DATA130030.01, Natural Language Processing (*15-30 minutes to read*) 03/06/2024

Lecture-01: Minimum Edit Distance (MED)

Lecturer: Baojian Zhou *The School of Data Science, Fudan University*

Assessing the similarity between two strings is essential in many NLP tasks. For spell correction, one must choose the most similar word to a misspelled input from a set of candidates. For example, if "graffe" is the input, among the options "graf", "graft", "grail", and "giraffe", the task is to select the word closest to the input based on a certain criterion. Similarly, comparing a translated sentence to a reference translation helps evaluate the performance of translation models in machine translation. This process may involve calculating the minimum edit distance to determine the similarity. This note gives a dynamic programming algorithm for MED.

1.1 String similarity problem and MED

Definition 1 (Minimum Edit Distance (MED)) *Given two strings $X = [x_1, x_2, \ldots, x_n]$ and $Y = [y_1, y_2, \ldots, y_m]$, the minimum edit distance between X and Y is the minimum number of edit operations required to convert X into Y. The allowable edit operations are: 1. Insertion - Adding a character to X; 2. Deletion - Removing a character from X; and 3. Substitution - Replacing a character in X with another character. These operations are performed sequentially from the first (leftmost) character to the last (rightmost) in X to obtain the string Y.*

Example. Suppose $X =$ INTENTION and $Y =$ EXECUTION, a possible process of $X \rightarrow Y$ is

$$X = \underline{\text{I}}\text{NTENTION} \xrightarrow{o_1=\text{Del}(I)} \cancel{\text{N}}\text{TENTION} \xrightarrow{o_2=\text{Sub}(N,E)} \text{E}\cancel{\text{T}}\text{ENTION} \xrightarrow{o_3=\text{Sub}(T,X)}$$

$$\text{EXE}\wedge\text{NTION} \xrightarrow{o_4=\text{Ins}(C)} \text{EXEC}\cancel{\text{N}}\text{TION} \xrightarrow{o_5=\text{Sub}(N,U)} \text{EXECUTION} = Y.$$

We assume that Ins and Del each count as 1 operation, while Sub counts 2 operations. Consequently, the total number of edit operations above is 8. Let $O_k = [o_1, o_2, \ldots, o_k]$ represent the sequence of operations transforming X into Y, where each o_i is one of the operations $\{\text{Ins, Del, Sub}\}$. The sequence O_k is referred to as an *alignment* of X into Y, as illustrated in Fig. 1.1.

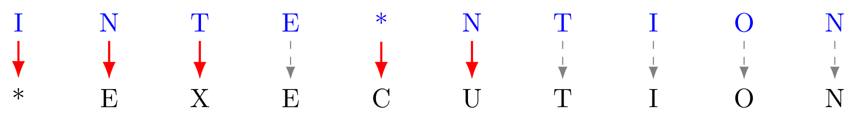

Figure 1.1: An alignment of $X \rightarrow Y$. Operations O_k are from left of X to right. The total number of operations is denoted as $\text{cost}(O_k) = 8$. The dashed arrows copy characters without any cost.

1.2 Dynamic programming for MED

A naive approach would involve examining all possible edit sequences O_k and choosing the one with the minimal cost. However, this method has an exponentially large time complexity. Instead, one might consider breaking the problem into smaller subproblems and take advantage of optimal substructure.

Problem settings. For $i = 0, 1, \ldots, n$ and $j = 0, 1, \ldots, m$, let $X_i = [x_1, x_2, \cdots, x_i]$ and $Y_j = [y_1, y_2, \cdots, y_j]$ be substrings of X and Y, respectively. By default, $X_0 = \emptyset$ and $Y_0 = \emptyset$. Define $D[i, j]$ as MED of $X_i \to Y_j$. Clearly, $D[n, m]$ is MED of $X \to Y$. When $i = 0$ or $j = 0$, one can trivially calculate $D[0, j]$ or $D[i, 0]$ by inserting all y_j into $\emptyset$ or deleting all x_i from X_i. Hence, $D[i, 0] = i$ and $D[0, j] = j$. In the rest, we assume $i, j \geq 1$.

Optimal substructure. Let $O_k = [o_1, o_2, \ldots, o_k]$ be minimal operations of $X_i \to Y_j$, then we have

$$X_i = X_i^0 \xrightarrow{o_1} X_i^1 \xrightarrow{o_2} X_i^2 \to \cdots \xrightarrow{o_{k-1}} X_i^{k-1} \xrightarrow{o_k} X_i^k = Y_j.$$

By cutting out the last operation o_k, one key observation is that operations $O_{k-1} = [o_1, o_2, \ldots, o_{k-1}]$ is an optimal process for $X_i \to X_i^{k-1}$. We can prove this by making a contradiction: let us assume the O_{k-1} is not the optimal operation of $X_i \to X_i^{k-1}$, then there must be optimal operations O'_{k-1} for $X_i \to X_i^{k-1}$ with $\text{cost}(O'_{k-1}) < \text{cost}(O_{k-1})$. Then the total cost of O'_{k-1} plus $\text{cost}(o_k)$ is less than $\text{cost}(O_k)$, which makes a contradiction since $\text{cost}(O_k)$ is the minimal cost for $X_i \to Y_j$. This property is called the *optimal substructure* property.

As operations are from left of X_i to right, there are only 3 possibilities for o_k:

- **Case 1.** x_i is deleted from X_i. Then $D[i, j]$ can be calculated as $D[i, j] = D[i-1, j] + \text{Del}(x_i)$.
- **Case 2.** y_j is inserted into Y_{j-1}. Then $D[i, j] = D[i, j-1] + \text{Ins}(y_j)$.
- **Case 3.** x_i is substituted by y_j. Then $D[i, j] = D[i-1, j-1] + \text{Sub}(x_i, y_j)$.

Therefore, we can solve the problem of MED for $X_i \to Y_j$ by leveraging three subproblems $D[i-1, j], D[i, j-1]$, and $D[i-1, j-1]$. That is, we need to choose the minimal one among these 3 cases

$$D[i, j] = \min \begin{cases} D[i-1, j] + \text{Del}(x_i) \\ D[i, j-1] + \text{Ins}(y_j) \\ D[i-1, j-1] + \text{Sub}(x_i, y_j). \end{cases}$$

The above formula is called dynamic procedure or dynamic programming. Given the above-mentioned example, we can calculate matrix $\boldsymbol{D}$ using this dynamic procedure as shown in the following tables.

N	9									
O	8									
I	7									
T	6									
N	5									
E	4									
T	3									
N	2									
I	1									
#	0	1	2	3	4	5	6	7	8	9
	#	E	X	E	C	U	T	I	O	N

$D[\cdot, \cdot]$ at the initial stage.

N	9	8	9	10	11	12	11	10	9	8
O	8	7	8	9	10	11	10	9	8	9
I	7	6	7	8	9	10	9	8	9	10
T	6	5	6	7	8	9	8	9	10	11
N	5	4	5	6	7	8	9	10	11	10
E	4	3	4	5	6	7	8	9	10	9
T	3	4	5	6	7	8	7	8	9	8
N	2	3	4	5	6	7	8	7	8	7
I	1	2	3	4	5	6	7	6	7	8
#	0	1	2	3	4	5	6	7	8	9
	#	E	X	E	C	U	T	I	O	N

$D[n, m] = 8$ at the final stage.

Remark 1 *We call the above method a dynamic programming method. 1) Try to prove or disprove the uniqueness of the optimal operations; and 2). Can you find approximate solutions if n and m are large? (Hint: find "approximate string matching").*