1 Quick note: log-probabilities

- Multiplying probabilities can quickly lead to very small numbers, and single-precision floats (C float) only go down to about $10^{-45}$
- Typically people use log-probabilities instead:
  \[
  \log(pq) = \log p + \log q \quad \text{outdated bonus: + used to be faster than } \times
  \]
  \[
  p \leq q \leftrightarrow \log p \leq \log q
  \]
- More esoteric:
  \[
  \log(p + q) = \begin{cases} 
  \log p + \log(1 + \exp(\log q - \log p)) & \text{if } p > q \\
  \log q + \log(1 + \exp(\log p - \log q)) & \text{otherwise}
  \end{cases}
  \]
  where $\log(1 + x)$ often has a special implementation in math libraries called \texttt{log1p} that is more precise for small $x$.

2 Beam search

Motivation

- Parsing takes $O(n^3|G|)$ time, and $|G|$ can get very large. So we prune the search space to speed things up, possibly at the cost of search errors
- Suppose our input sentence is $w_1 \cdots w_{10}$ and we have $[\text{NP}, 0, 5] : 0.001$ and $[\text{RRC}, 0, 5] : 10^{-30}$ in the same cell
- Viterbi CKY doesn’t let us throw out the RRC item, because they are not comparable. It could be that there are lots of high-probability rules that can get from $S$ down to RRC, and only low-probability rules to get from $S$ down to NP.
- But intuitively we know that the chances are slim of this item becoming part of the best parse.
- So: we need a way of (1) making items approximately comparable and then (2) abandoning the items that don’t look like they can compete with the others
Heuristic functions

• The probability on item \([NP, 0, 5]\) : \(p\) is the probability of the best derivation \(NP \Rightarrow^* w_1 \cdots w_5\).

• We would like to approximate the best probability of \(S \Rightarrow^* NPw_6 \cdots w_{10}\) (sometimes called the Viterbi outside probability). If we multiply \(p\) by that, then the products will all be comparable.

• There are lots of ways to do this, but the very easiest is just to use a separate distribution \(P(X)\), the probability that any given node in a tree will be \(X\). (We’ll talk next week about how to estimate values for both \(P(X \rightarrow \alpha \mid X)\) and \(P(X)\).)

• Call the probability of an item times its heuristic function its score.

Pruning cells

• Threshold pruning
  – Discard any item whose score is worse than \(\beta\) times the best score in the same cell.
  – Typical values of \(\beta\) are \(10^{-1}\) to \(10^{-5}\)

• Histogram pruning
  – Discard all but the best \(b\) items
  – Typical values of \(b\) are 10 to 100

• Implementation notes
  – Easy way: just do threshold pruning on the fly, using the best score seen so far. This means that some bad items could sneak in early. (Not acceptable for full credit on Exercise 7.)
  – In CKY, you know when you are done with a cell, so you can prune it at that point. For histogram pruning, use a linear-time \(b\)-best selection algorithm, don’t sort!
  – A cleaner method is to use a worst-first priority heap, which lets us efficiently do both threshold and histogram pruning at any time.
  – Links:
    * http://en.wikipedia.org/wiki/Binary_heap