

# INF4820: Theoretical Answers to Problem Set (3b)

## Estimating a PCFG from a Treebank

Rule	Rule Count	LHS Count	P(RHS LHS)	Rule	Rule Count	LHS Count	P(RHS LHS)
S → NP VP	2	2	$\frac{2}{2}$	NNP → <i>Frodo</i>	2	4	$\frac{2}{4}$
NP → NNP	4	6	$\frac{4}{6}$	NNP → <i>Sam</i>	2	4	$\frac{2}{4}$
NP → DT NN	2	6	$\frac{2}{6}$	VBD → <i>sent</i>	2	2	$\frac{2}{2}$
VP → VBD NP PP	1	2	$\frac{1}{2}$	DT → <i>the</i>	2	2	$\frac{2}{2}$
VP → VBD NP NP	1	2	$\frac{1}{2}$	NN → <i>ring</i>	2	2	$\frac{2}{2}$
PP → P NP	1	1	$\frac{1}{1}$	P → <i>to</i>	1	1	$\frac{1}{1}$

## Generalized Chart Parsing

There are three places in our code where new edges get added to the agenda:

1. Initialization Stage: for each word in the input, we add all lexical rules in the grammar which have that word as their RHS. These are passive edges, with a PoS category and a single daughter which is an edge representing the word. The probability is the probability of the lexical rule.
2. Fundamental Rule: The fundamental rule will add an edge to the agenda when it has been called with compatible active and passive edges. For the edges to be compatible, the category of the passive edge must be the same as the first element of the unanalyzed categories in the active edge. Furthermore, the edges must be adjacent. There are two points in the algorithm where the fundamental rule can be called:
  - (a) when processing a passive edge, if an edge with the same span and category has not been seen, the fundamental rule will be called for each active edge already in the chart whose `to` index equals the `from` index of the passive edge.
  - (b) when processing an active edge, the fundamental rule will be called for all passive edge already in the chart whose `from` index equals the `to` index of the active edge.

The edges added by the fundamental rule will have the `category` of the active edge; `daughters` equal to the `daughters` of the active edge, plus the passive edge; and `unanalysed` equal to the `unanalysed` of the active edge, without the first element. The `from` index will be the `from` from the active edge, while the `to` index of the new edge will be the `to` of the passive edge. The probability will be the probability of the active edge.

3. Main Loop, Prediction: After processing a passive edge, we add edges for each grammar rule whose RHS starts with the category of our just-added passive edge. These edges will have a single daughter consisting of the passive edge, the `category` of the rule LHS, and `unanalysed` the rule RHS, without the first element. The span (`to` and `from`) will be the same as that of the passive edge. The probability will be the probability of the rule.

## Viterbi over a PCFG Parse Forest

- (a) The supplied `viterbi()` function is a recursive function that operates on an edge. If the `cache` slot of the edge contains an edge, then we have already calculated the optimal sub-tree and probability for the category and span represented by this edge, and thus we just return that cached edge.

Otherwise, we first calculate the probability for the sub-tree represented by the host edge and its daughter edges (the ones in the `daughters` slot). We do this by adding the probability of the host edge (which will initially be the log probability of the grammar rule it represents) to the Viterbi scores in each of the daughter edges, which are calculated by running `viterbi()` recursively on those edges.

Once we have an initial score, we iterate over the alternate edges that have been packed into the host edge, making the same calculation for each of them, and checking to see whether they have a higher probability than our host edge (the Viterbi maximize step). If so, we set the probability of our host edge to that of the alternate edge, and set the edge daughters to the daughters of the alternate edge (we need not change the `category` value, as the host edge and all alternate edges packed into it, by definition, have the same category).

Once we have found the edge with the highest probability, whether from the host edge or one of its alternates, we store that edge in its own cache, so the calculation does not need to be repeated.

- (b) The Viterbi function always calculates optimal solutions for smaller sub-problems, and builds on them to form the solution to a larger problem. It also records solutions to previously seen sub-problems for re-use. This is the same for both HMM and a PCFG packed forest.

In the HMM, the smaller sub-problems were the highest probability sub-sequences (from 0 to  $i$ ), and these were built on by exploring all ways to extend the observation/label sequence by one step ( $i + 1$ ).

For the packed forest, the sub-problems are the highest probability sub-trees for sub-spans of the input. The larger problems are the larger trees which cover a wider span and use these sub-trees as daughters. The Viterbi score for a tree then is the product of the probability of the rule that derives the left hand side from the daughters on the right, and the Viterbi scores of each of those daughters. Rather than build up the full solution in a linear left-to-right fashion, as in HMM, Viterbi over a packed forest builds up hierarchically from the bottom up.