① Referential Ambiguity → when it's unclear what a word or phrase refers to. Ex- "The old man and the young woment were sitting on the bench" - who is sitting with whom?

② Two strings and their alignment.

I	N	T	E	*	N	T	I	O	N
|	|	|	|	|	|	|	|	|	|
*	E	X	E	C	U	T	I	O	N

The gap between intention and execution is 5.

if each operation has a cost of 1 (levenshtein) distance between this is 5

if substitution costs 2 (alternate version) distance between these is 8.

Searching for a path (sequence of edits) from the start to the final string:

initial state : the word we are tranforming.

Operators : insert, delete, substitute.

Goal state : the word we are trying to get to

Path Cost : what we want to minimize: the no of edits.

intention

del ins subst

ntention intecntion inxention

in t en tion ← delete ;

n t e n t i o n ← substitute n by e

e t en tion ← substitute t by x

exention

exenution ← insert u

execution ← substitute n by c

Dynamic programming

A tabular computation of D(n; m)

Solving problems by combining solutions to subproblems Bottom-up

Compute D(i, j) for small i; j

compute large D(i; j) based on previously computed smaller values.

Compute D(i; j) for all i and j till get D(n; m)

minimum edit distance between two strings -

$$D[i,j] = \min \begin{cases} D[i-1,j]+1 \\ D[i,j-1]+1 \\ D[i-1,j-1] + \begin{cases} 2; & \text{if } source[i] \neq target; \\ 0; & \text{if } source[i] = target; \end{cases} \end{cases}$$

function MIN-EDIT-DISTANCE (target, source)
returns min-distance

$n \leftarrow$ LENGTH (target)

$m \leftarrow$ LENGTH (source)

Create a distance matrix distance $[n+1, m+1]$

Initialize the zeroth row and column to be the distance from the empty string

distance $[0,0] = 0$

for each column i from 1 to n do
    distance $[i,0] \leftarrow$ distance $[i-1,0]$ + ins-cost (target[i])

for each row j from 1 to m do
    distance $[0,j] \leftarrow$ distance $[0,j-1]$ + del-cost (source[j])

for each column j from 1 to n do
    for each row j from 1 to m do
        distance $[i,j] \leftarrow$ MIN (distance $[i-1,j]$ + ins-cost (target$_{i-1}$),

distance [i-1, j-1] + subst-cost(source'j-1, target i-1)
distance [i, j-1] + del-cost(source j-1)
return distance [n, m]

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	8
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	~~8~~ 6	7	8
#	0	1	2	3	4	5	6	7	8	9
	#	e	x	e	c	a	t	i	o	n.