Learning objectives

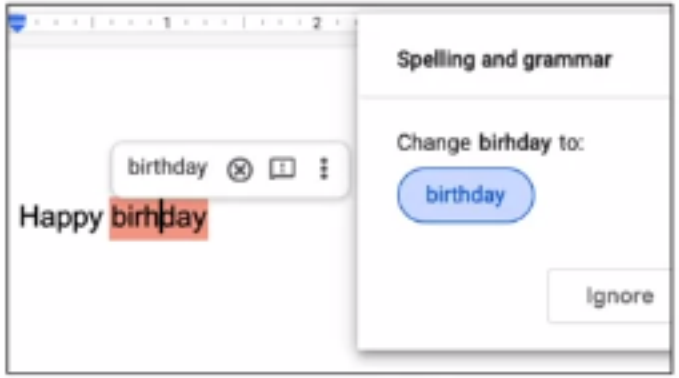
- What is autocorrect?
- Building the model
- Minimum edit distance
- Minimum edit distance algorithm

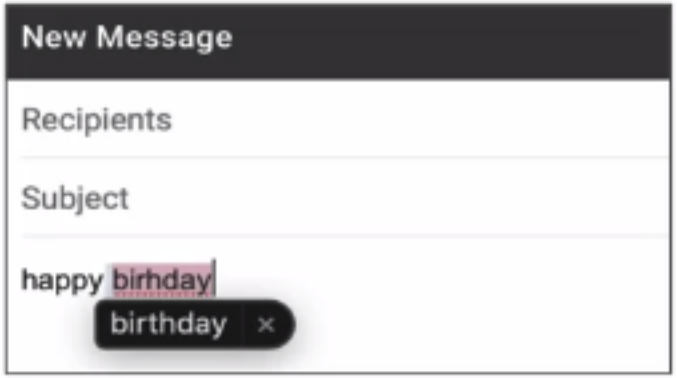
deah → dear ✅
yeah
dear
dean
… *etc*

	#	s	t	a	y
#	0	1	2	3	4
p	1	2	3	4	5
l	2	3	4	5	6
a	3	4	5	4	5
y	4	5	6	5	4

What is autocorrect?

- Phones
- Tablets
- Computers





What is autocorrect?

- Example:

Happy birthday deah friend! 

What is autocorrect ?

- Example:

What is autocorrect?

- Example:

Happy birthday deer friend!  ??

How it works

1. Identify a misspelled word
2. Find strings n edit distance away
3. Filter candidates
4. Calculate word probabilities

How it works

1. Identify a misspelled word
2. Find strings n edit distance away
3. Filter candidates
4. Calculate word probabilities

deah

How it works

1. Identify a misspelled word
2. Find strings n edit distance away
3. Filter candidates
4. Calculate word probabilities

deah

_eah

d_ar

de_r

... *etc*

How it works

1. Identify a misspelled word
2. Find strings n edit distance away
3. Filter candidates
4. Calculate word probabilities

deah
yeah
dear
dean
... *etc*

How it works

1. Identify a misspelled word
2. Find strings n edit distance away
3. Filter candidates
4. Calculate word probabilities

deah

yeah

dear

dean

... *etc*

How it works

1. Identify a misspelled word
2. Find strings n edit distance away
3. Filter candidates
4. Calculate word probabilities

Building the model

1. Identify a misspelled word

deah ?? 

Building the model

1. Identify a misspelled word

```
if word not in vocab:
    misspelled = True
```

deah ?? 🤔

Building the model

1. Identify a misspelled word

```
if word not in vocab:
    misspelled = True
```

deah ❌

deer 🦌 ✅

Building the model

1. Identify a misspelled word

```
if word not in vocab:
    misspelled = True
```

deah ❌

Happy birthday deer ! 🦌 ✅ 🎂

Building the model

2. Find strings n edit distance away

Building the model

2. Find strings n edit distance away

- Edit: an operation performed on a string to change it

Building the model

2. Find strings n edit distance away

- Edit: an operation performed on a string to change it
- Insert (add a letter)

Building the model

2. Find strings n edit distance away

- Edit: an operation performed on a string to change it
- Insert (add a letter)

‘to’: ‘top’, ‘two’ …

Building the model

2. Find strings n edit distance away

- Edit: an operation performed on a string to change it

- Insert (add a letter) ‘to’: ‘top’, ‘two’ …
- Delete (remove a letter)

Building the model

2. Find strings n edit distance away

- Edit: an operation performed on a string to change it

- Insert (add a letter)
- Delete (remove a letter)

'to': 'top', 'two' ...

'hat': 'ha', 'at', 'ht'

Building the model

2. Find strings n edit distance away

- Edit: an operation performed on a string to change it

- Insert (add a letter) ‘to’: ‘top’, ‘two’ …
- Delete (remove a letter) ‘hat’: ‘ha’, ‘at’, ‘ht’
- Switch (swap 2 adjacent letters)

Building the model

2. Find strings n edit distance away

- Edit: an operation performed on a string to change it

- Insert (add a letter) 'to': 'top', 'two' ...
- Delete (remove a letter) 'hat': 'ha', 'at', 'ht'
- Switch (swap 2 adjacent letters) 'eta': 'eat', 'tea'

Building the model

2. Find strings n edit distance away

- Edit: an operation performed on a string to change it

- Insert (add a letter) 'to': 'top', 'two' ...
- Delete (remove a letter) 'hat': 'ha', 'at', 'ht'
- Switch (swap 2 adjacent letters) 'eta': 'eat', 'tea' 'ate' ✗

Building the model

2. Find strings n edit distance away

- Edit: an operation performed on a string to change it

- Insert (add a letter) 'to': 'top', 'two' ...
- Delete (remove a letter) 'hat': 'ha', 'at', 'ht'
- Switch (swap 2 adjacent letters) 'eta': 'eat', 'tea'
- Replace (change 1 letter to another)

Building the model

2. Find strings n edit distance away

- Edit: an operation performed on a string to change it

- Insert (add a letter) 'to': 'top', 'two' ...
- Delete (remove a letter) 'hat': 'ha', 'at', 'ht'
- Switch (swap 2 adjacent letters) 'eta': 'eat', 'tea'
- Replace (change 1 letter to another) 'jaw': 'jar', 'paw', ...

Building the model

2. Find strings n edit distance away

- Given a string find all possible strings that are n edit distance away using
  - Input
  - Delete
  - Switch
  - Replace

Building the model

2. Find strings n edit distance away

- Given a string find all possible strings that are n edit distance away using
  - Input
  - Delete
  - Switch
  - Replace

Building the model

1. Identify a misspelled word
2. Find strings n edit distance away
3. Filter candidates
4. Calculate word probabilities

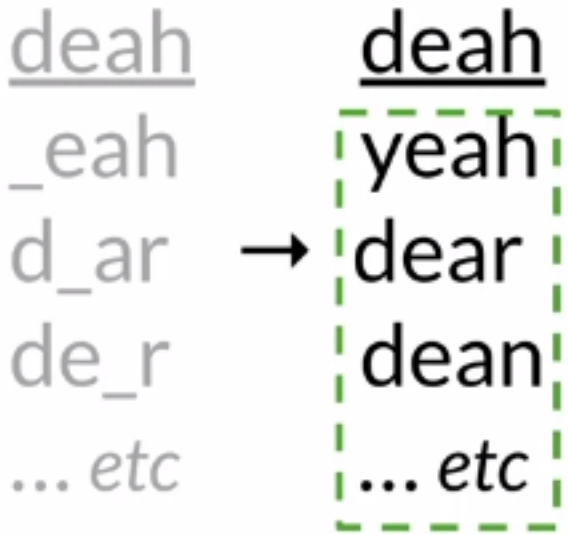
Building the model

3. Filter candidates

deah
_eah
d_ar
de_r
... *etc*

Building the model

3. Filter candidates



Building the model

1. Identify a misspelled word
2. Find strings n edit distance away
3. Filter candidates
4. Calculate word probabilities

Building the model

4. Calculate word probabilities

Building the model

4. Calculate word probabilities

   Example: “I am happy because I am learning”

Word	Count
I	2
am	2
happy	1
because	1
learning	1

Total : 7

Building the model

4. Calculate word probabilities

Example: "I am happy because I am learning"

Word	Count
I	2
am	2
happy	1
because	1
learning	1

Total : 7

Building the model

4. Calculate word probabilities

Example: “I am happy because I am learning”

Word	Count
I	2
am	2
happy	1
because	1
learning	1

Total : 7

Building the model

4. Calculate word probabilities

   Example: “I am happy because I am learning”

Word	Count
I	2
am	2
happy	1
because	1
learning	1
Total :	7

Building the model

4. Calculate word probabilities

Example: "I am happy because I am learning"

$$P(w) = \frac{C(w)}{V}$$

$P(w)$ Probability of a word

$C(w)$ Number of times the word appears

V Total size of the corpus

Word	Count
I	2
am	2
happy	1
because	1
learning	1

Total : 7

Building the model

4. Calculate word probabilities

Example: "I am happy because I am learning"

$$P(w) = \frac{C(w)}{V}$$

$$P(\text{am}) = \frac{C(\text{am})}{V} = \frac{2}{7}$$

$P(w)$ Probability of a word

$C(w)$ Number of times the word appears

V Total size of the corpus

Word	Count
I	2
am	2
happy	1
because	1
learning	1

Total : 7

Building the model

4. Calculate word probabilities

Building the model

4. Calculate word probabilities

Summary

1. Identify a misspelled word
2. Find strings n edit distance away
   - Insert
   - Delete
   - Switch
   - Replace
3. Filter candidates
4. Calculate word probabilities

$$P(w) = \frac{C(w)}{V}$$

deah → dear ✅

yeah

dear

dean

... etc

Summary

1. Identify a misspelled word
2. Find strings n edit distance away
   - Insert
   - Delete
   - Switch
   - Replace
3. Filter candidates
4. Calculate word probabilities

$$P(w) = \frac{C(w)}{V}$$

deah → dear ✅

yeah

dear

dean

... *etc*

Summary

1. Identify a misspelled word
2. Find strings n edit distance away
   - Insert
   - Delete
   - Switch
   - Replace
3. Filter candidates
4. Calculate word probabilities

$$P(w) = \frac{C(w)}{V}$$

deah → dear ✅

yeah

dear

dean

... *etc*

Summary

1. Identify a misspelled word
2. Find strings n edit distance away
   - Insert
   - Delete
   - Switch
   - Replace
3. Filter candidates
4. Calculate word probabilities

$$P(w) = \frac{C(w)}{V}$$

deah → dear ✅

_eah

d_ar

de_r

... *etc*

Summary

1. Identify a misspelled word
2. Find strings n edit distance away
   - Insert
   - Delete
   - Switch
   - Replace
3. Filter candidates
4. Calculate word probabilities

$$P(w) = \frac{C(w)}{V}$$

deah → dear ✅

yeah

dear

dean

... *etc*

Summary

1. Identify a misspelled word
2. Find strings n edit distance away
   - Insert
   - Delete
   - Switch
   - Replace
3. Filter candidates
4. Calculate word probabilities

$$P(w) = \frac{C(w)}{V}$$

deah → dear ✅

yeah

dear

dean

... *etc*

Summary

1. Identify a misspelled word
2. Find strings n edit distance away
   - Insert
   - Delete
   - Switch
   - Replace
3. Filter candidates
4. Calculate word probabilities

$$P(w) = \frac{C(w)}{V}$$

deah → dear ✅

yeah

dear

dean

... etc

Minimum edit distance

- How to evaluate similarity between 2 strings?
- Minimum number of edits needed to transform 1 string into the other
- Spelling correction, document similarity, machine translation, DNA sequencing, and more

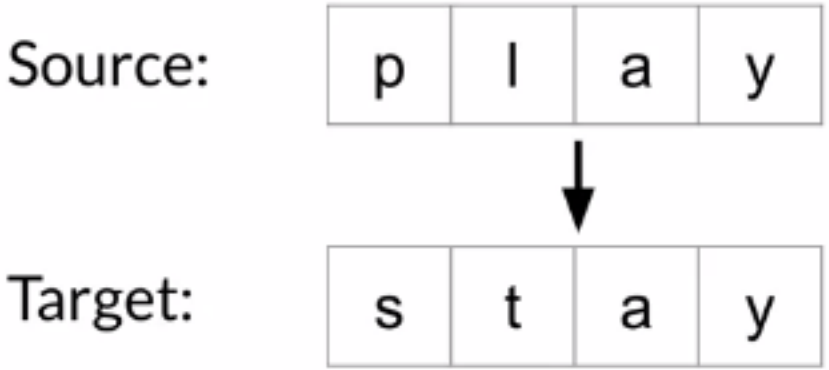
Minimum edit distance

- Edits:

• Insert	(add a letter)	‘to’: ‘top’, ‘two’ …
• Delete	(remove a letter)	‘hat’: ‘ha’, ‘at’, ‘ht’
• Replace	(change 1 letter to another)	‘jaw’: ‘jar’, ‘paw’, …

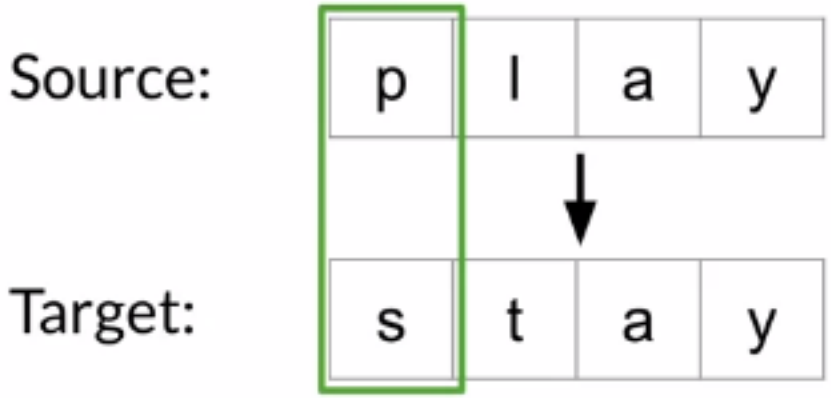
Minimum edit distance

- Example:



Minimum edit distance

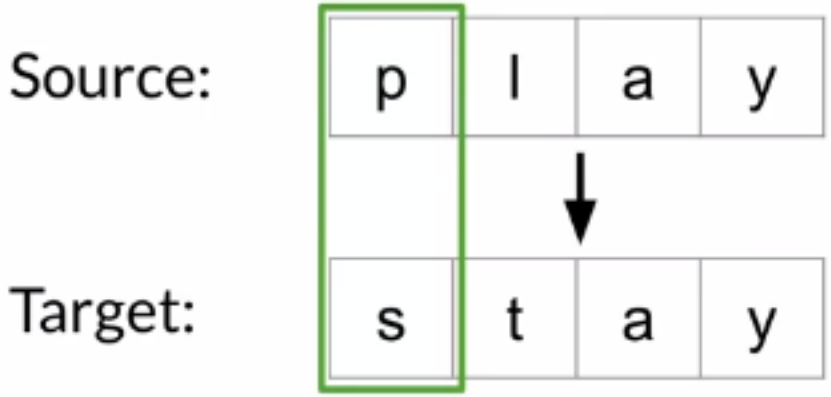
- Example:



p → s : replace

Minimum edit distance

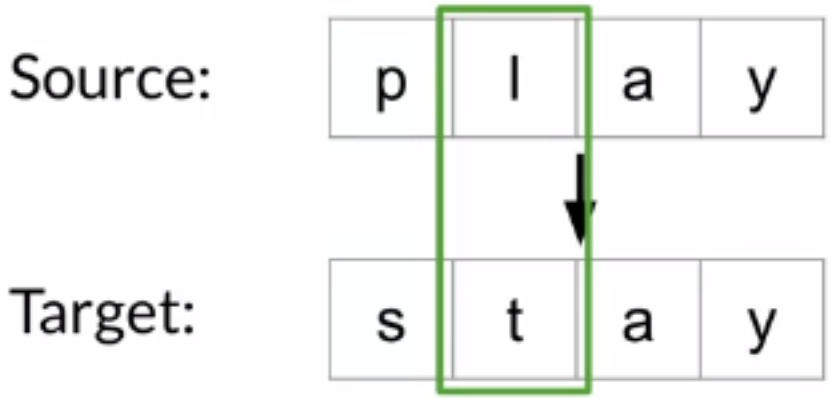
- Example:



p → s : replace

Minimum edit distance

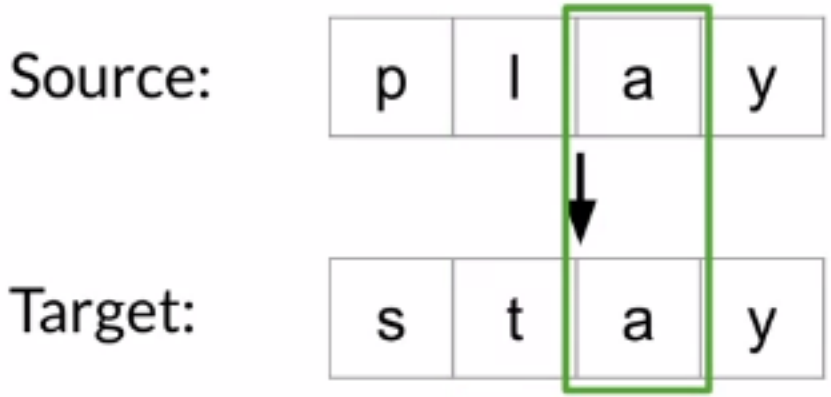
- Example:



p → s : replace
l → t : replace

Minimum edit distance

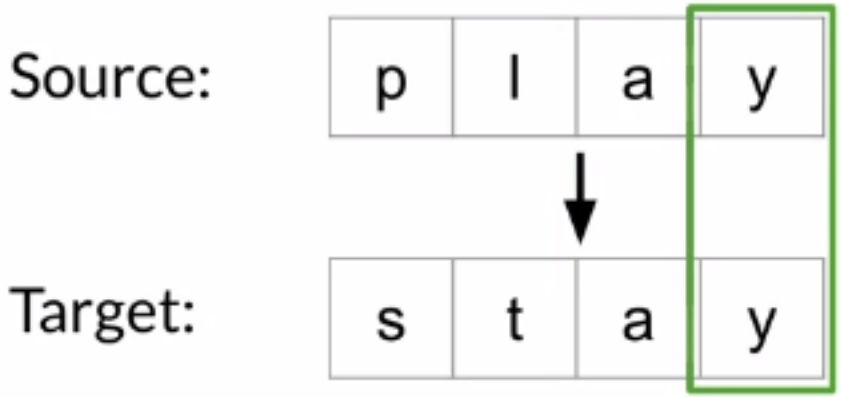
- Example:



p → s : replace
l → t : replace

Minimum edit distance

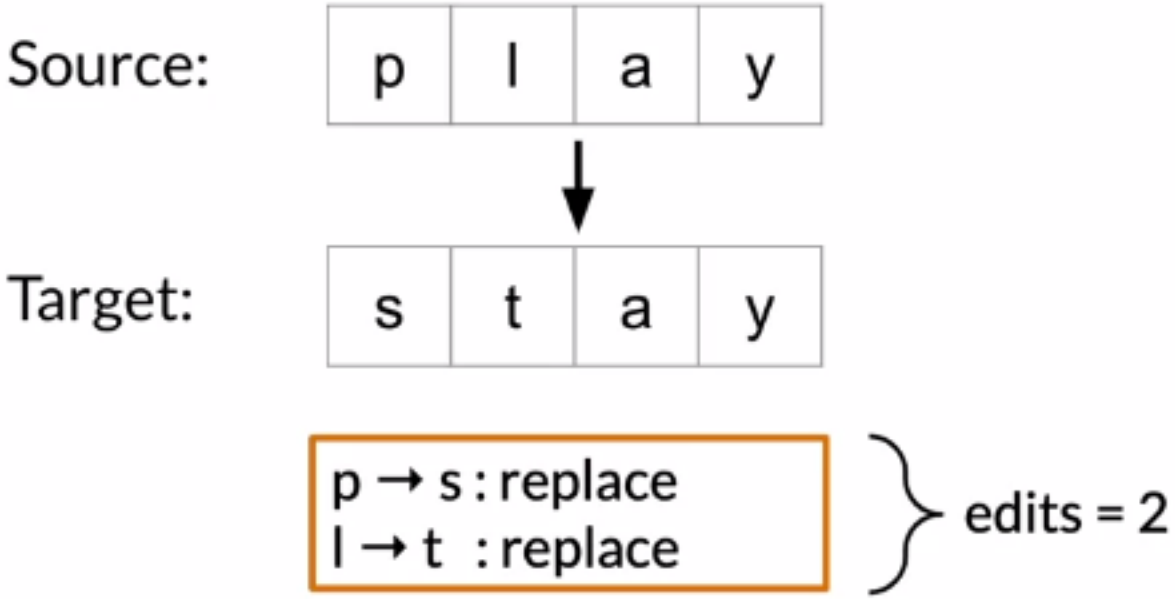
- Example:



p → s : replace
l → t : replace

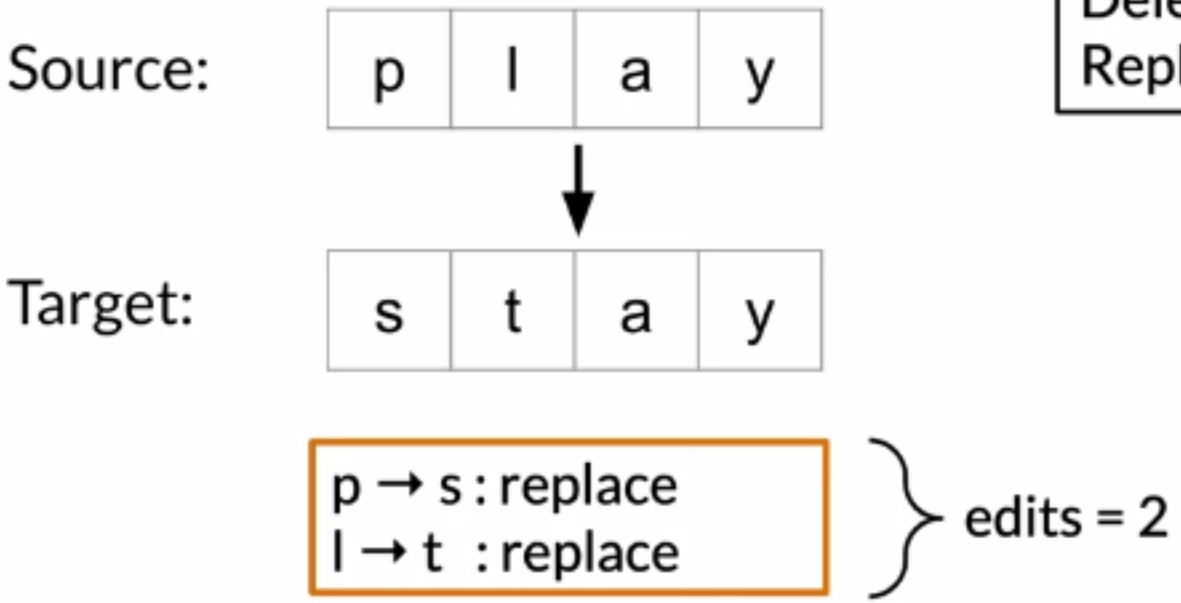
Minimum edit distance

- Example:



Minimum edit distance

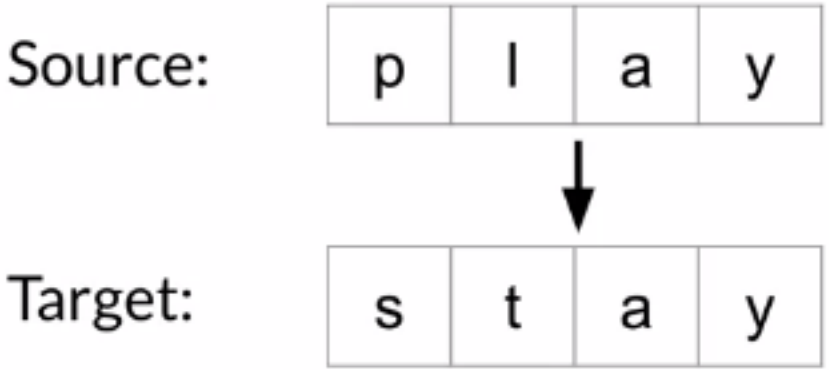
- Example:



Edit cost:
Insert 1
Delete 1
Replace 2

Minimum edit distance

- Example:



Edit cost:
Insert 1
Delete 1
Replace 2

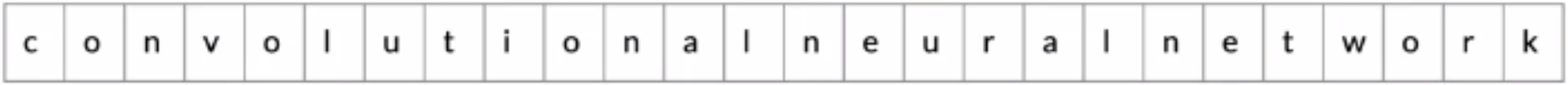
edit distance = 2 * 2 = 4

p → s : replace
l → t : replace

edits = 2

Minimum edit distance

- Example:



Minimum edit distance

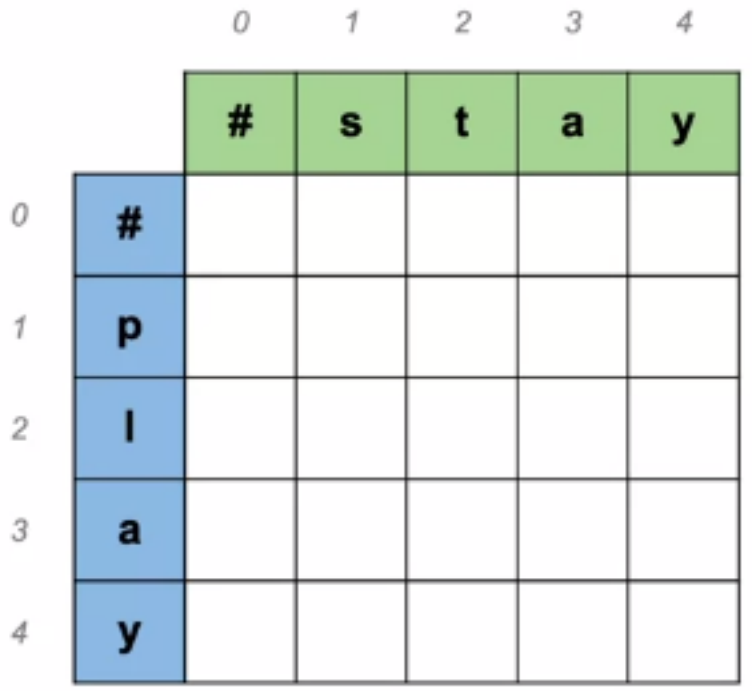
- Example:

c	o	n	v	o	l	u	t	i	o	n	a	l	n	e	u	r	a	l	n	e	t	w	o	r	k

CCAAGGGGTGACTCTAGTTTAATATAACTGAGATCAAATTATATGGGTGAT  !!

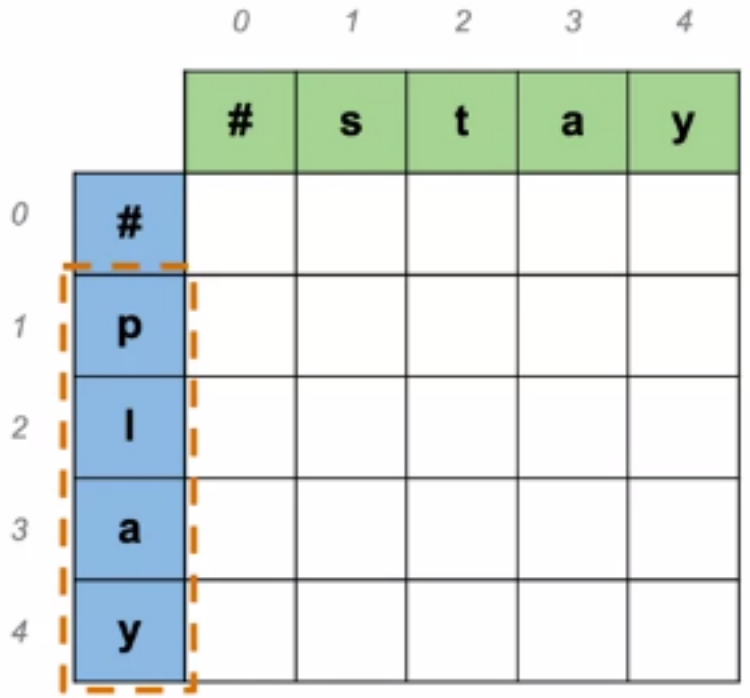
Minimum edit distance

Source: play → Target: stay



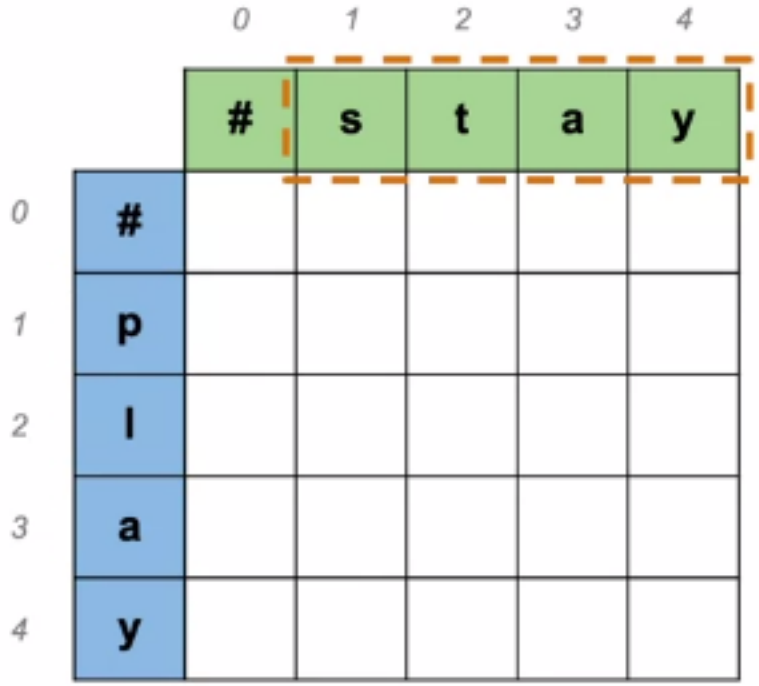
Minimum edit distance

Source: play → Target: stay



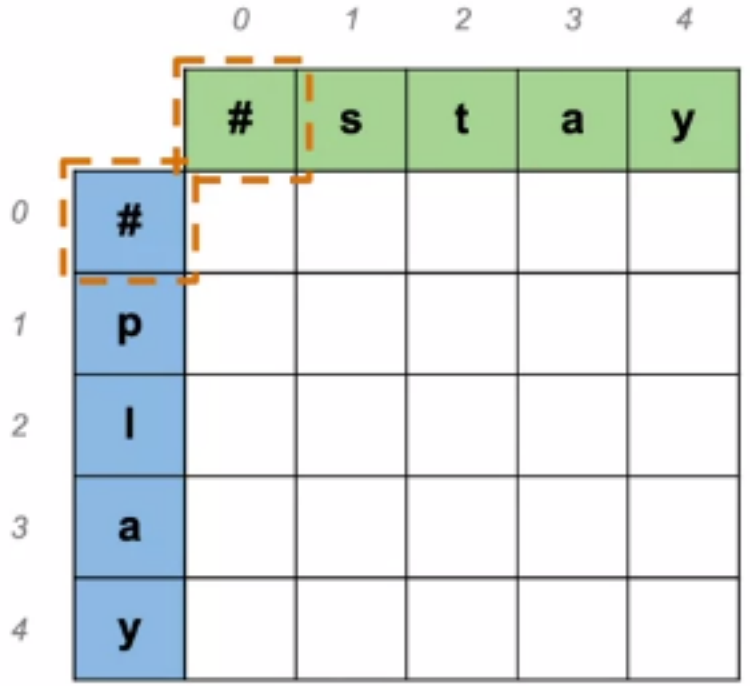
Minimum edit distance

Source: play → Target: stay



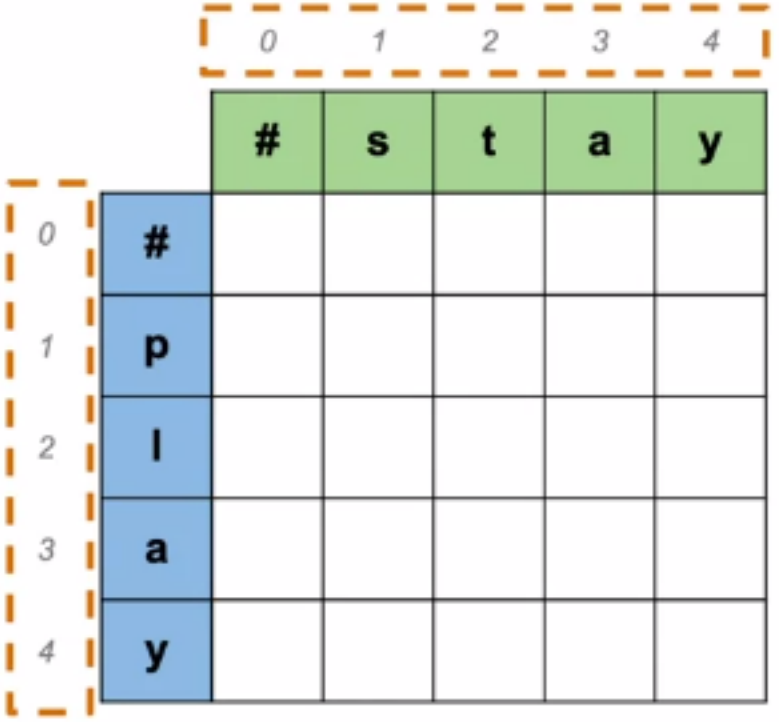
Minimum edit distance

Source: play → Target: stay



Minimum edit distance

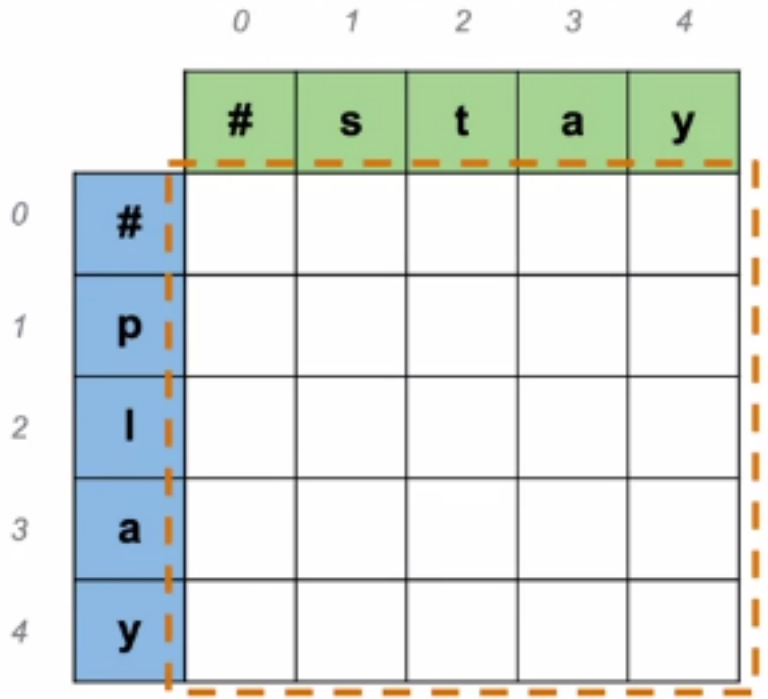
Source: play → Target: stay



Minimum edit distance

Source: play → Target: stay

$D[\,]$

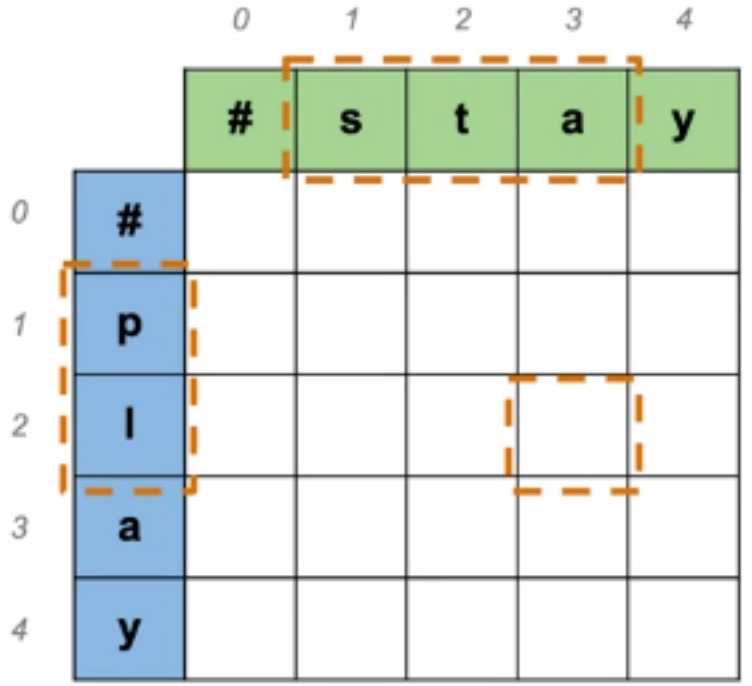


Minimum edit distance

Source: play → Target: stay

$D[\ ]$

$D[2,3]$ = pl → sta



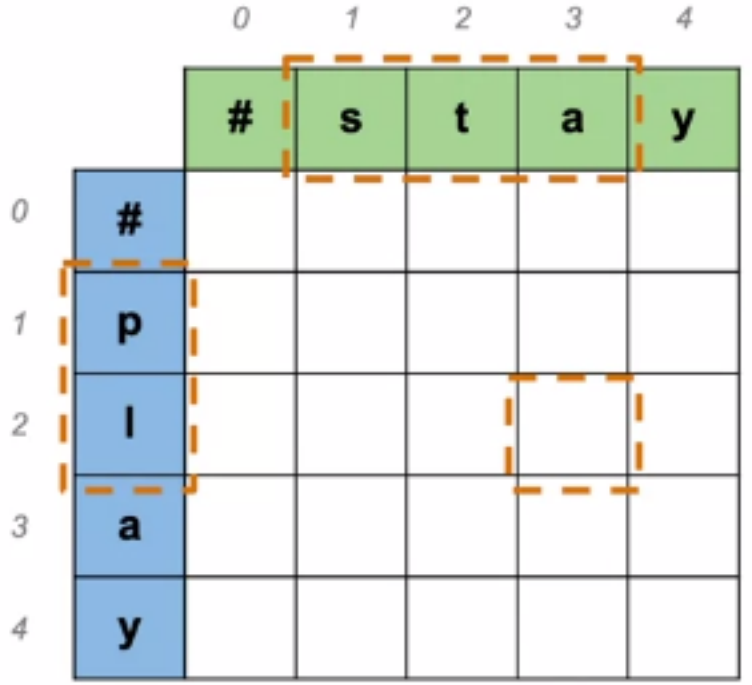
Minimum edit distance

Source: play → Target: stay

$D[\ ]$

$D[2,3]$ = pl → sta

$D[2,3]$ = source[:2] → target[:3]



Minimum edit distance

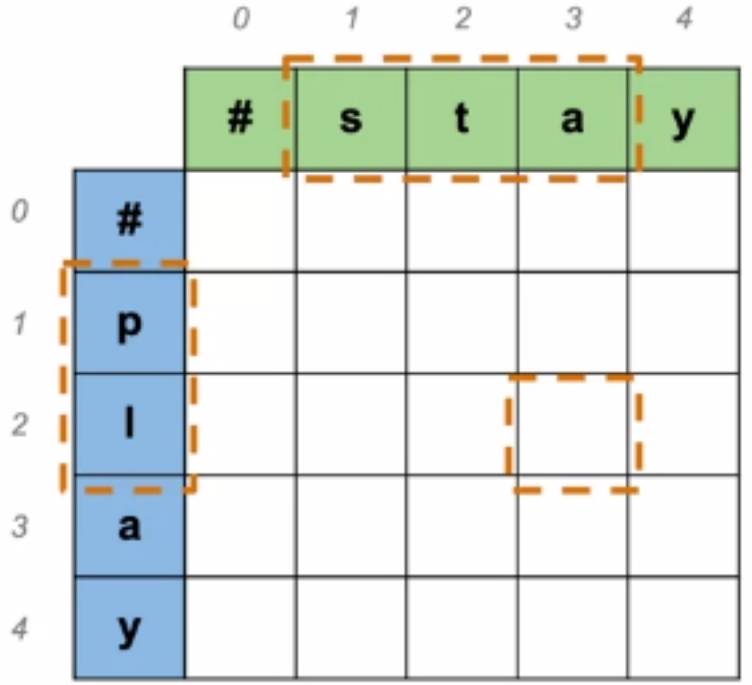
Source: play → Target: stay

$D[\ ]$

$D[2,3]$ = pl → sta

$D[2,3]$ = source[:2] → target[:3]

$D[\ i, j\ ]$ = source[: i] → target[: j]

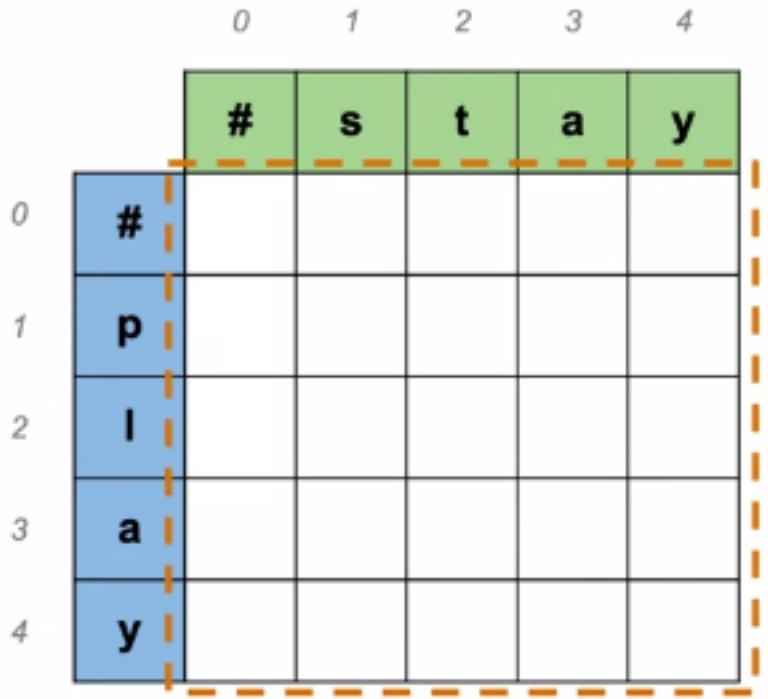


Minimum edit distance

Source: play → Target: stay

$D[\,]$

$D[\,i, j\,] =$ source$[\,:i\,] \rightarrow$ target$[\,:j\,]$



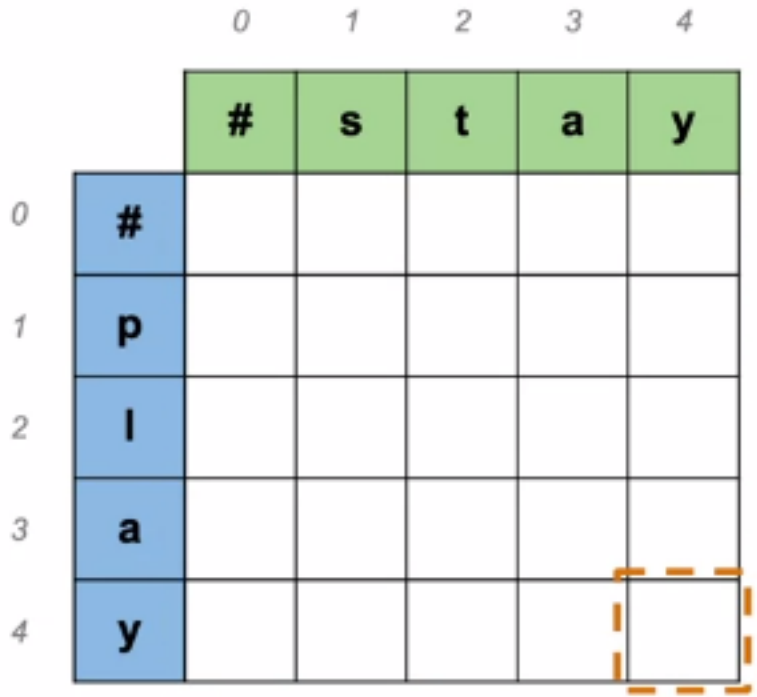
Minimum edit distance

Source: play → Target: stay

$D[\,]$

$D[\,i, j\,] =$ source$[\,:i\,] \rightarrow$ target$[\,:j\,]$

$D[\,m, n\,] =$ source → target



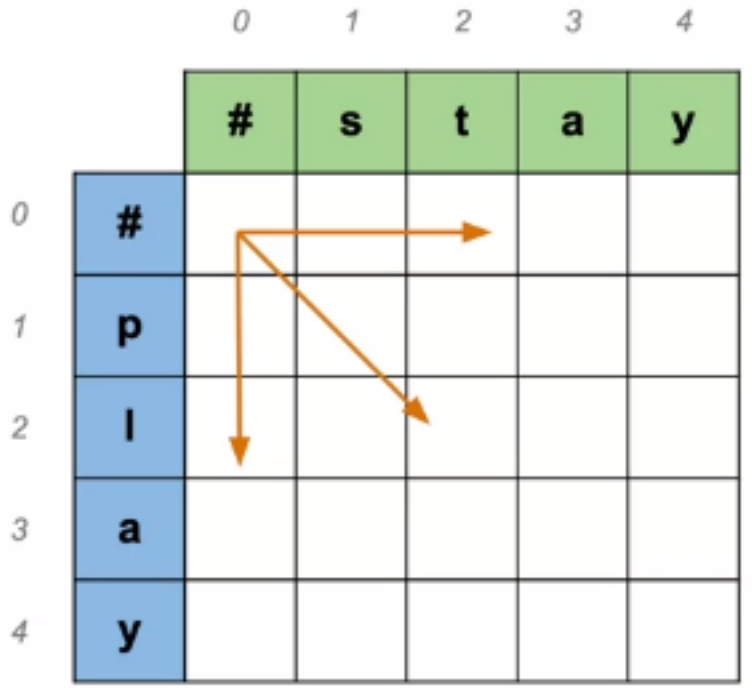
Minimum edit distance

Source: play → Target: stay

$D[\,]$

$D[\,i, j\,] =$ source$[\,:i\,] \rightarrow$ target$[\,:j\,]$

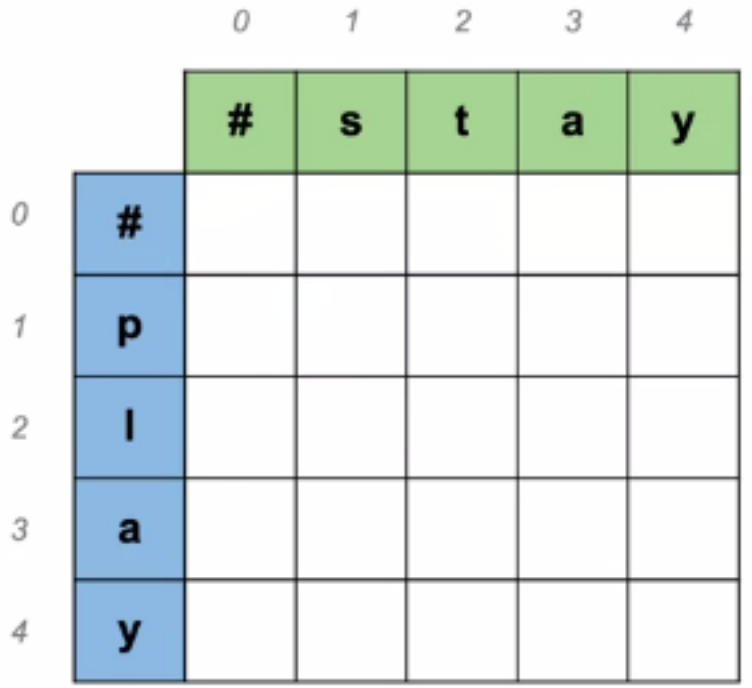
$D[\,m, n\,] =$ source $\rightarrow$ target



Minimum edit distance

Source: play → Target: stay

Cost: insert: 1, delete: 1, replace: 2

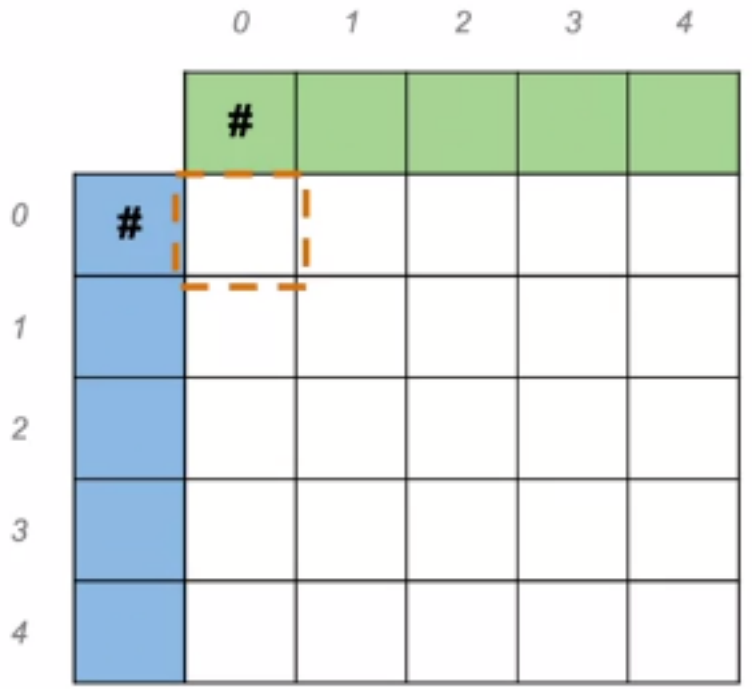


Minimum edit distance

Source: play → Target: stay

Cost: insert: 1, delete: 1, replace: 2

→

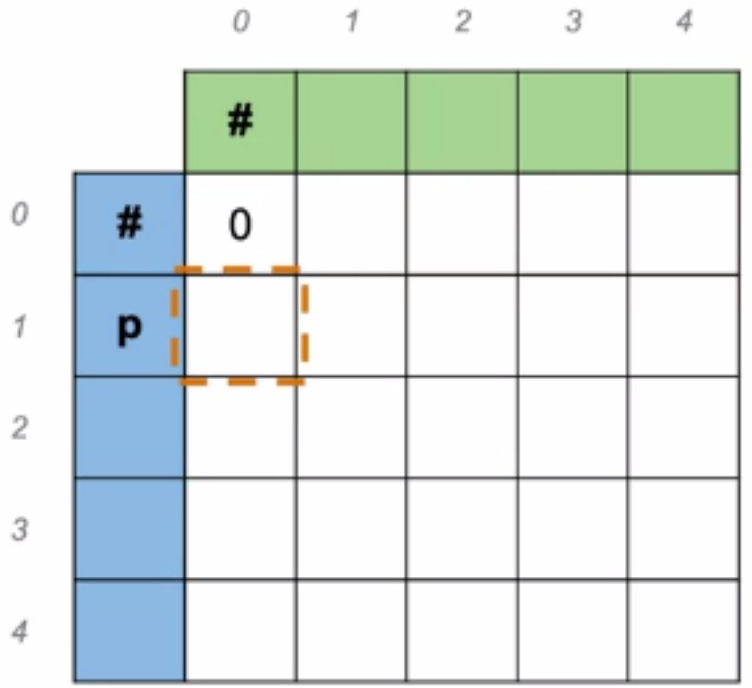


Minimum edit distance

Source: play → Target: stay

Cost: insert: 1, delete: 1, replace: 2

p → #



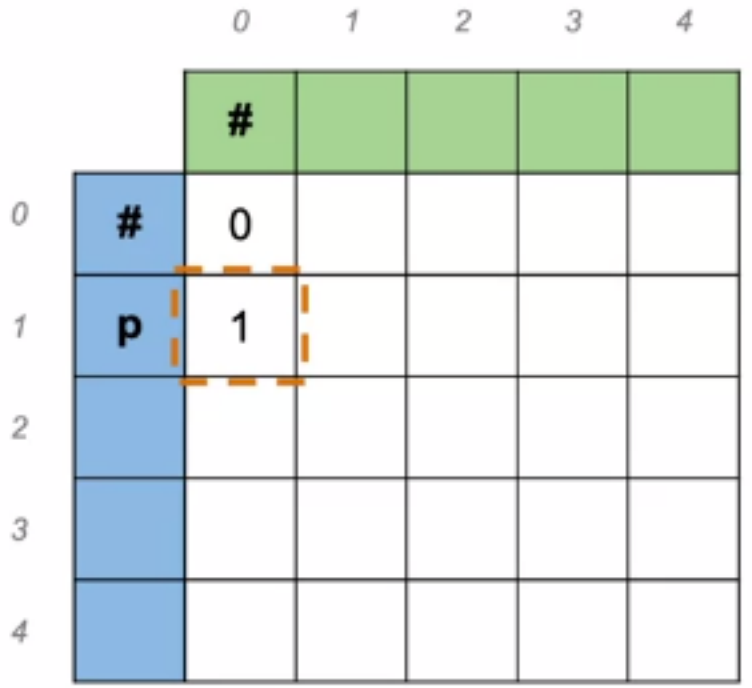
Minimum edit distance

Source: play → Target: stay

Cost: insert: 1, delete: 1, replace: 2

p → #

delete

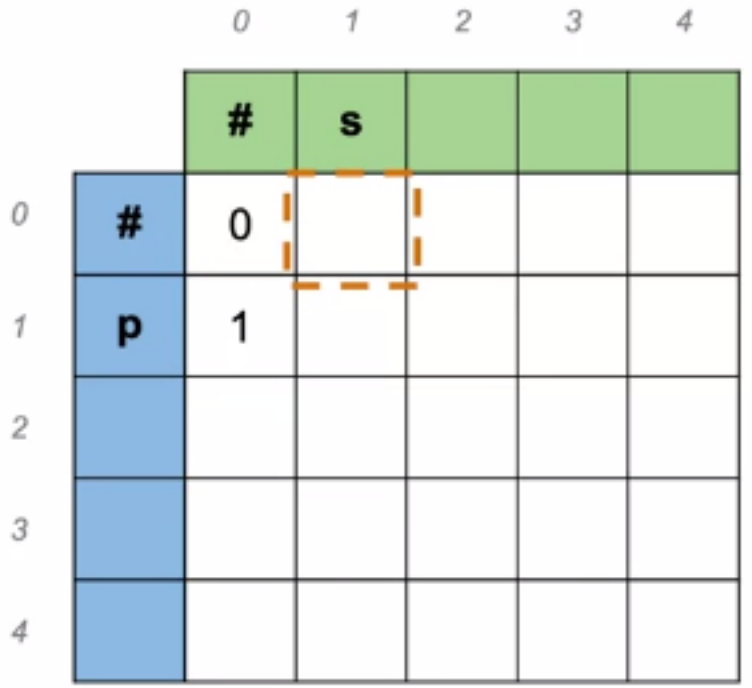


Minimum edit distance

Source: play → Target: stay

Cost: insert: 1, delete: 1, replace: 2

→ s



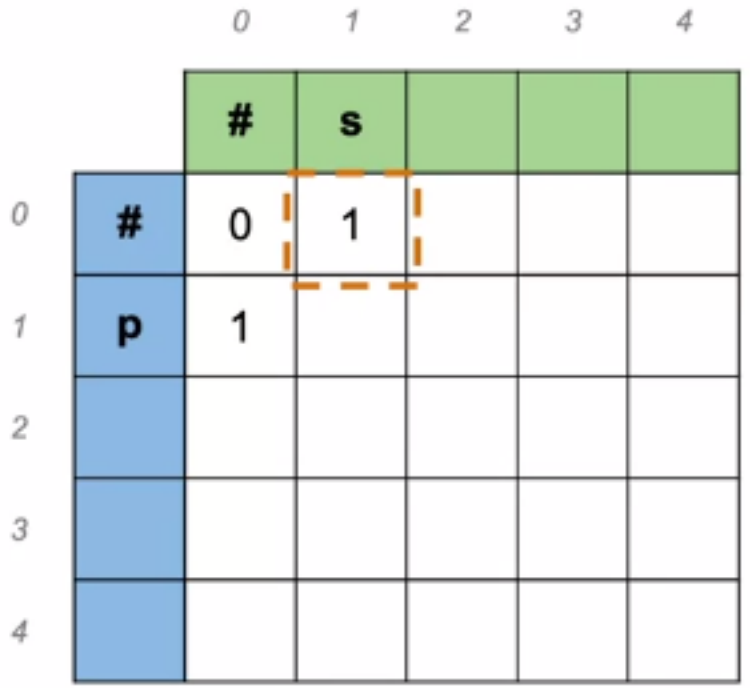
Minimum edit distance

Source: play → Target: stay

Cost: insert: 1, delete: 1, replace: 2

→ s

insert

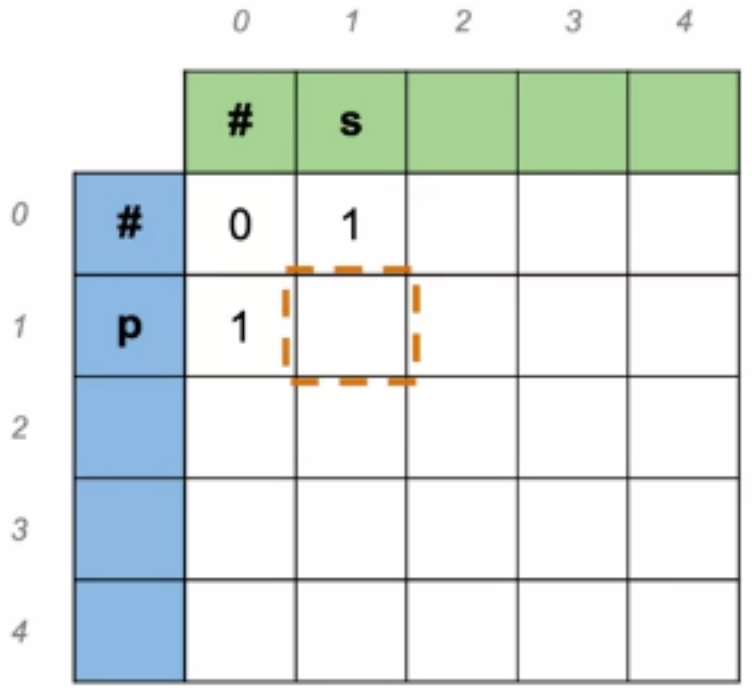


Minimum edit distance

Source: play → Target: stay

Cost: insert: 1, delete: 1, replace: 2

p → s



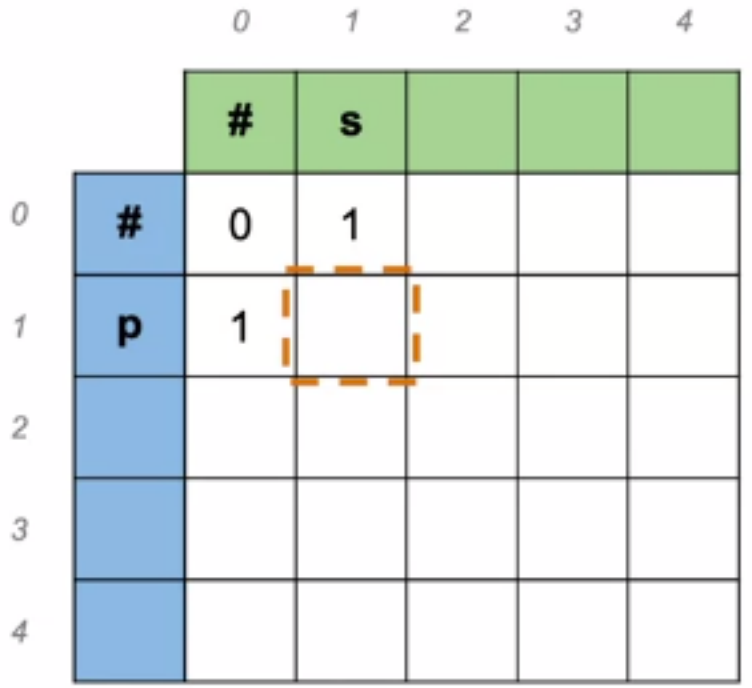
Minimum edit distance

Source: play → Target: stay

Cost: insert: 1, delete: 1, replace: 2

p → s

insert + delete: p → ps → s



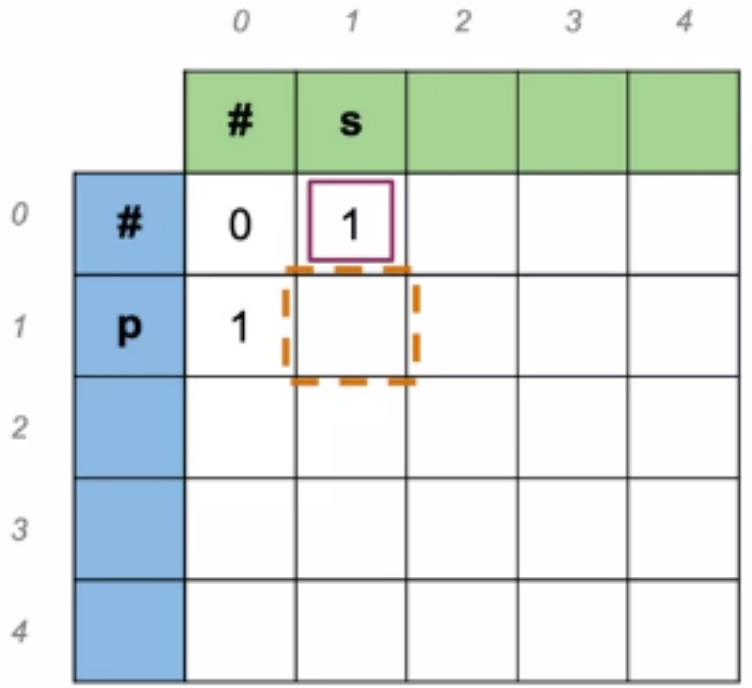
Minimum edit distance

Source: play → Target: stay

Cost: insert: 1, delete: 1, replace: 2

p → s

insert + *delete*: p → ps → s



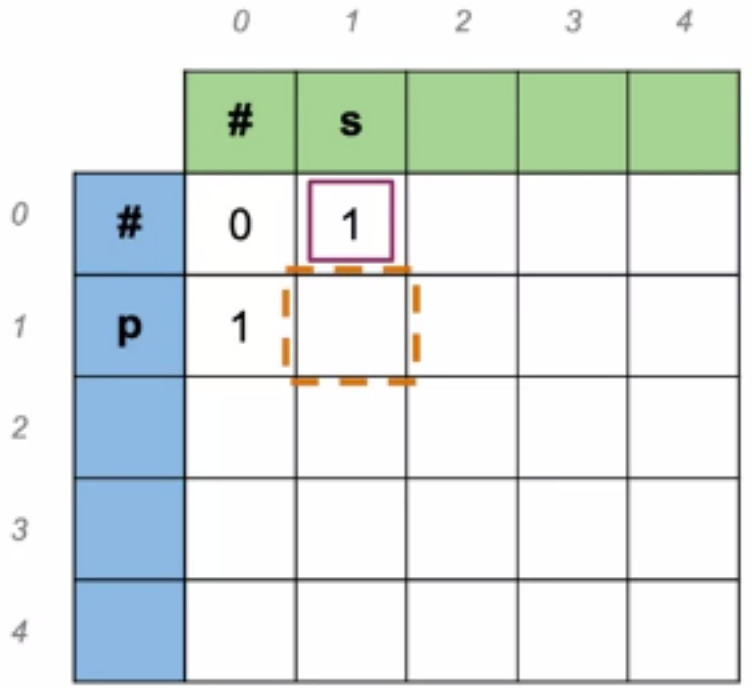
Minimum edit distance

Source: play → Target: stay

Cost: insert: 1, delete: 1, replace: 2

p → s

insert + *delete*: p → ps → s: 2



Minimum edit distance

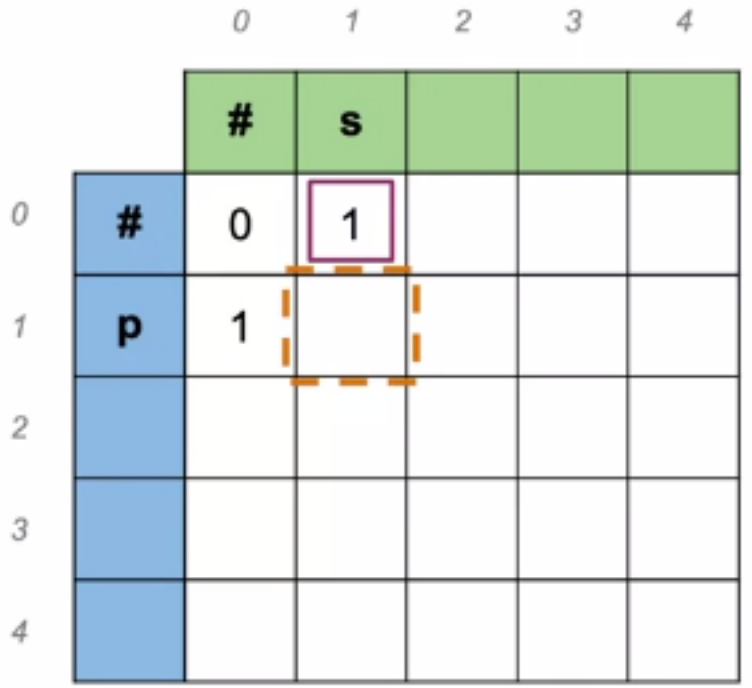
Source: play → Target: stay

Cost: insert: 1, delete: 1, replace: 2

p → s

insert + *delete*: p → ps → s: 2

delete + *insert*: p → # → s



Minimum edit distance

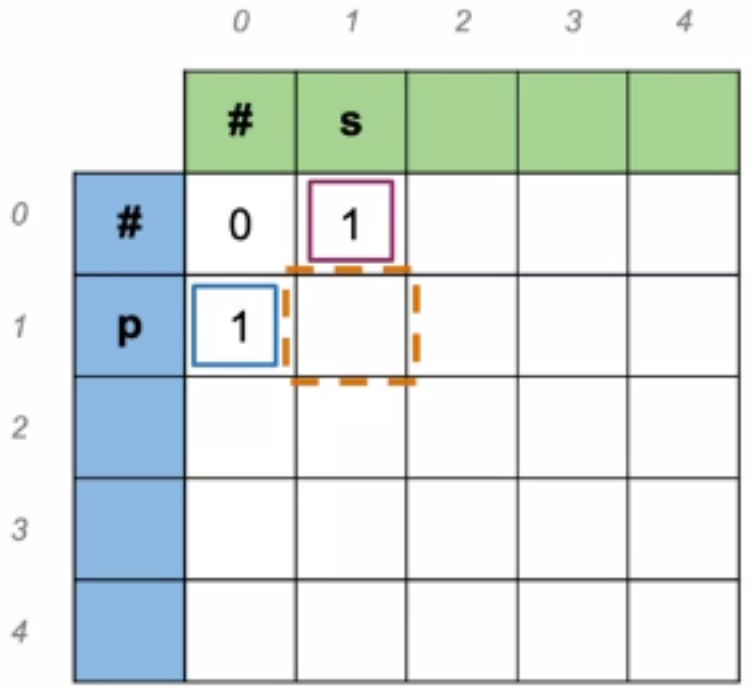
Source: play → Target: stay

Cost: insert: 1, delete: 1, replace: 2

p → s

insert + *delete*: p → ps → s: 2

delete + *insert*: p → # → s



Minimum edit distance

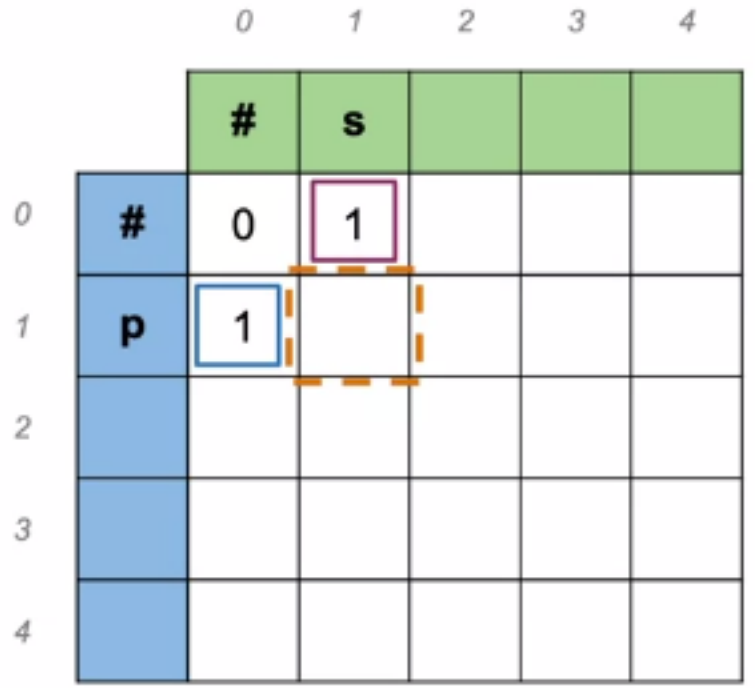
Source: play → Target: stay

Cost: insert: 1, delete: 1, replace: 2

p → s

insert + *delete*: p → ps → s: 2

delete + *insert*: p → # → s: 2



Minimum edit distance

Source: play → Target: stay

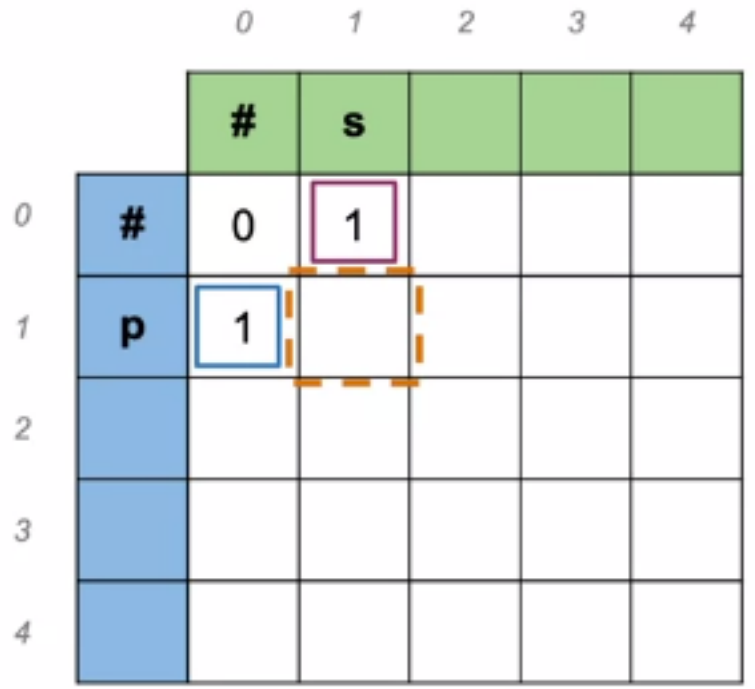
Cost: insert: 1, delete: 1, replace: 2

p → s

insert + *delete*: p → ps → s: 2

delete + *insert*: p → # → s: 2

replace: p → s



Minimum edit distance

Source: play → Target: stay

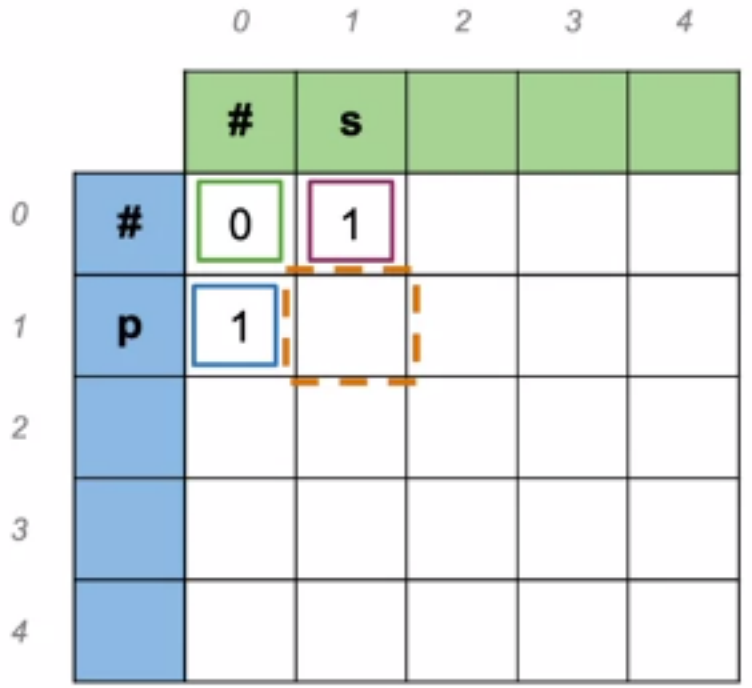
Cost: insert: 1, delete: 1, replace: 2

p → s

insert + *delete*: p → ps → s: 2

delete + *insert*: p → # → s: 2

replace: p → s: 2



Minimum edit distance

Source: play → Target: stay

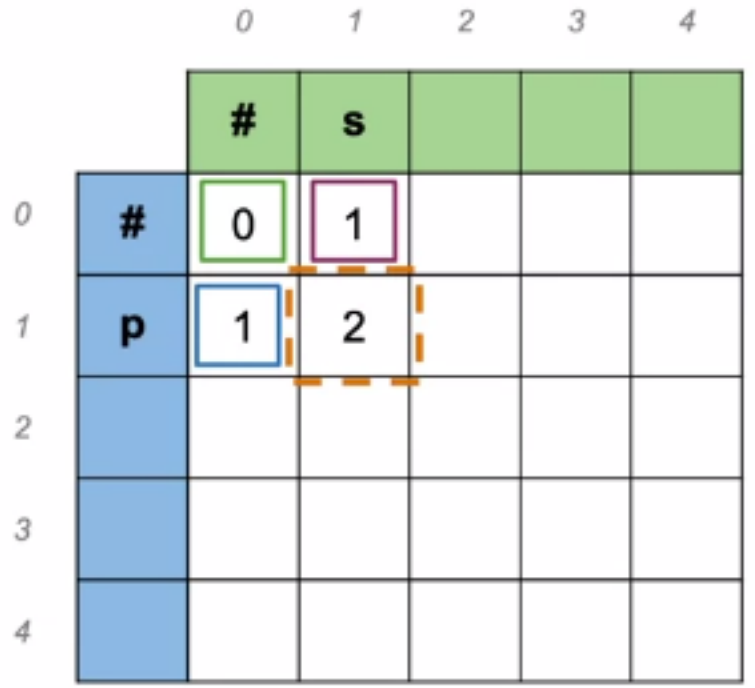
Cost: insert: 1, delete: 1, replace: 2

p → s

insert + *delete*: p → ps → s: 2

delete + *insert*: p → # → s: 2

replace: p → s: 2



Minimum edit distance

Source: play → Target: stay

Cost: insert: 1, delete: 1, replace: 2

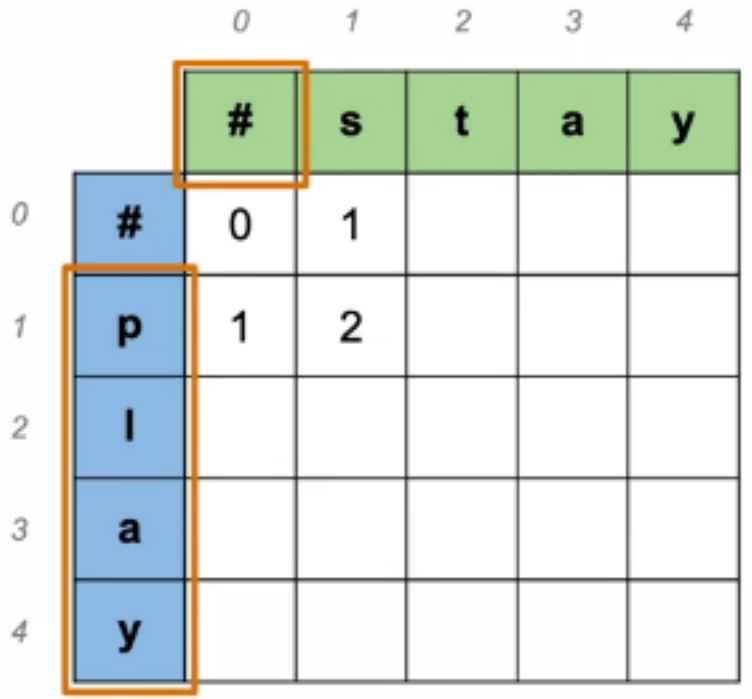
		0	1	2	3	4
		#	s	t	a	y
0	#	0	1			
1	p	1	2			
2	l					
3	a					
4	y					

Minimum edit distance

Source: play → Target: stay

Cost: insert: 1, delete: 1, replace: 2

play → #



		0	1	2	3	4
		#	s	t	a	y
0	#	0	1			
1	p	1	2			
2	l					
3	a					
4	y					

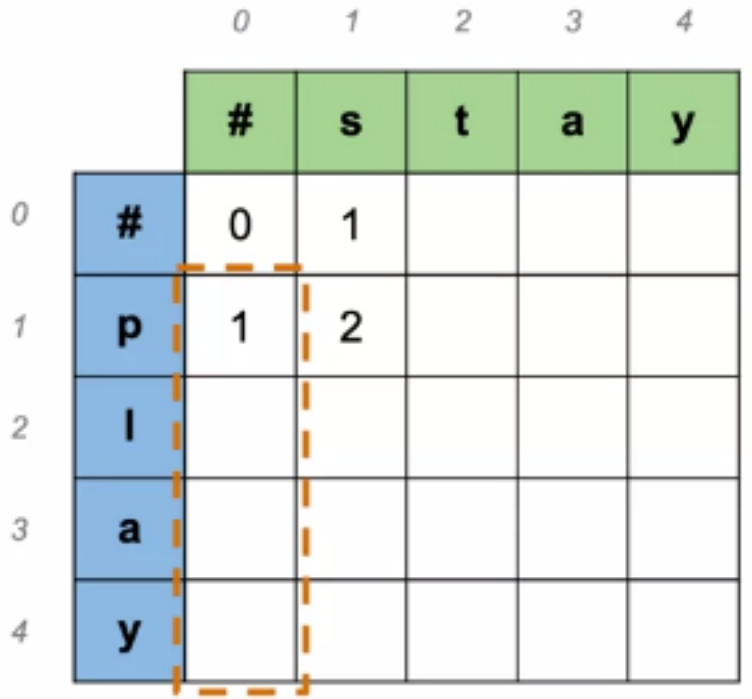
Minimum edit distance

Source: play → Target: stay

Cost: insert: 1, delete: 1, replace: 2

play → #

$D[i, j] = D[i-1, j] + del\_cost$



Minimum edit distance

Source: play → Target: stay

Cost: insert: 1, delete: 1, replace: 2

play → #

$D[i, j] = D[i-1, j] + del\_cost$

		0	1	2	3	4
		#	s	t	a	y
0	#	0	1			
1	p	1	2			
2	l	2				
3	a	3				
4	y	4				

Minimum edit distance

Source: play → Target: stay
Cost: insert: 1, delete: 1, replace: 2

play → #

$D[i,j] = D[i\text{-}1,j] + del\_cost$

$D[4,0]$ = play → #
= source[:4] → target[0]

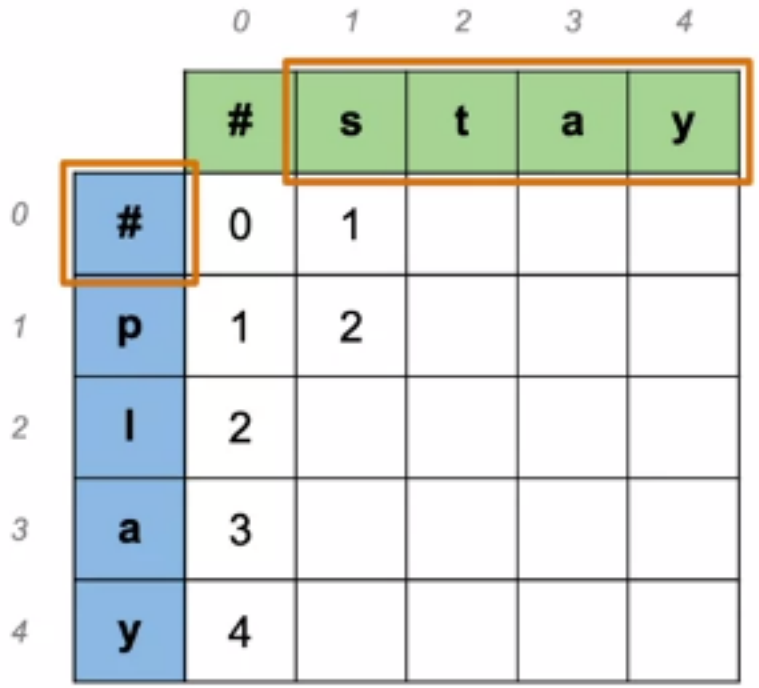
		0	1	2	3	4
		#	s	t	a	y
0	#	0	1			
1	p	1	2			
2	l	2				
3	a	3				
4	y	4				

Minimum edit distance

Source: play → Target: stay

Cost: insert: 1, delete: 1, replace: 2

→ play



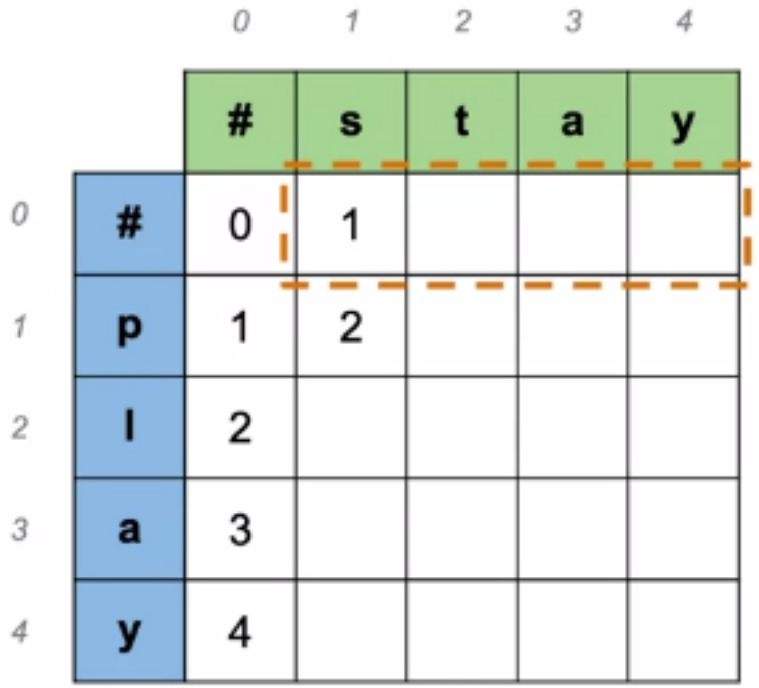
Minimum edit distance

Source: play → Target: stay

Cost: insert: 1, delete: 1, replace: 2

→ play

$D[i,j] = D[i,j\text{-}1] + ins\_cost$



Minimum edit distance

Source: play → Target: stay

Cost: insert: 1, delete: 1, replace: 2

→ play

$D[i, j] = D[i, j\text{-}1] + ins\_cost$

		0	1	2	3	4
		#	s	t	a	y
0	#	0	1	2	3	4
1	p	1	2			
2	l	2				
3	a	3				
4	y	4				

Minimum edit distance

Source: play → Target: stay

Cost: insert: 1, delete: 1, replace: 2

p → s

$$D[i,j] = \min \begin{cases} D[i-1,j] + del\_cost \\ D[i,j-1] + ins\_cost \\ D[i-1,j-1] + \begin{cases} rep\_cost; & \text{if } src[i] \neq tar[j] \\ 0; & \text{if } src[i] = tar[j] \end{cases} \end{cases}$$

		0	1	2	3	4
		#	s	t	a	y
0	#	0	1	2	3	4
1	p	1	2			
2	l	2				
3	a	3				
4	y	4				

Minimum edit distance

Source: play → Target: stay

Cost: insert: 1, delete: 1, replace: 2

p → s

$$D[i,j] = \min \begin{cases} D[i-1,j] + del\_cost \\ D[i,j-1] + ins\_cost \\ D[i-1,j-1] + \begin{cases} rep\_cost; & \text{if } src[i] \neq tar[j] \\ 0; & \text{if } src[i] = tar[j] \end{cases} \end{cases}$$

		0	1	2	3	4
		#	s	t	a	y
0	#	0	1	2	3	4
1	p	1	2			
2	l	2				
3	a	3				
4	y	4				

Minimum edit distance

Source: play → Target: stay

Cost: insert: 1, delete: 1, replace: 2

p → s

$$D[i,j] = \min \begin{cases} D[i-1,j] + del\_cost \\ D[i,j-1] + ins\_cost \\ D[i-1,j-1] + \begin{cases} rep\_cost; & \text{if } src[i] \neq tar[j] \\ 0; & \text{if } src[i] = tar[j] \end{cases} \end{cases}$$

		0	1	2	3	4
		#	s	t	a	y
0	#	0	1	2	3	4
1	p	1	2			
2	l	2				
3	a	3				
4	y	4				

Minimum edit distance

Source: play → Target: stay

Cost: insert: 1, delete: 1, replace: 2

p → s

$$D[i,j] = \min \begin{cases} D[i-1,j] + del\_cost \\ D[i,j-1] + ins\_cost \\ D[i-1,j-1] + \begin{cases} rep\_cost; & \text{if } src[i] \neq tar[j] \\ 0; & \text{if } src[i] = tar[j] \end{cases} \end{cases}$$

		0	1	2	3	4
		#	s	t	a	y
0	#	0	1	2	3	4
1	p	1	2			
2	l	2				
3	a	3				
4	y	4				

Minimum edit distance

Source: play → Target: stay

Cost: insert: 1, delete: 1, replace: 2

p → s

$$D[i,j] = \min \begin{cases} D[i-1,j] + del\_cost \\ D[i,j-1] + ins\_cost \\ D[i-1,j-1] + \begin{cases} rep\_cost; & \text{if } src[i] \neq tar[j] \\ 0; & \text{if } src[i] = tar[j] \end{cases} \end{cases}$$

		0	1	2	3	4
		#	s	t	a	y
0	#	0	1	2	3	4
1	p	1	2			
2	l	2				
3	a	3				
4	y	4				

Minimum edit distance

Source: play → Target: stay

Cost: insert: 1, delete: 1, replace: 2

p → s

$$D[i,j] = \min \begin{cases} D[i-1,j] + del\_cost \\ D[i,j-1] + ins\_cost \\ D[i-1,j-1] + \begin{cases} rep\_cost; & \text{if } src[i] \neq tar[j] \\ 0; & \text{if } src[i] = tar[j] \end{cases} \end{cases}$$

		0	1	2	3	4
		#	s	t	a	y
0	#	0	1	2	3	4
1	p	1	2			
2	l	2				
3	a	3				
4	y	4				

Minimum edit distance

Source: play → Target: stay

Cost: insert: 1, delete: 1, replace: 2

p → s

$D[i\text{-}1, j] + 1 = 2$

$D[i, j\text{-}1] + 1 = 2$

$D[i\text{-}1, j\text{-}1] + 2 = 2$

		0	1	2	3	4
		#	s	t	a	y
0	#	0	1	2	3	4
1	p	1	2			
2	l	2				
3	a	3				
4	y	4				

Minimum edit distance

Source: play → Target: stay

Cost: insert: 1, delete: 1, replace: 2

p → s

$D[i\text{-}1, j] + 1 = 2$

$D[i, j\text{-}1] + 1 = 2$

$D[i\text{-}1, j\text{-}1] + 2 = 2$

		0	1	2	3	4
		#	s	t	a	y
0	#	0	1	2	3	4
1	p	1	2			
2	l	2				
3	a	3				
4	y	4				

Minimum edit distance

Source: play → Target: stay

Cost: insert: 1, delete: 1, replace: 2

p → s

$D[i-1, j] + 1 = 2$

$D[i, j-1] + 1 = 2$

$D[i-1, j-1] + 2 = 2$

} min = 2

		0	1	2	3	4
		#	s	t	a	y
0	#	0	1	2	3	4
1	p	1	2			
2	l	2				
3	a	3				
4	y	4				

Minimum edit distance

Source: play → Target: stay

Cost: insert: 1, delete: 1, replace: 2

play → stay

D[*m*, *n*] = 4

		0	1	2	3	4
		#	s	t	a	y
0	#	0	1	2	3	4
1	p	1	2	3	4	5
2	l	2	3	4	5	6
3	a	3	4	5	4	5
4	y	4	5	6	5	4

Minimum edit distance

Source: play → Target: stay

Cost: insert: 1, delete: 1, replace: 2

play → stay

D[*m*, *n*] = 4

		0	1	2	3	4
		#	s	t	a	y
0	#	0	1	2	3	4
1	p	1	2	3	4	5
2	l	2	3	4	5	6
3	a	3	4	5	4	5
4	y	4	5	6	5	4

Minimum edit distance

Source: play → Target: stay

Cost: insert: 1, delete: 1, replace: 2

play → stay

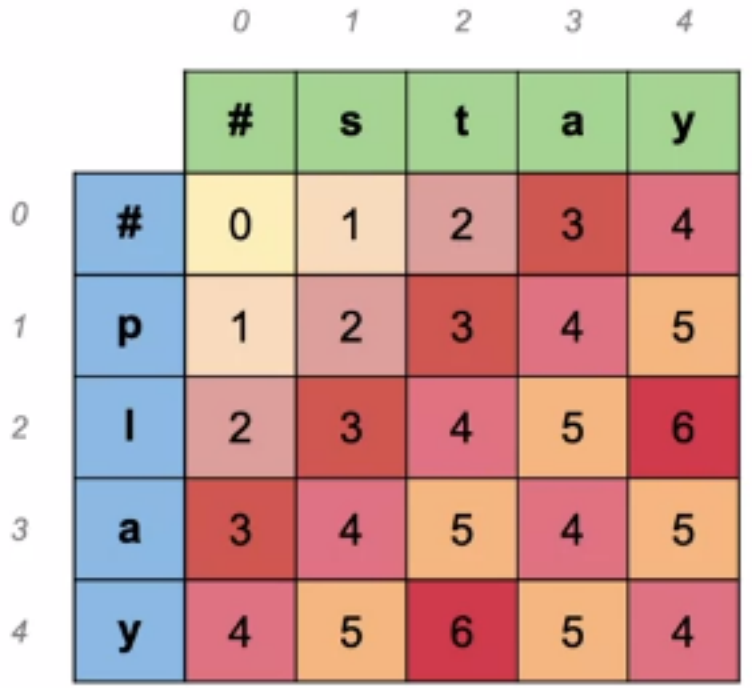
D[*m*, *n*] = 4

		0	1	2	3	4
		#	s	t	a	y
0	#	0	1	2	3	4
1	p	1	2	3	4	5
2	l	2	3	4	5	6
3	a	3	4	5	4	5
4	y	4	5	6	5	4

Minimum edit distance

Source: play → Target: stay
Cost: insert: 1, delete: 1, replace: 2

- Levenshtein distance



		0	1	2	3	4
		#	s	t	a	y
0	#	0	1	2	3	4
1	p	1	2	3	4	5
2	l	2	3	4	5	6
3	a	3	4	5	4	5
4	y	4	5	6	5	4

Minimum edit distance

Source: play → Target: stay
Cost: insert: 1, delete: 1, replace: 2

- Levenshtein distance
- Backtrace

		0	1	2	3	4
		#	s	t	a	y
0	#	0	1	2	3	4
1	p	1	2	3	4	5
2	l	2	3	4	5	6
3	a	3	4	5	4	5
4	y	4	5	6	5	4

Minimum edit distance

Source: play → Target: stay
Cost: insert: 1, delete: 1, replace: 2

- Levenshtein distance
- Backtrace
- Dynamic programming

		0	1	2	3	4
		#	s	t	a	y
0	#	0	1	2	3	4
1	p	1	2	3	4	5
2	l	2	3	4	5	6
3	a	3	4	5	4	5
4	y	4	5	6	5	4

Summary - learning objectives

- What is autocorrect ?
- Building the model
- Minimum edit distance
- Minimum edit distance algorithm

deah → dear ✅
yeah
dear
dean
... *etc*

	#	s	t	a	y
#	0	1	2	3	4
p	1	2	3	4	5
l	2	3	4	5	6
a	3	4	5	4	5
y	4	5	6	5	4