Practice Exam Problems: Parsing

Natural Language Processing (JHU 601.465/665)

Prof. Jason Eisner

1. Recall that a grammar is in Chomsky Normal Form (CNF) iff every rule has the form $X \to a$ or $X \to YZ$, where lowercase a denotes a terminal and capital X, Y, Z denote nonterminals. Every nonterminal node in a parse tree therefore has 1 or 2 children.

   A grammar is in Frankenstein Form (FF) iff every rule has the form $X \to a$ or $X \to YaZ$. Every node in a parse tree therefore has 1 or 3 children.

   (a) [7 points] Pseudocode for the CKY recognition algorithm is shown in Figure 1. Modify it to handle FF grammars instead of CNF grammars. (Show your changes directly on Figure 1 on page 2.)

   (b) [3 points] Instead of changing the recognizer algorithm, you could just change the grammar. Given an FF grammar F, how could you convert it to an *equivalent* CNF grammar C?

   (A sentence should be grammatical under F if and only if it is grammatical under C—and you can check *that* using CKY on C.)

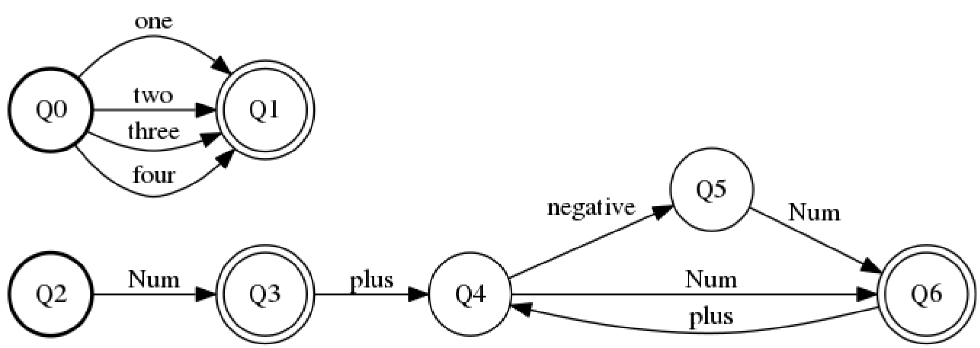
```
RECOGNIZE(Input, Nonterminals, Rules):
n := length(Input)
Chart := a new array with all entries initialized to false
for i := 1 to n
    foreach X ∈ Nonterminals
        if (X → Input[i]) ∈ Rules
            then Chart[i − 1, i, X] := true
for width := 2 to n
    for start := 0 to n − width
        let end := start + width
        for mid := start + 1 to end − 1
            foreach (X → Y Z) ∈ Rules
                if Chart[start, mid, Y] and Chart[mid, end, Z]
                    then Chart[start, end, X] := true
return Chart[0, n, ROOT]
```

Figure 1: The CKY recognition algorithm, to be modified for question 1.

2. [9 points] The context-free grammar formalism can be augmented so that the right-hand side of each rule is not necessarily a string but could be any regular expression, as in this little grammar:[1]

$$
\begin{aligned}
\text{Num} &\rightarrow \text{one} \mid \text{two} \mid \text{three} \mid \text{four} \\
\text{Root} &\rightarrow \text{Num (plus (negative)? Num)*}
\end{aligned}
$$

The right-hand sides of these rules compile into the following DFAs (deterministic finite-state automata):



In class, we discussed how to modify Earley's algorithm to run on this kind of grammar. This involved generalizing the notion of "dotted rule." Suppose the input to the modified Earley's algorithm is the grammatical sentence

one plus negative four plus two

(a) List all entries that appear in column 2 of the parse table. (*Hint:* At this point, the system has consumed the input *one plus* and is preparing for what might come next.)

(b) Now list all entries that appear in column 3.

[1] As in question 1, capitalized identifiers denote nonterminals. The ? suffix denotes optionality while the * suffix denotes repetition and the | operator denotes disjunction.

3. [5 points] The "inside algorithm" is the variant of CKY, discussed in lecture, that keeps track of a *total* probability for each entry in the chart.

   Suppose it computes a probability of 10^{-32} for "NP from 3 to 7" and a probability of only 10^{-33} for "VP from 3 to 7." Does this imply that the correct parse tree (the one that the speaker used to produce the sentence) is more likely to contain an NP than a VP from 3 to 7? *(circle the single best answer)*

   (a) Yes.

   (b) No, because these probabilities can change as the algorithm continues.

   (c) No, because there is no guarantee that the correct parse contains any constituent at all from 3 to 7.

   (d) No, because the inside probability 10^{-32} does not represent the probability that the sentence contains an NP from 3 to 7.

   (e) No, because the inside probability is about the sum of many parses, not about the highest-probability parse.

4. You have a probabilistic context-free grammar of English, G, in which the probability of each rule is given by a conditional log-linear model.

   (a) [3 points] For example, the probability of the rule `S[head=eat,tense=past]` $\rightarrow$ `NP[head=lion,num=plural] VP[head=eat,num=plural,tense=past]` is given by

   $$p(\underline{\hspace{3cm}} \mid \underline{\hspace{3cm}}) = \underline{\hspace{12cm}}$$

   Fill in the blanks to show that you know how to define a conditional log-linear model. Assume that you are given feature functions $f_1, f_2, \ldots, f_K$ and a parameter vector $\vec{\theta}$; you don't have to define those.

   Some of your input sentences are not in English and you would rather not waste time parsing them with your English grammar G.

   To detect quickly that a sentence will have a very low probability under G, you could first try parsing it with a "coarsened" version of the grammar, G'.

   You will derive G' from G by simplifying the nonterminals by throwing away all their attributes. Thus, G' has simplified rules like `S` $\rightarrow$ `NP VP`.

(b) [3 points] The idea is that it's much faster to parse with the simple G' than with G. What is the worst-case runtime of CKY as a function of the number of nonterminals V, for a fixed-length sentence?

$$O(\underline{\hspace{4cm}})$$

(c) [3 points] Your idea is that if the sentence has a very low probability p under the coarse grammar G', such as $p = 10^{-34567}$, then you can conclude that it will also have probability $\leq p$ under the original grammar G.

For this conclusion to be valid, how do you need to define the probabilities of rules in G', such as $\texttt{S} \rightarrow \texttt{NP VP}$?

(d) [2 points] An alternative way to define the probability of a rule in G' is to use the same log-linear model as you used for G, but to ignore all features that look at the attributes.

Suppose the attribute-specific features all have *negative* weights (e.g., they serve to discourage bad attribute combinations rather than encouraging good attribute combinations). Does this imply that the sentence will have an even lower probability under G than it did under G', as desired by the previous question? Explain.