14.1 Search tree

Recall speech example recognizer, with large marginals for forward messages in inference. The size is exponential on the number of variables in message!

But, can approximate full search using $k$ best beam per word. (Best path could leave beam, but probably won’t.)

Then don’t need exhaustive search, stop at $k$ best.

- sort model tables,
- search in order of decreasing probability
- winner is one keeping most probability, all others fall below before end

14.2 A*/best-first search algorithm for sequence models

When using a beam, we can find solutions in probability order, then stop.

- Keep partial paths as elements $e$ in a priority queue
  - priority queue ‘add’ operation in $O(\log n)$
  - priority queue ‘remove best’ operation in $O(\log n)$
- Partial paths consist of an index to the $e_v$th best outcome for each random variable $v$, plus the probability of the partial path so far
- A partial path’s probability decreases monotonically (at every step), so the set of partial paths in the queue form an ‘equi-probable frontier.’
- At each iteration, the algorithm only has to explore the best partial path on the frontier.
\( Q = [\langle 1.0, 0, -1, -1, -1 \rangle] \)  

(first field is probability)

while \( Q \) not empty and \( B_t \) not full:

\[ e = \langle e_0, e_1, \ldots, e_V \rangle = \text{remove best element from } Q \]

if \( e_V \neq -1 \):

add \( e \) to \( B_t \)  

(finish)

else:

\[ v = \arg \max_{v'} e_{v'} \neq -1 \]

\[ e' = e; \quad e'_{v+1} = 0; \quad e'_0 = e'_0 \cdot P_{\theta_{v+1}}(e'_{v+1} | \ldots) \]

add \( e' \) to \( Q \)  

(forward step)

\[ e' = e; \quad e'_v = e'_v + 1; \quad e'_{v+1} = -1; \quad e'_0 = e'_0 \cdot P_{\theta_v}(e'_v | \ldots) \]

add \( e' \) to \( Q \)  

(side step)

14.3 Example

Given the following tables in the speech recognizer model from the previous notes:

\[ \theta_{FP} = \begin{array}{ccc} W & \bar{P} & \bar{F} \\ /wi/ & 0 & 1 \\ /wi/ & 1 & .6 .4 \\ /wik/ & 0 & 1 \\ /wik/ & 1 & 1 \\ /wik/ & 2 & .5 .5 \end{array} \]

\[ \theta_{FW} = \begin{array}{ccc} W & \bar{P} & \bar{F} \\ /wi/ & false & true \\ /wi/ & false & /wi/ true \\ /wik/ & false & /wik/ true .1 .9 \\ /wik/ & false & /wik/ true .5 .5 0 0 0 \end{array} \]

\[ \theta_W = \begin{array}{ccc} W & \bar{F} & \bar{P} \\ /wi/ & false & /wi/ false false /wi/ true true /wik/ false true false /wik/ true true false /wik/ true false false /wik/ true \end{array} \]

\[ \theta_P = \begin{array}{cccc} P & \bar{W} & \bar{P} & \bar{F} \\ /wi/ & 0 & 1 & 2 \\ /wi/ & 0 & 1 & 0 \\ /wik/ & 0 & .5 & .5 \\ /wik/ & 0 & 0 & 1 \\ /wik/ & 0 & 0 & 1 \\ /wik/ & 2 & 0 & .5 \\ /du/ & 0 & 0 & 1 \\ /du/ & 0 & 0 & 1 \\ /si/ & 0 & 0 & 1 \\ /si/ & 1 & 0 & .9 \end{array} \]

\[ \theta_X = \begin{array}{cccc} W & \bar{P} & \bar{F} & \bar{W} \\ /wi/ & 0 & 1 & 2 \\ /wi/ & 0 & 1 & 0 .9 .9 .9 .9 .9 \end{array} \]

Here is the set of queue elements that get generated:
<table>
<thead>
<tr>
<th>prev</th>
<th>$F^p$</th>
<th>$F^W$</th>
<th>$W$</th>
<th>$P$</th>
<th>$X$</th>
</tr>
</thead>
<tbody>
<tr>
<td>/wi/</td>
<td>0.6</td>
<td>/wi/</td>
<td>0.1</td>
<td>1.3</td>
<td>0.9</td>
</tr>
<tr>
<td>1.4</td>
<td>false</td>
<td>3</td>
<td></td>
<td></td>
<td>[i]:.27</td>
</tr>
<tr>
<td>/wik/</td>
<td>0.1</td>
<td>true</td>
<td>2</td>
<td>0.3</td>
<td>[i]:0</td>
</tr>
<tr>
<td>1.4</td>
<td>false</td>
<td>4</td>
<td>0.5</td>
<td>0.1</td>
<td>[i]:0</td>
</tr>
<tr>
<td>false</td>
<td>0.1</td>
<td>false</td>
<td>2</td>
<td>0.9</td>
<td>[i]:18</td>
</tr>
<tr>
<td>/wik/</td>
<td>0.5</td>
<td>1.5</td>
<td>1.2</td>
<td>[i]:0.2</td>
<td></td>
</tr>
<tr>
<td>/wik/</td>
<td>2.2</td>
<td>0.1</td>
<td>[i]:0.02</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>