

Minimum Edit Distance

Definition of Minimum Edit Distance

How similar are two strings?

Spell correction

- The user typed “graffe”

Which is closest?

- graf
- graft
- grail
- giraffe

Which candidate would require the minimum number of letter changes?

Similarity and Alignment in Computational Biology

We can compute similarity of two sequences of bases:

```
AGGCTATCACCTGACCTCCAGGCCGATGCCC  
TAGCTATCACGACCGCGGTTCGATTTGCCCGAC
```

And we can compute an **alignment** between them:

```
-AGGCTATCACCTGACCTCCAGGCCGA--TGCCC---  
TAG-CTATCAC--GACCGC--GGTCGATTTGCCCGAC
```

I.e., given two sequences, align each letter to a letter or gap

Evaluating Automatic Speech Recognition (ASR) and Machine Translation (MT)

- We want to know which hypothesis is closer to a "reference" transcript
 - Measure edit distance (in words, or tokens) between hypotheses and referent
 - The better hypothesis is closer (has a lower edit distance) to the referent

Reference Spokesman confirms senior government adviser was replaced

Hypothesis1 Spokesman confirms the senior adviser was replaced

I D

Hypothesis2 Spokesman said the older adviser was fired

S I S D S

Edit Distance

The minimum edit distance between two strings

Is the minimum number of editing operations

- Insertion
- Deletion
- Substitution

Needed to transform one into the other

Minimum Edit Distance

Two strings and their **alignment**:

Given two sequences, an alignment is a correspondence between substrings of the two sequences, like the individual letters in this case

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

We can read off the edit distance from the alignment

```
  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 is 5

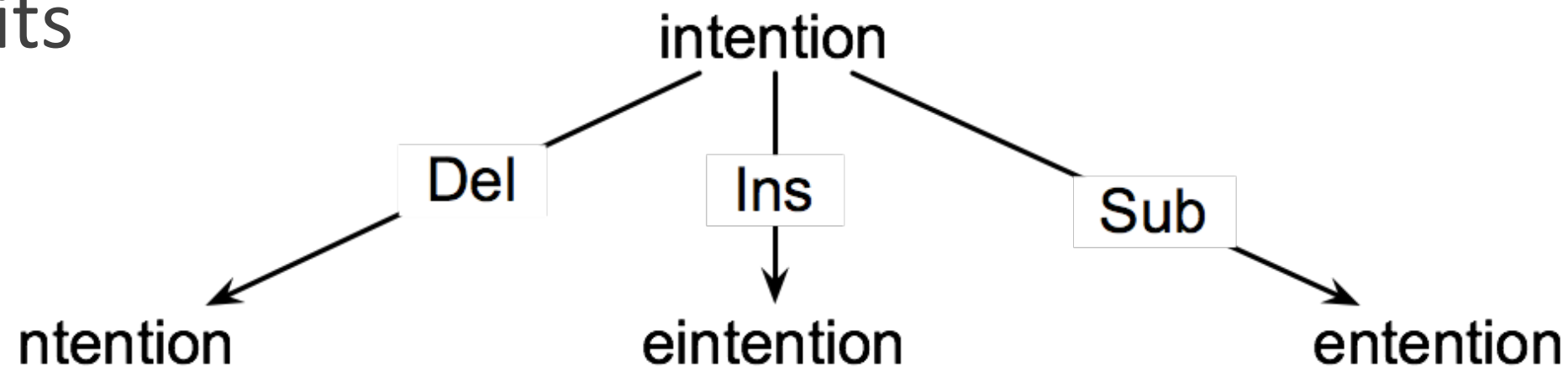
If substitutions cost 2 (a version of Levenshtein distance)

- Distance between them is 8

How to find the Min Edit Distance?

Searching for a path (a 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 naively

Luckily:

- 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.
- We'll see a dynamic programming solution in the next lecture

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 $D(n,m)$

Minimum Edit Distance

Definition of Minimum Edit Distance



Dynamic Programming for Minimum Edit Distance

- **Dynamic programming:** A tabular computation of $D(n,m)$
- Solving problems by combining solutions to subproblems.
- Bottom-up
 - We 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

$$D(i, 0) = i$$

$$D(0, j) = j$$

- Recurrence Relation:

For each $i = 1..M$

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



The Edit Distance Table

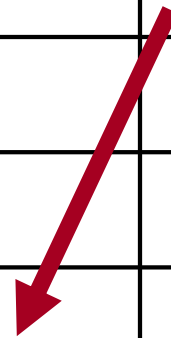
N	9									
O	8									
I	7									
T	6									
N	5									
E	4									
T	3									
N	2									
I	1									
#	0	1	2	3	4	5	6	7	8	9
	#	E	X	E	C	U	T	I	O	N



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 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



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



MinEdit 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:

$$D(i, 0) = i$$

$$D(0, j) = j$$

- Termination:

$$D(N, M) \text{ is distance}$$

- Recurrence Relation:

For each $i = 1..M$

For each $j = 1..N$

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

$$\begin{cases} 2; & \text{if } X(i) \neq Y(j) \\ 0; & \text{if } X(i) = Y(j) \end{cases}$$

$$\text{ptr}(i, j) = \begin{cases} \text{LEFT} \\ \text{DOWN} \\ \text{DIAG} \end{cases}$$

deletion

insertion

substitution

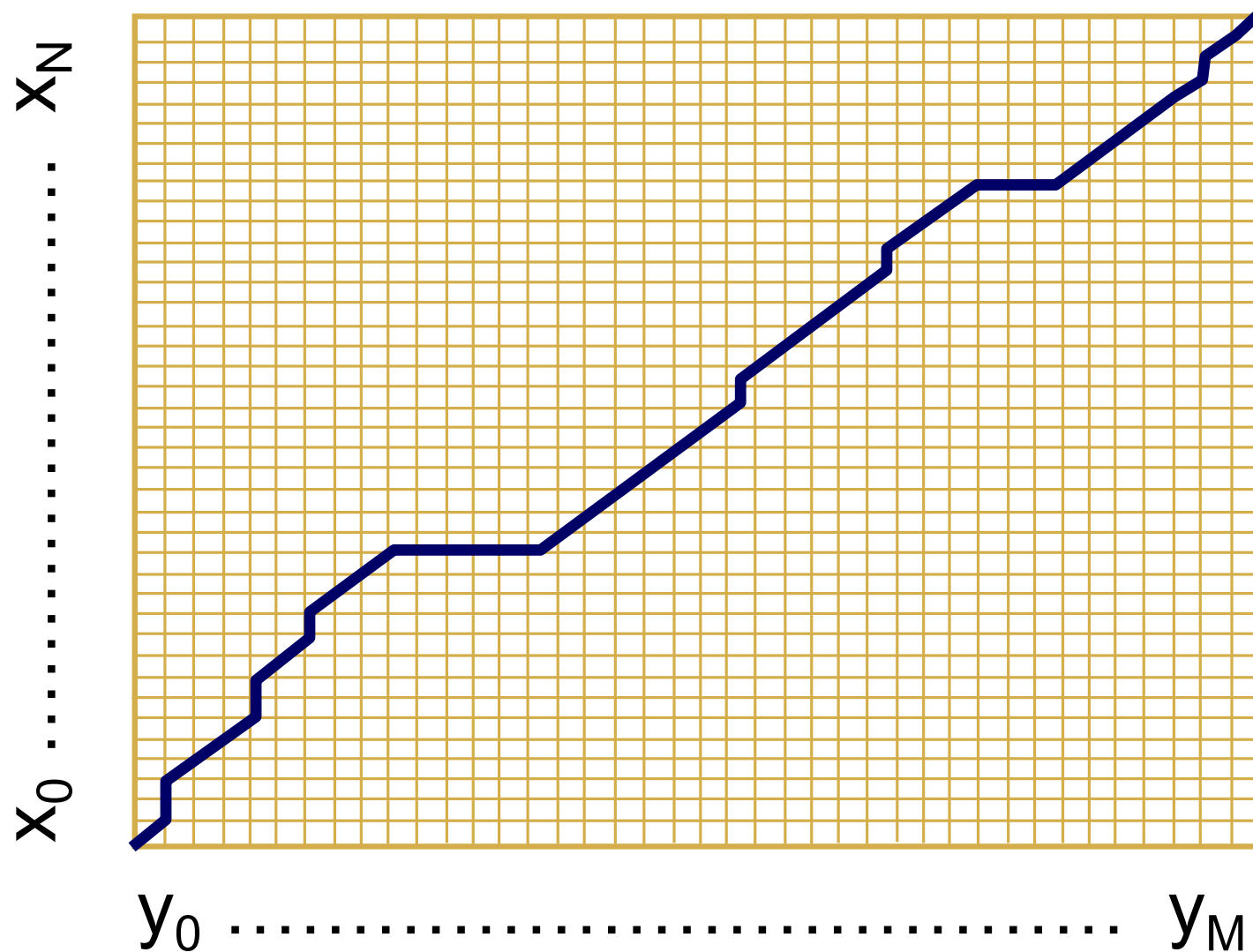
insertion

deletion

substitution



The Distance Matrix



Every non-decreasing path
from $(0,0)$ to (M, N)
corresponds to
an alignment
of the two sequences

An optimal alignment is composed
of optimal subalignments



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

$$O(n+m)$$

