TDM02 Sequence Alignment Correction

Advanced Human Machine Interaction and Interaction Data Analysis

Part 1 Problem formalisation

One possible formalisation is:

- **Inputs:** two traces $X = (x_1, \ldots, x_m)$ and $Y = (y_1, \ldots, y_n)$ over a finite event alphabet A.
- **Output:** two subsequences of consecutive events, an alignment containing possible gaps, a similarity score and the half-open source intervals $[s_X, e_X)$ and $[s_Y, e_Y)$.
- **Problem class:** similarity-based local pattern extraction.
- **Model parameters:** match reward, substitution scores, gap costs and any minimum score or length used to accept a result.
- **Evaluation:** precision and recall against annotated patterns, stability when scores change, runtime, and usefulness for the intended interaction analysis.

The scoring model is part of the problem definition. It describes which behavioural differences should be treated as plausible variations of the same pattern.

Part 2 Smith–Waterman by hand

For $X = (A, B, C)$ and $Y = (A, X, C)$:

	–	A	X	C
–	0	0	0	0
A	0	2	0	0
B	0	0	1	0
C	0	0	0	3

Examples of cell computations:

$$H[1,1] = \max(0, 0+2, 0-2, 0-2) = 2,$$
$$H[2,2] = \max(0, 2-1, 0-2, 0-2) = 1,$$
$$H[3,3] = \max(0, 1+2, 0-2, 0-2) = 3.$$

The traceback follows three diagonal moves and produces

$$\begin{array}{ccc} A & B & C \\ A & X & C \end{array}$$

with score $2 - 1 + 2 = 3$. Both intervals are $[0, 3)$. Zero allows the algorithm to discard a negative prefix and start a new local alignment at any position.

Part 3 Implementation

The supplied file `smith_waterman_solution.py` provides a complete reference implementation. Important points include:

- the matrix has one extra row and column;
- the maximal position is recorded while the matrix is filled;
- traceback tests the diagonal move before deletion and insertion to make ties deterministic;
- traceback stops on a zero and returns the remaining indices as start positions;
- the aligned lists are reversed before being returned.

Different tie-breaking orders may produce different optimal alignments with the same score. This is expected and should not be treated as an implementation error.

Part 4 Influence of the scoring model

For $X = (1, 2, 3, 4)$ and $Y = (1, 2, 9, 3, 4)$, the reference implementation gives:

Match	Substitution	Gap	Score	One selected alignment
$+2$	-1	-1	7	`1 2 - 3 4` / `1 2 9 3 4`
$+2$	-1	-2	6	`1 2 - 3 4` / `1 2 9 3 4`
$+2$	-1	-4	4	`1 2` / `1 2`
$+2$	-3	-2	6	`1 2 - 3 4` / `1 2 9 3 4`

A strong gap penalty expresses the assumption that missing or additional events are unlikely. Here it makes a short exact fragment preferable to an alignment spanning event 9. The appropriate scores depend on the observation process. Gesture recognition may produce substitutions and missing detections, while page visits are usually observed exactly but may contain unrelated navigation events.

A simple normalisation is $score/L$, where L is the number of alignment columns. This facilitates comparisons across lengths but favours short exact matches. A minimum length or statistical significance model may therefore be necessary.

Part 5 Generated traces

Running the provided solution with seed 42, the first 30 traces of each scenario, and the default scores gives:

Scenario	Pairs	Mean score	Mean span	Single motif	Multiple motifs
Clean	435	14.28	11.15	4.6%	95.4%
Separated	435	9.76	8.37	23.2%	65.3%
Corrupted	435	6.72	4.78	43.4%	29.0%
Random control	435	6.45	5.91	0.0%	0.0%

Here, *single motif* means that both selected intervals lie inside occurrences of the same planted motif. *Multiple motifs* means that at least one selected interval overlaps more than one planted occurrence. These measures describe alignment boundaries; they do not measure complete motif recovery.

Interpretation

The clean dataset produces the highest scores, but most maximal alignments cross several motif boundaries. Concatenating exact motifs creates longer regular sequences and repeated cross-boundary fragments. A learner may therefore discover combinations created by the generator rather than the five intended motifs.

Separator events disrupt these accidental concatenations. They reduce the maximal scores but make individual motif boundaries easier to isolate. The corrupted scenario produces even shorter alignments, which explains its high single-motif containment rate. This does not mean that it recovers complete motifs best. Its mean score is close to the random control, so chance similarities become a serious concern.

The random control establishes how much similarity Smith–Waterman finds without planted structure. Without this baseline, a positive score or an apparently plausible fragment provides weak evidence.

The numerical results depend on the motif alphabet, motif lengths, separator alphabet, trace lengths, corruption rates and scoring model. For example, changing separator events to the same alphabet as motif events should increase accidental similarities and the proportion of boundary-crossing alignments.

Optional BLAST extension

Smith–Waterman guarantees an optimal local alignment under the selected scoring model and requires $O(mn)$ time for two sequences of lengths m and n. BLAST first searches for promising short seeds and then extends them. It reduces computation on large databases but may miss an alignment if no suitable seed triggers an extension. For the short synthetic traces used here, exhaustive dynamic programming remains appropriate. Seed-and-extend methods become relevant when the number or length of traces makes all pairwise matrices too expensive.