

Natural Language Processing

Problem Set 2: Analytical

Linan Qiu (lq2137)

October 5, 2014

1. PCFG Transformation to Chomsky Normal Form

a. Transformation Algorithm

The following recursive algorithm reduces rules with more than 2 non-terminals to rules with 2 non-terminals (Chomsky Normal Form)

- Let rule $A \rightarrow B_1 B_2 B_3 \dots B_n$
 - If $n = 2$ (e.g. $A \rightarrow B_1 B_2$), leave it as it is
 - If $n > 2$,
 - * Delete rule $A \rightarrow B_1 \dots B_n$
 - * Create new rule $A \rightarrow B_1 C$ where C is a newly created non-terminal. Set $P(A \rightarrow B_1 C) = P(A \rightarrow B_1 B_2 B_3 \dots B_n)$
 - * Create new rule $C \rightarrow B_2 \dots B_n$. Set $P(C \rightarrow B_2 \dots B_n) = 1$ since we are sure that this newly created rule goes only to the subsequence originally on the right
 - * Apply this algorithm on rule $C \rightarrow B_2 \dots B_n$ until $n = 2$
- Set $P(A \rightarrow B_1 B_2 B_3 \dots B_n) = P(A \rightarrow B_1 C_1) P(C_1 \rightarrow B_2 C_2) \dots P(C_{n-2} \rightarrow B_{n-1} B_n)$

b. Transforming G'

There are 3 rules we need to expand

- $S \rightarrow NP \ NP \ VP$
 - $S \rightarrow NP \ NP\text{-}VP, P = 0.3$

- NP-VP \rightarrow NP VP, $P = 1$
- VP \rightarrow Vt NP PP
 - VP \rightarrow Vt NP-PP, $P = 0.2$
 - NP-PP \rightarrow NP PP, $P = 1$
- NP \rightarrow DT NN NN
 - NP \rightarrow DT NN-NN, $P = 0.3$
 - NN-NN \rightarrow NN NN, $P = 1$

Hence the resulting non-terminal rules for grammar G is

Rule				Probability
S	\rightarrow	NP	VP	0.7
S	\rightarrow	NP	NP-VP	0.3
NP-VP	\rightarrow	NP	VP	1
VP	\rightarrow	Vt	NP	0.8
VP	\rightarrow	Vt	NP-PP	0.2
NP-PP	\rightarrow	NP	PP	1
NP	\rightarrow	DT	NN-NN	0.3
NN-NN	\rightarrow	NN	NN	1
NP	\rightarrow	NP	PP	0.7
PP	\rightarrow	IN	NP	1.0

Table 1: Non-Terminal Rules

The terminal rules are unchanged.

2. Treebank

a. PCFG

Rule				Probability
S	→	NP	VP	1
SBAR	→	COMP	S	1
VP	→	V1	SBAR	$\frac{1}{3}$
VP	→	V2		$\frac{1}{3}$
VP	→	VP	ADVP	$\frac{1}{3}$
V1	→	said		$\frac{1}{3}$
V1	→	declared		$\frac{1}{3}$
V1	→	pronounced		$\frac{1}{3}$
V2	→	snored		$\frac{1}{3}$
V2	→	ran		$\frac{1}{3}$
V2	→	swarm		$\frac{1}{3}$
ADVP	→	loudly		$\frac{1}{3}$
ADVP	→	quickly		$\frac{1}{3}$
ADVP	→	elegantly		$\frac{1}{3}$
COMP	→	that		1
NP	→	John		$\frac{1}{6}$
NP	→	Sally		$\frac{1}{3}$
NP	→	Bill		$\frac{1}{6}$
NP	→	Fred		$\frac{1}{6}$
NP	→	Jeff		$\frac{1}{6}$

Table 2: PCFG for Treebank

b. Parse Trees

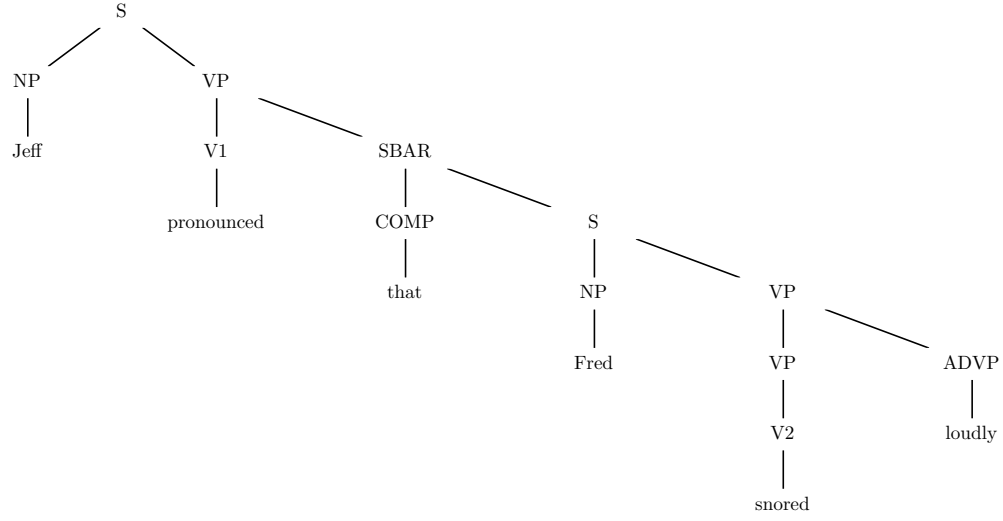


Figure 1: Parse Tree 1: Jeff accusing Fred of snoring loudly.

$$\begin{aligned}
 P_{\tau_1} &= [P(S \rightarrow NP VP) * P(VP \rightarrow V1 SBAR) * P(SBAR \rightarrow COMP S) * P(S \rightarrow NP VP) \\
 &\quad * P(VP \rightarrow VP ADVP) * P(VP \rightarrow V2) \\
 &\quad * P(NP \rightarrow Jeff) * P(V1 \rightarrow pronounced) * P(COMP \rightarrow that) * P(NP \rightarrow Fred) \\
 &\quad * P(V2 \rightarrow snored) * P(ADVP \rightarrow loudly)] \\
 &= \left[1 * \frac{1}{3} * 1 * 1 * \frac{1}{3} * \frac{1}{3} * \frac{1}{6} * \frac{1}{3} * 1 * \frac{1}{6} * \frac{1}{3} * \frac{1}{3} \right] \\
 &= \frac{1}{26244}
 \end{aligned}$$

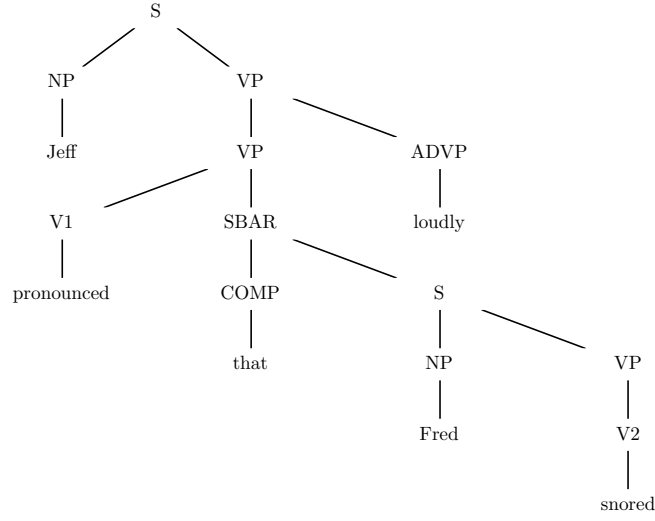


Figure 2: Parse Tree 2: Jeff saying loudly that Fred snored.

$$\begin{aligned}
P_{\tau_1} &= [P(S \rightarrow NP VP) * P(VP \rightarrow VP ADVP) * P(VP \rightarrow V1 SBAR) * P(SBAR \rightarrow COMP S) \\
&\quad * P(S \rightarrow NP VP) * P(VP \rightarrow V2) \\
&\quad * P(NP \rightarrow Jeff) * P(V1 \rightarrow pronounced) * P(COMP \rightarrow that) * P(NP \rightarrow Fred) \\
&\quad * P(V2 \rightarrow snored) * P(ADVP \rightarrow loudly)] \\
&= \left[1 * \frac{1}{3} * 1 * 1 * \frac{1}{3} * \frac{1}{3} * \frac{1}{6} * \frac{1}{3} * 1 * \frac{1}{6} * \frac{1}{3} * \frac{1}{3} \right] \\
&= \frac{1}{26244}
\end{aligned}$$

c. Modifying Non-Terminals

Modify rule $VP \rightarrow VP ADVP$ to $VP \rightarrow V2 ADVP$ such that it is impossible for ADVP to follow a verb phrase (and only follows single verbs, hence eliminating "high" attachments.)

This will allow the first parse tree in Figure 1 to retain its probability since the ADVP portion of the parse tree will simply be modified to $VP \rightarrow V2 ADVP$. However, the second parse tree in Figure 2 will have a probability of 0 since $VP \rightarrow VP ADVP$ does not exist anymore, hence the ADVP loudly cannot possibly refer to the entire VP "pronounced that Fred snored".

3. CKY Algorithm

Since the tree is balanced and since sentences are only of length $n = 2^x$, then instead of searching over all s , we can just divide the sentence length into half each time. Hence, the algorithm can be

- **Base Case:** for all $i = 1 \dots n$, for all $X \in N$

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

- **Recursive Case:**

– For all $X \in N$, calculate

$$\pi(i, j, X) = \max_{X \rightarrow YZ \in R} * \pi\left(i, \frac{j}{2}, Y\right) * \pi\left(\frac{j}{2} + 1, j, Z\right)$$

- **Return:**

$$\pi(1, n, S) = \max_{t \in \tau_G(s)} p(t)$$

This speeds up the algorithm significantly, giving it a complexity of $O(\log n |N|^3)$ (please don't kill me if I get the complexity wrong).