Implementation

Similar to the 1-order Viterbi Algorithm, the calculation of emission parameters is the same as in part 2 and part 3. The transition parameters used in the decoding process are taken to be the MLE of the transition probabilities of a trigram of tag sequence, expressed as follows:

$$p(y_i \mid y_{i-2}, y_{i-1}) = \frac{p(y_{i-2}, y_{i-1}, y_i)}{p(y_{i-2}, y_{i-1})} = \frac{Count(y_{i-2}, y_{i-1}, y_i)}{Count(y_{i-2}, y_{i-1})}$$

It could also be written as

$$p(v \mid w, u) = \frac{Count(w, u, v)}{Count(w, u)}$$

, where the tag sequence is w -> u -> v.

2-order Viterbi Algorithm

- Base case:

$$\pi(0, u, START) = 1$$

- Moving forward recursively: for any $j \in \{1, 2, \ldots, n-1\}$

$$\pi(j, y_j, y_{j+1}) = max_{y_{j-1} \in T}\ \pi(j-1, y_{j-1}, y_j) \cdot a(y_{j+1} | y_{j-1}, y_j) \cdot b(x_j, y_j)$$

- Final case:

$$\pi(n, y_n, STOP) = max_{y_{n-1} \in T}\ \pi(n-1, y_{n-1}, y_n) \cdot a(STOP | y_{n-1}, y_n) \cdot b(x_n, y_n)$$

Most likely tag sequences

It is easier to obtain the most likely tag sequence with given next tag:

$$y_n^* = argmax_{y_n \in T}\, \pi(n, y_n, STOP)$$

$$y_j^* = argmax_{y_j \in T}\, \pi(j, y_j, y_{j+1}^*)$$

Time complexity

To calculate the score for each possible tag at a time, we need to do the calculation for all of the combinations of (w,u,v), where w is the previous tag, u is the current tag and v is the next tag. For time=1 to time=n, we will have three loops at each time step. Therefore, the total complexity is given by $O(nT^3)$