

Introduction to Algorithms and Data Structures

Tutorial 7

your tutor

University of Edinburgh

3rd-7th February, 2025

Q1: Dynamic programming *Edit Distance*

$$d = \begin{matrix} & & A & T & G & G & C & T \\ \begin{matrix} A \\ C \\ T \\ G \\ G \\ T \end{matrix} & \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 0 & 1 & 2 & 3 & 4 & 5 \\ 2 & 1 & 1 & 2 & 3 & 3 & 4 \\ 3 & & & & & & \\ 4 & & & & & & \\ 5 & & & & & & \\ 6 & & & & & & \end{bmatrix} \end{matrix}, a = \begin{bmatrix} - & 2 & 2 & 2 & 2 & 2 & 2 \\ 3 & 0 & 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 1 & 1/2 & 1/2 & 0 & 2 \\ 3 & & & & & & \\ 3 & & & & & & \\ 3 & & & & & & \\ 3 & & & & & & \end{bmatrix}$$

row 2: for cell $[2, 1]$ recurrence says $d[2, 1] \leftarrow 1 + \min\{d[1, 1], d[2, 0], d[1, 0]\}$

looking up already-computed values, $d[2, 1] \leftarrow 1 + \min\{0, 2, 1\}$

So $d[2, 1] \leftarrow 1$, last op was a "deletion" ($a[2, 1] \leftarrow 3$).

cell $[2, 3]$: again $s_2 \neq t_3$, recurrence $d[2, 3] \leftarrow 1 + \min\{d[1, 3], d[2, 2], d[1, 2]\}$

looking up already-computed values, $d[2, 3] \leftarrow 1 + \min\{2, 1, 1\}$

$\Rightarrow d[2, 3] \leftarrow 1$, last op was a substitution or an insertion ($a[2, 3] \leftarrow 1/2$)

Q1: Dynamic programming *Edit Distance*

$$d = \begin{matrix} & & A & T & G & G & C & T \\ \begin{matrix} A \\ C \\ T \\ G \\ G \\ T \end{matrix} & \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 0 & 1 & 2 & 3 & 4 & 5 \\ 2 & 1 & 1 & 2 & 3 & 3 & 4 \\ 3 & 2 & 1 & 2 & 3 & 4 & 3 \\ 4 & 3 & 2 & 1 & 2 & 3 & 4 \\ 5 & 4 & 3 & 2 & 1 & 2 & 3 \\ 6 & & & & & & \end{bmatrix} \end{matrix}, a = \begin{bmatrix} - & 2 & 2 & 2 & 2 & 2 & 2 \\ 3 & 0 & 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 1 & 1/2 & 1/2 & 0 & 2 \\ 3 & 3 & 0 & 1/2 & 1/2 & 1/2/3 & 0 \\ 3 & 3 & 3 & 0 & 0 & 2 & 2/3 \\ 3 & 3 & 3 & 0 & 0 & 2 & 2 \\ 3 & & & & & & \end{bmatrix}$$

Q1: Dynamic programming *Edit Distance*

$$d = \begin{matrix} & & A & T & G & G & C & T \\ \begin{matrix} A \\ C \\ T \\ G \\ G \\ T \end{matrix} & \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 0 & 1 & 2 & 3 & 4 & 5 \\ 2 & 1 & 1 & 2 & 3 & 3 & 4 \\ 3 & 2 & 1 & 2 & 3 & 4 & 3 \\ 4 & 3 & 2 & 1 & 2 & 3 & 4 \\ 5 & 4 & 3 & 2 & 1 & 2 & 3 \\ 6 & 5 & 4 & 3 & 2 & 2 & 2 \end{bmatrix} \end{matrix}, a = \begin{bmatrix} - & 2 & 2 & 2 & 2 & 2 & 2 \\ 3 & 0 & 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 1 & 1/2 & 1/2 & 0 & 2 \\ 3 & 3 & 0 & 1/2 & 1/2 & 1/2/3 & 0 \\ 3 & 3 & 3 & 0 & 0 & 2 & 2/3 \\ 3 & 3 & 3 & 0 & 0 & 2 & 2 \\ 3 & 3 & 0 & 3 & 3 & 1 & 0 \end{bmatrix}$$

Q1: Dynamic programming *Edit Distance*

(b) Find the optimum alignment using the $a[i,j]$ values:

$$d = \begin{matrix} & & A & T & G & G & C & T \\ \begin{matrix} A \\ C \\ T \\ G \\ G \\ T \end{matrix} & \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 0 & 1 & 2 & 3 & 4 & 5 \\ 2 & 1 & 1 & 2 & 3 & 3 & 4 \\ 3 & 2 & 1 & 2 & 3 & 4 & 3 \\ 4 & 3 & 2 & 1 & 2 & 3 & 4 \\ 5 & 4 & 3 & 2 & 1 & 2 & 3 \\ 6 & 5 & 4 & 3 & 2 & 2 & 2 \end{bmatrix} \end{matrix}, a = \begin{matrix} & & - & 2 & 2 & 2 & 2 & 2 & 2 \\ \begin{matrix} 3 & 0 & 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 1 & 1/2 & 1/2 & 0 & 2 \\ 3 & 3 & 0 & 1/2 & 1/2 & 1/2/3 & 0 \\ 3 & 3 & 3 & \mathbf{0} & 0 & 2 & 2/3 \\ 3 & 3 & 3 & 0 & \mathbf{0} & \mathbf{2} & 2 \\ 3 & 3 & 0 & 3 & 3 & 1 & \mathbf{0} \end{matrix} \end{matrix}$$

match (so next we will check $a[3,2]$)

'G' 'G' - 'T'
 'G' 'G' 'C' 'T'

Q1: Dynamic programming *Edit Distance*

(b) Find the optimum alignment using the $a[i,j]$ values:

$$d = \begin{matrix} & & A & T & G & G & C & T \\ \begin{matrix} A \\ C \\ T \\ G \\ G \\ T \end{matrix} & \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 0 & 1 & 2 & 3 & 4 & 5 \\ 2 & 1 & 1 & 2 & 3 & 3 & 4 \\ 3 & 2 & 1 & 2 & 3 & 4 & 3 \\ 4 & 3 & 2 & 1 & 2 & 3 & 4 \\ 5 & 4 & 3 & 2 & 1 & 2 & 3 \\ 6 & 5 & 4 & 3 & 2 & 2 & 2 \end{bmatrix} \end{matrix}, a = \begin{matrix} & & - & 2 & 2 & 2 & 2 & 2 & 2 \\ \begin{matrix} 3 & 0 & 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 1 & 1/2 & 1/2 & 0 & 2 \\ 3 & 3 & \mathbf{0} & 1/2 & 1/2 & 1/2/3 & 0 \\ 3 & 3 & 3 & \mathbf{0} & 0 & 2 & 2/3 \\ 3 & 3 & 3 & 0 & \mathbf{0} & \mathbf{2} & 2 \\ 3 & 3 & 0 & 3 & 3 & 1 & \mathbf{0} \end{matrix} \end{matrix}$$

match (so next we will check $a[2,1]$)

'T' 'G' 'G' - 'T'
 'T' 'G' 'G' 'C' 'T'

Q1: Dynamic programming *Edit Distance*

(b) Find the optimum alignment using the $a[i,j]$ values:

$$d = \begin{matrix} & & A & T & G & G & C & T \\ \begin{matrix} A \\ C \\ T \\ G \\ G \\ T \end{matrix} & \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 0 & 1 & 2 & 3 & 4 & 5 \\ 2 & 1 & 1 & 2 & 3 & 3 & 4 \\ 3 & 2 & 1 & 2 & 3 & 4 & 3 \\ 4 & 3 & 2 & 1 & 2 & 3 & 4 \\ 5 & 4 & 3 & 2 & 1 & 2 & 3 \\ 6 & 5 & 4 & 3 & 2 & 2 & 2 \end{bmatrix} \end{matrix}, a = \begin{matrix} & & - & 2 & 2 & 2 & 2 & 2 & 2 \\ \begin{matrix} 3 & 0 & 2 & 2 & 2 & 2 & 2 \\ 3 & \mathbf{3} & 1 & 1/2 & 1/2 & 0 & 2 \\ 3 & 3 & \mathbf{0} & 1/2 & 1/2 & 1/2/3 & 0 \\ 3 & 3 & 3 & \mathbf{0} & 0 & 2 & 2/3 \\ 3 & 3 & 3 & 0 & \mathbf{0} & \mathbf{2} & 2 \\ 3 & 3 & 0 & 3 & 3 & 1 & \mathbf{0} \end{matrix} \end{matrix}$$

deletion (and next we check $a[1,1]$)

'C' 'T' 'G' 'G' - 'T'
 - 'T' 'G' 'G' 'C' 'T'

Q1: Dynamic programming *Edit Distance*

(b) Find the optimum alignment using the $a[i,j]$ values:

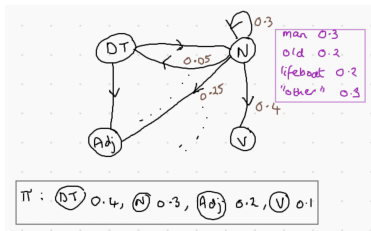
$$d = \begin{matrix} & & \text{A} & \text{T} & \text{G} & \text{G} & \text{C} & \text{T} \\ \text{A} & \left[\begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 0 & 1 & 2 & 3 & 4 & 5 \\ 2 & 1 & 1 & 2 & 3 & 3 & 4 \\ 3 & 2 & 1 & 2 & 3 & 4 & 3 \\ 4 & 3 & 2 & 1 & 2 & 3 & 4 \\ 5 & 4 & 3 & 2 & 1 & 2 & 3 \\ 6 & 5 & 4 & 3 & 2 & 2 & 2 \end{array} \right] & , & a = \left[\begin{array}{cccccc} - & 2 & 2 & 2 & 2 & 2 & 2 \\ 3 & \mathbf{0} & 2 & 2 & 2 & 2 & 2 \\ 3 & \mathbf{3} & 1 & 1/2 & 1/2 & 0 & 2 \\ 3 & 3 & \mathbf{0} & 1/2 & 1/2 & 1/2/3 & 0 \\ 3 & 3 & 3 & \mathbf{0} & 0 & 2 & 2/3 \\ 3 & 3 & 3 & 0 & \mathbf{0} & \mathbf{2} & 2 \\ 3 & 3 & 0 & 3 & 3 & 1 & \mathbf{0} \end{array} \right] \end{matrix}$$

match

'A' 'C' 'T' 'G' 'G' - 'T'
 'A' - 'T' 'G' 'G' 'C' 'T'

Q2: Viterbi Algorithm example

We have an HMM to model sentences in English.



Full transition matrix p , “emissions probabilities” (b_q for every state q) are:

	DT	N	V	Adj		lifeboats	man	old	the	“other”
DT	0	0.6	0	0.4	DT	0	0	0	0.5	0.5
N	0.05	0.3	0.4	0.25	N	0.2	0.3	0.2	0	0.3
V	0.4	0.3	0.1	0.2	V	0	0.1	0	0	0.9
Adj	0.1	0.5	0.2	0.2	Adj	0	0	0.4	0	0.6
Transitions					Emissions					

Q2: Viterbi Algorithm example

The “start state” distribution π assigns probabilities as follows:

$$\pi(\text{DT}) = 0.4, \pi(\text{N}) = 0.3, \pi(\text{Adj}) = 0.2, \pi(\text{V}) = 0.1.$$

The old man the lifeboats.

$s_1 = \text{the}$, $s_1 s_2 = \text{the old}$, $s_1 \dots s_3 = \text{the old man}$, $s_1 \dots s_4 = \text{the old man the}$.

“bottom-up”:

- ▶ consider $s_1 = \text{the}$ to find $mlp[1, q]$ for every state $q \in Q \dots$

$$\pi(q) \times b_{q, \text{the}} \text{ for every } q \in Q$$

- ▶ then consider $mlp[2, q]$ for every state $q \in Q$
- ▶ ... then consider $mlp[3, q]$ for every state $q \in Q$
- ▶ ...

$$mlp[i, q] = \begin{cases} \pi_q \cdot b_{q, s_1} & i = 1 \\ \max_{q^* \in Q} \{ mlp[i-1, q^*] \cdot p_{q^*, q} \cdot b_{q, s_i} \} & i > 1 \end{cases}$$

Q2: Viterbi Algorithm example

row 1: Initialisation for the base case, where *the* might be emitted from a specific q .
Probability $\pi(q) \times b_{q,the}$ for every $q \in Q$

$b_{q,the}$ is 0 for every q except DT.

	DT	N	V	Adj
the	$.4 \times .5 = .2$	0	0	0
old				
man				
the				
lifeboats				

Q2: Viterbi Algorithm example

For subsequent rows, use the recurrence

$$mlp[i, q] = \begin{cases} \pi_q \cdot b_{q, s_1} & i = 1 \\ \max_{q^* \in Q} \{ mlp[i-1, q^*] \cdot p_{q^*, q} \cdot b_{q, s_i} \} & i > 1 \end{cases}$$

	DT	N	V	Adj
the	.4x.5 = .2	0	0	0
old				
man				
the				
lifeboats				

Q2: Viterbi Algorithm example

For subsequent rows, use the recurrence

$$mlp[i, q] = \begin{cases} \pi_q \cdot b_{q, s_1} & i = 1 \\ \max_{q^* \in Q} \{ mlp[i-1, q^*] \cdot p_{q^*, q} \cdot b_{q, s_i} \} & i > 1 \end{cases}$$

	DT	N	V	Adj
the	.4x.5 = .2	0	0	0
old	0		0	
man				
the				
lifeboats				

the old: *old* only has positive emission probability at N and Adj

Q2: Viterbi Algorithm example

For subsequent rows, use the recurrence

$$mlp[i, q] = \begin{cases} \pi_q \cdot b_{q, s_1} & i = 1 \\ \max_{q^* \in Q} \{ mlp[i-1, q^*] \cdot p_{q^*, q} \cdot b_{q, s_i} \} & i > 1 \end{cases}$$

	DT	N	V	Adj
the	.4x.5 = .2	0	0	0
old	0	.2x.6x.2 = .024	0	.2x.4x.4 = .032
man				
the				
lifeboats				

the old: the only positive probability option for *the* is DT, so consider $mlp[1, DT] = 0.2$ followed by the options DT \rightarrow N, DT \rightarrow Adj to emit *old*

Q2: Viterbi Algorithm example

For subsequent rows, use the recurrence

$$mlp[i, q] = \begin{cases} \pi_q \cdot b_{q, s_1} & i = 1 \\ \max_{q^* \in Q} \{ mlp[i-1, q^*] \cdot p_{q^*, q} \cdot b_{q, s_i} \} & i > 1 \end{cases}$$

	DT	N	V	Adj
the	.4x.5 = .2	0	0	0
old	0	.2x.6x.2 = .024	0	.2x.4x.4 = .032
man				
the				
lifeboats				

Q2: Viterbi Algorithm example

For subsequent rows, use the recurrence

$$mlp[i, q] = \begin{cases} \pi_q \cdot b_{q,s_1} & i = 1 \\ \max_{q^* \in Q} \{ mlp[i-1, q^*] \cdot p_{q^*,q} \cdot b_{q,s_i} \} & i > 1 \end{cases}$$

	DT	N	V	Adj
the	.4x.5 = .2	0	0	0
old	0	.2x.6x.2 = .024	0	.2x.4x.4 = .032
man	0			0
the				
lifeboats				

the old man: notice *man* only possible at states N and V

Q2: Viterbi Algorithm example

For subsequent rows, use the recurrence

$$mlp[i, q] = \begin{cases} \pi_q \cdot b_{q,s_1} & i = 1 \\ \max_{q^* \in Q} \{ mlp[i-1, q^*] \cdot p_{q^*,q} \cdot b_{q,s_i} \} & i > 1 \end{cases}$$

	DT	N	V	Adj
the	.4x.5 = .2	0	0	0
old	0	.2x.6x.2 = .024	0	.2x.4x.4 = .032
man	0	.3x.032x.5 = .0048		0
the				
lifeboats				

the old man: notice *man* only possible at states N and V

Ending at N: $b_{N,man} = 0.3$. Take \max of $mlp[2, Adj] \times p_{Adj,N} = 0.032 \times 0.5 = 0.016$ against $mlp[2, N] \times p_{N,N} = 0.024 \times 0.3 = 0.0072$, multiply the 0.016 by $b_{N,man} = 0.3$

Q2: Viterbi Algorithm example

For subsequent rows, use the recurrence

$$mlp[i, q] = \begin{cases} \pi_q \cdot b_{q,s_1} & i = 1 \\ \max_{q^* \in Q} \{ mlp[i-1, q^*] \cdot p_{q^*,q} \cdot b_{q,s_i} \} & i > 1 \end{cases}$$

	DT	N	V	Adj
the	.4x.5 = .2	0	0	0
old	0	.2x.6x.2 = .024	0	.2x.4x.4 = .032
man	0	.3x.032x.5 = .0048	.1x.024x.4 = .00096	0
the				
lifeboats				

the old man: notice *man* only possible at states N and V

Ending at V: $b_{V,man} = 0.1$. Take max of $mlp[2, Adj] \times p_{Adj,V} = 0.032 \times 0.2 = 0.0064$ against $mlp[2, N] \times p_{N,V} = 0.024 \times 0.4 = 0.0096$, multiply the 0.0096 by $b_{N,man} = 0.1$

Q2: Viterbi Algorithm example

For subsequent rows, use the recurrence

$$mlp[i, q] = \begin{cases} \pi_q \cdot b_{q,s_1} & i = 1 \\ \max_{q^* \in Q} \{ mlp[i-1, q^*] \cdot \rho_{q^*,q} \cdot b_{q,s_i} \} & i > 1 \end{cases}$$

	DT	N	V	Adj
the	.4x.5 = .2	0	0	0
old	0	.2x.6x.2 = .024	0	.2x.4x.4 = .032
man	0	.3x.032x.5 = .0048	.1x.024x.4 = .00096	0
the				
lifeboats				

Q2: Viterbi Algorithm example

For subsequent rows, use the recurrence

$$mlp[i, q] = \begin{cases} \pi_q \cdot b_{q,s_1} & i = 1 \\ \max_{q^* \in Q} \{ mlp[i-1, q^*] \cdot p_{q^*,q} \cdot b_{q,s_i} \} & i > 1 \end{cases}$$

	DT	N	V	Adj
the	.4x.5 = .2	0	0	0
old	0	.2x.6x.2 = .024	0	.2x.4x.4 = .032
man	0	.3x.032x.5 = .0048	.1x.024x.4 = .00096	0
the		0	0	0
lifeboats				

the old man the: notice final item *the* is only possible at state DT

Q2: Viterbi Algorithm example

For subsequent rows, use the recurrence

$$mlp[i, q] = \begin{cases} \pi_q \cdot b_{q, s_1} & i = 1 \\ \max_{q^* \in Q} \{ mlp[i-1, q^*] \cdot p_{q^*, q} \cdot b_{q, s_i} \} & i > 1 \end{cases}$$

	DT	N	V	Adj
the	.4x.5 = .2	0	0	0
old	0	.2x.6x.2 = .024	0	.2x.4x.4 = .032
man	0	.3x.032x.5 = .0048	.1x.024x.4 = .00096	0
the	.5x.00096x.4 = 0.000192	0	0	0
lifeboats				

the old man the: notice final item *the* is only possible at state DT

$b_{DT, the} = 0.5$. Take max of $mlp[3, N] \times p_{N, DT} = 0.0048 \times 0.05$ against $mlp[3, V] \times p_{V, DT} = 0.00096 \times 0.4 = 0.000384$, multiply the 0.000384 by $b_{N, the} = 0.5$

Q2: Viterbi Algorithm example

For subsequent rows, use the recurrence

$$mlp[i, q] = \begin{cases} \pi_q \cdot b_{q,s_1} & i = 1 \\ \max_{q^* \in Q} \{ mlp[i-1, q^*] \cdot p_{q^*,q} \cdot b_{q,s_i} \} & i > 1 \end{cases}$$

	DT	N	V	Adj
the	.4x.5 = .2	0	0	0
old	0	.2x.6x.2 = .024	0	.2x.4x.4 = .032
man	0	.3x.032x.5 = .0048	.1x.024x.4 = .00096	0
the	.5x.00096x.4 = 0.000192	0	0	0
lifeboats	0	.00002304	0	0

lifeboats can only be output at N, *the old man the* can only end at DT $\Rightarrow 0.000192 \times 0.6 \times 0.2$

Q2: Viterbi Algorithm example

Thus the most probable tagging is:

The/DT old/N man/V the/DT lifeboats/N

(The backtrace pointers can be read off from the above matrix in an ad hoc fashion: e.g. in the cell for (man, N), the carried factor is .032 which comes from the cell for (old, Adj) ... though actually our “most likely” path doesn't label *man* with N)

Or we could have built the `prev` array as in the pseudocode for Viterbi.

Q2: Viterbi Algorithm example

prev array for students to verify their own.

	DT	N	V	Adj
the	-	-	-	-
old	-	DT	-	DT
man	-	Adj	N	-
the	V	-	-	-
lifeboats	-	DT	-	-

Q3: The Travelling Salesman Path

TSP problem: find a minimum cost path from the source to destination visiting all vertices **exactly once**.

- ▶ undirected graph $G = (V, E)$, start node s , end node t
- ▶ weight function $w : E \rightarrow \mathbb{Q}^+$
- ▶ ordering π s.t. $\pi_1 = s, \pi_n = t$ and $(\pi_i, \pi_{i+1}) \in E$ for all i .

$$TSP((V, E), s, t) = \begin{cases} w(s, t) & \text{if } V = \{s, t\} \\ \min_{\substack{u \in V \setminus \{s, t\}, \\ (u, t) \in E}} \{w(u, t) + TSP((V \setminus \{t\}, E_{\setminus \{t\}}, s, u))\} & \text{if } |V| \geq 3 \end{cases}$$

- ▶ Notice the recurrence in the above formula
- ▶ Can we build a dynamic programming algorithm for this problem?
- ▶ What are the four properties of a dynamic programming algorithm?