ASR Assignment 2

Ayush Agrawal, Anshul Verma, Anand Narasimhan, Akshat Goyal

March 2024

Part 1 - Vitterbi Variants

Consider an HMM $\lambda = (A, B)$ with a sequence of hidden states Q, a sequence of observations O, transition probabilities $a_{ij} = \Pr(q_t = j | q_{t-1} = i)$ and emission probabilities $b_j(o_t) = \Pr(o_t | q_t = j)$.

1.1 Efficient Viterbi

Problem 1.1

Solution The modification occurs only during computation of $v_t(j) = \max_{i=1}^{N} v_{t-1}(i) a_{ij} b_j(o_t)$. The quantity a_{ij} can only either be p or q, and so it is enough if we store the largest value of $v_{t-1}(i)$, in order to compute $v_t(j)$.

Essentially, to compute $v_t(j)$, we only have to check $v_{t-1}(j) a_{jj}$, which is $v_{t-1}(j) \times p$, along with $(\max_i v_{t-1}(i)) \times a_{ij}$, which is $(\max_i v_{t-1}(i)) \times q$.

We compute this at the start of each iteration over t, saving us a loop over N to compute the max, and just checking between 2 values instead.

Let $a = \operatorname{argmax}_i v_{t-1}(i)$

$$v_t(i) = b_i(o_t) \max(v_{t-1}(a) \times q, v_{t-1}(i) \times p)$$

Also for computing the backtrace array.

$$bt_t(i) = \begin{cases} a, & v_{t-1}(a) \times q \geq v_{t-1}(i) \times p \\ i, & \text{otherwise} \end{cases}$$

1.2 k-repeat Viterbi

Problem 1.2

Solution Maintain another array, called $P_t(N,k)$, where N,k are the dimensions of the array. The first dimension has the usual meaning, whereas the second dimension is supposed to store the number of times the current state has been repeated. If the current state has been repeated k times, we move it to the v array, since now the repetition does not matter.

Firstly, the entries of $v_t(j)$ for $t < k$ will all be 0, since there is no sequence of observations with length less than k that satisfies the hypothesis in the question.

Initialisation of $P_1(j,1)$ is done by the initial probability distribution, $\Pi_j b_j(o_1)$

Updating $P_t(N,k)$ as follows :

$$P_t(j,l+1) = P_{t-1}(j,l)a_{jj}b_j(o_t) \quad 0 < l < k$$
$$P_t(j,1) = \max_{(i=1,i\neq j,l=0)}^{(i=N,l=k-1)} P_{t-1}(i,l)a_{ij}b_j(o_t)$$

The first equation talks about a sequence where j has appeared l consecutive times already, and we are computing the probability of it appearing for the $l+1^{\text{th}}$ consecutive time.

The second equation talks about being in a state j, but for the first time, the previous observation being some state $i \neq j$. We consider all the values in the timestep $t-1$ for this, varying over both the first and second dimension.

Updating $v_t(j)$ as follows :

$$v_t(j) = \max_{i=1}^{N} \left(\max\left\{v_{t-1}(i)a_{ij}b_j(o_t), P_{t-1}(i,k)a_{ij}b_j(o_t)\right\}\right)$$

1.3 Extended HMMs

Problem 1.3

Solution Here, the length of the emitted sequence is not always 1, so we build the recurrence from $v_{t-l}(i)$, instead of $v_{t-1}(i)$

The recurrence will be :

$$v_t(j) = \max_{i=1,l=1}^{N,l_{\max}} v_{t-l}(i) \times a_{ij} \times L(j,l) \times B(j,o_{t-l+1},\ldots,o_t)$$

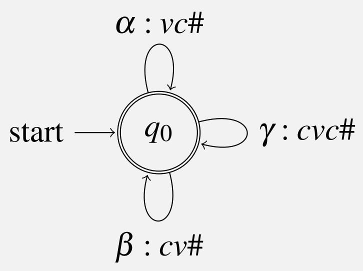
The a_{ij} term is present because we're transitioning from state i to state j, doesn't depend on the observation timestep. The other two terms represent the probability of emitting that sequence of length l on reaching state i.
$L(j,l)$ is the probability of the length of the sequence being l and $B(j, o_{t-l+1}, \dots, o_t)$ is the probability of emission of the last l observations given that the length of the sequence was l.

WFSTs

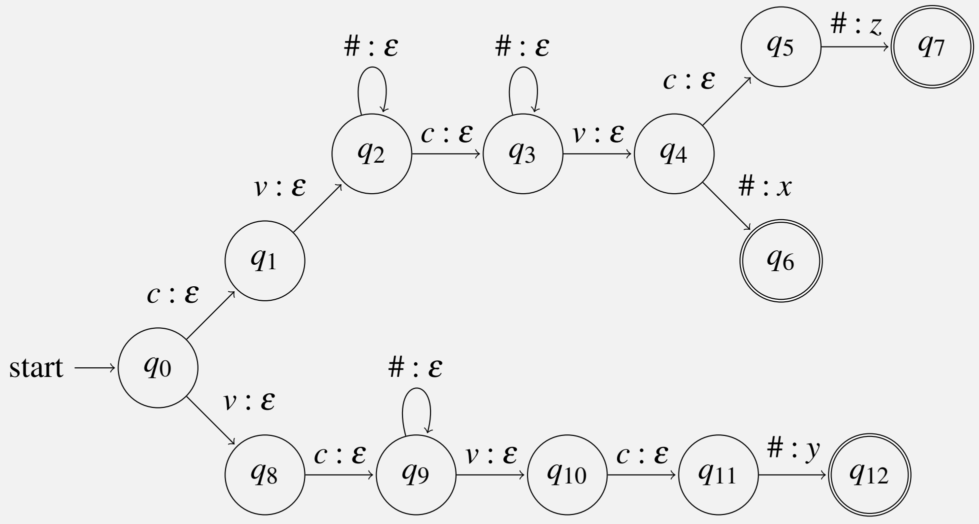
2.1 Defining C and L

Problem 2.1

Solution This is the FST for C



This accepts the empty string and loops back on inputs α, β and γ, outputting the corresponding phones, vc, cv, cvc, along with a *breaker* symbol #. It accepts $(\alpha + \beta + \gamma)^*$ and rejects everything else.
The # symbol is to ensure that words do not cross syllable boundaries.
The FST for L is also straightforward, we only output and reach to a final state when we see the *breaker* symbol at the end, we also add loops on the *breaker* because # can come between two syllables in the same word.

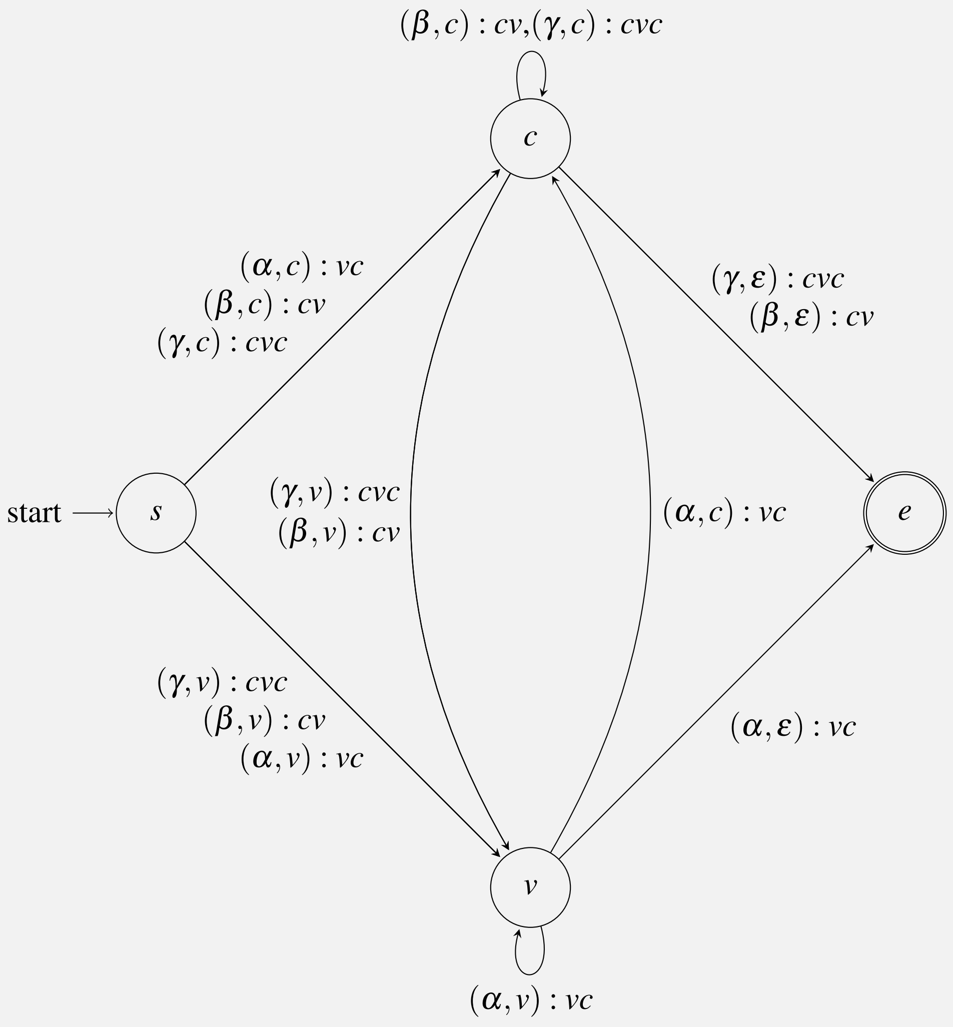


2.2 Contextualized syllables

Problem 2.2

Solution This WFST helps us remove possible bad transitions in our final transducer. For example (α, c) can't be followed by (α, c) because the first phone in α isn't c.

s and e here are the start and end states, we make two states corresponding to both c and v, the state c should only have those outgoing edges where the syllable S starts with c and only those incoming edges where the phone P is c, a similar statement should hold for the state v, So the transitions follow naturally and when the phone P is ε we go to the final state.


$(\beta, c) : cv, (\gamma, c) : cvc$
c
$(\alpha, c) : vc$
$(\beta, c) : cv$
$(\gamma, c) : cvc$
$(\gamma, \varepsilon) : cvc$
$(\beta, \varepsilon) : cv$
start
s
$(\gamma, v) : cvc$
$(\beta, v) : cv$
$(\alpha, c) : vc$
e
$(\gamma, v) : cvc$
$(\beta, v) : cv$
$(\alpha, v) : vc$
$(\alpha, \varepsilon) : vc$
v
$(\alpha, v) : vc$