Comp 408 Advanced Topics in Artificial Intelligence

Lecture 3

Minimum Edit Distance

22/ 4 / 2025

Most of the Slides are by D. Jurafsky and J. M. Martin

How similar are two strings?

- Spell correction
  - The user type “graffe”

  Which is closest?
    - graf
    - graft
    - grail
    - giraffe

Text similarities also required, for Machine Translation, Information Extraction, Speech Recognition

Edit Distance

- The minimum edit distance between two strings
- Is the minimum number of editing operations
  - Insertion
  - Deletion
  - Substitution
- Needed to transform one string into the other

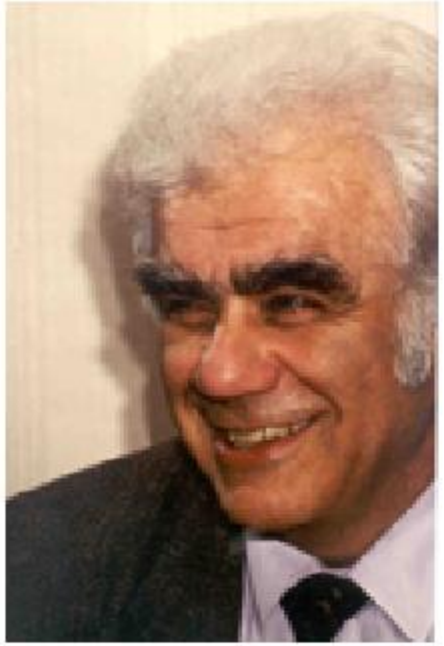
Minimum Edit Distance

- Two strings and their **alignment**:

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

Minimum Edit Distance

```
I N T E * N T I O N
| | | | | | | | | |
* E X E C U T I O N
d s s   i s
```



- If each operation has cost of 1
  - Distance between these two strings is 5
- If substitutions cost 2 (Levenshtein) and other operations cost 1
  - Distance between them is 8 (1 +2 +2 +1+2)

Uses of Edit Distance in NLP

- Evaluating Machine Translation and speech recognition

R Spokesman confirms senior government adviser was shot human translation

H Spokesman said the senior adviser was shot dead machine translation

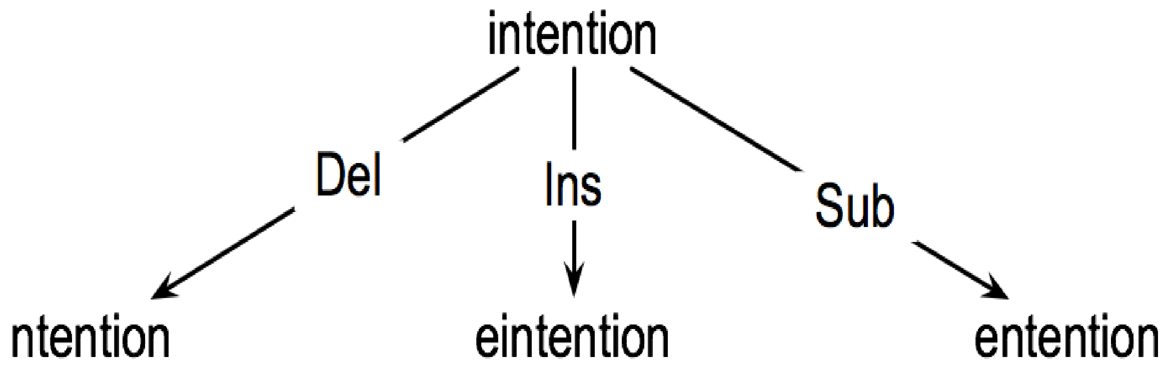
S I D I

S: Substitution, I: insertion, D: deletion

- Named Entity Extraction and Entity Coreference
  - IBM Inc. announced today
  - IBM profits
  - Stanford President John Hennessy announced yesterday
  - for Stanford University President John Hennessy

How to find the Min Edit Distance?

- Searching for a path (sequence of edits) from the start string to the final string:
  - **Initial state**: the word we're transforming
  - **Operators**: insert, delete, substitute
  - **Goal state**: the word we're trying to get to
  - **Path cost**: what we want to minimize: the number of edits



Minimum Edit as Search

- But the space of all edit sequences is huge!
  - We can’t afford to navigate naïvely
  - Lots of distinct paths wind up at the same state.
    - We don’t have to keep track of all of them
    - Just the shortest path to each of those revisited states.

Defining Min Edit Distance

- For two strings
  - X of length n
  - Y of length m
- We define $D(i, j)$
  - the edit distance between $X[1..i]$ and $Y[1..j]$
    - i.e., the first i characters of X and the first j characters of Y
  - The edit distance between X and Y is thus $D(n, m)$

Dynamic Programming for Minimum Edit Distance

- **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
  - And compute larger D(i, j) based on previously computed smaller values
  - i.e., compute D(i, j) for all i ($0 < i < n$) and j ($0 < j < m$)

Defining Min Edit Distance (Levenshtein)

- **Initialization**

  D(i, 0) = i   // the cost of i and null string is the cost of deleting those i

  D(0, j) = j // the cost of the null string and j is the cost of inserting j

- **Recurrence Relation:**

  For each i = 1…N

  For each j = 1…M

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

- **Termination:**

  D(N, M) is the distance

Minimum edit distance algorithm

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

$n \leftarrow$ LENGTH(*source*)
$m \leftarrow$ LENGTH(*target*)
Create a distance matrix $D[n+1, m+1]$

\# *Initialization: the zeroth row and column is the distance from the empty string*
$D[0,0] = 0$
for each row i **from** 1 **to** n **do**
  $D[i,0] \leftarrow D[i\text{-}1,0] + \textit{del-cost}(\textit{source}[i])$
for each column j **from** 1 **to** m **do**
  $D[0,j] \leftarrow D[0,j\text{-}1] + \textit{ins-cost}(\textit{target}[j])$

Filling the first column

Filling the first row

\# *Recurrence relation:*
for each row i **from** 1 **to** n **do**
  for each column j **from** 1 **to** m **do**
    $D[i,j] \leftarrow \text{MIN}(\, D[i-1,j] + \textit{del-cost}(\textit{source}[i]),$
        $D[i-1,j-1] + \textit{sub-cost}(\textit{source}[i], \textit{target}[j]),$
        $D[i,j-1] + \textit{ins-cost}(\textit{target}[j]))$
\# *Termination*
return D[n,m]

The Edit Distance Table

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

The Edit Distance Table

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

$$D(i,j) = \min \begin{cases} D(i-1,j) + 1 \\ D(i,j-1) + 1 \\ D(i-1,j-1) + \begin{cases} 2; & \text{if } S_1(i) \neq S_2(j) \\ 0; & \text{if } S_1(i) = S_2(j) \end{cases} \end{cases}$$

Edit Distan

$$D(i,j) = \min \begin{cases} D(i-1,j) + 1 \\ D(i,j-1) + 1 \\ D(i-1,j-1) + \begin{cases} 2; & \text{if } S_1(i) \neq S_2(j) \\ 0; & \text{if } S_1(i) = S_2(j) \end{cases} \end{cases}$$

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

The Edit Distance Table

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

Computing alignments

- Edit distance isn't sufficient
  - We often need to **align** each character of the two strings to each other
- We do this by keeping a "backtrace"
- Every time we enter a cell, remember where we came from
- When we reach the end,
  - Trace back the path from the upper right corner to read off the alignment

Edit Distance

$$D(i,j) = \min \begin{cases} D(i-1,j) + 1 \\ D(i,j-1) + 1 \\ D(i-1,j-1) + \begin{cases} 2; & \text{if } S_1(i) \neq S_2(j) \\ 0; & \text{if } S_1(i) = S_2(j) \end{cases} \end{cases}$$

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

Minimum Edit with Backtrace

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**	

Adding Backtrace to Minimum Edit Distance

- **Base conditions:**

  $D(i, 0) = i \qquad D(0, j) = j$

Termination:

$D(N, M)$ is distance

- **Recurrence Relation:**

  For each $i = 1 \ldots N$

  For each $j = 1 \ldots M$

$$D(i, j) = \min \begin{cases} D(i-1, j) + 1 & \text{deletion} \\ D(i, j-1) + 1 & \text{insertion} \\ D(i-1, j-1) + \begin{cases} 2; & \text{if } X(i) \neq Y(j) \\ 0; & \text{if } X(i) = Y(j) \end{cases} & \text{substitution} \end{cases}$$

$$\mathtt{ptr(i,j)} = \begin{cases} \mathtt{Left} & \text{insertion} \\ \mathtt{DOWN} & \text{deletion} \\ \mathtt{DIAG} & \text{substitution} \end{cases}$$

Result of Backtrace

- Two strings and their **alignment**:

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

Performance

- Time:

  $O(nm)$
- Space:

  $O(nm)$
- Backtrace

  $O(n+m)$

Note

- Minimum cost edit distance can be accomplished in multiple ways:
  - Source: actress; target: crest

Cost = 4

```
a c t r e s _ s
  |   | | |
_ c _ r e s t _
```

Cost = 4

```
a c t r e s s _
  |   | |   |
_ c _ r e _ s t
```

```
a c t r e s s _
  |   | | |
_ c _ r e s _ t
```

Cost = 4
3 deletion + 1 insertion

```
a c t r e s s
  |   | | |
_ c _ r e s t
```

Cost = 4;
2 deletion + 1 substitution

s	7	6	5	4	3	4	↓
s	6	5	4	3	2	←3	
e	5	4	3	2	3	4	
r	4	3	2	3	4	5	
t	3	↓2	3	4	5	4	
c	2	1	2	3	4	5	
a	↓1	2	3	4	5	6	
#	0	1	2	3	4	5	
	#	c	r	e	s	t	

The alignment corresponding to the red tracing

```
a c t r e s _ s
  |   | | |
_ c _ r e s t _
```

d d i d

First, s is deleted, we have a down arrow, then t is inserted, …

H. W.

- Compute the Levenshtein minimum edit distance of each of the following pairs of source and target strings. Then, find out the alignment using backtrace.
  1. azced and abcdef
  2. Saturday and Sunday
  3. kitten and sitting
  4. horse and rose
  5. leda to deal

H. W.

- Figure out whether drive is closer to brief or to divers and what the edit distance is to each.