Natural Language Processing

Problem Set 2: Analytical

Linan Qiu (lq2137)

October 10, 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 $\mathrm{A} \rightarrow \mathrm{B}_1\mathrm{B}_2\mathrm{B}_3...\mathrm{B}_n$
  - If $n = 2$ (e.g. $\mathrm{A} \rightarrow \mathrm{B}_1\mathrm{B}_2$), leave it as it is
  - If $n > 2$,
    * Delete rule $\mathrm{A} \rightarrow \mathrm{B}_1...\mathrm{B}_n$
    * Create new rule $A \rightarrow \mathrm{B}_1\mathrm{C}$ where C is a newly created non-terminal. Set $\mathrm{P}(\mathrm{A} \rightarrow \mathrm{B}_1\mathrm{C}) = P(\mathrm{A} \rightarrow \mathrm{B}_1\mathrm{B}_2\mathrm{B}_3...\mathrm{B}_n)$
    * Create new rule $\mathrm{C} \rightarrow \mathrm{B}_2...\mathrm{B}_n$. Set $P(\mathrm{C} \rightarrow \mathrm{B}_2...\mathrm{B}_n) = 1$ since we are sure that this newly created rule goes only to the subsequence originally on the right
    * Apply this algorithm on rule $\mathrm{C} \rightarrow \mathrm{B}_2...\mathrm{B}_n$ until $n = 2$
- Set $P(A \rightarrow \mathrm{B}_1\mathrm{B}_2\mathrm{B}_3...\mathrm{B}_n) = P(A \rightarrow \mathrm{B}_1\mathrm{C}_1)P(\mathrm{C}_1 \rightarrow \mathrm{B}_2\mathrm{C}_2)...P(\mathrm{C}_{n-2} \rightarrow \mathrm{B}_{n-1}\mathrm{B}_n)$

b. Transforming G'

There are 3 rules we need to expand

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

- NP-VP → NP VP, $P = 1$

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

Hence the resulting non-terminal rules for grammar G is

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

Table 1: Non-Terminal Rules

The terminal rules are unchanged.

2. Treebank

a. PCFG

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

Table 2: PCFG for Treebank

b. Parse Trees

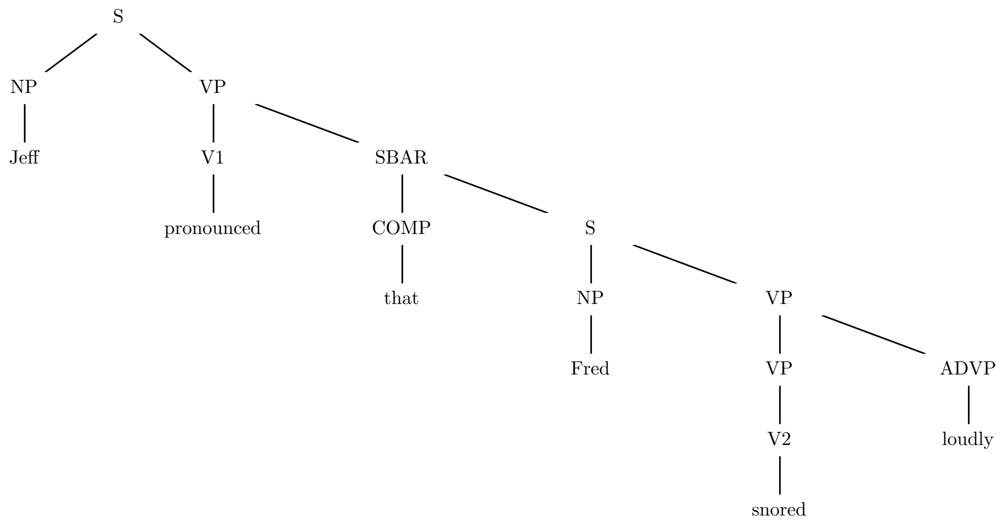

Figure 1: Parse Tree 1: Jeff accusing Fred of snoring loudly.

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

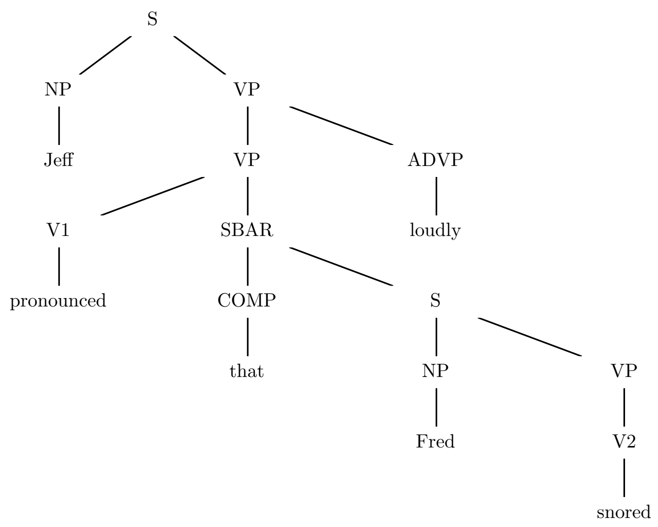

Figure 2: Parse Tree 2: Jeff saying loudly that Fred snored.

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

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 $i = 1...n$, for all $X \in N$

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

- **Recursive Case:**
  - For all $X \in N$, calculate

$$\pi(i, j, X) = \max_{X \to Y Z \in R} \left[q(\mathrm{X} \to \mathrm{Y\ Z}) * \pi\left(i, \frac{j}{2}, Y\right) * \pi\left(\frac{j}{2} + 1, j, Z\right)\right]$$

- **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).