1.1 Edit (Levenshtein) distance

Edit distance, also known as Levenshtein distance or evolutionary distance is a concept from information retrieval and it describes the number of edits (insertions, deletions and substitutions) that have to be made in order to change one string to another. It is the most common measure to expose the dissimilarity between two strings (Levenshtein 1966; Navarro & Raffinot 2002).

The edit distance *ed(x, y)* between strings $x=x_1 \ldots x_m$ and $y=y_1 \ldots y_n$, where $x, y \in \Sigma*$ is the minimum cost of a sequence of editing steps required to convert *x* into *y*. The alphabet Σ of possible characters *ch* gives Σ∗, the set of all possible sequences of $ch \in \Sigma$. Edit distance can be calculated using dynamic programming (Navarro & Raffinot 2002). Dynamic programming is a method of solving a large problem by regarding the problem as the sum of the solution to its recursively solved subproblems. Dynamic programming is different to recursion however. In order to avoid recalculating the solutions to subproblems, dynamic programming makes use of a technique called *memoisation*, whereby the solutions to subproblems are stored once calculated, to save recalculation.

To compute the edit distance *ed(x,y)* between strings *x* and *y*, a matrix $M_{1\ldots m+1,1\ldots n+1}$ is constructed where $M_{i,j}$ is the minimum number of edit operations needed to match $x_{1\ldots i}$ to $y_{1\ldots j}$. Each matrix element $M_{i,j}$ is calculated as per Equation 1, where $\delta(a,b) = 0$ if $a = b$ and 1 otherwise. The matrix element $M_{1,1}$ is the edit distance between two empty strings.

$$M_{1,1} \leftarrow 0$$

$$M_{i,j} \leftarrow \min \begin{cases} M_{i-1,j} + 1 \\ M_{i,j-1} + 1 \\ M_{i-1,j-1} + \delta(x_i, y_j) \end{cases}$$

Equation 1

The algorithm considers the last characters, x_i and y_j. If they are equal, then $x_{1..i}$ can be converted into $y_{1..j}$ at a cost of $M_{i-1,j-1}$. If they are not equal, x_i can be converted to y_j by substitution at a cost of $M_{i-1,j-1} + 1$, or x_i can be deleted at a cost of $M_{i-1,j} + 1$ or y_j can be appended to *x* at a cost of $M_{i,j-1} + 1$. The minimum edit distance between *x* and *y* is given by the matrix entry at position $M_{m+1,n+1}$.

Table 1 is an example of the matrix produced to calculate the edit distance between the strings "DFGDGBDEGGAB" and "DGGGDGBDEFGAB". The edit distance between these strings given as $M_{m+1,n+1}$ is 3.

		D	**G**	**G**	**G**	**D**	**G**	**B**	**D**	**E**	**F**	**G**	**A**	**B**
	0	1	2	3	4	5	6	7	8	9	10	11	12	13
D	1	0	1	2	3	4	5	6	7	8	9	10	11	12
F	2	1	1	2	3	4	5	6	7	8	8	9	10	11
G	3	2	1	1	2	3	4	5	6	7	8	8	9	10
D	4	3	2	2	2	2	3	4	5	6	7	8	9	10
G	5	4	3	2	2	3	2	3	4	5	6	7	8	9
B	6	5	4	3	3	3	3	2	3	4	5	6	7	8
D	7	6	5	4	4	3	4	3	2	3	4	5	6	7
E	8	7	6	5	5	4	4	4	3	2	3	4	5	6
G	9	8	7	6	5	5	4	5	4	3	3	3	4	5
G	10	9	8	7	6	6	5	5	5	4	4	3	4	5
A	11	10	9	8	7	7	6	6	6	5	5	4	3	4
B	12	11	10	9	8	8	7	6	7	6	6	5	4	3

Table 1: Edit distance matrix for the strings "DFGDGBDEGGAB" and "DGGGDGBDEFGAB" with the minimum edit distance position highlighted

An alternative expression of the edit distance equation which gives identical results is given in Equation 2, which is equivalent to Equation 1 because neighbouring cells in *M* differ by at most 1.

$$M_{i,1} \leftarrow i-1, M_{1,j} \leftarrow j-1$$

$$M_{i,j} \leftarrow \begin{cases} M_{i-1,j-1} & \text{if } x_i = y_i \\ 1 + \min\left(M_{i-1,j-1}, M_{i-1,j}, M_{i,j-1}\right) & \text{else} \end{cases}$$

Equation 2

The algorithm can be adapted to find the lowest edit distances for *x* in substrings of *y*. This is achieved by setting $M_{1,j} = 0$ for all $j \in 1...n+1$. In contrast to the edit distance algorithm described above, the last row $M_{m+i,j}$ is then used to give a *sliding window* edit distance for *x* in substrings of *y* as per Equation 3 (Navarro & Raffinot 2002).

$$eds(x,y) = \min_{1 \geq j \geq n+1}\left(M_{m+1,j}\right)$$

Equation 3

An example of this variation on the edit distance applied to search for the pattern "BDEE" in "DGGGDGBDEFGAB" is given in Table 2. The minimum edit distance positions are highlighted. Variations on the edit distance algorithm have been applied in domains such as DNA analysis and automated spell checking and are commonly used in MIR systems (Birmingham et al. 2001; Lemstrom & Perttu 2000; Rho & Hwang 2004; McPherson & Bainbridge 2001; Prechelt & Typke 2001).

		D	**G**	**G**	**G**	**D**	**G**	**B**	**D**	**E**	**F**	**G**	**A**	**B**
	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	1	1	1	1	1	1	1	0	1	1	1	1	1	0
D	2	1	2	2	2	1	2	1	0	1	2	2	2	1
E	3	2	2	3	3	2	2	2	1	0	1	2	3	2
E	4	3	3	3	4	3	3	3	2	1	1	2	3	3

Table 2: Edit distance for the string "BDEE" in "DGGGDGBDEFGAB". This string represents the first 13 notes from the tune "Jim Coleman's" in normalised ABC format