Mod 2 - CO2 - 2024-08-20 - Apply word level analysis

20 August 2024 09:23

-> I am writing this email on behaf of ....

↳ user : <u>behaf</u>

-> Close words : behalf & behave

-> <u>Spelling Correction</u> :

-> <u>Isolated Word Error Correction</u> ;

-> '<u>behaf</u>'

-> Define '<u>closest</u>' ?

↳ Distance Metric

↳ <u>Edit Distance</u> ;

-> Edit Distance : Spelling Correction Algo .

↳ Minimum edit distance b/w 2 strings .

↳ Is the minimum number of editing operations.

↳ Insertion, Deletion & Substitution.

-> Minimum Edit Distance :

'intention' to 'execution'.

	I	N	T	E	*	N	T	I	O	N
del	*	E	X	E	C	U	T	I	O	N
		sub	sub		ins	sub				

d s s i s

-> If each operation has a cost of 1 (Levenshtein)

-> Distance between these is 5

-> If substitution costs 2

-> Distance between these is 8

-> Searching for a path (Sequence of Edits) from start string to final string.

-> Initial State : The word we are transforming.

-> Operators : Insert, Delete, Substitute.

- Operations : insert, delete, substitute.

-> Goal : Word we are trying to get.

-> Path Cost : Minimize the number of edits.

-> Navigation:

- -> The space of all edit sequences is huge.
- -> Distinct paths that end up on same state.
- -> Cannot keep track of all.
- -> Keeping track of shortest path.

-> For 2 strings:

- -> X of length n
- -> Y of length m
- -> D(i, j)
- -> Edit distance b/w x[1...i] & y[1...j]

-> Dynamic Programming:

-> **Dynamic Programming** :

-> Tabular Computation (Memoization)

-> Combining solutions to subproblems .

-> Bottom-up :

↳ Compute $D(i,j)$ for small i & j .

↳ Compute longer $D(i,j)$ based on previously computed smaller values .

↳ Compute $D(i,j)$ for all i & j till you get to $D(N,M)$.

-> Initialization, Recurrence, Termination.

-> Initialization :

$D(i, 0) = i$

$D(0, j) = j$

$n = 1024$

2^{1024} & 1024^2

-> Recurrence Relation :

```
for each $i = 1 \ldots M$
    for each $j = 1 \ldots N$
```

j=

for each j=1 ..... N

$$D(i,j) = \min \begin{cases} D(i-1,j)+1 \\ D(i,j-1)+1 \\ D(i-1,j-1)+ \begin{cases} 2 & \text{if } x(i) \neq y(j) \\ 0 & \text{if } x(i) = y(j) \end{cases} \end{cases}$$

Termination:

D(N,M) is distance.

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

i

(N,M) D

D(i-1,j)+1

1+1=2

D(i,j-1)+1

1+1=2

D(i-1,j-1)=

0+2=2

3,3,3

4,4,4

i

I N T E N T I O N

E X E C U T I O N

j

-> **Computing Alignments:** (Backtracking)

Base Conditions

$D(i, 0) = i$

$D(0, j) = j$

Recurrence Relation:

```
for each $i = 1 \dots M$
    for each $j = 1 \dots N$
```

$$D(i,j) = \min \begin{cases} D(i-1,j)+1 & \boxed{\text{del}} \\ D(i,j-1)+1 & \boxed{\text{ins}} \\ D(i-1,j-1) + \begin{cases} 2 & \text{if } x(i) \neq y(j) \\ 0 & \text{if } x(i) = y(j) \end{cases} & \boxed{\text{sub}} \end{cases}$$

$$\text{ptr}(i,j) = \begin{cases} \text{left} & \boxed{\text{ins}} \\ \text{down} & \boxed{\text{del}} \\ \text{diagonal} & \boxed{\text{sub}} \end{cases}$$

Termination

$D(N, M)$ is distance.

e	3	2	3 ←	2
t	2	3	3	1 ↓
->a	1	2	1 ↓	2
#	0 ←	1 ↙	2	3

$$\begin{cases} D(i-1, j) + 1 \\ \quad = 3 \\ D(i, j-1) + 1 = 3 \end{cases}$$

		1	2
#	0 ← 1 ←	2	3
#	e	a	t

$$\begin{cases} D(i, j-1)+1 \to 3 \\ D(i-1, j-1) \to \textcircled{1} \end{cases}$$

$D(N, M) = \underline{\underline{2}}$

Performance:

Time $O(nm)$

Space $O(nm)$

Backtracking $O(n+m)$

→ Mistyped Words : Weighted Minimum Edit Distance:

$y = mx + b$

$\hat{y} = wx + b$ independent var

weight bias

(arbitrary parameter)

Initialization:

$D(0, 0) = 0$

$D(i, 0) = D(i-1, 0) + del[x[i]]; \quad 1 < i \leq N$

$D(i,0) = D(i-1,0) + del[x[i]]; \quad 1 < i \le N$

$D(0,j) = D(0,j-1) + ins[y[j]]; \quad 1 < j \le M$

RR :

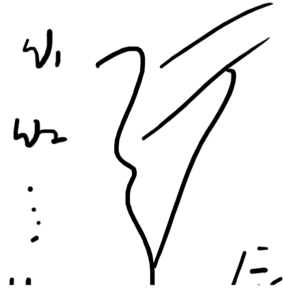
$$D(i,j) = \min \begin{cases} D(i-1,j) + del[x(i)] \\ D(i,j-1) + ins[y(j)] \\ D(i-1,j-1) + sub[[x(i), y(j)] \end{cases}$$

-> Metathesis : (Modification w/ transpose)

-> Transpose $(x,y) = (y,x)$

$$D(i,j) = \min \begin{cases} ins \\ del \\ sub \\ \\ D(i-2,j-2)+1 \quad : \text{Transpose} \\ \qquad \text{if } x[i-1] = y[j] \\ \qquad \&\& \\ \qquad x[i] = y[j-1] \end{cases}$$

Wold cup



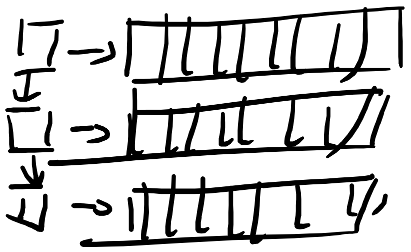
world
would } Suitable candidate

Edit Distanc

⋮
wn } Edit Distance

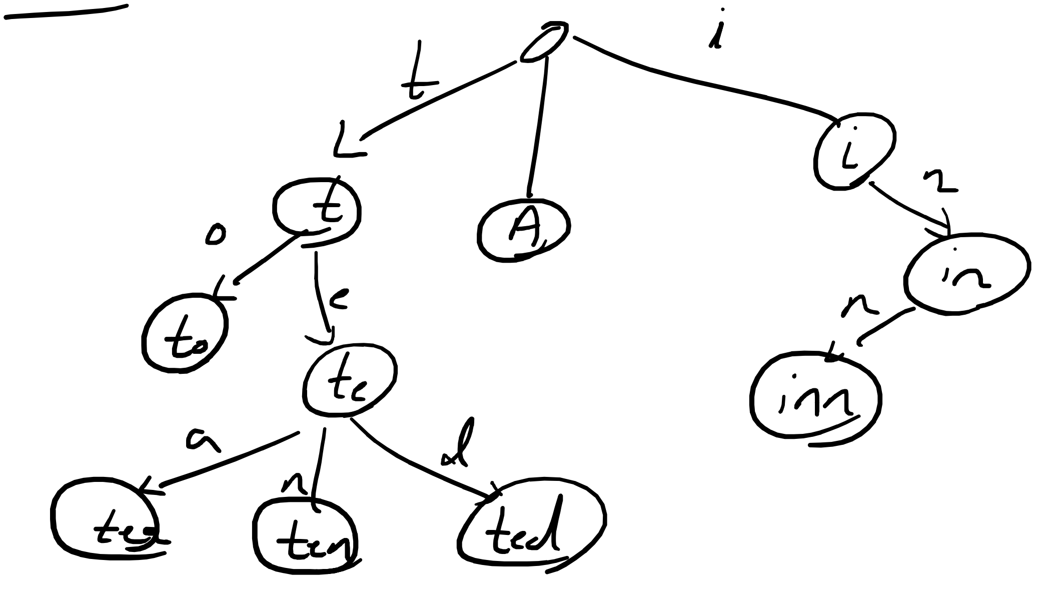
-> Dictionary Entries with smallest edit distance.

-> String : ~~Array~~, ~~LL~~, ~~Queue~~, ~~Stack~~, Tree & ~~Graph~~



Tree
{ Binary Search Tree
&
Red-Black Tree }

Trie :



-> Symmetric Delete Spelling Correction

-> Generate terms with edit distance ≤ 2 from each dict term.

-> Generate terms with edit distance ≤ 2

-> Generate terms with edit distance ≤ 4 from their input terms & search in dict.