Computational Linguistics 1 - Fall 2011
CMSC/LING723, LBSC 744
Homework 4 - Due 3 November 2011

Submission Guidelines

1. Print out any written portion of the assignment and submit in class on November 3, *as well as* emailing your writeup, in .pdf format, to `compling723.fall2011@gmail.com`.
2. Any code you write for this homework should also be submitted to `compling723.fall2011@gmail.com`.
3. Be sure include your full name and e-mail in your submission, and clearly indicate each problem number in your solution set.

Data

I've provided a new version of our favorite corpus, f2-21.train.parse and f2-21.test.parse (available from the class mailing list). These files contain the parse trees from the Penn WSJ Treebank, split into our usual train and test sets.

Please note! For our work on parsing (this homework and next), we will be ignoring the lexical items in the tree. I have left these items in the parse files, because I think it makes the parse a little more interesting to read. Before you start coding, however, please replace the lexical items with lower-cased representation of their POS tag, to reduce memory requirements on our parsers. You can do so with the following sed command:
cat f2-21.train.parse | sed 's/(\([^]*\) \([^()]*\))/(\1 W\L\1)/g' > f2-21.train.parse.noLEX
Please do so for each of your parse files *before* beginning Problem 3.

Problem 1 (5 points)

I've removed the unary productions from the trees for you, and I've done so by simply replacing any unary production A → B with B. So, for example, the original tree:
(TOP (S (NP (PRP we)) (VP (VBD helped))))
became:
(TOP (S (PRP we) (VBD helped)))
by replacing NP → PRP with PRP, and replacing VP → VBD with VBD.

Does this seem like a reasonable way to remove unary productions? Can you think of a "better" way to transform unaries? Tell me why it would be better, discuss any possible ramifications it might have on your probability distribution, and give an example using your unary transformation process.

Problem 2 (15 points)

(a) Draw the parse (which is the fourth parse in our training corpus):
(TOP (S (NP (DT The) (NN luxury) (NN auto) (NN maker))
(NP (JJ last) (NN year))
(VP (VBD sold) (NP (CD 1,214) (NNS cars))
(PP (IN in) (NP (DT the) (NNP U.S.))))))
as a tree, similar to slide 14 from lecture 15.

(b) For the same parse given above, write each of the context-free rules that will be extracted from this parse.

(c) Left-factor the context-free rules from (b) into Chomsky Normal Form (CNF), following the example on slide 20 of lecture 15. Make an exception for the rule with "TOP" on its left-hand side, which does not need to be put into CNF. Write each of the resulting left-factored rules.

Problem 3 (30 points)

Induce a probabilistic context-free grammar from the training set, using relative frequency estimation for conditional probabilities, i.e., $\text{P(RHS} \mid \text{LHS)} = \frac{\text{count(LHS}\rightarrow\text{RHS)}}{\text{count(LHS)}}$.

(a) How many rules are in your grammar? (not including POS → Wpos rules)

(b) Print out all rules with either "ADVP" or "NAC" on the left-hand side, with their relative frequency. For example:

ADJP → ADVP JJ	=	0.006371150
ADJP → ADJP , CONJP JJ , VP	=	0.000271113
NAC → CD NN PP	=	0.003278690
NAC → NNP PP	=	0.222951000

Problem 4 (30 points)

Left-factor the grammar from Problem 3 into CNF (you may either do this by tranforming the treebank itself following slide 20 of lecture 15, or by factorizing the context-free rules themselves following slide 22 of lecture 15). Rules with "TOP" on the left-hand side are an exception to the rule, and do not need to be put into CNF.

(a) How many rules are in your left-factored grammar? (not including POS → Wpos rules)

(b) Print each left-factored rule with either "ADVP", "NAC", "ADVP-something", or "NAC-something" on the left-hand side, with its relative frequency. For example:

ADJP → ADVP JJ	=	0.006371150
ADJP∼ADJP∼, → CONJP ADJP∼ADJP∼,∼CONJP	=	0.071428600
NAC∼CD → NN PP	=	1.0
NAC → NNP PP	=	0.222951000