HOMEWORK 6

In this assignment you will implement the probabilistic version of CKY and parse a small corpus of sentences. You should write a program called "cky.py" to be run as follows:

```
python cky.py GRAMMAR_FILE SENTENCE_FILE
```

Look in the provided english_cnf.gr and sentences.sen files for the formats of the grammar and sentences files, respectively. The overall format of the grammar is the same as in Homework 5. However, the grammar is in Chomsky Normal Form, and the intended interpretation of the weights of the rules is different. In this assignment, you should treat the weights as a penalty. As we've seen in previous assignments, actually multiplying probabilities in your code is a bad idea due to round-off error. Rule probabilities of a PCFG can be converted to weights such as these by taking the negative log of the probabilities. In this case, we sum the weights rather than multiplying (since we're dealing with logs) and we want the parse with the lowest total weight (since we're taking the negative).

PROGRAMMING (60 POINTS)

There is only one programming component of the assignment: to implement probabilistic CKY.

I suggest approaching the programming in pieces (as usual). Start by implementing the non-probabilistic version of CKY without backpointers. Then add the backpointers and functionality for printing the set of resulting trees (those with ROOT as the top node). Finally, add the probabilistic component and print out the set of trees and their total weights. **The program should print out ALL the valid trees and their weights rather than the single best parse.**

The provided files, test.sen and test.out, give you sample input and output. When run on test.sen, your parser should print output like that shown in test.out.

Coding suggestions:

- Represent your grammar so it is easy to access rules given the *right hand side* of the rule. You should do this so that you can quickly check for a cell at [i,j] whether the grammar has a rule that combines a nonterminal at [i,k] with a nonterminal at [k,j].
- Write CKY as a function that takes the grammar and the sentence as arguments and returns the completed chart.
- Write a function that prints out the trees given the completed chart as an argument.
- Each cell in the CKY chart should be a list of nodes
- Each node is a collection of pieces: the probability, the nonterminal symbol, and the backpointers. You may find it handy to represent each node as a tuple with components for each of these pieces. This means what gets stored in each cell is a list of tuples.

You may find it helpful to use the NLTK tree drawing module from HW 5 to process your output so that the trees are easier to read, but be sure to hand in the actual output of your program.

HOMEWORK 6

After completing your programming and running your parser on sentences.sen, answer the following questions.

QUESTIONS

Q1 (10 POINTS)
For the sentence "every fine fly and monkey want to understand him .", there are two parses with the same weight. What meanings do these parses correspond to? Why does the grammar assign the same weight to the two parses? Could you change the weights in this grammar to prefer one of the parses over the other? Explain. Be specific!

Q2 (10 POINTS)
For the sentence "the monkey think -ed that the president want -ed his pickle on his sandwich .", there are three parses. Explain what meaning each of these parses corresponds to. Are any of these parses the one you get when you read the sentence? Explain.

Q3 (20 POINTS)
How does this grammar prefer to parse the PP in sentences of the form NP V NP PP?

a) Construct a sentence your grammar can generate of this form and parse it. Save this sentence as Q3.sen and your parser's output on this sentence as Q3.out.
b) Why does the grammar prefer this parse? (hint: consider the rules involved)
c) Keeping the rules the same, change the weights in the grammar so the other parse is preferred for your sentence. What rule weights did you have to change and how? Save your updated grammar as Q3.gr.
d) Consider the meanings of sentences like "I ate soup with a spoon" and "I kissed the puppy with a tan coat". Why are both grammars (the original one and your modified one) inadequate for parsing sentences like these? Be specific!

WHAT TO SUBMIT

Submit your cky.py script, your output on sentences.sen (e.g. sentences.out), Q3.sen, Q3.out, Q3.gr, and your answers to the questions above.