QUESTION 2

PART (a)

(i) $$\Pr(q_{t+1}=k \mid q_t=j, O_1, \dots, O_T) = \frac{\Pr(q_{t+1}=k, q_t=j, O_1, \dots, O_T)}{\Pr(q_t=j, O_1, \dots, O_T)}$$

Taking numerator,

$$\Pr(q_{t+1}=k, q_t=j, O_1, \dots, O_T) = \Pr(O_{t+2}, \dots, O_T \mid q_{t+1}=k, q_t=j, O_1 \dots O_{t+1}) \cdot \Pr(q_{t+1}=k, q_t=j, O_1 \dots, O_{t+1})$$

$$= \underbrace{\beta_{t+1}(k)}_{\text{due to output independence}} \cdot \Pr(q_{t+1}=k, O_{t+1} \mid q_t=j, O_1, \dots, O_t) \cdot \Pr(q_t=j, O_1, \dots, O_t)$$

$$= \beta_{t+1}(k) \cdot \underbrace{\Pr(O_{t+1} \mid q_{t+1}=k)}_{\text{due to output independence}} \underbrace{\Pr(q_{t+1}=k \mid q_t=j)}_{\text{due to Markov property}} \cdot \alpha_t(j)$$

$$= \beta_{t+1}(k) \cdot b_k(O_{t+1}) \cdot a_{jk} \cdot \alpha_t(j)$$

Taking denominator,

$$\Pr(q_t=j, O_1, \dots, O_T) = \underbrace{\Pr(O_{t+1}, \dots O_T \mid q_t=j)}_{\text{due to output independence}} \Pr(q_t=j, O_1, \dots O_t)$$

$$= \beta_t(j)\, \alpha_t(j)$$

$\therefore$ Finally,

$$\Pr(q_{t+1}=k \mid q_t=j, O_1, \dots O_T) = \frac{\beta_{t+1}(k)\, b_k(O_{t+1})\, a_{jk}\, \cancel{\alpha_t(j)}}{\beta_t(j)\, \cancel{\alpha_t(j)}}$$

$$= \frac{[\beta_{t+1}(k)][b_k(O_{t+1})][a_{jk}]}{\beta_t(j)}$$

(ii) $Pr(q_{t-1}=i, q_t=j, q_{t+1}=k \mid O_1, \dots, O_T)$

$$= \frac{Pr(q_{t+1}=k, q_t=j, q_{t-1}=i, O_1, \dots, O_T)}{Pr(O_1, \dots, O_T)}$$

Taking numerator,

$Pr(q_{t+1}=k, q_t=j, q_{t-1}=i, O_1, \dots, O_T)$

$= Pr(O_{t+2} \dots O_T \mid q_{t+1}=k, q_t=j, q_{t-1}=i, O_1 \dots O_{t+1})$

$\quad Pr(q_{t+1}=k, q_t=j, q_{t-1}=i, O_1 \dots O_{t+1})$

$= \beta_{t+1}(k) \; Pr(q_{t+1}=k, O_{t+1} \mid q_t=j) \cdot Pr(q_t=j, O_T \mid q_{t-1}=i)$

$\quad \cdot Pr(q_{t-1}=i, O_1 \dots O_{t-1})$

due to output independence

due to both properties (markov, output)

$= [\beta_{t+1}(k)][b_k(O_{t+1}) \cdot a_{jk}][b_j(O_t)\, a_{ij}][\alpha_{t-1}(i)]$

Taking denominator,

$Pr(O_1, \dots, O_T) = \sum_{i=1}^{N} Pr(O_1, \dots, O_T, q_t = i)$ (N is number of states, t is any number in $[1,T]$)

$= \sum_{i=1}^{N} Pr(O_{t+1} \dots O_T \mid O_1 \dots O_t, q_t = i)$

$\quad Pr(O_1 \dots O_t, q_t = i)$

$= \sum_{i=1}^{N} Pr(O_{t+1} \dots O_T \mid q_t = i)\; Pr(O_1 \dots O_T, q_t = i)$

$= \sum_{i=1}^{N} \beta_t(i)\, \alpha_t(i)$ ← Note that for any $t \in \{1, \dots, T\}$, this value must be same

$\therefore$ Finally,

$Pr(q_{t-1}=i, q_t=j, q_{t+1}=k \mid O_1 \dots O_T)$

$$= \frac{\beta_{t+1}(k)\, b_k(O_{t+1})\, a_{jk}\, b_j(O_t)\, a_{ij}\, \alpha_{t-1}(i)}{\sum_{i=1}^{N} \alpha_t(i)\, \beta_t(i)}$$

(b) The original Viterbi recursion involves finding,

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

$$= b_j(o_t) \max_{i=1}^{N} v_{t-1}(i)\, a_{ij}$$

Now, we have $a_{ii} = p$, $a_{ij} = q$ for $i \neq j$ and $p > q$.

* Note that while computing $v_t(j)$ themselves, we can keep track of the top two maximas of v_t without any additional overhead. Let them be α, β for v_{t-1} array.

* Now, if we find the value of $v_t(j)$ in $O(1)$, we are done and left with an $O(NT)$ algorithm.

* Now, for that we need to find $\max_{i=1}^{N} v_{t-1}(i) a_{ij}$ in $O(1)$.

  i.e., we want $\max\left(p\, v_{t-1}[j],\ q \max_{\substack{i=1 \\ i\neq j}}^{N} v_{t-1}[i]\right)$

* So, if we find $\max_{\substack{i=1, \\ i\neq j}}^{N} v_{t-1}(i)$ in $O(1)$, we are done.

  Now, this can be found as follows,

  if $v_{t-1}(j) == \alpha$, then $\max_{\substack{i=1 \\ i\neq j}}^{N} v_{t-1}(i) = \beta$.

  else $\max_{\substack{i=1 \\ i\neq j}}^{N} v_{t-1}(i) = \alpha$.

  [This is obvious since α, β are the top two maximas of v_{t-1}. Note that if the maximum value α occurs multiple times, then $\beta = \alpha$]

* Now, while we get the values of $v_t(j)$, we keep track of top two maximum values to help in next iteration.

* Similarly, argmax can be computed along the same lines.

Thus, we have an <u>$O(NT)$ algorithm</u>.

(C) Viterbi recursion is essentially a dynamic programming method (DP)

~~So, we extend the states of the DP to as follows~~

~~$v'_{t,x,y}(j)$ to denote $\max p(q_1 \dots q_{t-1}, o_1, o_2 \dots o_t, q_t = j \mid \lambda)$~~

~~under the constraint that~~

~~if $y = 1$, $\exists x <$~~

Let ~~$v_{t,0}(j)$ denote the usual $v_t(j)$ and~~ $v'_t(j)$ denotes that

$\max_{q_1 \dots q_{t-1}} P(q_1 \dots q_{t-1}, o_1 \dots o_t, q_t = j \mid \lambda)$ under the constraint that

$\exists$ a consecutive k-length subsequence in $q_1 \dots q_{t-1}$ in which all are equal.

~~Now, the usual recursion of $v_t(j)$ will hold, i.e.,~~

~~Now, we have, the usual~~

We compute the usual Viterbi recursion,

$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i)\, a_{ij}\, b_j(o_t).$$

Additionally, we compute $v_t'(j)$ as follows,

$$v_t'(j) = \max \begin{pmatrix} \max_{i=1}^{N} v'_{t-1}(i)\, a_{ij}\, b_j(o_t), \\ \underbrace{[a_{jj} b_j(o_t)][a_{jj} b_j(o_{t-1})] \cdots [a_{jj} b_j(o_{t-k+2})]}_{k-1 \text{ terms}} \times v_{t-k+1}(j) \end{pmatrix}$$

this part is considered if $t \geq k$.

$= a_{jj}^{k-1} \prod_{i=1}^{k-1} b_j(o_{t-i+1})$

The explanation of the above expression is ~~as follows~~,

* Either we could simply consider all $q_1 \dots q_{t-1}$ which already ~~contains~~ contain a k-length subsequence with equal values,

(or)

* we consider the possibility that q_t to q_{t-k+1} are all equal to j ~~and~~, which is nothing but finding,

$$\max_{q_1 \dots q_{t-k}} p(q_1 \dots q_{t-k}, o_1 \dots o_t, q_{t-k+1} = j, \dots, q_t = j \mid \lambda)$$

~~$= \max_{q_1 \dots q_{t-k}} \left[p(q_1 \dots q_{t-k}, o_1 \dots o_t, q_{t-k+1} = j) \times p[q_{t-k+2} = j, \dots, q_t = j] \mid q_{t-k+1} = j \right]$~~

due to Markov property.

$$= \max_{q_1 \dots q_{t-k}} P(q_1 \dots q_{t-k}, O_1 \dots O_{t-1}, q_{t-k+1} = j, \dots q_{t-1} = j \mid \lambda)$$

$$\times \underbrace{P(O_t, q_t = j \mid q_{t-1} = j)}_{= a_{jj} b_j(O_t)}$$

$\vdots$ doing this $k-1$ times

$$= \left[a_{jj}^{k-1} \prod_{i=1}^{k-1} b_j(O_{t-i+1}) \right] \underbrace{\max_{q_1 \dots q_{t-k}} P(q_1, \dots q_{t-k}, O_1, \dots, O_{t-k}, q_{t-k+1} = j \mid \lambda)}_{= v_{t-k+1}(j) \text{ (by definition)}}$$

$$= \left[a_{jj}^{k-1} \prod_{i=1}^{k-1} b_j(O_{t-i+1}) \right] V_{t-k+1}(j).$$

Now, finally we get v' which is what we wanted.

To, backtrack, just take

(we have, $v_t'(j) = \max \left(\underset{\text{(A)}}{\max_{i=1}^{N} v_{t-1}'(i) a_{ij} b_j(O_t)}, \underset{\text{(B)}}{\left[a_{jj}^{k-1} \prod_{i=1}^{k-1} b_j(O_{t-i+1}) \right] V_{t-k+1}(j)} \right)$

Now,

$$bt_t(j) = \begin{cases} \operatorname{argmax}_{i=1}^{N} v_{t-1}'(i) a_{ij} b_j(O_t) & \text{if } A > B \\ -1 & \text{otherwise} \end{cases}$$

Now, while backtracking, if we encounter -1, we simply take the current state k times and then continue from bt_{t-k+1} onwards.

Thus, the algorithm is complete, by computing both $V_t(j)$ & $V_t'(j)$ simultaneously.