PCFG algorithms

A. Overview

(1) a. PCFGs
b. CNF
c. Chart parsing
d. Forward probabilities
e. Backward probabilities
f. Inside-outside algorithm

B. What is a PCFG?

(2) Sentence probability: \( p(s) = \sum_j p(t_j)p(s|t_j) \)

(3) \[
\begin{align*}
\text{NP} & \rightarrow \text{NP C NP} \quad .4 \\
\text{NP} & \rightarrow \text{Mary} \quad .3 \\
\text{NP} & \rightarrow \text{Mindy} \quad .2 \\
\text{NP} & \rightarrow \text{Mark} \quad .1 \\
\text{C} & \rightarrow \text{and} \quad 1 \\
\end{align*}
\]

(4) The probability of each parse is: 
\(.3 \times .2 \times .1 \times 1 \times 1 \times .4 \times .4 = .00096\). The overall probability of the string is 
\(.00096 + .00096 = .00192\).

(5) 

```
NP
   /   \
  NP   C
     /   \
  Mary and NP
       /   \
      NP   C
        /   \
       NP   NP
         /   /\
        NP Mindy and Mark
```
C. Chomsky-normal form

(7) Any context-free grammar can be expressed as in Chomsky-normal form (CNF) without any change in expressive power.

(8) Chomsky-Normal Form
   a. $A \rightarrow BC$, where $A$, $B$, and $C$ are non-terminal symbols.
   b. $A \rightarrow a$, where $A$ is a non-terminal and $a$ is a single terminal.

(9) Demonstrating the inside-outside algorithm is much easier with a CFG in CNF.

D. Chart parsing

(10) How do we find whether a sentence is legal with respect to some CFG?

(11) This is a hard problem because, in principle, we must keep track of every possible application of every rewrite rule as we proceed from left to right.

(12) *The horse raced past the barn fell.*

(13) Chart parsing (dynamic programming) is an efficient way to do this.

(14) Chart parsing
   Keep track of all successful rewrite rule applications in a chart as you parse left to right.
(15) \[
\begin{array}{c}
Mary \\
NP
\end{array} \rightarrow \begin{array}{c}
Mary \\
NP
\end{array} \quad \begin{array}{c}
and \\
C
\end{array} \rightarrow \begin{array}{c}
Mary \\
NP
\end{array} \quad \begin{array}{c}
and \\
Mindy \\
C \\
NP
\end{array} \rightarrow \begin{array}{c}
and \\
NP
\end{array}
\]

\[
\begin{array}{c}
Mary \\
NP
\end{array} \rightarrow \begin{array}{c}
NP
\end{array} \\
\begin{array}{c}
and \\
C
\end{array} \rightarrow \begin{array}{c}
NP
\end{array} \\
\begin{array}{c}
Mindy \\
NP \\
and
\end{array} \rightarrow \begin{array}{c}
NP
\end{array}
\]

\[
\begin{array}{c}
Mary \\
NP
\end{array} \rightarrow \begin{array}{c}
NP
\end{array} \\
\begin{array}{c}
and \\
C
\end{array} \rightarrow \begin{array}{c}
NP
\end{array} \\
\begin{array}{c}
Mindy \\
NP \\
and
\end{array} \rightarrow \begin{array}{c}
NP
\end{array} \\
\begin{array}{c}
and \\
Mark
\end{array} \rightarrow \begin{array}{c}
NP
\end{array}
\]

(16) Why is this efficient?
   a. We do not keep track of all complete parses.
   b. Once we know a rule can succeed, we record that success for all future parses.

E. Inside and outside probabilities

(17) Backward probability = “inside” probability
\[
\beta_j(k, l) \overset{\text{def}}{=} p(w_{k,l} | N^j_{k,l})
\]

(18) Base case for inside probability for PCFG in CNF
\[
\beta_j(k, k) = p(w_k | N^j_{k,k}) = p(N^j \rightarrow w_k)
\]

(19) Recursive case for inside probability for PCFG in CNF
\[
\beta_j(k, l) = \sum_{p,q,m} p(N^j \rightarrow N^p N^q) \beta_p(k, m) \beta_q(m + 1, l)
\]

(20) Forward probability = “outside” probability
\[
\alpha_j(k, l) \overset{\text{def}}{=} p(w_{1,k-1}, N^j_{k,l}, w_{l+1,n})
\]

(21) Base case for outside probability for PCFG in CNF
\[
\alpha_1(1, n) = p(N^1_{1,n}) = 1
\]

(22) Recursive case for outside probability for PCFG in CNF
\[
\alpha_j(k, l) = \sum_{h,p,q} \alpha_p(h, l) p(N^p \rightarrow N^q N^j) \beta_q(h, k - 1) + \sum_{m,p,q} \alpha_p(k, m) p(N^p \rightarrow N^j N^q) \beta_q(l + 1, m)
\]
F. Training a PCFG

\[ |N^j \rightarrow N^p N^q| \overset{\text{def}}{=} \frac{1}{p(w_1,w)} \sum_{k,l,m} \alpha_j(k, l) \ p(N^j \rightarrow N^p N^q) \ \beta_p(k, m) \ \beta_q(m + 1, l) \]

References


