we can return

$$\arg \max_{t \in \mathcal{T}_G(s)} p(t)$$

as the output from our parser—this is the most likely parse tree for s under the model. Thus if our distribution $p(t)$ is a good model for the probability of different parse trees in our language, we will have an effective way of dealing with ambiguity.

PCFG (Definition)

1. *A context-free grammar* $G = (N, \Sigma, S, R)$.

2. *A parameter*

$$q(\alpha \rightarrow \beta)$$

for each rule $\alpha \rightarrow \beta \in R$. *The parameter* $q(\alpha \rightarrow \beta)$ *can be interpreted as the conditional probabilty of choosing rule* $\alpha \rightarrow \beta$ *in a left-most derivation, given that the non-terminal being expanded is* α. *For any* $X \in N$, *we have the constraint*

$$\sum_{\alpha \rightarrow \beta \in R : \alpha = X} q(\alpha \rightarrow \beta) = 1$$

In addition we have $q(\alpha \rightarrow \beta) \geq 0$ *for any* $\alpha \rightarrow \beta \in R$.

Given a parse-tree $t \in \mathcal{T}_G$ *containing rules* $\alpha_1 \rightarrow \beta_1, \alpha_2 \rightarrow \beta_2, \ldots, \alpha_n \rightarrow \beta_n$, *the probability of* t *under the PCFG is*

$$p(t) = \prod_{i=1}^{n} q(\alpha_i \rightarrow \beta_i)$$

PCFG- Example

$N = \{\text{S}, \text{NP}, \text{VP}, \text{PP}, \text{DT}, \text{Vi}, \text{Vt}, \text{NN}, \text{IN}\}$

$S = \text{S}$

$\Sigma = \{\text{sleeps}, \text{saw}, \text{man}, \text{woman}, \text{dog}, \text{telescope}, \text{the}, \text{with}, \text{in}\}$

$R, q =$

S	→	NP	VP	1.0
VP	→	Vi		0.3
VP	→	Vt	NP	0.5
VP	→	VP	PP	0.2
NP	→	DT	NN	0.8
NP	→	NP	PP	0.2
PP	→	IN	NP	1.0

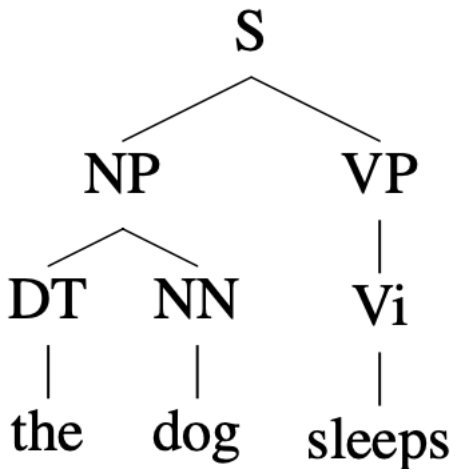
Vi	→	sleeps	1.0
Vt	→	saw	1.0
NN	→	man	0.1
NN	→	woman	0.1
NN	→	telescope	0.3
NN	→	dog	0.5
DT	→	the	1.0
IN	→	with	0.6
IN	→	in	0.4

$$\sum_{\alpha \rightarrow \beta \in R : \alpha = \texttt{VP}} q(\alpha \rightarrow \beta) = q(\texttt{VP} \rightarrow \texttt{Vi}) + q(\texttt{VP} \rightarrow \texttt{Vt NP}) + q(\texttt{VP} \rightarrow \texttt{VP PP})$$

$$= 0.3 + 0.5 + 0.2$$

$$= 1.0$$

PCFG- Example



$$p(t) = q(\texttt{S} \rightarrow \texttt{NP VP}) \times q(\texttt{NP} \rightarrow \texttt{DT NN}) \times q(\texttt{DT} \rightarrow \texttt{the}) \times q(\texttt{NN} \rightarrow \texttt{dog}) \times q(\texttt{VP} \rightarrow \texttt{Vi}) \times q(\texttt{Vi} \rightarrow \texttt{sleeps})$$

PCFG - Generate Parse Trees

Intuitively, PCFGs make the assumption that parse trees are generated stochastically, according to the following process:

- Define $s_1 = S, i = 1$.
- While s_i contains at least one non-terminal:
  - Find the left-most non-terminal in s_i, call this X.
  - Choose one of the rules of the form $X \rightarrow \beta$ from the distribution $q(X \rightarrow \beta)$.
  - Create s_{i+1} by replacing the left-most X in s_i by β.
  - Set $i = i + 1$.

Deriving a PCFG from a Corpus

Having defined PCFGs, the next question is the following: how do we derive a PCFG from a corpus? We will assume a set of training data, which is simply a set of parse trees $t_1, t_2, \ldots, t_m$. As before, we will write $\textsf{yield}(t_i)$ to be the yield for the i'th parse tree in the sentence, i.e., $\textsf{yield}(t_i)$ is the i'th sentence in the corpus.

Each parse tree t_i is a sequence of context-free rules: we assume that every parse tree in our corpus has the same symbol, S, at its root. We can then define a PCFG (N, Σ, S, R, q) as follows:

Deriving a PCFG from a Corpus

- N is the set of all non-terminals seen in the trees $t_1 \ldots t_m$.
- Σ is the set of all words seen in the trees $t_1 \ldots t_m$.
- The start symbol S is taken to be S.
- The set of rules R is taken to be the set of all rules $\alpha \rightarrow \beta$ seen in the trees $t_1 \ldots t_m$.
- The maximum-likelihood parameter estimates are

$$q_{ML}(\alpha \rightarrow \beta) = \frac{\text{Count}(\alpha \rightarrow \beta)}{\text{Count}(\alpha)}$$

where $\text{Count}(\alpha \rightarrow \beta)$ is the number of times that the rule $\alpha \rightarrow \beta$ is seen in the trees $t_1 \ldots t_m$, and $\text{Count}(\alpha)$ is the number of times the non-terminal α is seen in the trees $t_1 \ldots t_m$.

Parsing with PCFGs

A crucial question is the following: given a sentence s, how do we find the highest scoring parse tree for s, or more explicitly, how do we find

$$\arg \max_{t \in \mathcal{T}(s)} p(t) \quad ?$$

Definition 2 (Chomsky Normal Form) *A context-free grammar $G = (N, \Sigma, R, S)$ is in Chomsky form if each rule $\alpha \rightarrow \beta \in R$ takes one of the two following forms:*

- $X \rightarrow Y_1 Y_2$ *where* $X \in N, Y_1 \in N, Y_2 \in N$.
- $X \rightarrow Y$ *where* $X \in N, Y \in \Sigma$.

Hence each rule in the grammar either consists of a non-terminal X rewriting as exactly two non-terminal symbols, $Y_1 Y_2$; or a non-terminal X rewriting as exactly one terminal symbol Y. $\square$

Parsing with PCFGs - CYK Algorithm

- For a given sentence $x_1 \dots x_n$, define $\mathcal{T}(i, j, X)$ for any $X \in N$, for any (i, j) such that $1 \leq i \leq j \leq n$, to be the set of all parse trees for words $x_i \dots x_j$ such that non-terminal X is at the root of the tree.

- Define

$$\pi(i, j, X) = \max_{t \in \mathcal{T}(i,j,X)} p(t)$$

(we define $\pi(i, j, X) = 0$ if $\mathcal{T}(i, j, X)$ is the empty set).

Parsing with PCFGs - CYK Algorithm

The key observation in the CKY algorithm is that we can use a recursive definition of the π values, which allows a simple bottom-up dynamic programming algorithm. The algorithm is "bottom-up", in the sense that it will first fill in $\pi(i, j, X)$ values for the cases where $j = i$, then the cases where $j = i + 1$, and so on.

The base case in the recursive definition is as follows: for all $i = 1 \ldots n$, for all $X \in N$,

$$\pi(i, i, X) = \begin{cases} q(X \rightarrow x_i) & \text{if } X \rightarrow x_i \in R \\ 0 & \text{otherwise} \end{cases}$$

The recursive definition is as follows: for all (i, j) such that $1 \leq i < j \leq n$, for all $X \in N$,

$$\pi(i, j, X) = \max_{\substack{X \rightarrow YZ \in R, \\ s \in \{i \ldots (j-1)\}}} (q(X \rightarrow YZ) \times \pi(i, s, Y) \times \pi(s + 1, j, Z)) \quad (1)$$

Input: a sentence $s = x_1 \dots x_n$, a PCFG $G = (N, \Sigma, S, R, q)$.
Initialization:
For all $i \in \{1 \dots n\}$, for all $X \in N$,

$$\pi(i, i, X) = \begin{cases} q(X \rightarrow x_i) & \text{if } X \rightarrow x_i \in R \\ 0 & \text{otherwise} \end{cases}$$

Algorithm:

- For $l = 1 \dots (n-1)$
  - For $i = 1 \dots (n-l)$
    - Set $j = i + l$
    - For all $X \in N$, calculate

$$\pi(i, j, X) = \max_{\substack{X \rightarrow YZ \in R, \\ s \in \{i \dots (j-1)\}}} (q(X \rightarrow YZ) \times \pi(i, s, Y) \times \pi(s+1, j, Z))$$

and

$$bp(i, j, X) = \arg \max_{\substack{X \rightarrow YZ \in R, \\ s \in \{i \dots (j-1)\}}} (q(X \rightarrow YZ) \times \pi(i, s, Y) \times \pi(s+1, j, Z))$$

Output: Return $\pi(1, n, S) = \max_{t \in \mathcal{T}(s)} p(t)$, and backpointers bp which allow recovery of $\arg \max_{t \in \mathcal{T}(s)} p(t)$.

Weaknesses of PCFGs as Parsing Models

- Lack of Sensitivity to Lexical Information
  - The PCFGs we have described essentially generate lexical items as an afterthought, conditioned only on the Part Of Speech directly above them in the tree. This is a very strong independence assumption, which leads to non-optimal decisions being made by the parser in many important cases of ambiguity.
- Lack of Sensitivity to Structural Preferences
  - For two trees with the same set of rules, the PCFG assigns identical probabilities to them, because it can not capture the bias towards close-attachment in this case.

Lack of Sensitivity to Lexical Information

(a)

Rules
S → NP VP
NP → NNS
VP → VP PP
VP → VBD NP
NP → NNS
PP → IN NP
NP → DT NN
NNS → workers
VBD → dumped
NNS → sacks
IN → into
DT → a
NN → bin

(b)

Rules
S → NP VP
NP → NNS
NP → NP PP
VP → VBD NP
NP → NNS
PP → IN NP
NP → DT NN
NNS → workers
VBD → dumped
NNS → sacks
IN → into
DT → a
NN → bin

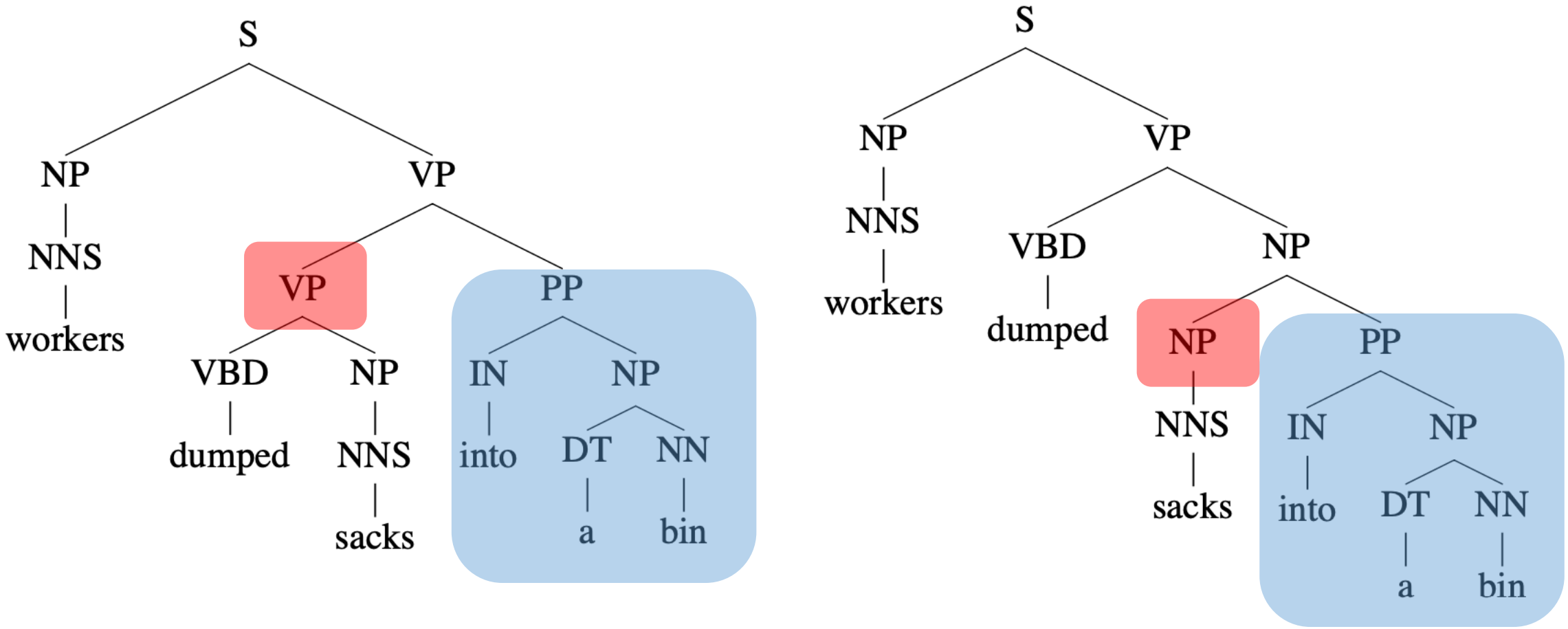
when choosing between the two parse trees, will pick tree (a) if

$$q(\text{VP} \rightarrow \text{VP PP}) > q(\text{NP} \rightarrow \text{NP PP})$$

and will pick tree (b) if

$$q(\text{NP} \rightarrow \text{NP PP}) > q(\text{VP} \rightarrow \text{VP PP})$$

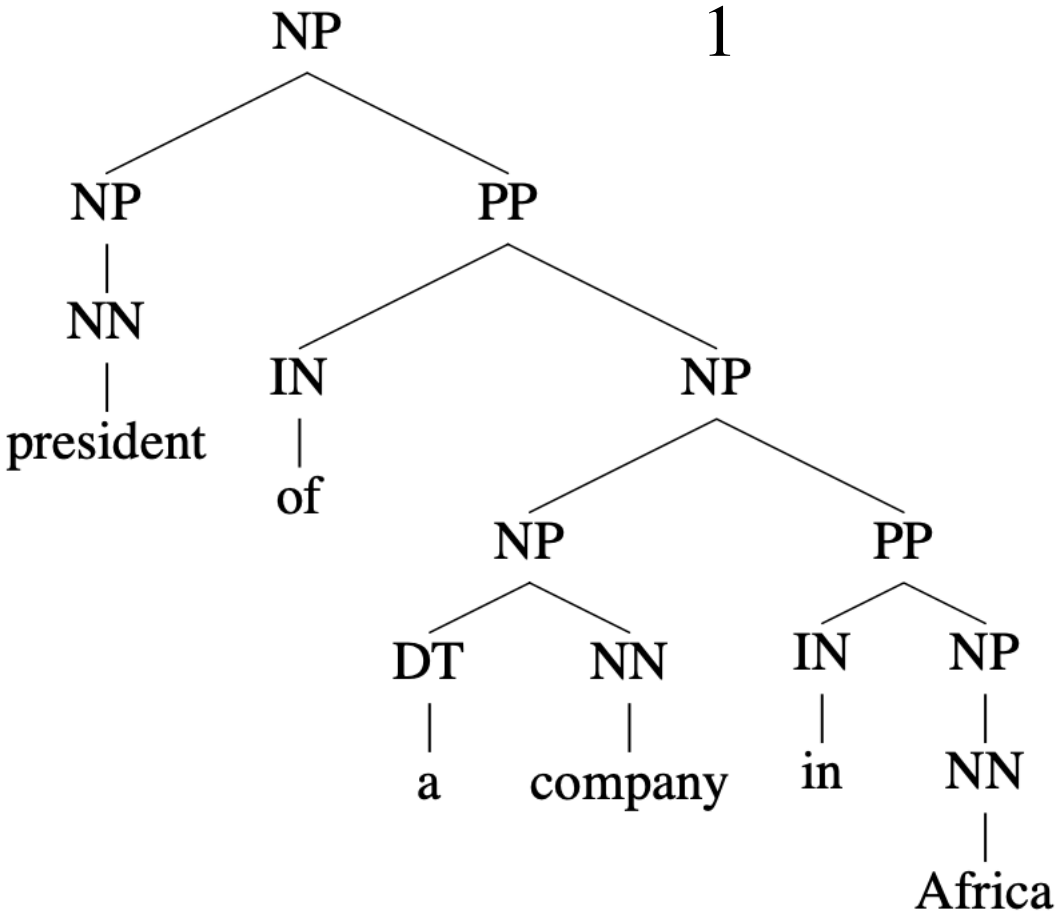
Lack of Sensitivity to Lexical Information



If we look at the preposition, *into*, alone, we find that PPs with *into* as the preposition are almost nine times more likely to attach to a VP rather than an NP (this statistic is taken from the Penn treebank data).

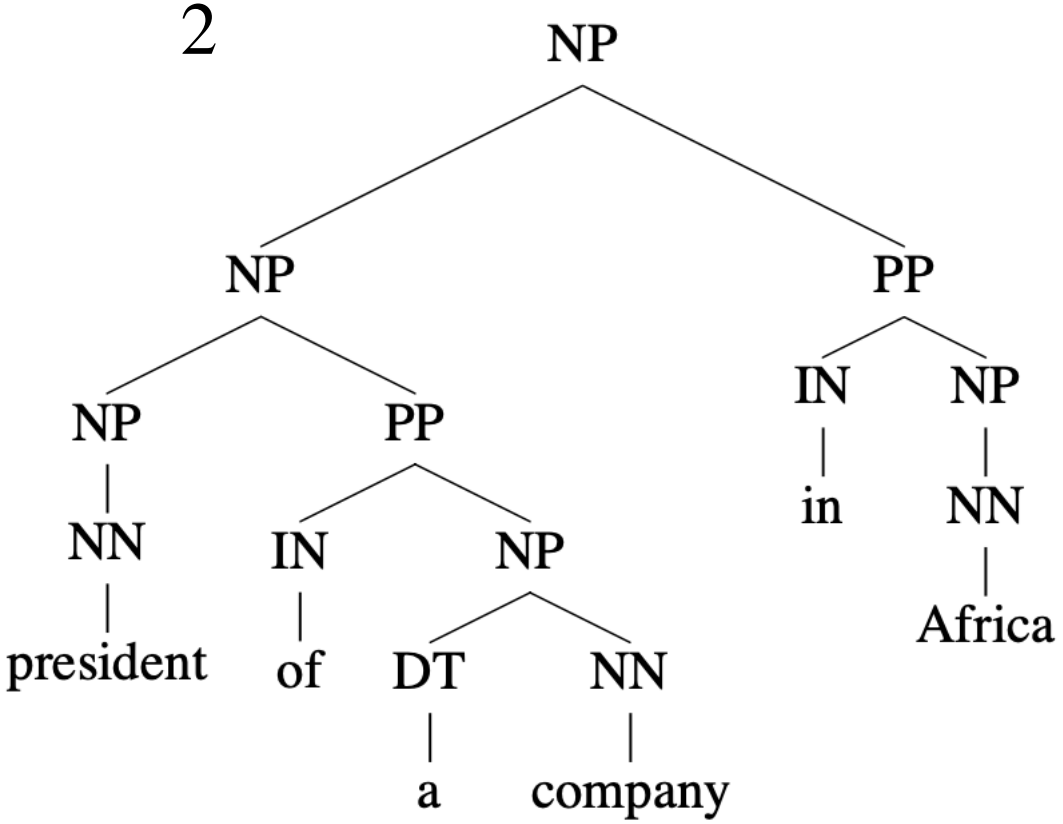
Lack of Sensitivity to Structural Preferences

1



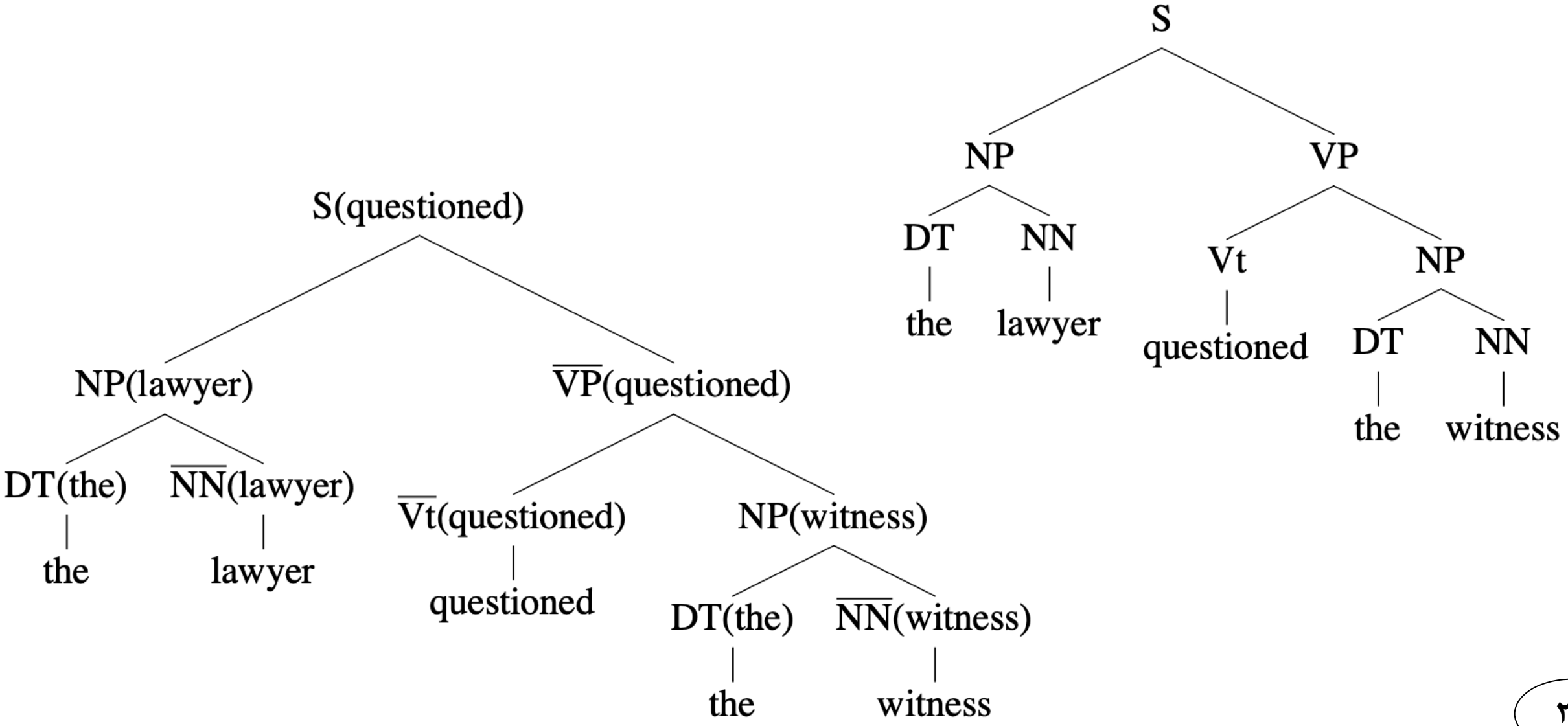
close attachment

2



structure 1 is roughly twice as frequent as structure 2

Lexicalization of a Treebank



Lexicalization of a Treebank

- *head* of the rule
  - the "center" or the most important child of the rule
    - S → NP VP h = 2 (corresponding to the VP)
    - NP → NP PP PP PP h = 1 (corresponding to the NP)
    - PP → IN NP h = 1 (corresponding to the IN)
- Once the head of each context-free rule has been identified, lexical information can be propagated bottom-up through parse trees in the treebank.
  - Researchers have generally used a simple set of rules to automatically identify the head of each context-free rule.

Identify the Head of a Rule

If the rule contains NN, NNS, or NNP:
    Choose the rightmost NN, NNS, or NNP

Else If the rule contains an NP: Choose the leftmost NP

Else If the rule contains a JJ: Choose the rightmost JJ

Else If the rule contains a CD: Choose the rightmost CD

Else Choose the rightmost child

Figure 6: Example of a set of rules that identifies the head of any rule whose left-hand-side is an NP.

Lexicalized PCFGs in Chomsky Normal Form

Definition 1 (Lexicalized PCFGs in Chomsky Normal Form) *A lexicalized PCFG in Chomsky normal form is a 6-tuple* $G = (N, \Sigma, R, S, q, \gamma)$ *where:*

- N *is a finite set of non-terminals in the grammar.*
- Σ *is a finite set of lexical items in the grammar.*
- R *is a set of rules. Each rule takes one of the following three forms:*

  1. $X(h) \rightarrow_1 Y_1(h)\ Y_2(m)$ *where* $X, Y_1, Y_2 \in N$, $h, m \in \Sigma$.
  2. $X(h) \rightarrow_2 Y_1(m)\ Y_2(h)$ *where* $X, Y_1, Y_2 \in N$, $h, m \in \Sigma$.
  3. $X(h) \rightarrow h$ *where* $X \in N$, $h \in \Sigma$.

Lexicalized PCFGs in Chomsky Normal Form

- *For each rule $r \in R$ there is an associated parameter*

$$q(r)$$

The parameters satisfy $q(r) \geq 0$, and for any $X \in N, h \in \Sigma$,

$$\sum_{r \in R : LHS(r) = X(h)} q(r) = 1$$

where we use $LHS(r)$ to refer to the left hand side of any rule r.

- *For each $X \in N$, $h \in \Sigma$, there is a parameter $\gamma(X, h)$. We have $\gamma(X, h) \geq 0$, and $\sum_{X \in N, h \in \Sigma} \gamma(X, h) = 1$.*

Given a left-most derivation $r_1, r_2, \ldots r_N$ under the grammar, where each r_i is a member of R, the probability of the derivation is

$$\gamma(LHS(r_1)) \times \prod_{i=1}^{N} q(r_i)$$

Example

S(questioned) $\rightarrow_2$ NP(lawyer) VP(questioned)
NP(lawyer) $\rightarrow_2$ DT(the) NN(lawyer)
DT(the) $\rightarrow$ the
NN(lawyer) $\rightarrow$ lawyer
VP(questioned) $\rightarrow_1$ Vt(questioned) NP(witness)
NP(witness) $\rightarrow_2$ DT(the) NN(witness)
DT(the) $\rightarrow$ the
NN(witness) $\rightarrow$ witness

$$
\begin{aligned}
&\gamma(\text{S}, \text{questioned}) \\
&\times q(\text{S(questioned)} \rightarrow_2 \text{NP(lawyer) VP(questioned)}) \\
&\times q(\text{NP(lawyer)} \rightarrow_2 \text{DT(the) NN(lawyer)}) \\
&\times q(\text{DT(the)} \rightarrow \text{the}) \\
&\times q(\text{NN(lawyer)} \rightarrow \text{lawyer}) \\
&\times q(\text{VP(questioned)} \rightarrow_1 \text{Vt(questioned) NP(witness)}) \\
&\times q(\text{NP(witness)} \rightarrow_2 \text{DT(the) NN(witness)}) \\
&\times q(\text{DT(the)} \rightarrow \text{the}) \\
&\times q(\text{NN(witness)} \rightarrow \text{witness})
\end{aligned}
$$

Parameter Estimation in Lexicalized PCFGs

First, for a given rule of the form

$$X(h) \rightarrow_1 Y_1(h)\ Y_2(m)$$

or

$$X(h) \rightarrow_2 Y_1(m)\ Y_2(h)$$

define the following variables: X is the non-terminal on the left-hand side of the rule; H is the head-word of that non-terminal; R is the rule used, either of the form $X \rightarrow_1 Y_1\ Y_2$ or $X \rightarrow_2 Y_1\ Y_2$; M is the modifier word.

For example, for the rule

$$\text{S(examined)} \rightarrow_2 \text{NP(lawyer)}\ \text{VP(examined)}$$

we have

$$\begin{aligned} X &= \text{S} \\ H &= \text{examined} \\ R &= \text{S} \rightarrow_2 \text{NP VP} \\ M &= \text{lawyer} \end{aligned}$$

Parameter Estimation in Lexicalized PCFGs

$$
\begin{aligned}
& q(\text{S(examined)} \rightarrow_2 \text{NP(lawyer) VP(examined)}) \\
= \quad & P(R = \text{S} \rightarrow_2 \text{NP VP}, M = \text{lawyer} | X = \text{S}, H = \text{examined})
\end{aligned}
$$

The first step in deriving an estimate of $q(\text{S(examined)} \rightarrow_2 \text{NP(lawyer) VP(examined)})$ will be to use the chain rule to decompose the above expression into two terms:

$$
\begin{aligned}
& P(R = \text{S} \rightarrow_2 \text{NP VP}, M = \text{lawyer} | X = \text{S}, H = \text{examined}) \\
= \quad & P(R = \text{S} \rightarrow_2 \text{NP VP} | X = \text{S}, H = \text{examined}) \qquad (3) \\
& \times P(M = \text{lawyer} | R = \text{S} \rightarrow_2 \text{NP VP}, X = \text{S}, H = \text{examined}) \qquad (4)
\end{aligned}
$$

$$
\begin{aligned}
& q(\text{S(examined)} \rightarrow_2 \text{NP(lawyer) VP(examined)}) \\
= \quad & (\lambda_1 \times q_{ML}(\text{S} \rightarrow_2 \text{NP VP} | \text{S}, \text{examined}) + (1 - \lambda_1) \times q_{ML}(\text{S} \rightarrow_2 \text{NP VP} | \text{S})) \\
& \times (\lambda_2 \times q_{ML}(\text{lawyer} | \text{S} \rightarrow_2 \text{NP VP}, \text{examined}) + (1 - \lambda_2) \times q_{ML}(\text{lawyer} | \text{S} \rightarrow_2 \text{NP VP}))
\end{aligned}
$$

Parameter Estimation in Lexicalized PCFGs

$$P(R = \text{S} \rightarrow_2 \text{NP VP} | X = \text{S}, H = \text{examined})$$

is then defined as

$$\lambda_1 \times q_{ML}(\text{S} \rightarrow_2 \text{NP VP} | \text{S}, \text{examined}) + (1 - \lambda_1) \times q_{ML}(\text{S} \rightarrow_2 \text{NP VP} | \text{S})$$

$$q_{ML}(\text{S} \rightarrow_2 \text{NP VP} | \text{S}, \text{examined}) = \frac{\text{count}(R = \text{S} \rightarrow_2 \text{NP VP}, X = \text{S}, H = \text{examined})}{\text{count}(X = \text{S}, H = \text{examined})}$$

$$q_{ML}(\text{S} \rightarrow_2 \text{NP VP} | \text{S}) = \frac{\text{count}(R = \text{S} \rightarrow_2 \text{NP VP}, X = \text{S})}{\text{count}(X = \text{S})}$$

Parameter Estimation in Lexicalized PCFGs

$$q_{ML}(\text{lawyer}|\text{S} \rightarrow_2 \text{NP VP}, \text{examined}) = \frac{\text{count}(M = \text{lawyer}, R = \text{S} \rightarrow_2 \text{NP VP}, H = \text{examined})}{\text{count}(R = \text{S} \rightarrow_2 \text{NP VP}, H = \text{examined})}$$

$$q_{ML}(\text{lawyer}|\text{S} \rightarrow_2 \text{NP VP}) = \frac{\text{count}(M = \text{lawyer}, R = \text{S} \rightarrow_2 \text{NP VP})}{\text{count}(R = \text{S} \rightarrow_2 \text{NP VP})}$$

The estimate of

$$P(M = \text{lawyer}|R = \text{S} \rightarrow_2 \text{NP VP}, X = \text{S}, H = \text{examined})$$

is then

$$\lambda_2 \times q_{ML}(\text{lawyer}|\text{S} \rightarrow_2 \text{NP VP}, \text{examined}) + (1 - \lambda_2) \times q_{ML}(\text{lawyer}|\text{S} \rightarrow_2 \text{NP VP})$$

Parsing with Lexicalized PCFGs

1. $\pi(i, j, h, X) = 0$
2. For $s = h \ldots (j-1)$, for $m = (s+1) \ldots j$, for $X(x_h) \rightarrow_1 Y(x_h)Z(x_m) \in R$,
   (a) $p = q(X(x_h) \rightarrow_1 Y(x_h)Z(x_m)) \times \pi(i, s, h, Y) \times \pi(s+1, j, m, Z)$
   (b) If $p > \pi(i, j, h, X)$,
   $$\pi(i, j, h, X) = p$$
   $$bp(i, j, h, X) = \langle s, m, Y, Z \rangle$$
3. For $s = i \ldots (h-1)$, for $m = i \ldots s$, for $X(x_h) \rightarrow_2 Y(x_m)Z(x_h) \in R$,
   (a) $p = q(X(x_h) \rightarrow_2 Y(x_m)Z(x_h)) \times \pi(i, s, m, Y) \times \pi(s+1, j, h, Z)$
   (b) If $p > \pi(i, j, h, X)$,
   $$\pi(i, j, h, X) = p$$
   $$bp(i, j, h, X) = \langle s, m, Y, Z \rangle$$

Parsing with Lexicalized PCFGs

Input: a sentence $s = x_1 \dots x_n$, a lexicalized PCFG $G = (N, \Sigma, S, R, q, \gamma)$.
Initialization:
For all $i \in \{1 \dots n\}$, for all $X \in N$,

$$\pi(i, i, i, X) = \begin{cases} q(X(x_i) \rightarrow x_i) & \text{if } X(x_i) \rightarrow x_i \in R \\ 0 & \text{otherwise} \end{cases}$$

Algorithm:

- For $l = 1 \dots (n-1)$
  - For $i = 1 \dots (n-l)$
    - Set $j = i + l$
    - For all $X \in N, h \in \{i \dots j\}$, calculate $\pi(i, j, h, X)$ using the algorithm in figure 8.

Output:

$$(X^*, h^*) = \arg \max_{S \in N, h \in \{1 \dots n\}} \gamma(X, h) \times \pi(1, n, h, X)$$

Use backpointers starting at $bp(1, n, h^*, X^*)$ to obtain the highest probability tree.