Natural Language Processing - 2nd Semester
(2024-2025)
1038141

1.6 - Spelling Correction and Minimum Edit Distance

SAPIENZA
UNIVERSITÀ DI ROMA

Prof. Stefano Faralli
faralli@di.uniroma1.it

Prof. Iacopo Masi
masi@di.uniroma1.it

**credits are reported in the last slide

6 - Spelling correction and Minimum Edit Distance

- detecting and correcting word errors
- spelling correction
- minimum edit distance
- weighted minimum edit distance
- Q&A

Milestones in NLP

today topic is mainly related to the way we can assess strings' similarities and differences.

rule-based systems

statistical classical machine learning models

deep learning models

Detecting and correcting word errors

- A typical feature of modern word processors, search engines and OCR
- [Kukich, 1992] proposes three increasingly broader problems:
  - Detection of non-words (e.g. graffe)
  - Isolated word error correction (e.g. graffe => giraffe)
  - Context dependent error detection and correction where the error may result in a valid word (e.g. there => three)
- According to [Damereau, 1964] 80% of all misspelled words are caused by single-error misspellings which fall into the following categories:
  - Insertion (the => ther)
  - Deletion (the => th)
  - Substitution (the => thw)
  - Transposition (the => teh)

[Kukich, 1992]
[Damerau, 1964]

The intuition

For many applications (e.g., Spelling Correction, Machine Translation, Information Extraction, Speech Recognition, Computational Biology) it is crucial to assess:
<u>"How similar are two string?"</u>

Spelling correction:

The user typed "graffe", which is closest?

- graf
- graft
- grail
- giraffe

given a dictionary of correct words, similarities are used to find the most similar correct spelling

[Kukich, 1992]
[Damerau, 1964]

The intuition

For many applications (e.g., Spelling Correction, Machine Translation, Information Extraction, Speech Recognition, Computational Biology) it is crucial to assess:

"How similar are two string?"

Computational Biology:

- Align two sequences of nucleotides:

AGGCTATCACCTGACCTCCAGGCCGATGCCC
TAGCTATCACGACCGCGGTCGATTTGCCCGAC

- Resulting alignment:

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

given two nucleotide sequences, similarities are used to perform the optimal alignment

[Kukich, 1992]
[Damerau, 1964]

Minimum Edit Distance

definition

the minimum edit distance is a measure of the minimum cost for the application of editing operations to be performed in order to align a source string X to a target string Y.

editing operations are:

- Insertion (**i**)
- Deletion (**d**)
- Substitution (**s**)

and **cost(z)**, $z \in \{i,d,s\}$ is the function providing the cost in the application of a given editing operation.

[Jurafsky&Martin, 2022]

Minimum Edit Distance

example

X= INTENTION
Y= EXECUTION
|X|= **n** = 9
|Y|= **m** = 9

In general, **n** and **m** are not equal.

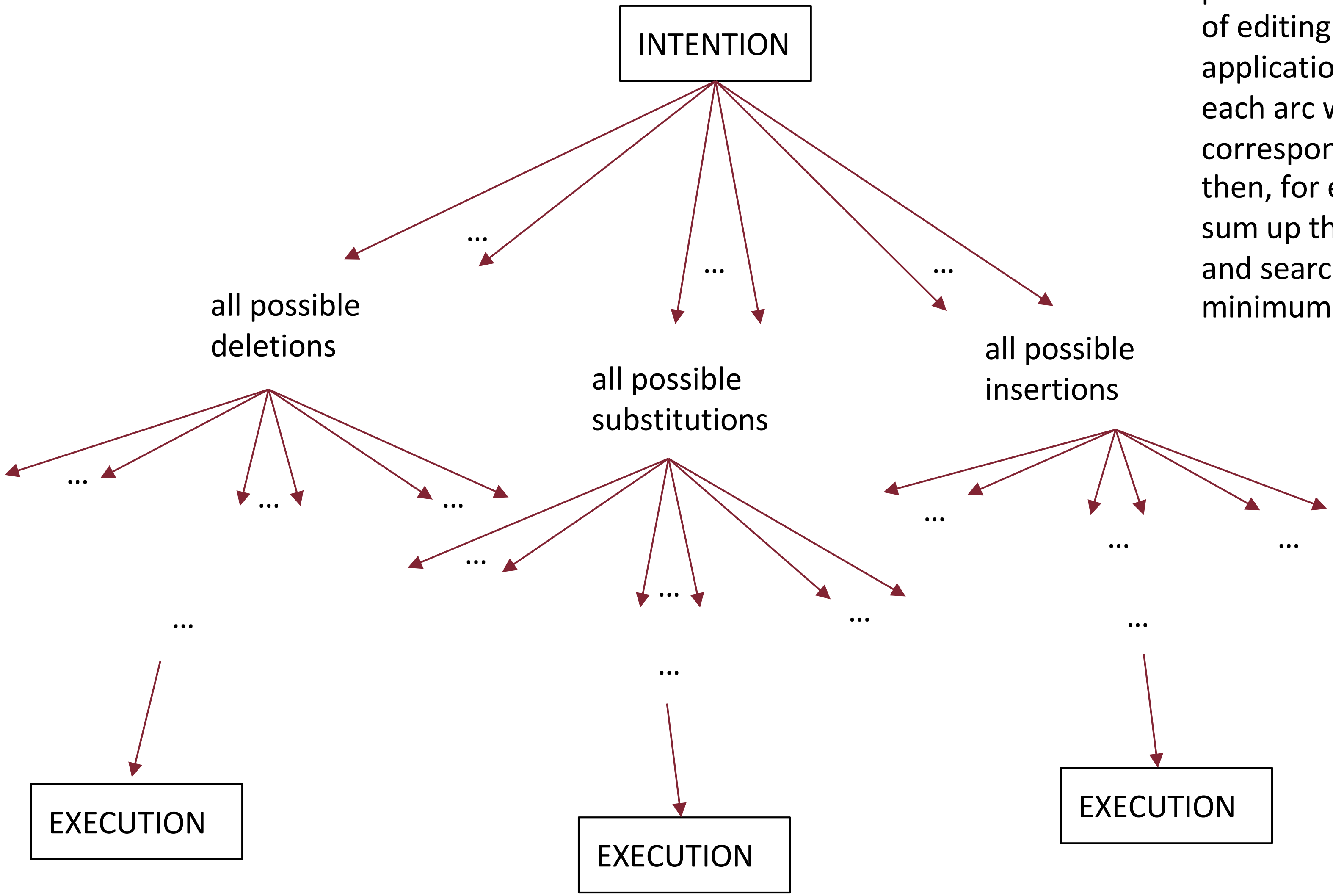
```
I N T E * N T I O N
| | | | | | | | | |
* E X E C U T I O N
d s s   i s
```

If each operation has cost of 1 (cost(d)=cost(i)=cost(s)=1):
Minimum Edit Distance between X and Y is 5

If substitutions cost 2 (Levenshtein: (cost(d)=cost(i)=1, cost(s)=2):
Minimum Edit Distance between X and Y is 8

[Jurafsky&Martin, 2022]

How to find the Minimum Edit Distance: Brute Force



we generate the tree of possible combinations of editing operation applications, we weigh each arc with the corresponding cost, we then, for each branch sum up the costs for and search for the minimum cost.

[Jurafsky&Martin, 2022]

How to find the Minimum Edit Distance: Brute Force

Brute Force:

we generate the tree of possible combinations of editing operation applications, we weigh each arc with the corresponding cost, we then, for each branch sum up the costs for d search for the nimum cost.

INTENTION

...

...

...

WAY TOO EXPENSIVE

EXECUTION

EXECUTION

[Jurafsky&Martin, 2022]

How to find the Minimum Edit Distance: Dynamic programming

Given two strings:

- X of length n
- Y of length m

We define $D_{X,Y}(i,j)$

- 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 $D_{X,Y}(n,m)$

The dynamic programming approach consists of a tabular computation of $D_{X,Y}(n,m)$;

Intuition: Solving problems by combining solutions to subproblems.

Bottom-up

- We compute $D_{X,Y}(i,j)$ for small i,j
- And compute larger $D_{X,Y}(i,j)$ based on previously computed smaller values
- i.e., compute $D_{X,Y}(i,j)$ for all i $(0 < i < n)$ and j $(0 < j < m)$

[Jurafsky&Martin, 2022]

How to find the Minimum Edit Distance: Dynamic programming

Initialization

create a matrix $D_{X,Y}$ with n+1 rows (n=|X|) and m+1 columns (m=|Y|)

$D_{X,Y}(i,0) = i$ ↑

$D_{X,Y}(0,j) = j$ ←

Recurrence Relation:

For each i = 1…n

For each j = 1…m

when computing the minimum keep track of the cells with the minimum value with backtrack pointers (↑, ←, ↖)

$$D_{X,Y}(i,j) = \min \begin{cases} D_{X,Y}(i-1,j) + \text{cost}(\mathbf{d}) & \uparrow \\ D_{X,Y}(i,j-1) + \text{cost}(\mathbf{i}) & \leftarrow \\ D_{X,Y}(i-1,j-1) + \begin{cases} \text{cost}(\mathbf{s}); & \text{if } X[i] \neq Y[j] \\ 0; & \text{if } X[i] = Y[j] \end{cases} & \nwarrow \end{cases}$$

Termination:

$D_{X,Y}(n,m)$ is the minimum edit distance;

Optimal alignment:

start form $D_{X,Y}(n,m)$ and follow the backtrack pointers;

How to find the Minimum Edit Distance: Dynamic programming

Exercise:

Compute the $D_{X,Y}$ for, X=hey and Y=hello and provide an optimal alignment. assume the Levenshtein costs for editing operations.

Initialization

create a matrix $D_{X,Y}$ with n+1 rows (n=|X|) and m+1 columns (m=|Y|)

$D_{X,Y}(i,0) = i$

$D_{X,Y}(0,j) = j$

How to find the Minimum Edit Distance: Dynamic programming

Exercise:

Compute the $D_{X,Y}$ for, X=hey and Y=hello and provide an optimal alignment. assume the Levenshtein costs for editing operations.

		j=0	j=1	j=2	j=3	j=4	j=5
		#	h	e	l	l	o
i=0	#						
i=1	h						
i=2	e						
i=e	y						

Initialization

create a matrix $D_{X,Y}$ with n+1 rows (n=|X|) and m+1 columns (m=|Y|)

$D_{X,Y}(i,0) = i$ ↑

$D_{X,Y}(0,j) = j$ ←

[Jurafsky&Martin, 2022]

How to find the Minimum Edit Distance: Dynamic programming

Exercise:

Compute the $D_{X,Y}$ for, X=hey and Y=hello and provide an optimal alignment. assume the Levenshtein costs for editing operations.

		j=0	j=1	j=2	j=3	j=4	j=5
		#	h	e	l	l	o
i=0	#						
i=1	h						
i=2	e						
i=e	y						

Initialization

create a matrix $D_{X,Y}$ with n+1 rows (n=|X|) and m+1 columns (m=|Y|)

$D_{X,Y}(i,0) = i$

$D_{X,Y}(0,j) = j$

[Jurafsky&Martin, 2022]

How to find the Minimum Edit Distance: Dynamic programming

Exercise:

Compute the $D_{X,Y}$ for, X=hey and Y=hello. and provide an optimal alignment. assume the Levenshtein costs for editing operations.

		j=0	j=1	j=2	j=3	j=4	j=5
		#	h	e	l	l	o
i=0	#	0					
i=1	h	↑ 1					
i=2	e	↑ 2					
i=e	y	↑ 3					

Initialization

create a matrix $D_{X,Y}$ with n+1 rows (n=|X|) and m+1 columns (m=|Y|)

$D_{X,Y}(i,0) = i$ ↑

$D_{X,Y}(0,j) = j$ ←

[Jurafsky&Martin, 2022]

How to find the Minimum Edit Distance: Dynamic programming

Exercise:

Compute the $D_{X,Y}$ for, X=hey and Y=hello. and provide an optimal alignment. assume the Levenshtein costs for editing operations.

		j=0	j=1	j=2	j=3	j=4	j=5
		#	h	e	l	l	o
i=0	#	0					
i=1	h	↑ 1					
i=2	e	↑ 2					
i=e	y	↑ 3					

Initialization

```
create a matrix D_{X,Y} with n+1 rows (n=|X|) and m+1 columns (m=|Y|)
    D_{X,Y}(i,0) = i   ↑
    D_{X,Y}(0,j) = j  ←
```


[Jurafsky&Martin, 2022]

How to find the Minimum Edit Distance: Dynamic programming

Exercise:

Compute the $D_{X,Y}$ for, X=hey and Y=hello. and provide an optimal alignment. assume the Levenshtein costs for editing operations.

		j=0	j=1	j=2	j=3	j=4	j=5
		#	h	e	l	l	o
i=0	#	0	← 1	← 2	← 3	← 4	← 5
i=1	h	↑ 1					
i=2	e	↑ 2					
i=e	y	↑ 3					

Initialization

create a matrix $D_{X,Y}$ with n+1 rows (n=|X|) and m+1 columns (m=|Y|)

$D_{X,Y}(i,0) = i$ ↑

$\mathbf{D_{X,Y}(0,j) = j}$ ←

[Jurafsky&Martin, 2022]

How to find the Minimum Edit Distance: Dynamic programming

		j=0	j=1	j=2	j=3	j=4	j=5
		#	h	e	l	l	o
i=0	#	0	← 1	← 2	← 3	← 4	← 5
i=1	h	↑ 1	↖ 0				
i=2	e	↑ 2					
i=e	y	↑ 3					

Recurrence Relation:

For each i = 1…n **i=1**

For each j = 1…m **j=1**

$$
D_{X,Y}(i,j) = \min \begin{cases} D_{X,Y}(i-1,j) + \mathrm{cost}(\mathbf{d}) & \uparrow \\ D_{X,Y}(i,j-1) + \mathrm{cost}(\mathbf{i}) & \leftarrow \\ D_{X,Y}(i-1,j-1) + \begin{cases} \mathrm{cost}(\mathbf{s}); & \text{if } X[i] \neq Y[j] \\ 0; & \text{if } X[i] = Y[j] \end{cases} & \nwarrow \end{cases}
$$

when computing the minimum keep track of the cells with the minimum value with backtrack pointers (↑, ←, ↖)

[Jurafsky&Martin, 2022]

How to find the Minimum Edit Distance: Dynamic programming

		j=0	j=1	j=2	j=3	j=4	j=5
		#	h	e	l	l	o
i=0	#	0	← 1	← 2	← 3	← 4	← 5
i=1	h	↑ 1	↖ 0	← 1			
i=2	e	↑ 2					
i=e	y	↑ 3					

Recurrence Relation:

```
For each  i = 1…n
     For each  j = 1…m
```

i=1

j=2

$$D_{X,Y}(i,j) = \min \begin{cases} D_{X,Y}(i-1,j) + cost(\mathbf{d}) & \uparrow \\ D_{X,Y}(i,j-1) + cost(\mathbf{i}) & \leftarrow \\ D_{X,Y}(i-1,j-1) + \begin{cases} cost(\mathbf{s}); & \text{if } X[i] \neq Y[j] \\ 0; & \text{if } X[i] = Y[j] \end{cases} & \nwarrow \end{cases}$$

when computing the minimum keep track of the cells with the minimum value with backtrack pointers (↑, ←, ↖)

[Jurafsky&Martin, 2022]

How to find the Minimum Edit Distance: Dynamic programming

		j=0	j=1	j=2	j=3	j=4	j=5
		#	h	e	l	l	o
i=0	#	0	← 1	← 2	← 3	← 4	← 5
i=1	h	↑ 1	↖ 0	← 1	← 2		
i=2	e	↑ 2					
i=e	y	↑ 3					

Recurrence Relation:

```
For each  i = 1…n
     For each  j = 1…m
```

i=1

j=3

$$D_{X,Y}(i,j) = \min \begin{cases} D_{X,Y}(i-1,j) + \text{cost}(\mathbf{d}) & \uparrow \\ D_{X,Y}(i,j-1) + \text{cost}(\mathbf{i}) & \leftarrow \\ D_{X,Y}(i-1,j-1) + \begin{cases} \text{cost}(\mathbf{s}); & \text{if } X[i] \neq Y[j] \\ 0; & \text{if } X[i] = Y[j] \end{cases} & \nwarrow \end{cases}$$

when computing the minimum keep track of the cells with the minimum value with backtrack pointers (↑, ←, ↖)

[Jurafsky&Martin, 2022]

How to find the Minimum Edit Distance: Dynamic programming

		j=0	j=1	j=2	j=3	j=4	j=5
		#	h	e	l	l	o
i=0	#	0	← 1	← 2	← 3	← 4	← 5
i=1	h	↑ 1	↖ 0	← 1	← 2	← 3	
i=2	e	↑ 2					
i=e	y	↑ 3					

Recurrence Relation*:*

For each i = 1…n

For each j = 1…m

i=1

j=4

$$D_{X,Y}(i,j) = \min \begin{cases} D_{X,Y}(i-1,j) + cost(\mathbf{d}) & \uparrow \\ D_{X,Y}(i,j-1) + cost(\mathbf{i}) & \leftarrow \\ D_{X,Y}(i-1,j-1) + \begin{cases} cost(\mathbf{s}); & \text{if } X[i] \neq Y[j] \\ 0; & \text{if } X[i] = Y[j] \end{cases} & \nwarrow \end{cases}$$

when computing the minimum keep track of the cells with the minimum value with backtrack pointers (↑, ←, ↖)

How to find the Minimum Edit Distance: Dynamic programming

		j=0	j=1	j=2	j=3	j=4	j=5
		#	h	e	l	l	o
i=0	#	0	← 1	← 2	← 3	← 4	← 5
i=1	h	↑ 1	↖ 0	← 1	← 2	← 3	← 4
i=2	e	↑ 2					
i=e	y	↑ 3					

Recurrence Relation:

```
For each  i = 1…n
     For each  j = 1…m
```

i=1

j=5

$$D_{X,Y}(i,j) = \min \begin{cases} D_{X,Y}(i-1,j) + cost(\mathbf{d}) & \uparrow \\ D_{X,Y}(i,j-1) + cost(\mathbf{i}) & \leftarrow \\ D_{X,Y}(i-1,j-1) + \begin{cases} cost(\mathbf{s}); & \text{if } X[i] \neq Y[j] \\ 0; & \text{if } X[i] = Y[j] \end{cases} & \nwarrow \end{cases}$$

when computing the minimum keep track of the cells with the minimum value with backtrack pointers (↑, ←, ↖)

[Jurafsky&Martin, 2022]

How to find the Minimum Edit Distance: Dynamic programming

		j=0	j=1	j=2	j=3	j=4	j=5
		#	h	e	l	l	o
i=0	#	0	← 1	← 2	← 3	← 4	← 5
i=1	h	↑ 1	↖ 0	← 1	← 2	← 3	← 4
i=2	e	↑ 2	↑ 1				
i=e	y	↑ 3					

Recurrence Relation:

For each i = 1…n

For each j = 1…m

i=2

j=1

$$D_{X,Y}(i,j) = \min \begin{cases} D_{X,Y}(i-1,j) + cost(\mathbf{d}) & \uparrow \\ D_{X,Y}(i,j-1) + cost(\mathbf{i}) & \leftarrow \\ D_{X,Y}(i-1,j-1) + \begin{cases} cost(\mathbf{s}); & \text{if } X[i] \neq Y[j] \\ 0; & \text{if } X[i] = Y[j] \end{cases} & \nwarrow \end{cases}$$

when computing the minimum keep track of the cells with the minimum value with backtrack pointers (↑, ←, ↖)

[Jurafsky&Martin, 2022]

How to find the Minimum Edit Distance: Dynamic programming

		j=0	j=1	j=2	j=3	j=4	j=5
		#	h	e	l	l	o
i=0	#	0	← 1	← 2	← 3	← 4	← 5
i=1	h	↑ 1	↖ 0	← 1	← 2	← 3	← 4
i=2	e	↑ 2	↑ 1	↖ 0			
i=e	y	↑ 3					

Recurrence Relation:

For each i = 1…n **i=2**

For each j = 1…m **j=2**

$$D_{X,Y}(i,j) = \min \begin{cases} D_{X,Y}(i-1,j) + \text{cost}(\mathbf{d}) \uparrow \\ D_{X,Y}(i,j-1) + \text{cost}(\mathbf{i}) \leftarrow \\ D_{X,Y}(i-1,j-1) + \begin{cases} \text{cost}(\mathbf{s}); & \text{if } X[i] \neq Y[j] \\ 0; & \text{if } X[i] = Y[j] \end{cases} \nwarrow \end{cases}$$

when computing the minimum keep track of the cells with the minimum value with backtrack pointers (↑, ←, ↖)

[Jurafsky&Martin, 2022]

How to find the Minimum Edit Distance: Dynamic programming

		j=0	j=1	j=2	j=3	j=4	j=5
		#	h	e	l	l	o
i=0	#	0	← 1	← 2	← 3	← 4	← 5
i=1	h	↑ 1	↖ 0	← 1	← 2	← 3	← 4
i=2	e	↑ 2	↑ 1	↖ 0	← 1		
i=e	y	↑ 3					

Recurrence Relation:

For each i = 1…n   **i=2**

    For each j = 1…m   **j=3**

$$D_{X,Y}(i,j) = \min \begin{cases} D_{X,Y}(i-1,j) + \text{cost}(\mathbf{d}) & \uparrow \\ D_{X,Y}(i,j-1) + \text{cost}(\mathbf{i}) & \leftarrow \\ D_{X,Y}(i-1,j-1) + \begin{cases} \text{cost}(\mathbf{s}); & \text{if } X[i] \neq Y[j] \\ 0; & \text{if } X[i] = Y[j] \end{cases} & \nwarrow \end{cases}$$

when computing the minimum keep track of the cells with the minimum value with backtrack pointers (↑, ←, ↖)

How to find the Minimum Edit Distance: Dynamic programming

	j=0	j=1	j=2	j=3	j=4	j=5
	#	h	e	l	l	o
i=0	#	0	← 1	← 2	← 3	← 4	← 5
i=1	h	↑ 1	↖ 0	← 1	← 2	← 3	← 4
i=2	e	↑ 2	↑ 1	↖ 0	← 1	← 2	
i=e	y	↑ 3					

Recurrence Relation:

For each i = 1…n

For each j = 1…m

i=2

j=4

$$D_{X,Y}(i,j) = \min \begin{cases} D_{X,Y}(i-1,j) + cost(\mathbf{d}) & \uparrow \\ D_{X,Y}(i,j-1) + cost(\mathbf{i}) & \leftarrow \\ D_{X,Y}(i-1,j-1) + \begin{cases} cost(\mathbf{s}); & \text{if } X[i] \neq Y[j] \\ 0; & \text{if } X[i] = Y[j] \end{cases} & \nwarrow \end{cases}$$

when computing the minimum keep track of the cells with the minimum value with backtrack pointers (↑, ←, ↖)

[Jurafsky&Martin, 2022]

How to find the Minimum Edit Distance: Dynamic programming

	j=0	j=1	j=2	j=3	j=4	j=5
	#	h	e	l	l	o
i=0 #	0	← 1	← 2	← 3	← 4	← 5
i=1 h	↑ 1	↖ 0	← 1	← 2	← 3	← 4
i=2 e	↑ 2	↑ 1	↖ 0	← 1	← 2	← 3
i=e y	↑ 3					

Recurrence Relation:

```
For each  i = 1…n
     For each  j = 1…m
```

i=2
j=5

$$
D_{X,Y}(i,j) = \min \begin{cases} D_{X,Y}(i-1,j) + cost(\mathbf{d}) & \uparrow \\ D_{X,Y}(i,j-1) + cost(\mathbf{i}) & \leftarrow \\ D_{X,Y}(i-1,j-1) + \begin{cases} cost(\mathbf{s}); & \text{if } X[i] \neq Y[j] \\ 0; & \text{if } X[i] = Y[j] \end{cases} & \nwarrow \end{cases}
$$

when computing the minimum keep track of the cells with the minimum value with backtrack pointers (↑, ←, ↖)

[Jurafsky&Martin, 2022]

How to find the Minimum Edit Distance: Dynamic programming

		j=0	j=1	j=2	j=3	j=4	j=5
		#	h	e	l	l	o
i=0	#	0	← 1	← 2	← 3	← 4	← 5
i=1	h	↑ 1	↖ 0	← 1	← 2	← 3	← 4
i=2	e	↑ 2	↑ 1	↖ 0	← 1	← 2	← 3
i=e	y	↑ 3	↑ 2				

Recurrence Relation:

```
For each  i = 1…n
     For each  j = 1…m
```

i=3

j=1

$$D_{X,Y}(i,j) = \min \begin{cases} D_{X,Y}(i-1,j) + \mathrm{cost}(\mathbf{d}) & \uparrow \\ D_{X,Y}(i,j-1) + \mathrm{cost}(\mathbf{i}) & \leftarrow \\ D_{X,Y}(i-1,j-1) + \begin{cases} \mathrm{cost}(\mathbf{s}); & \text{if } X[i] \neq Y[j] \\ 0; & \text{if } X[i] = Y[j] \end{cases} & \nwarrow \end{cases}$$

when computing the minimum keep track of the cells with the minimum value with backtrack pointers (↑, ←, ↖)

[Jurafsky&Martin, 2022]

How to find the Minimum Edit Distance: Dynamic programming

		j=0	j=1	j=2	j=3	j=4	j=5
		#	h	e	l	l	o
i=0	#	0	← 1	← 2	← 3	← 4	← 5
i=1	h	↑ 1	↖ 0	← 1	← 2	← 3	← 4
i=2	e	↑ 2	↑ 1	↖ 0	← 1	← 2	← 3
i=3	y	↑ 3	↑ 2	↑ 1			

Recurrence Relation:

For each i = 1…n

For each j = 1…m

i=3

j=2

$$D_{X,Y}(i,j) = \min \begin{cases} D_{X,Y}(i-1,j) + \text{cost}(\mathbf{d}) & \uparrow \\ D_{X,Y}(i,j-1) + \text{cost}(\mathbf{i}) & \leftarrow \\ D_{X,Y}(i-1,j-1) + \begin{cases} \text{cost}(\mathbf{s}); & \text{if } X[i] \neq Y[j] \\ 0; & \text{if } X[i] = Y[j] \end{cases} & \nwarrow \end{cases}$$

when computing the minimum keep track of the cells with the minimum value with backtrack pointers (↑, ←, ↖)

How to find the Minimum Edit Distance: Dynamic programming

		j=0	j=1	j=2	j=3	j=4	j=5
		#	h	e	l	l	o
i=0	#	0	← 1	← 2	← 3	← 4	← 5
i=1	h	↑ 1	↖ 0	← 1	← 2	← 3	← 4
i=2	e	↑ 2	↑ 1	↖ 0	← 1	← 2	← 3
i=3	y	↑ 3	↑ 2	↑ 1	← ↖ ↑ 2		

Recurrence Relation:

For each i = 1…n **i=3**

For each j = 1…m **j=3**

$$D_{X,Y}(i,j) = \min \begin{cases} D_{X,Y}(i-1,j) + cost(\mathbf{d}) & \uparrow \\ D_{X,Y}(i,j-1) + cost(\mathbf{i}) & \leftarrow \\ D_{X,Y}(i-1,j-1) + \begin{cases} cost(\mathbf{s}); & \text{if } X[i] \neq Y[j] \\ 0; & \text{if } X[i] = Y[j] \end{cases} & \nwarrow \end{cases}$$

when computing the minimum keep track of the cells with the minimum value with backtrack pointers (↑, ←, ↖)

How to find the Minimum Edit Distance: Dynamic programming

		j=0	j=1	j=2	j=3	j=4	j=5
		#	h	e	l	l	o
i=0	#	0	← 1	← 2	← 3	← 4	← 5
i=1	h	↑ 1	↖ 0	← 1	← 2	← 3	← 4
i=2	e	↑ 2	↑ 1	↖ 0	← 1	← 2	← 3
i=3	y	↑ 3	↑ 2	↑ 1	←↖↑ 2	←↖↑ 3	

Recurrence Relation:

For each i = 1…n

For each j = 1…m

i=3

j=4

$$D_{X,Y}(i,j) = \min \begin{cases} D_{X,Y}(i-1,j) + cost(\mathbf{d}) & \uparrow \\ D_{X,Y}(i,j-1) + cost(\mathbf{i}) & \leftarrow \\ D_{X,Y}(i-1,j-1) + \begin{cases} cost(\mathbf{s}); & \text{if } X[i] \neq Y[j] \\ 0; & \text{if } X[i] = Y[j] \end{cases} & \nwarrow \end{cases}$$

when computing the minimum keep track of the cells with the minimum value with backtrack pointers (↑, ←, ↖)

How to find the Minimum Edit Distance: Dynamic programming

		j=0	j=1	j=2	j=3	j=4	j=5
		#	h	e	l	l	o
i=0	#	0	← 1	← 2	← 3	← 4	← 5
i=1	h	↑ 1	↖ 0	← 1	← 2	← 3	← 4
i=2	e	↑ 2	↑ 1	↖ 0	← 1	← 2	← 3
i=3	y	↑ 3	↑ 2	↑ 1	← ↖ ↑ 2	← ↖ ↑ 3	← ↖ ↑ 4

Recurrence Relation:

For each i = 1…n

For each j = 1…m

i=3

j=5

$$D_{X,Y}(i,j) = \min \begin{cases} D_{X,Y}(i-1,j) + \text{cost}(\mathbf{d}) & \uparrow \\ D_{X,Y}(i,j-1) + \text{cost}(\mathbf{i}) & \leftarrow \\ D_{X,Y}(i-1,j-1) + \begin{cases} \text{cost}(\mathbf{s}); & \text{if } X[i] \neq Y[j] \\ 0; & \text{if } X[i] = Y[j] \end{cases} & \nwarrow \end{cases}$$

when computing the minimum keep track of the cells with the minimum value with backtrack pointers (↑, ←, ↖)

[Jurafsky&Martin, 2022]

How to find the Minimum Edit Distance: Dynamic programming

		j=0	j=1	j=2	j=3	j=4	j=5
		#	h	e	l	l	o
i=0	#	0	← 1	← 2	← 3	← 4	← 5
i=1	h	↑ 1	↖ 0	← 1	← 2	← 3	← 4
i=2	e	↑ 2	↑ 1	↖ 0	← 1	← 2	← 3
i=3	y	↑ 3	↑ 2	↑ 1	←↖↑ 2	←↖↑ 3	←↖↑ 4

Termination:

$D_{X,Y}(n,m)$ is the minimum edit distance;

How to find the Minimum Edit Distance: Dynamic programming

		j=0	j=1	j=2	j=3	j=4	j=5
		#	h	e	l	l	o
i=0	#	0	← 1	← 2	← 3	← 4	← 5
i=1	h	↑ 1	↖ 0	← 1	← 2	← 3	← 4
i=2	e	↑ 2	↑ 1	↖ 0	← 1	← 2	← 3
i=3	y	↑ 3	↑ 2	↑ 1	←↖↑2	←↖↑3	←↖↑4

X : # h e y
Y : # _ _ _
op: _ _ _ _

↑ deletion
↖ substitution
← insertion

Optimal alignment*:*
`start form` $D_{X,Y}(n,m)$ `and follow the backtrack pointers;`

How to find the Minimum Edit Distance: Dynamic programming

		j=0	j=1	j=2	j=3	j=4	j=5
		#	h	e	l	l	o
i=0	#	0	← 1	← 2	← 3	← 4	← 5
i=1	h	↑ 1	↖ 0	← 1	← 2	← 3	← 4
i=2	e	↑ 2	↑ 1	↖ 0	← 1	← 2	← 3
i=3	y	↑ 3	↑ 2	↑ 1	← ↖ ↑ 2	← ↖ ↑ 3	← ↖ ↑ 4

X : # h e y
Y : # _ _ _ **o**
op: _ _ _ _ **i**

↑ deletion
↖ substitution
← insertion

Optimal alignment:

`start form` $D_{X,Y}(n,m)$ `and follow the backtrack pointers;`

How to find the Minimum Edit Distance: Dynamic programming

		j=0	j=1	j=2	j=3	j=4	j=5
		#	h	e	l	l	o
i=0	#	0	← 1	← 2	← 3	← 4	← 5
i=1	h	↑ 1	↖ 0	← 1	← 2	← 3	← 4
i=2	e	↑ 2	↑ 1	↖ 0	← 1	← 2	← 3
i=3	y	↑ 3	↑ 2	↑ 1	←↖↑ 2	←↖↑ 3	←↖↑ 4

X : # h e y
Y : # _ _ _ **l** o
op: _ _ _ _ **i** i

↑ deletion
↖ substitution
← insertion

Optimal alignment*:*

`start form` $D_{X,Y}$`(n,m) and follow the backtrack pointers;`

[**Jurafsky&Martin, 2022**]

How to find the Minimum Edit Distance: Dynamic programming

		j=0	j=1	j=2	j=3	j=4	j=5
		#	h	e	l	l	o
i=0	#	0	← 1	← 2	← 3	← 4	← 5
i=1	h	↑ 1	↖ 0	← 1	← 2	← 3	← 4
i=2	e	↑ 2	↑ 1	↖ 0	← 1	← 2	← 3
i=3	y	↑ 3	↑ 2	↑ 1	←↖↑ 2	←↖↑ 3	←↖↑ 4

X : # h e y
Y : # _ _ **l** l o
op: _ _ _ **s** i i

↑ deletion
↖ substitution
← insertion

Optimal alignment*:*

`start form` $D_{X,Y}$`(n,m) and follow the backtrack pointers;`

[**Jurafsky&Martin, 2022**]

How to find the Minimum Edit Distance: Dynamic programming

		j=0	j=1	j=2	j=3	j=4	j=5
		#	h	e	l	l	o
i=0	#	0	← 1	← 2	← 3	← 4	← 5
i=1	h	↑ 1	↖ 0	← 1	← 2	← 3	← 4
i=2	e	↑ 2	↑ 1	↖ 0	← 1	← 2	← 3
i=3	y	↑ 3	↑ 2	↑ 1	← ↖ ↑ 2	← ↖ ↑ 3	← ↖ ↑ 4

X : # h e y
Y : # _ **e** l l o
op: _ _ _ s i i

↑ deletion
↖ substitution
← insertion

Optimal alignment*:*

`start form` $D_{X,Y}(n,m)$ `and follow the backtrack pointers;`

[**Jurafsky&Martin, 2022**]

How to find the Minimum Edit Distance: Dynamic programming

		j=0	j=1	j=2	j=3	j=4	j=5
		#	h	e	l	l	o
i=0	#	0	← 1	← 2	← 3	← 4	← 5
i=1	h	↑ 1	↖ 0	← 1	← 2	← 3	← 4
i=2	e	↑ 2	↑ 1	↖ 0	← 1	← 2	← 3
i=3	y	↑ 3	↑ 2	↑ 1	←↖↑ 2	←↖↑ 3	←↖↑ 4

X : # h e y
Y : # **h** e l l o
op: _ _ _ s i i

↑ deletion
↖ substitution
← insertion

Optimal alignment:

start form $D_{X,Y}(n,m)$ and follow the backtrack poi[illegible]s;

[Jurafsky&Martin, 2022]

How to find the Minimum Edit Distance: Dynamic programming

		j=0	j=1	j=2	j=3	j=4	j=5
		#	h	e	l	l	o
i=0	#	0	← 1	← 2	← 3	← 4	← 5
i=1	h	↑ 1	↖ 0	← 1	← 2	← 3	← 4
i=2	e	↑ 2	↑ 1	↖ 0	← 1	← 2	← 3
i=3	y	↑ 3	↑ 2	↑ 1	←↖↑ 2	←↖↑ 3	←↖↑ 4

X : # h e y
Y : # h e l l o
op: _ _ _ s i i

↑ deletion
↖ substitution
← insertion

Other paths are also possible !

Optimal alignment*:*

`start form` $D_{X,Y}(n,m)$ `and follow the backtrack pointers;`

Weighted Minimum Edit Distance

Why would we add weights to the computation?

- Spell Correction: some letters are more likely to be mistyped than others
- Biology: certain kinds of deletions or insertions are more likely than others

[Jurafsky&Martin, 2022]

Weighted Minimum Edit Distance

sub[X, Y] = Substitution of X (incorrect) for Y (correct)

X \ Y (correct)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	0	0	7	1	342	0	0	2	118	0	1	0	0	3	76	0	0	1	35	9	9	0	1	0	5	0
b	0	0	9	9	2	2	3	1	0	0	0	5	11	5	0	10	0	0	2	1	0	0	8	0	0	0
c	6	5	0	16	0	9	5	0	0	0	1	0	7	9	1	10	2	5	39	40	1	3	7	1	1	0
d	1	10	13	0	12	0	5	5	0	0	2	3	7	3	0	1	0	43	30	22	0	0	4	0	2	0
e	388	0	3	11	0	2	2	0	89	0	0	3	0	5	93	0	0	14	12	6	15	0	1	0	18	0
f	0	15	0	3	1	0	5	2	0	0	0	3	4	1	0	0	0	6	4	12	0	0	2	0	0	0
g	4	1	11	11	9	2	0	0	0	1	1	3	0	0	2	1	3	5	13	21	0	0	1	0	3	0
h	1	8	0	3	0	0	0	0	0	0	2	0	12	14	2	3	0	3	1	11	0	0	2	0	0	0
i	103	0	0	0	146	0	1	0	0	0	0	6	0	0	49	0	0	0	2	1	47	0	2	1	15	0
j	0	1	1	9	0	0	1	0	0	0	0	2	1	0	0	0	0	0	5	0	0	0	0	0	0	0
k	1	2	8	4	1	1	2	5	0	0	0	0	5	0	2	0	0	0	6	0	0	0	4	0	0	3
l	2	10	1	4	0	4	5	6	13	0	1	0	0	14	2	5	0	11	10	2	0	0	0	0	0	0
m	1	3	7	8	0	2	0	6	0	0	4	4	0	180	0	6	0	0	9	15	13	3	2	2	3	0
n	2	7	6	5	3	0	1	19	1	0	4	35	78	0	0	7	0	28	5	7	0	0	1	2	0	2
o	91	1	1	3	116	0	0	0	25	0	2	0	0	0	0	14	0	2	4	14	39	0	0	0	18	0
p	0	11	1	2	0	6	5	0	2	9	0	2	7	6	15	0	0	1	3	6	0	4	1	0	0	0
q	0	0	1	0	0	0	27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r	0	14	0	30	12	2	2	8	2	0	5	8	4	20	1	14	0	0	12	22	4	0	0	1	0	0
s	11	8	27	33	35	4	0	1	0	1	0	27	0	6	1	7	0	14	0	15	0	0	5	3	20	1
t	3	4	9	42	7	5	19	5	0	1	0	14	9	5	5	6	0	11	37	0	0	2	19	0	7	6
u	20	0	0	0	44	0	0	0	64	0	0	0	0	2	43	0	0	4	0	0	0	0	2	0	8	0
v	0	0	7	0	0	3	0	0	0	0	0	1	0	0	1	0	0	0	8	3	0	0	0	0	0	0
w	2	2	1	0	1	0	0	2	0	0	1	0	0	0	0	7	0	6	3	3	1	0	0	0	0	0
x	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0	0	0	0	0	0	0
y	0	0	2	0	15	0	1	7	15	0	0	0	2	0	6	1	0	7	36	8	5	0	0	1	0	0
z	0	0	0	7	0	0	0	0	0	0	0	7	5	0	0	0	0	2	21	3	0	0	0	0	3	0

[Jurafsky&Martin, 2022]


Q&A

Resources and References

[Jurafsky&Martin, 2022] Jurafsky and Martin. Speech and Language Processing, Prentice Hall, third edition https://web.stanford.edu/~jurafsky/slp3/ed3book.pdf

[Kukich, 1992] Kukich, K. (1992) Techniques for Automatically Correcting Words in Text. ACM Computing Surveys, 24, 377-439.
https://doi.org/10.1145/146370.146380

[Damerau, 1964] Fred J. Damerau. 1964. A technique for computer detection and correction of spelling errors. Commun. ACM 7, 3 (March 1964), 171–176. https://doi.org/10.1145/363958.363994

**Credits

The slides of this part of the course are the result of a personal reworking of the slides and of the course material from different sources:

1. The NLP course of Prof. Roberto Navigli, Sapienza University of Rome
2. The NLP course of Prof. Simone Paolo Ponzetto, University of Mannheim, Germany
3. The NLP course of Prof. Chris Biemann, University of Hamburg, Germany
4. The NLP course of Prof. Dan Jurafsky, Stanford University, USA