Introduction to Computational Learning Theory - Problem Set #2

Emma Ziegellaub Eichler - edz2103@columbia.edu

October 12, 2012

Problem 1

a

Given PCFG $G' = (N, \Sigma, R, S, q)$ in the form described, we will transform it into an equivalent PCFG G in Chomsky normal form leaving Σ and S as is and modifying N, R, and q as follows. For all rules in G', $r_1, ..., r_m \in R$, that do not satisfy Chomsky normal form (i.e., of the form $r_i = X \rightarrow Y_1Y_2...Y_n$, $n > 2$), we remove r_i from the set of rules and construct $n-1$ rules satisfying Chomsky normal form (i.e., of the form $r' = Y_1Y_2$) to replace it: $r_{i1} = X \rightarrow Y_1N_1$, $N_i \rightarrow Y_{i+1}N_{i+1} \forall 1 \leq i \leq n-1$, $N_{n-2} \rightarrow Y_{n-1}Y_n$, where $N_1, ..., N_{n-2} \notin N$ are new non-terminal symbols. Note that we can still generate $Y_1Y_2...Y_N$ from X and that we cannot generate anything but $Y_1Y_2...Y_N$ from $X \rightarrow Y_1N_1$ since each new non-terminal N_i only has one rule associated with it. Thus G and G' generate the same sentences, and it only remains to be shown that they do so with the same probability. Since there is only one rule $N_i \rightarrow AB \forall i$, the probability $q(N_i \rightarrow Y_{i+1}N_{i+1}) = 1$. Since we have not changed the number of rules $X \rightarrow AB$, the probability $q(X \rightarrow Y_1N_1) = q(X \rightarrow Y_1Y_2...Y_n)$. Suppose the probability of $q(X \rightarrow Y_1Y_2...Y_n) = x$ in G'. Then the probability of generating $Y_1Y_2...Y_N$ from X in G is $q(X \rightarrow Y_1N_1)(\Pi_{i=1}^{n-2} q(N_i \rightarrow Y_{i+1}N_{i+1}))q(N_{n-2} \rightarrow Y_{n-1}Y_n) = (x)(\Pi_{i=1}^{n-2} 1)(1) = x$. $\square$

b

Following the transformation described in part (a), we generate:
$S \rightarrow NP\ VP$ 0.7
$S \rightarrow A\ VP$ 0.3
$A \rightarrow NP\ NP$ 1.0
$VP \rightarrow Vt\ NP$ 0.8
$VP \rightarrow B\ PP$ 0.2
$B \rightarrow Vt\ NP$ 1.0

$NP \to NP\ PP$ 0.7
$NP \to C\ NN$ 0.3
$C \to DT\ NN$ 1.0
$PP \to IN\ NP$ 1.0
where A, B, C are the new non-terminals. All rules of the form $X \to x$, $x \in \Sigma$ remain unchanged. $\square$

2

a

$S \to NP\ VP$ 1.0
$VP \to V1\ SBAR$ $0.\bar{3}$
$VP \to V2$ $0.\bar{3}$
$VP \to VP\ ADVP$ $0.\bar{3}$
$SBAR \to COMP\ S$ 1.0
$NP \to John$ $0.1\bar{6}$
$NP \to Sally$ $0.\bar{3}$
$NP \to Fred$ $0.1\bar{6}$
$NP \to Bill$ $0.1\bar{6}$
$NP \to Jeff$ $0.1\bar{6}$
$V1 \to said$ $0.\bar{3}$
$V1 \to declared$ $0.\bar{3}$
$V1 \to pronounced$ $0.\bar{3}$
$COMP \to that$ 1.0
$V2 \to snored$ $0.\bar{3}$
$V2 \to swam$ $0.\bar{3}$
$V2 \to ran$ $0.\bar{3}$
$ADVP \to loudly$ $0.\bar{3}$
$ADVP \to quickly$ $0.\bar{3}$
$ADVP \to elegantly$ $0.\bar{3}$
$\square$

b

□

c

The "high" attachment problem is caused by the rule $VP \rightarrow VP\ ADVP$. $ADVP$ can only directly generate an adverb, but VP can either generate a verb or another verb phrase, in which case the adverb will modify the "higher" verb. To fix this, we need only remove the rule $VP \rightarrow VP\ ADVP$ and replace it with the rule $VP \rightarrow V2\ ADVP$, assigning the probability of the old rule to the new rule. Now we can only generate adverbs directly adjacent to the verbs they are modifying. Alternatively and equivalently, we can replace the removed rule with four new rules, introducing two new non-terminals, H (for high attachment) and L (for low attachment): $VP \rightarrow H$ $ADVP$, $VP \rightarrow L$ $ADVP$, $L \rightarrow V2$, and $H \rightarrow VP$. We assign $q(L \rightarrow V2) = 1$, $q(H \rightarrow VP) = 0$, and $q(VP \rightarrow H\ ADVP) = q(VP \rightarrow L\ ADVP) = q(VP \rightarrow VP\ ADVP)/2$. Now while a rule exists to generate "high" attachments, it has zero probability and will never occur; we always use the rule to generate "low" attachments. $\square$

3

Dynamic programming algorithm to return the maximum probability for any left-branching tree underlying a sentence $x_1, x_2, ..., x_n$.
Base case: $\forall i, 1 \leq i \leq n$, $\forall X \in N$
  if $X \rightarrow x_i \in R$, $\pi(i, i, X) = q(X \rightarrow x_i)$
  if $X \rightarrow x_i \notin R$, $\pi(i, i, X) = 0$
Recursive case: $\forall (i, j), 1 \leq i < j \leq n$, $\forall X \in N$
  $\pi(i, j, X) = max_{X \rightarrow YZ \in R}(q(X \rightarrow YZ) * \pi(i, j-1, Y) * \pi(j, j, Z))$
Return: $\pi(1, n, S) = max_{t \in T_G(s)} p(t)x$
Note that the only change from the original algorithm (used to find the maximum probability tree, left-branching or not) is that instead of allowing s to vary from 1 to $j-1$, we fix it at $j-1$. This is because in a left-branching tree, whenever a rule of the form $X \rightarrow YZ$ occurs, non-terminal Z must directly dominate a terminal symbol; i.e., can generate only one word in the sentence. $\square$