Spelling Correction: Edit Distance

Pawan Goyal

CSE, IITKGP

Week 2: Lecture 1

Spelling Correction

Spelling Correction

I am writing this email on behaf of ...

Spelling Correction

I am writing this email on behaf of ...

The user typed ‘behaf’.

Which are some close words?

Spelling Correction

I am writing this email on behaf of ...

The user typed 'behaf'.

Which are some close words?

- behalf
- behave
- ....

Spelling Correction

I am writing this email on behaf of ...

The user typed 'behaf'.

Which are some close words?

- behalf
- behave
- ....

Isolated word error correction

- Pick the one that is closest to 'behaf'

Spelling Correction

I am writing this email on behaf of ...

The user typed ‘behaf’.

Which are some close words?

- behalf
- behave
- ....

Isolated word error correction

- Pick the one that is closest to ‘behaf’
- How to define ‘closest’?

Spelling Correction

I am writing this email on behaf of ...

The user typed 'behaf'.

Which are some close words?

- behalf
- behave
- ....

Isolated word error correction

- Pick the one that is closest to 'behaf'
- How to define 'closest'?
- Need a **distance metric**

Spelling Correction

I am writing this email on behaf of ...

The user typed ‘behaf’.

Which are some close words?

- behalf
- behave
- ....

Isolated word error correction

- Pick the one that is closest to ‘behaf’
- How to define ‘closest’?
- Need a **distance metric**
- The simplest metric: **edit distance**

Edit Distance

- The minimum edit distance between two strings

Edit Distance

- The minimum edit distance between two strings
- Is the minimum number of editing operations

Edit Distance

- The minimum edit distance between two strings
- Is the minimum number of editing operations
  - Insertion
  - Deletion
  - Substitution

Minimum Edit Distance

Example

Edit distance from ‘intention’ to ‘execution’

Minimum Edit Distance

Example

Edit distance from ‘intention’ to ‘execution’

```
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
```

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 a cost of 1 (Levenshtein)
  - Distance between these is 5

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 a cost of 1 (Levenshtein)
  - Distance between these is 5
- If substitution costs 2 (alternate version)
  - Distance between these is 8

How to find the Minimum Edit Distance?

How to find the Minimum Edit Distance?

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

How to find the Minimum Edit Distance?

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

- **Initial state:** the word we are transforming

How to find the Minimum Edit Distance?

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

- **Initial state:** the word we are transforming
- **Operators:** insert, delete, substitute

How to find the Minimum Edit Distance?

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

- **Initial state:** the word we are transforming
- **Operators:** insert, delete, substitute
- **Goal state:** the word we are trying to get to

How to find the Minimum Edit Distance?

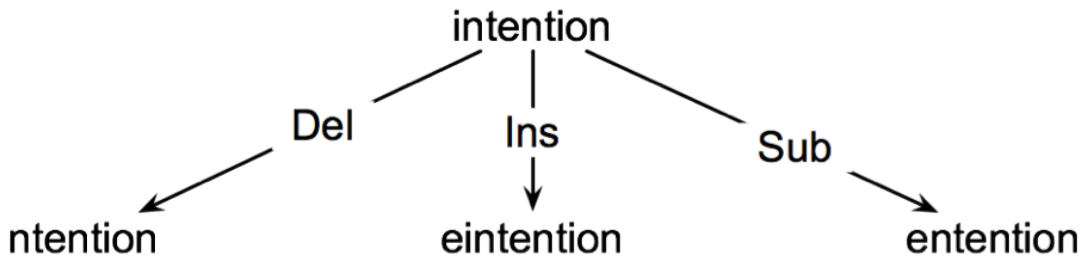
Searching for a path (sequence of edits) from the *start string* to the *final string*:

- **Initial state:** the word we are transforming
- **Operators:** insert, delete, substitute
- **Goal state:** the word we are trying to get to
- **Path cost:** what we want to minimize: the number of edits

How to find the Minimum Edit Distance?

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

- **Initial state:** the word we are transforming
- **Operators:** insert, delete, substitute
- **Goal state:** the word we are trying to get to
- **Path cost:** what we want to minimize: the number of edits



Minimum Edit as Search

How to navigate?

- The space of all edit sequences is huge

Minimum Edit as Search

How to navigate?

- The space of all edit sequences is huge
- Lot of distinct paths end up at the same state

Minimum Edit as Search

How to navigate?

- The space of all edit sequences is huge
- Lot of distinct paths end up at the same state
- Don't have to keep track of all of them

Minimum Edit as Search

How to navigate?

- The space of all edit sequences is huge
- Lot of distinct paths end up at the same state
- Don't have to keep track of all of them
- Keep track of the shortest path to each state

Defining Minimum Edit Distance Matrix

For two strings

- X of length n
- Y of length m

Defining Minimum Edit Distance Matrix

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

Defining Minimum Edit Distance Matrix

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

Thus, the edit distance between X and Y is $D(n,m)$

Computing Minimum Edit Distance

Dynamic Programming

- A tabular computation of $D(n, m)$

Computing Minimum Edit Distance

Dynamic Programming

- A tabular computation of $D(n, m)$
- Solving problems by combining solutions to subproblems

Computing Minimum Edit Distance

Dynamic Programming

- A tabular computation of $D(n, m)$
- Solving problems by combining solutions to subproblems
- Bottom-up

Computing 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

Computing 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
  - Compute larger $D(i,j)$ based on previously computed smaller values

Computing 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
  - Compute larger $D(i,j)$ based on previously computed smaller values
  - Compute $D(i,j)$ for all i and j till you get to $D(n,m)$

Dynamic Programming Algorithm

Initialization

$D(i,0) = i$

$D(0,j) = j$

Recurrence Relation:

For each $i = 1 \ldots M$

For each $j = 1 \ldots N$

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

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

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

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

Computing Alignments

- Computing edit distance may not be sufficient for some applications

Computing Alignments

- Computing edit distance may not be sufficient for some applications
  - We often need to align characters of the two strings to each other

Computing Alignments

- Computing edit distance may not be sufficient for some applications
  - We often need to align characters of the two strings to each other
- We do this by keeping a “backtrace”

Computing Alignments

- Computing edit distance may not be sufficient for some applications
  - We often need to align characters 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

Computing Alignments

- Computing edit distance may not be sufficient for some applications
  - We often need to align characters 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,

Computing Alignments

- Computing edit distance may not be sufficient for some applications
  - We often need to align characters 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

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

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

The Edit Distance Table

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

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

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

Base conditions:

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

Termination:

$D(N,M)$ is distance

Recurrence Relation:

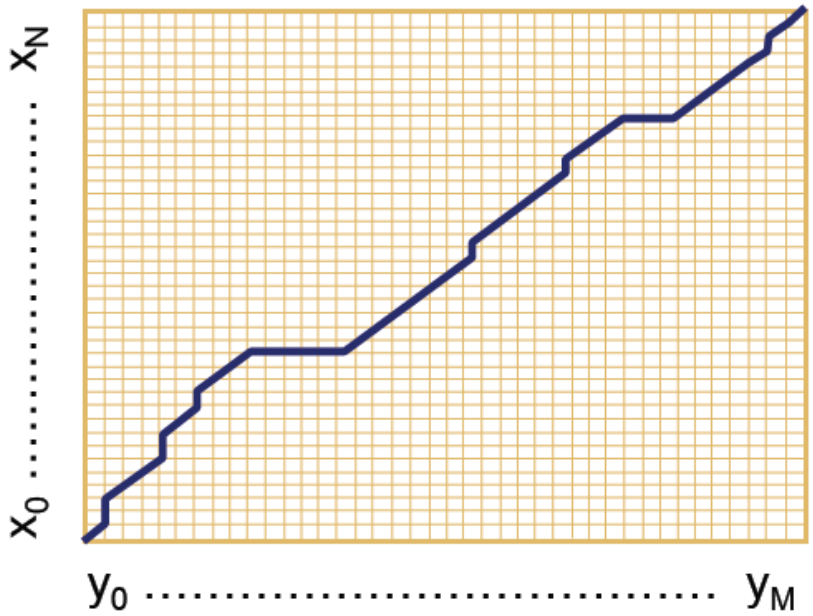
For each $i = 1 \ldots M$

For each $j = 1 \ldots N$

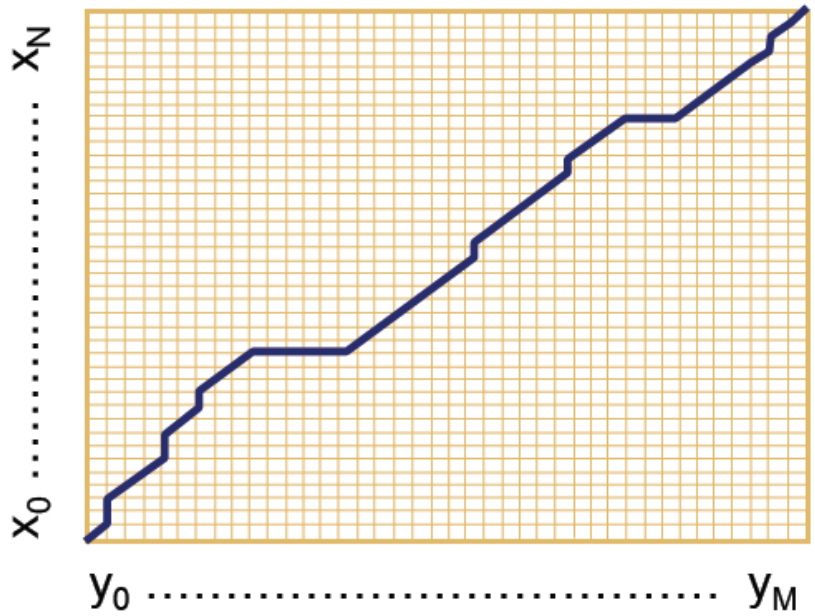
$$D(i,j) = \min \begin{cases} D(i-1,j) + 1 & \text{deletion} \\ D(i,j-1) + 1 & \text{insertion} \\ D(i-1,j-1) + \begin{cases} 2; & \text{if } X(i) \neq Y(j) \\ 0; & \text{if } X(i) = Y(j) \end{cases} & \text{substitution} \end{cases}$$

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

The distance matrix

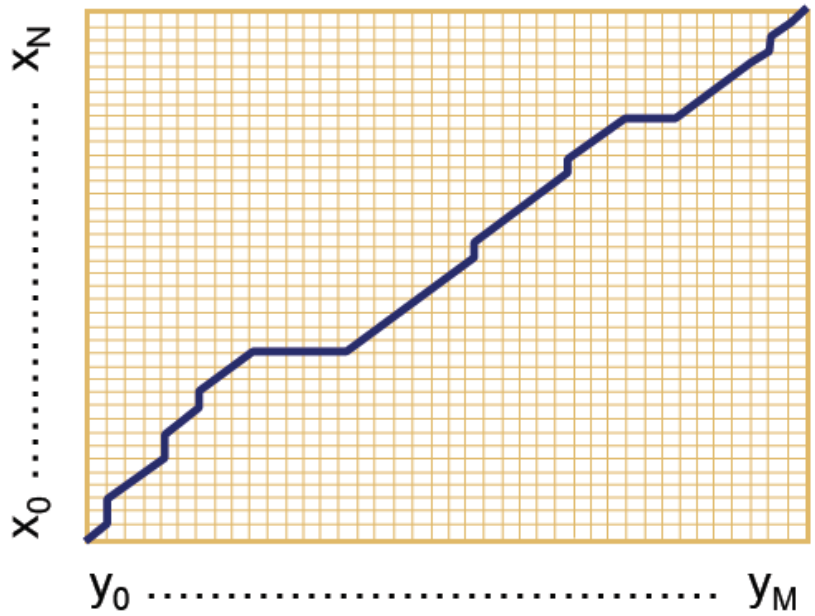


The distance matrix



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

The distance matrix



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

An optimal alignment is composed of optimal sub-alignments.

Result of Backtrace

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

Performance

Time

Performance

Time

$O(nm)$

Space

Performance

Time

$O(nm)$

Space

$O(nm)$

Backtrace

Performance

Time

$O(nm)$

Space

$O(nm)$

Backtrace

$O(n+m)$