# CSE 390 Assignment 3: PCFG Parsing

Due: Apr 20, 2017 by 11:59 PM EST

In this assignment you will implement a PCFG parser (almost) from scratch.

## 1 PCFG Induction (50 points)

You will induce the PCFG from the training data file (train.trees)[1]. The trees have already been binarized and are in Chomsky Normal form. The words that have appeared less than 2 times have been mapped to $\langle unk \rangle$ symbol in training and test. You don't need to do any special handling of $\langle unk \rangle$ symbols. Recall, PCFGs contain rules and the associated probabilities.

You will compute MLE probabilities for the observed rules in training data:

$$Pr(\alpha \rightarrow \beta_1\beta_2|\alpha) = \frac{\#(\alpha \rightarrow \beta_1\beta_2)}{\#(\alpha \rightarrow *)}$$

**A note on smoothing**: There are two types of rules:

1.Unary rules involving lexical items (e.g, VBZ → eats).

2.Binary rules involving POS or syntactic categories (e.g. NP → DT NN).

You only need to smooth the unary rules, which involve words. Use Laplace smoothing. For the binary rules use MLE probabilities.

Write down the grammar to a file. This file can be in any format. It just needs to be in a format that your CYK parser can use. In your report please list the ten most frequent rules in the training data along with their frequencies.

## 2 CYK Parser (30 points)

You will build a PCFG parser using the CYK algorithm we saw in class. You need to extend the algorithm to retain the best parse at each cell for each constituent type. For

[1] The data for this exercise is from Prof. David Chiang's NLP course while he was at ISI.

example let's say a span C[i,j] may be parsed as a NP or a VP. There are multiple ways in which C[i,j] could be parsed as an NP and multiple ways for it to be parsed as a VP. You need to score every such possibility but only retain the best possible way for C[i,j] to be parsed i) as an NP and ii) as a VP.

Each possible parse of a span is scored by multiplying the scores of the sub-parses (stored in the corresponding cells) and the score of the rule that applies. For e.g., score of a span $S_{ij}$ with a split point $k$, $C_k[i,j] = C[i,k] * C[k+1,j] * Pr(r)$, where rule $r$ is the rule that combines the phrases for spans $S_{ik}$ and $S_{k+1j}$. Again as before, you can use log transformation (i.e, sum of log probabilities) to avoid underflow.

**Pseudo-code and Details for Implementation** You are free to use any online source for finding a suitable guide for implementing the algorithm. If you want a guide for the implementation, you can follow the pseudo-code here:

https://www.usna.edu/Users/cs/nchamber/courses/nlp/s15/labs/cky-pseudo.html

Do not use existing code either directly or indirectly to guide your implementation.

## 3 Evaluation (20 points)

The output of your program should be in the same format as the train.trees and test.trees files provided. You will be provided with a script `evalb.py` which will evaluate the output of your program (it will calculate Precision, Recall, and F1).

You will evaluate the performance of your parser on the test.txt sentences. The generated trees will be compared with the test.trees file. To evaluate your program run the following:

```
python evalb.py <output.trees> <test.trees>
```

Where `<output.trees>` is the output of your program. You have also been provided with a file `tree.py`, which contains some helper functions for handling trees (it is used by `evalb.py`). Feel free to use any methods in the script.

## 4 Submission

Please implement in Python (3 is preferred, but 2.7 is fine as well) . You are free to implement in any language but then you will have to demo your code to the TA.

- You should submit your code to blackboard. Please document your code and use a README that tells us how to run your code if we need to.
- You need to demo your work to the grader. The grader will ask you to run a couple of sentences through your CYK parser. Your parser should produce the best parse in the .tree format used in the input files.

- Submit a report with relevant implementation details. Include the accuracy of the system on both the training and test data.
- Please list the top 10 frequent rules in your PCFG. Mention the number of binary and unary rules in your PCFG.
- Give some examples of the errors and make guesses on why those errors arise.