Assignment 4
CSEP 517: Natural Language Processing

University of Washington

Due: May 14, 2017

1 Exploring Existing Parsers (30%)

In this part of the assignment, you will run existing PCFG and dependency parsers and try to find some errors that they make. You can use whichever parsers you want, but here are some options that include a web interface for your convenience. It's also perfectly fine to download and install a parser.

Parsers with web interface:

- http://tomato.banatao.berkeley.edu:8080/parser/parser.html
- http://nlp.stanford.edu:8080/parser/
- http://demo.ark.cs.cmu.edu/parse
- https://spacy.io/demos/displacy
- http://lil.cs.washington.edu/easysrl/demo.cgi

These resources might be useful in decoding the part-of-speech tags, nonterminal labels, and dependency labels:

- http://cs.jhu.edu/~jason/465/hw-parse/treebank-notation.pdf
- http://www.surdeanu.info/mihai/teaching/ista555-fall13/readings/PennTreebankConst
  html
- http://nlp.stanford.edu/software/dependencies_manual.pdf

Keep in mind that you can complete this assignment without having to know all of these labels in detail! Start with simple sentences like "*my dog ate my homework*" or "*fruit flies like bananas*." Do the results seem reasonable to you? Try some longer sentences as well, e.g., "*two roads diverged in a wood, I took the one less traveled by, and that has made all the difference.*" You might notice that the output of different parsers is based on different conventions, label sets, etc. These details shouldn't matter for the purpose of this assignment.

Deliverables

- Explore at least two different parsers; one should generate some variation of phrase structure, and another should generate some variation of dependency structure.
- Find four different sentences whose parses that you determine as incorrect. Specify which parser gave each incorrect parse (please cite it properly, not just a URL, but a full citation). Explain why you think the parse is incorrect. Your four sentences should show different types of errors. You don't have to identify *all* errors in those sentences. It suffices to identify *some* errors. You must come up with your own unique example sentences. If we find your sentences are identical to others', we will invite you to our office to parse some very long sentences.

2 Variations to the CKY Algorithm (40%)

Recall the CKY algorithm, given here as equations for the scores in the chart (the back pointers are implicit). Base case: for $i \in \{1, \ldots, n\}$ and for each $N \in \mathcal{N}$:

$$s_{i:i}(N) = p(x_i \mid N) \tag{1}$$

For each i, k such that $1 \leq i < k \leq n$ and each $N \in \mathcal{N}$:

$$s_{i:k}(N) = \max_{L,R \in \mathcal{N}, j \in \{i,\ldots,k-1\}} p(L\ R \mid N) \cdot s_{i:j}(L) \cdot s_{(j+1):k}(R) \tag{2}$$

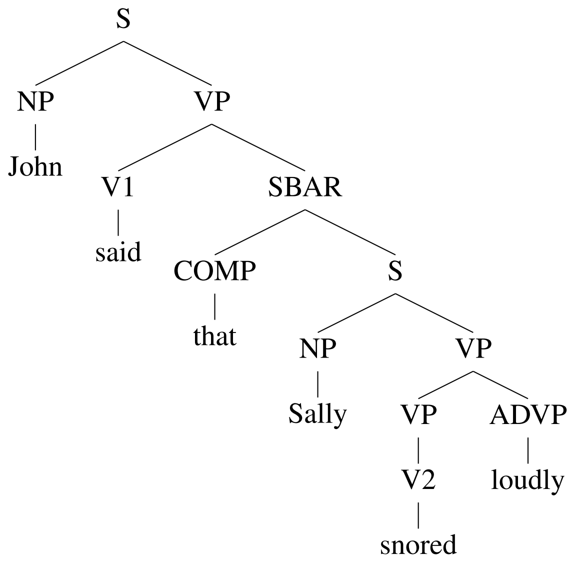
1. You find Chomsky normal form inelegant and propose to build a new parser that can also handle context-free rules with *three* nonterminals on the righthand side (in addition to the CNF rules). Write the recursive equation for a modified CKY algorithm that can handle these rules (in the style of Eq. 2). (You may be tempted to convert the three-child rules into two binary rules. Do not do this. Instead, modify the algorithm.) The base case is identical to Eq. 1.
2. Noting the incredible success of parent annotation [Johnson, 1998], you decide to derive a CKY-like algorithm that works with $p(\text{children} \mid N, G)$, where G is the parent of N, as opposed to the original form, $p(\text{children} \mid N)$. (For this problem, go back to the original CNF rules, no three-child rules.) Write the recursive equation for a modified CKY algorithm that can handle these probabilities (in the style of Eq. 2). Here is the base case:

$$\forall i \in \{1, \ldots, n\}, \forall N, G \in \mathcal{N}, \quad s_{i:i}(N^G) = p(x_i \mid N, G)$$

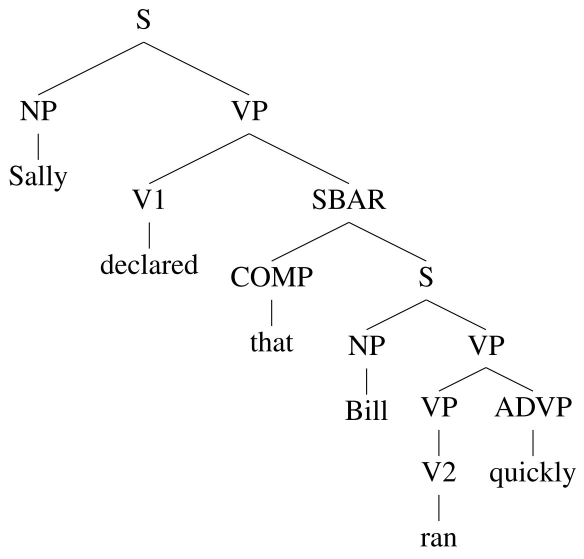
3. Bonus: what are the $O(\cdot)$ runtime and space requirements of your two new algorithms, as a function of the sentence length n and the number of nonterminals $|\mathcal{N}|$? Explain how you derived these expressions.

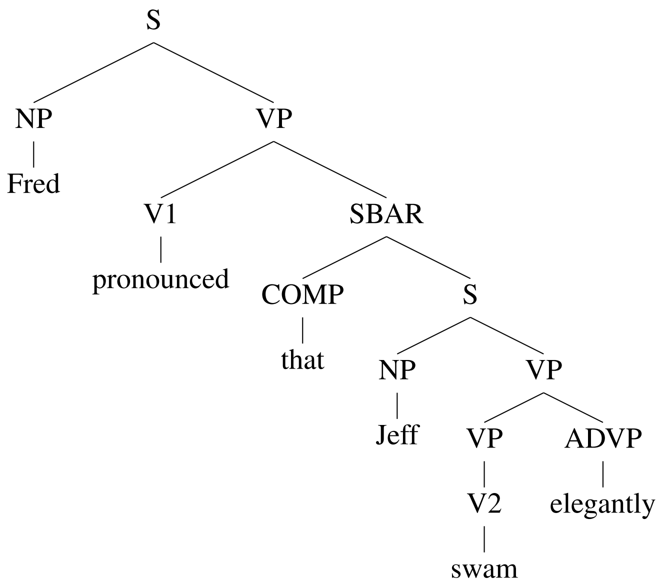
3 Grammar Refinement (30%)

Your friend decides to open a new business and treebank[1] corpora professionally. After much agony, he finally produces a corpus which contains the following three parse trees:



[1] Yes, you can verb *treebank*.





You then purchase this treebank for $2.99, and you decide to build a PCFG and a parser.

Deliverables

- Show the PCFG (CFG with rule probabilities) that you would derive from this treebank if you used relative frequency estimation (no smoothing).
- Consider the sentence "*Jeff pronounced that Bill ran elegantly*." List *all* of the parse trees and their probabilities (not just the best one). Note that you cannot use CKY as described to find the parse trees, because they are not in CNF, but you should still be able to work out what they are.
- You are shocked and dismayed that "*Jeff pronounced that Bill ran elegantly*" has two possible parses, and that one of them—that Jeff is doing the pronouncing elegantly—has relatively high probability, in spite of the fact that such "high" attachment of ADVP (i.e., an ADVP being the child of such an early VP / high in the tree) has never been seen in the corpus. Since your friend won't accept returns, you decide to fix the treebank yourself by refining the grammar, i.e., altering some nonterminal labels in the corpus. Show one such transformation to the grammar that would give zero probability (under relative frequency estimation) to parse trees with high attachments. (Hint: consider the options you've seen in

the class—parent annotation, markovization, label splitting, lexicalization—or propose a new one). While it is not necessary to enumerate all production rules after the grammar refinement, you must (1) spell out at least a subset of the rules with their corresponding probabilities that can clearly demonstrate what systematic changes you are making to the grammar, and (2) explain clearly how that change can ensure zero probability to the high attachment of ADVP.

- You get excited about building new treebanks and decide to build a specialized one to focus on the prepositional phrase attachment problem (see, for example, the "elephant in my pajamas" example in the lecture slides). To address such ambiguities, you decide to annotate many more sentences with similar ambiguities, for example:

  - I fixed a bug in my code.
  - I fixed a bug in my dream.
  - I cleaned dishes in my pajamas.
  - I cleaned dishes in the sink.
  - I ate noodles with tofu.
  - I ate noodles with chopsticks.

  Would parsing with the default PCFG (using, say, Earley's algorithm, which doesn't require CNF), if provided with a sufficient amount of annotated trees, learn to disambiguate the correct structure successfully? Explain why or why not. If not, then discuss grammar refinements that might help and explain why.

Submission Instructions

- **Code**: You should have noticed by now that this assignment has no code!
- **Report** (use the filename `A4.pdf` and upload to Canvas): Your writeup should be two to three pages long, or less, in pdf (one-inch margins, reasonable font sizes, preferably LaTeX-typeset). Part of the training we aim to give you in this class includes practice with technical writing. Organize your report as neatly as possible, and articulate your thoughts as clearly as possible. We prefer quality over quantity. Do not flood the report with tangential information. Similarly, when discussing the experimental results, do not copy and paste the entire system output directly to the report. Instead, create tables and figures to organize the experimental results.
- It is not our intention to have you spend many hours on LaTeX-typesetting of charts and trees. Feel free to use some other tool to draw trees (e.g., neatly hand-drawn and scanned, then included as an image, or drawn using a graphics editing tool). The `graphicx` package and `\includegraphics` command will help you tremendously.

References

Mark Johnson. PCFG models of linguistic tree representations. *Computational Linguistics*, 24(4):613–32, 1998.