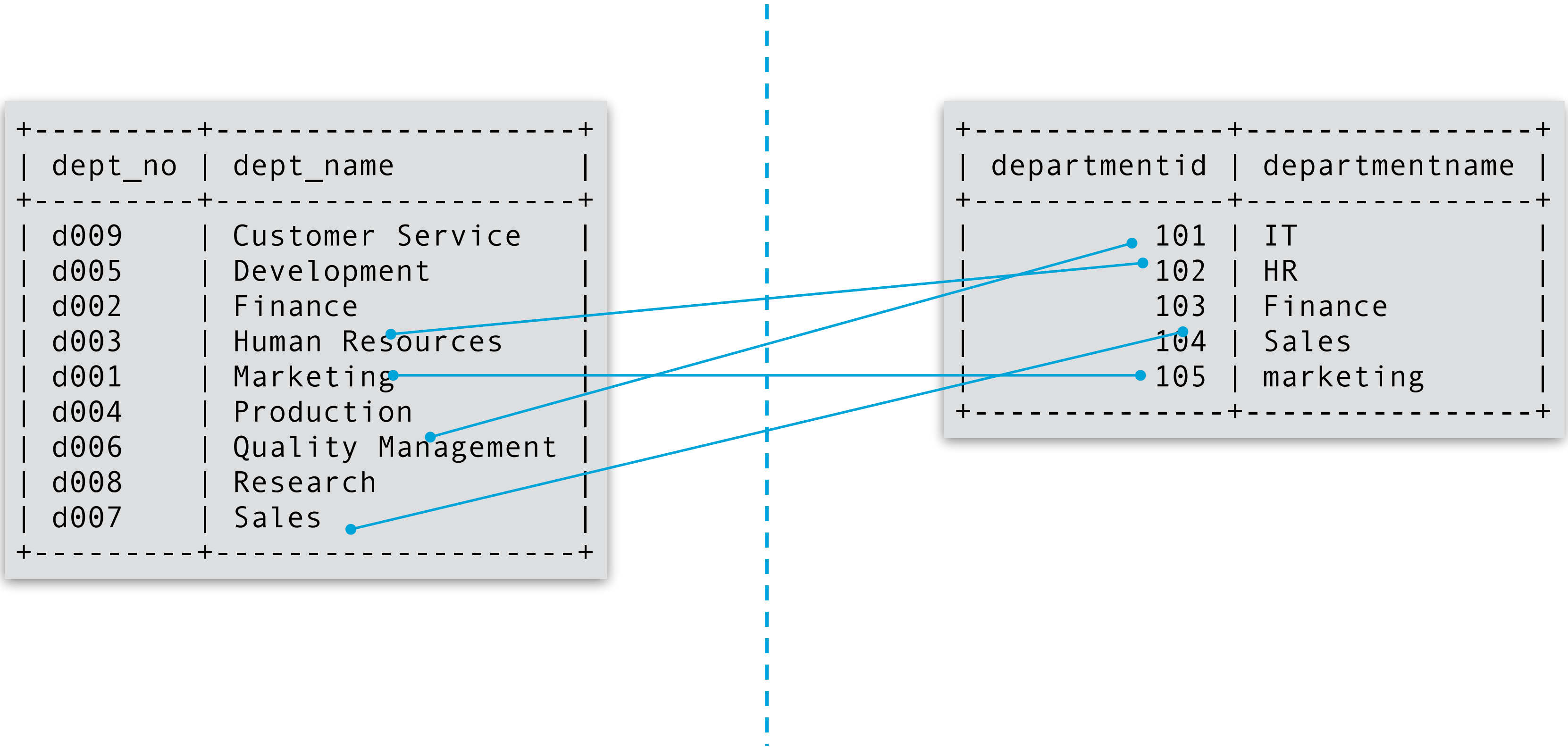
TÉCNICO LISBOA

Data Analysis and Integration

String matching

Introduction

- Comparing the two databases
  - departments

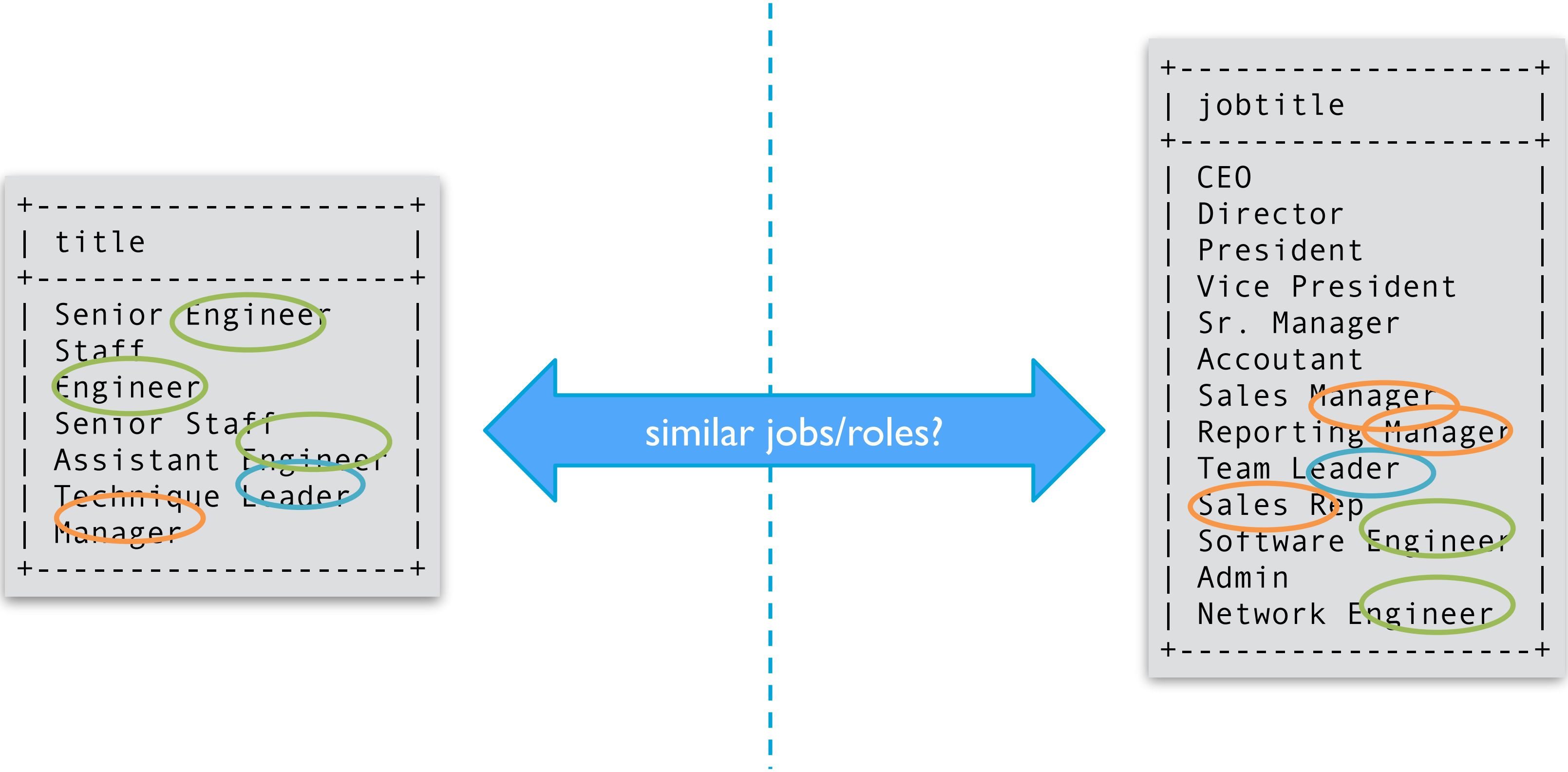


```
+---------+--------------------+
| dept_no | dept_name          |
+---------+--------------------+
| d009    | Customer Service   |
| d005    | Development        |
| d002    | Finance            |
| d003    | Human Resources    |
| d001    | Marketing          |
| d004    | Production         |
| d006    | Quality Management |
| d008    | Research           |
| d007    | Sales              |
+---------+--------------------+
```

```
+--------------+----------------+
| departmentid | departmentname |
+--------------+----------------+
|          101 | IT             |
|          102 | HR             |
|          103 | Finance        |
|          104 | Sales          |
|          105 | marketing      |
+--------------+----------------+
```

Introduction

- Comparing the two databases
  - job titles

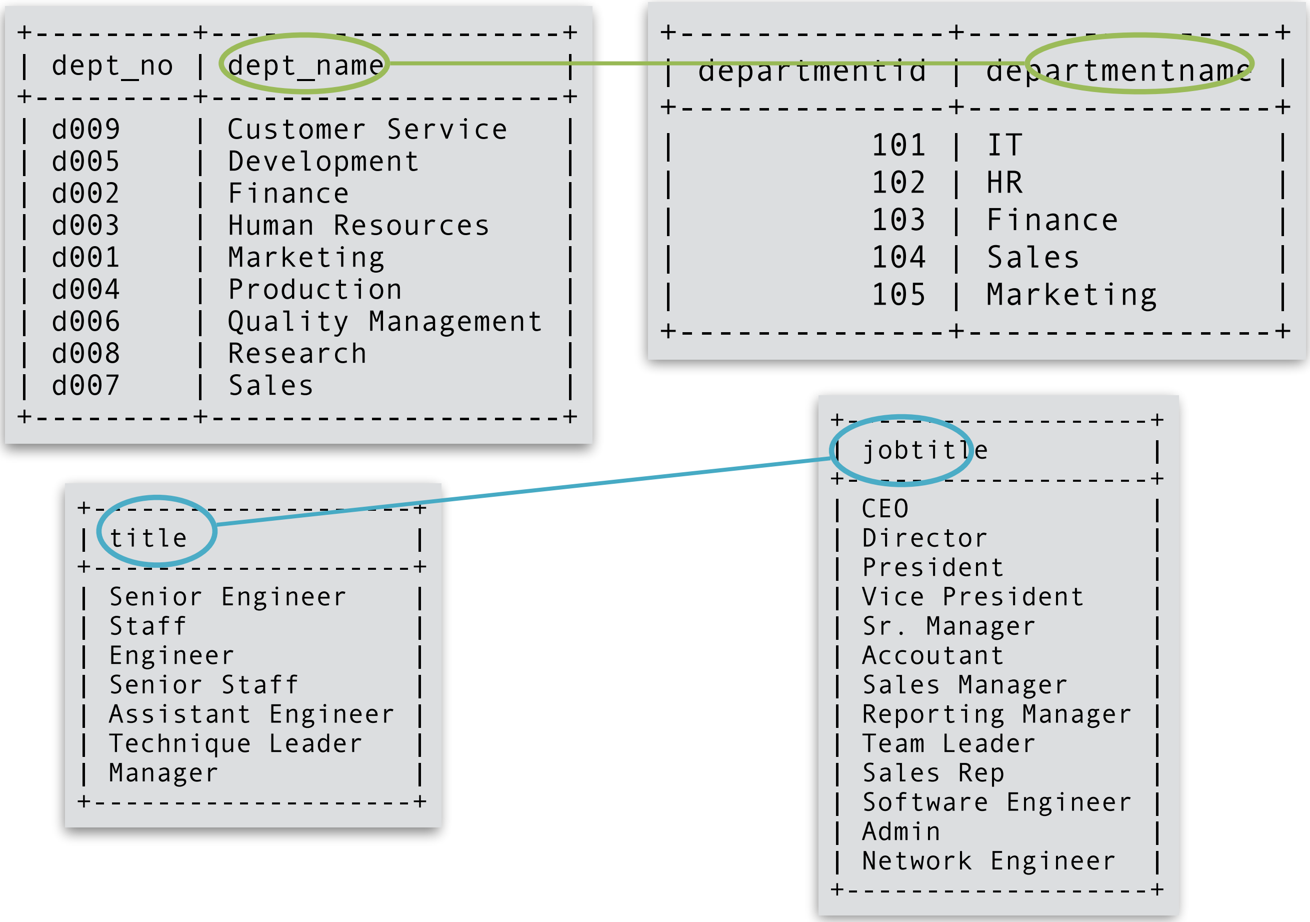


```
+--------------------+
| title              |
+--------------------+
| Senior Engineer    |
| Staff              |
| Engineer           |
| Senior Staff       |
| Assistant Engineer |
| Technique Leader   |
| Manager            |
+--------------------+
```

```
+--------------------+
| jobtitle           |
+--------------------+
| CEO                |
| Director           |
| President          |
| Vice President     |
| Sr. Manager        |
| Accountant         |
| Sales Manager      |
| Reporting Manager  |
| Team Leader        |
| Sales Rep          |
| Software Engineer  |
| Admin              |
| Network Engineer   |
+--------------------+
```

Introduction

- Applications of string matching in data integration
  - schema matching



```
+---------+--------------------+
| dept_no | dept_name          |
+---------+--------------------+
| d009    | Customer Service   |
| d005    | Development        |
| d002    | Finance            |
| d003    | Human Resources    |
| d001    | Marketing          |
| d004    | Production         |
| d006    | Quality Management |
| d008    | Research           |
| d007    | Sales              |
+---------+--------------------+
```

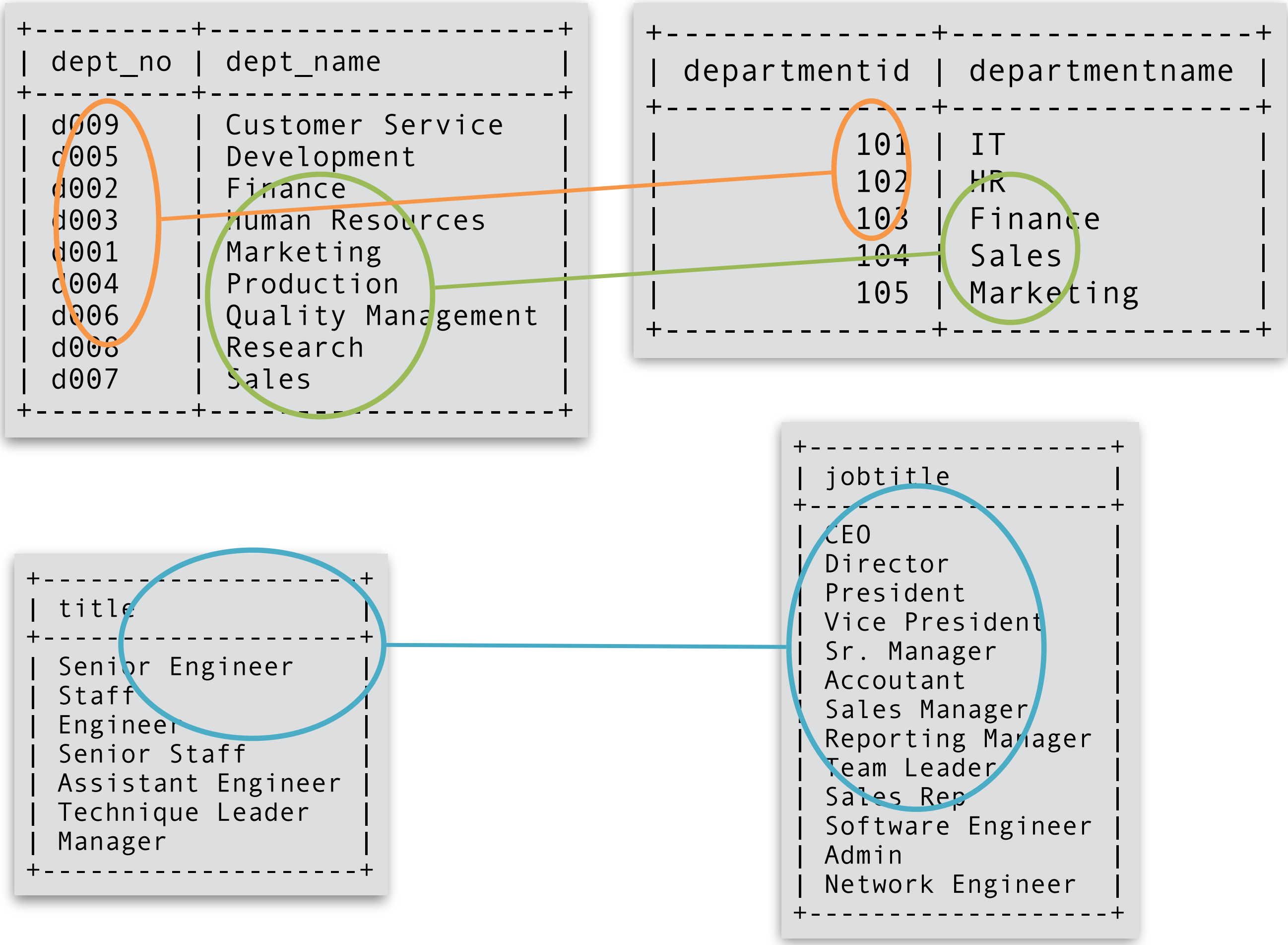
```
+--------------+----------------+
| departmentid | departmentname |
+--------------+----------------+
|          101 | IT             |
|          102 | HR             |
|          103 | Finance        |
|          104 | Sales          |
|          105 | Marketing      |
+--------------+----------------+
```

```
+--------------------+
| title              |
+--------------------+
| Senior Engineer    |
| Staff              |
| Engineer           |
| Senior Staff       |
| Assistant Engineer |
| Technique Leader   |
| Manager            |
+--------------------+
```

```
+-------------------+
| jobtitle          |
+-------------------+
| CEO               |
| Director          |
| President         |
| Vice President    |
| Sr. Manager       |
| Accoutant         |
| Sales Manager     |
| Reporting Manager |
| Team Leader       |
| Sales Rep         |
| Software Engineer |
| Admin             |
| Network Engineer  |
+-------------------+
```

Introduction

- Applications of string matching in data integration
  - schema matching via data matching



```
+----------+--------------------+
| dept_no  | dept_name          |
+----------+--------------------+
| d009     | Customer Service   |
| d005     | Development        |
| d002     | Finance            |
| d003     | Human Resources    |
| d001     | Marketing          |
| d004     | Production         |
| d006     | Quality Management |
| d008     | Research           |
| d007     | Sales              |
+----------+--------------------+
```

```
+--------------+----------------+
| departmentid | departmentname |
+--------------+----------------+
|          101 | IT             |
|          102 | HR             |
|          103 | Finance        |
|          104 | Sales          |
|          105 | Marketing      |
+--------------+----------------+
```

```
+--------------------+
| title              |
+--------------------+
| Senior Engineer    |
| Staff              |
| Engineer           |
| Senior Staff       |
| Assistant Engineer |
| Technique Leader   |
| Manager            |
+--------------------+
```

```
+------------------+
| jobtitle         |
+------------------+
| CEO              |
| Director         |
| President        |
| Vice President   |
| Sr. Manager      |
| Accoutant        |
| Sales Manager    |
| Reporting Manager|
| Team Leader      |
| Sales Rep        |
| Software Engineer|
| Admin            |
| Network Engineer |
+------------------+
```

String matching

- String matching is based on measures
  - measures of **distance** between strings
    - lower is better, higher is worse
  - measures of **similarity** between strings
    - higher is better, lower is worse

Outline

- String matching measures
  - sequence-based
  - set-based
  - phonetic

Sequence Based Metrics

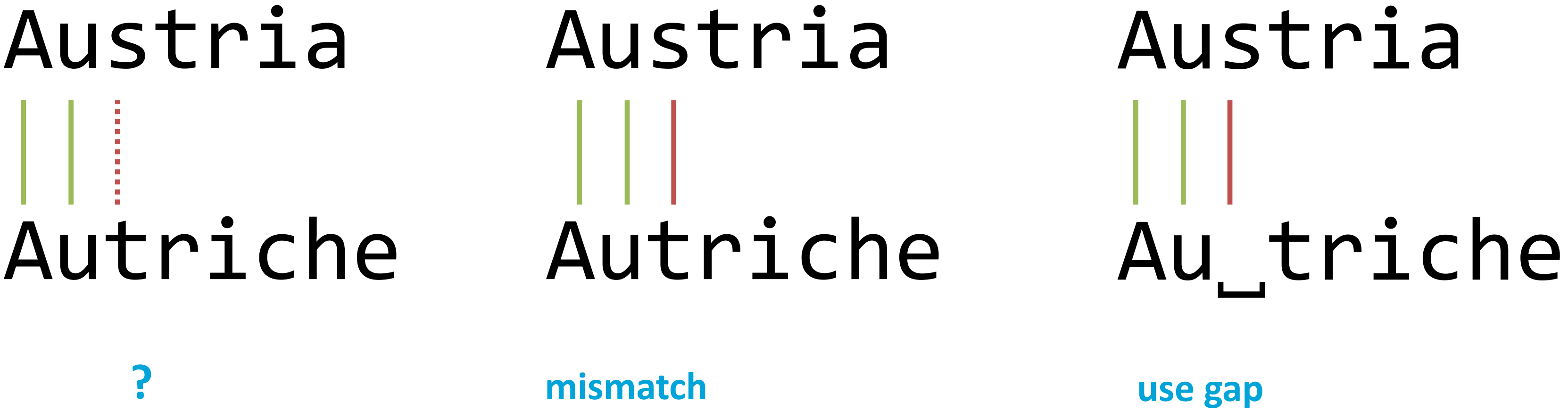
Sequence-based measures

- Edit distance (Levenshtein)
- Damerau-Levenshtein distance
- Needleman-Wunsch measure
- Jaro measure
- Jaro-Winkler measure

View the strings as sequences of characters

Edit distance (Levenshtein)

- Number of changes to transform one string into another
  - if the character matches: no change
  - if the character does not match: mismatch or use gap



Austria
Autriche

?

Austria
Autriche

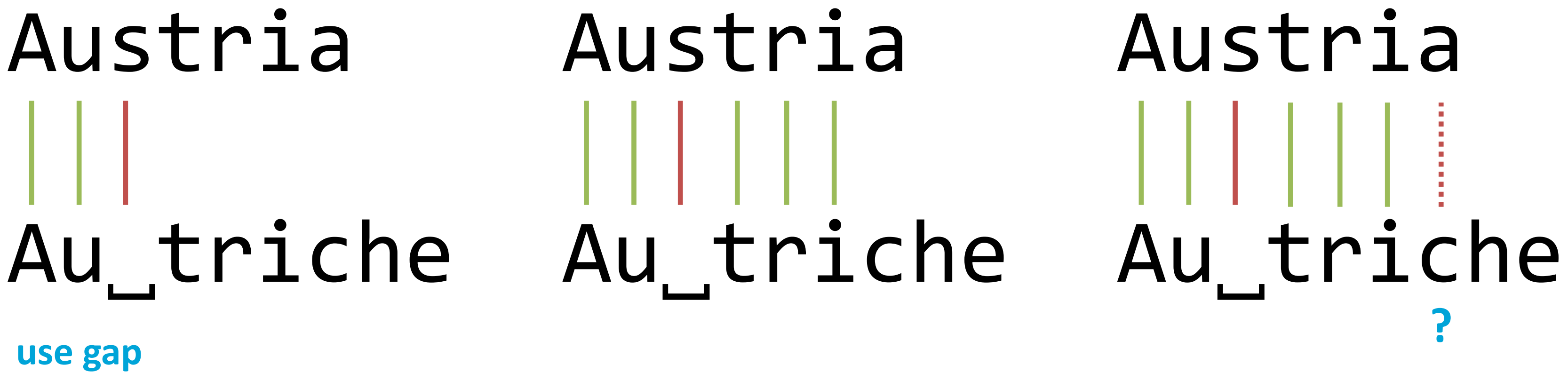
mismatch

Austria
Au␣triche

use gap

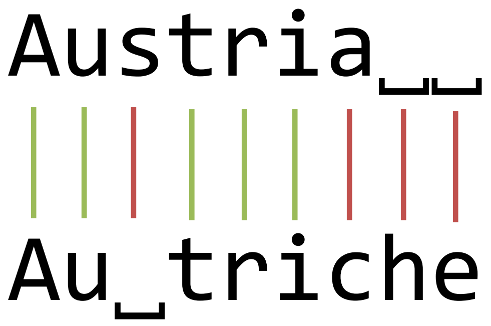
Edit distance (Levenshtein)

- Number of changes to transform one string into another
  - if the character matches: no change
  - if the character does not match: mismatch or use gap



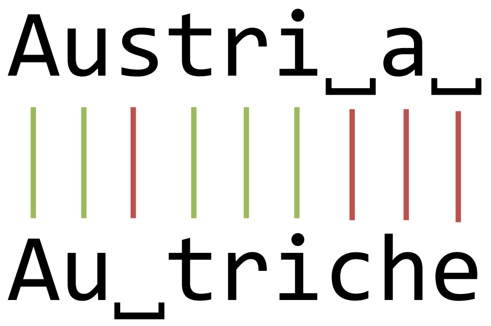
Edit distance (Levenshtein)

- Number of changes to transform one string into another
  - if the character matches: no change
  - if the character does not match: mismatch or use gap



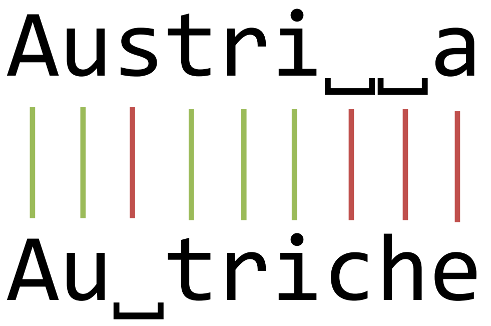
```
Austria␣␣
Au␣triche
```

4 changes



```
Austri␣a␣
Au␣triche
```

4 changes

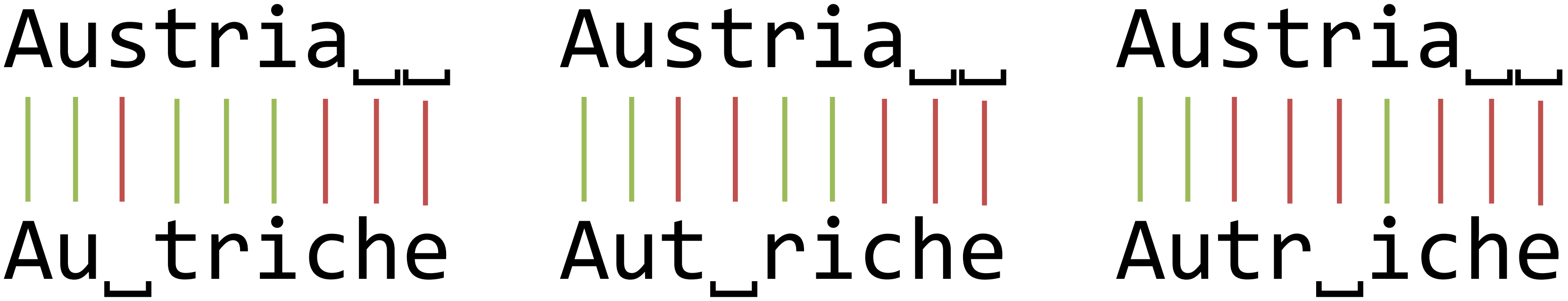


```
Austri␣␣a
Au␣triche
```

4 changes

Edit distance (Levenshtein)

- **Minimum** number of changes to transform one string into another
  - corresponds to an **optimal alignment**



```
Austria␣␣     Austria␣␣     Austria␣␣
Au␣triche     Aut␣riche     Autr␣iche
```

4 changes **5 changes** **6 changes**

Computing the edit distance

- Create a score matrix to find the best string alignment
  - edit distance is 4

		A	u	s	t	r	i	a
	0	1	2	3	4	5	6	7
A	1	0	1	2	3	4	5	6
u	2	1	0	1	2	3	4	5
t	3	2	1	1	1	2	3	4
r	4	3	2	2	2	1	2	3
i	5	4	3	3	3	2	1	2
c	6	5	4	4	4	3	2	2
h	7	6	5	5	5	4	3	3
e	8	7	6	6	6	5	4	4

Computing the edit distance

- Create a score matrix to find the best string alignment

		A	u	s	t	r	i	a
	0							
A								
u								
t								
r								
i								
c								
h								
e								

Computing the edit distance

- Create a score matrix to find the best string alignment

		A	u	s	t	r	i	a
	0	1	2	3	4	5	6	7
A								
u								
t								
r								
i								
c								
h								
e								

Austria…

␣␣␣␣␣␣␣…

Computing the edit distance

- Create a score matrix to find the best string alignment

Austria…

␣␣␣␣␣␣␣…

		A	u	s	t	r	i	a
	0	1	2	3	4	5	6	7
A	1							
u	2							
t	3							
r	4							
i	5							
c	6							
h	7							
e	8							

␣␣␣␣␣␣␣␣…

Autriche…

Computing the edit distance

- Create a score matrix to find the best string alignment
  - in each cell, choose minimum of 3 possibilities

		A	u	s	t	r	i	a
	0	1	2	3	4	5	6	7
A	1	0						
u	2							
t	3							
r	4							
i	5							
c	6							
h	7							
e	8							

A
A

(diagonally)
0 changes

A␣
␣A

(vertically)
2 changes

␣A
A␣

(horizontally)
2 changes

Computing the edit distance

- Create a score matrix to find the best string alignment
  - in each cell, choose minimum of 3 possibilities

		A	u	s	t	r	i	a
	0	1	2	3	4	5	6	7
A	1	0	1					
u	2							
t	3							
r	4							
i	5							
c	6							
h	7							
e	8							

Au
␣A

(diagonally)
2 changes

Au␣
␣␣A

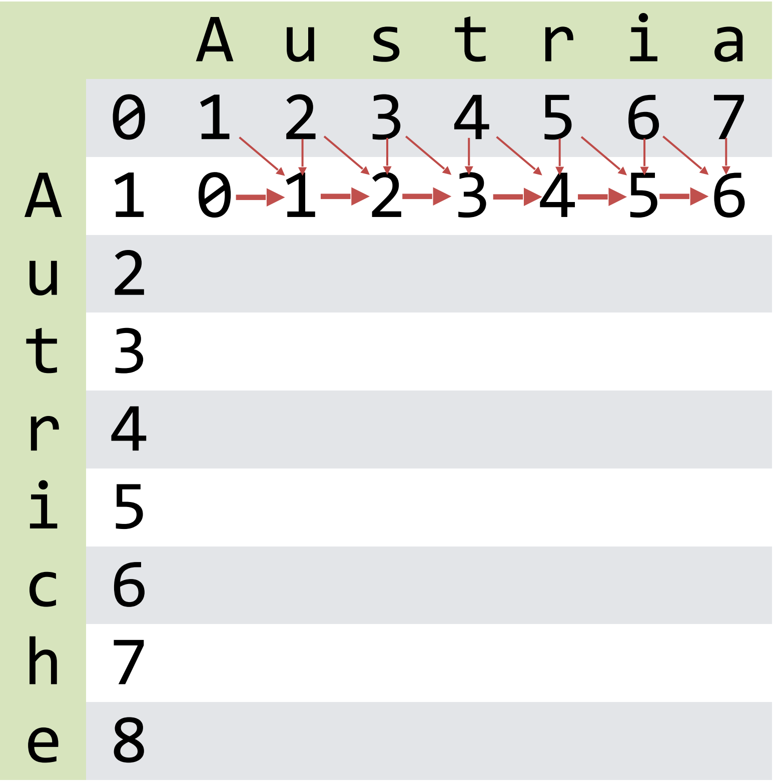
(vertically)
3 changes

Au
A␣

(horizontally)
1 change

Computing the edit distance

- Create a score matrix to find the best string alignment
  - in each cell, choose minimum of 3 possibilities

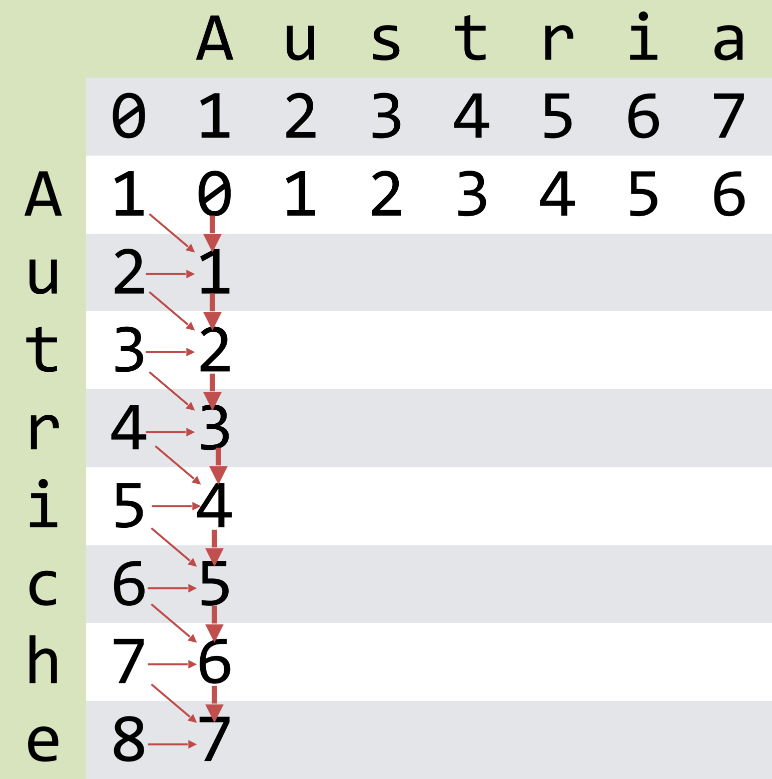


		A	u	s	t	r	i	a
	0	1	2	3	4	5	6	7
A	1	0	1	2	3	4	5	6
u	2							
t	3							
r	4							
i	5							
c	6							
h	7							
e	8							

```
Austria…
A␣␣␣␣␣␣…
```

Computing the edit distance

- Create a score matrix to find the best string alignment
  - in each cell, choose minimum of 3 possibilities



		A	u	s	t	r	i	a
	0	1	2	3	4	5	6	7
A	1	0	1	2	3	4	5	6
u	2	1						
t	3	2						
r	4	3						
i	5	4						
c	6	5						
h	7	6						
e	8	7						

```
Austria…
A␣␣␣␣␣␣…
```

```
A␣␣␣␣␣␣␣…
Autriche…
```

Computing the edit distance

- Create a score matrix to find the best string alignment
  - in each cell, choose minimum of 3 possibilities

		A	u	s	t	r	i	a
	0	1	2	3	4	5	6	7
A	1	0	1	2	3	4	5	6
u	2	1	0					
t	3	2						
r	4	3						
i	5	4						
c	6	5						
h	7	6						
e	8	7						

Au
Au

(diagonally)
0 changes

Au␣
A␣u

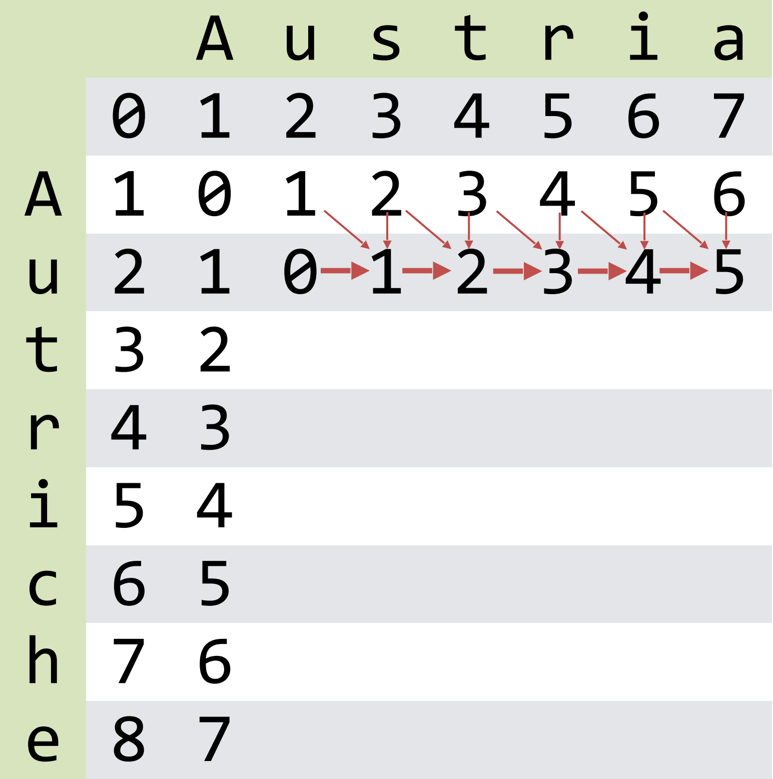
(vertically)
2 changes

A␣u
Au␣

(horizontally)
2 changes

Computing the edit distance

- Create a score matrix to find the best string alignment
  - in each cell, choose minimum of 3 possibilities



		A	u	s	t	r	i	a
	0	1	2	3	4	5	6	7
A	1	0	1	2	3	4	5	6
u	2	1	0	1	2	3	4	5
t	3	2						
r	4	3						
i	5	4						
c	6	5						
h	7	6						
e	8	7						

```
Austria…
Au␣␣␣␣␣…
```

Computing the edit distance

- Create a score matrix to find the best string alignment
  - in each cell, choose minimum of 3 possibilities

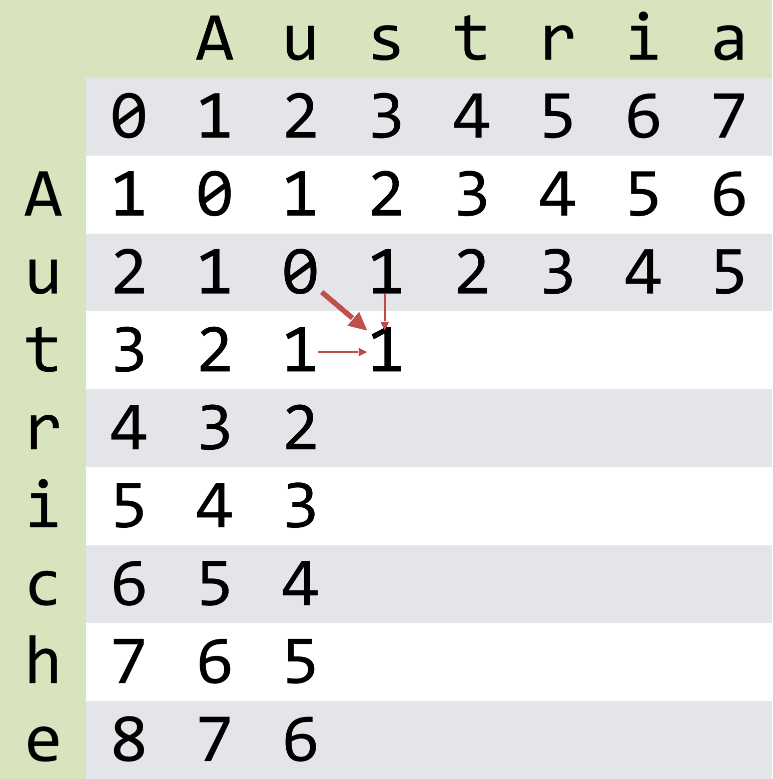
		A	u	s	t	r	i	a
	0	1	2	3	4	5	6	7
A	1	0	1	2	3	4	5	6
u	2	1	0	1	2	3	4	5
t	3	2	1					
r	4	3	2					
i	5	4	3					
c	6	5	4					
h	7	6	5					
e	8	7	6					

```
Austria…
Au␣␣␣␣␣…
```

```
Au␣␣␣␣␣␣…
Autriche…
```

Computing the edit distance

- Create a score matrix to find the best string alignment
  - in each cell, choose minimum of 3 possibilities



		A	u	s	t	r	i	a
	0	1	2	3	4	5	6	7
A	1	0	1	2	3	4	5	6
u	2	1	0	1	2	3	4	5
t	3	2	1	1				
r	4	3	2					
i	5	4	3					
c	6	5	4					
h	7	6	5					
e	8	7	6					

Aus
Aut

(diagonally)
1 change

Aus␣
Au␣t

(vertically)
2 changes

Au␣s
Aut␣

(horizontally)
2 changes

Computing the edit distance

- Create a score matrix to find the best string alignment
  - in each cell, choose minimum of 3 possibilities

		A	u	s	t	r	i	a
	0	1	2	3	4	5	6	7
A	1	0	1	2	3	4	5	6
u	2	1	0	1	2	3	4	5
t	3	2	1	1	1			
r	4	3	2					
i	5	4	3					
c	6	5	4					
h	7	6	5					
e	8	7	6					

Aust
Au␣t

(diagonally)
1 change

Aust␣
Au␣␣t

(vertically)
3 changes

Aust
Aut␣

(horizontally)
2 changes

Computing the edit distance

- Create a score matrix to find the best string alignment
  - in each cell, choose minimum of 3 possibilities

		A	u	s	t	r	i	a
	0	1	2	3	4	5	6	7
A	1	0	1	2	3	4	5	6
u	2	1	0	1	2	3	4	5
t	3	2	1	1	1	2	3	4
r	4	3	2					
i	5	4	3					
c	6	5	4					
h	7	6	5					
e	8	7	6					

```
Austria…
Au␣t␣␣␣…
```

Computing the edit distance

- Create a score matrix to find the best string alignment
  - in each cell, choose minimum of 3 possibilities

		A	u	s	t	r	i	a
	0	1	2	3	4	5	6	7
A	1	0	1	2	3	4	5	6
u	2	1	0	1	2	3	4	5
t	3	2	1	1	1	2	3	4
r	4	3	2	2				
i	5	4	3					
c	6	5	4					
h	7	6	5					
e	8	7	6					

```
Au␣s
Autr
```

(diagonally)
2 changes

```
Aus␣
Autr
```

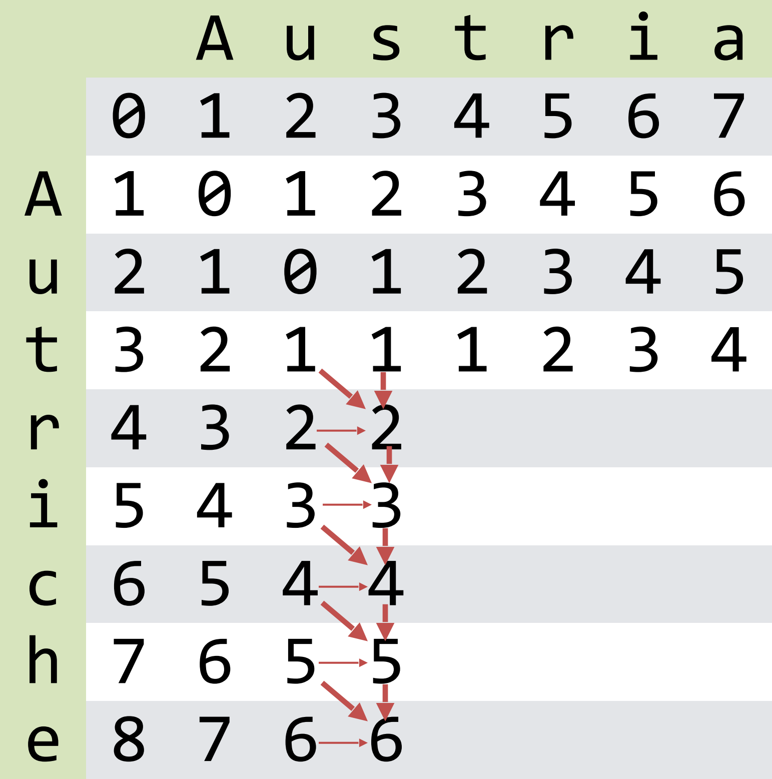
(vertically)
2 changes

```
Au␣␣s
Autr␣
```

(horizontally)
3 changes

Computing the edit distance

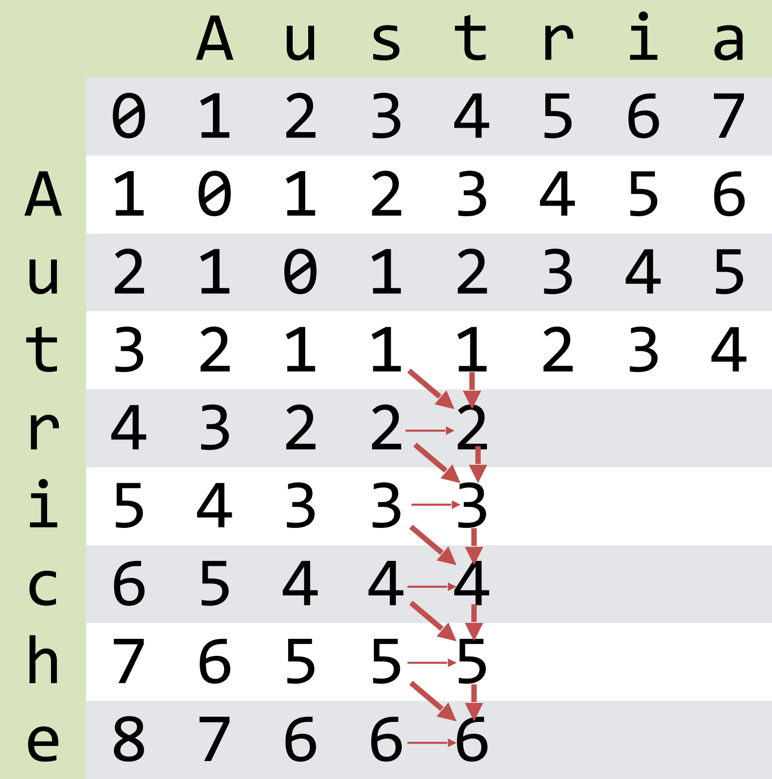
- Create a score matrix to find the best string alignment
  - in each cell, choose minimum of 3 possibilities



		A	u	s	t	r	i	a
	0	1	2	3	4	5	6	7
A	1	0	1	2	3	4	5	6
u	2	1	0	1	2	3	4	5
t	3	2	1	1	1	2	3	4
r	4	3	2	2				
i	5	4	3	3				
c	6	5	4	4				
h	7	6	5	5				
e	8	7	6	6				

Computing the edit distance

- Create a score matrix to find the best string alignment
  - in each cell, choose minimum of 3 possibilities



		A	u	s	t	r	i	a
	0	1	2	3	4	5	6	7
A	1	0	1	2	3	4	5	6
u	2	1	0	1	2	3	4	5
t	3	2	1	1	1	2	3	4
r	4	3	2	2	2			
i	5	4	3	3	3			
c	6	5	4	4	4			
h	7	6	5	5	5			
e	8	7	6	6	6			

Computing the edit distance

- Create a score matrix to find the best string alignment
  - in each cell, choose minimum of 3 possibilities

		A	u	s	t	r	i	a
	0	1	2	3	4	5	6	7
A	1	0	1	2	3	4	5	6
u	2	1	0	1	2	3	4	5
t	3	2	1	1	1	2	3	4
r	4	3	2	2	2	1		
i	5	4	3	3	3			
c	6	5	4	4	4			
h	7	6	5	5	5			
e	8	7	6	6	6			

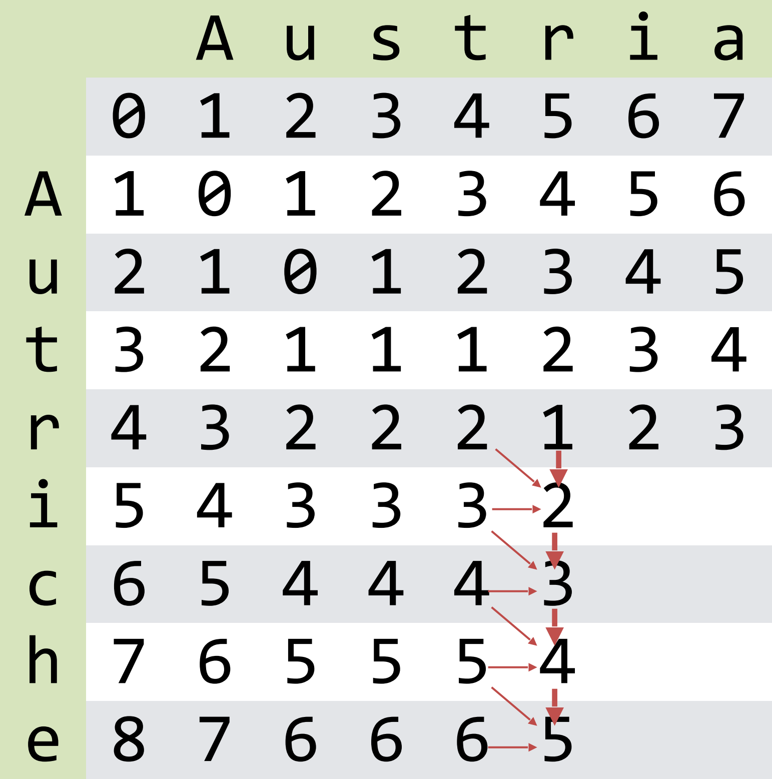
Computing the edit distance

- Create a score matrix to find the best string alignment
  - in each cell, choose minimum of 3 possibilities

		A	u	s	t	r	i	a
	0	1	2	3	4	5	6	7
A	1	0	1	2	3	4	5	6
u	2	1	0	1	2	3	4	5
t	3	2	1	1	1	2	3	4
r	4	3	2	2	2	1	2	3
i	5	4	3	3	3			
c	6	5	4	4	4			
h	7	6	5	5	5			
e	8	7	6	6	6			

Computing the edit distance

- Create a score matrix to find the best string alignment
  - in each cell, choose minimum of 3 possibilities



		A	u	s	t	r	i	a
	0	1	2	3	4	5	6	7
A	1	0	1	2	3	4	5	6
u	2	1	0	1	2	3	4	5
t	3	2	1	1	1	2	3	4
r	4	3	2	2	2	1	2	3
i	5	4	3	3	3	2		
c	6	5	4	4	4	3		
h	7	6	5	5	5	4		
e	8	7	6	6	6	5		

Computing the edit distance

- Create a score matrix to find the best string alignment
  - in each cell, choose minimum of 3 possibilities

		A	u	s	t	r	i	a
	0	1	2	3	4	5	6	7
A	1	0	1	2	3	4	5	6
u	2	1	0	1	2	3	4	5
t	3	2	1	1	1	2	3	4
r	4	3	2	2	2	1	2	3
i	5	4	3	3	3	2	1	
c	6	5	4	4	4	3		
h	7	6	5	5	5	4		
e	8	7	6	6	6	5		

Computing the edit distance

- Create a score matrix to find the best string alignment
  - in each cell, choose minimum of 3 possibilities

		A	u	s	t	r	i	a
	0	1	2	3	4	5	6	7
A	1	0	1	2	3	4	5	6
u	2	1	0	1	2	3	4	5
t	3	2	1	1	1	2	3	4
r	4	3	2	2	2	1	2	3
i	5	4	3	3	3	2	1	2
c	6	5	4	4	4	3		
h	7	6	5	5	5	4		
e	8	7	6	6	6	5		

Computing the edit distance

- Create a score matrix to find the best string alignment
  – in each cell, choose minimum of 3 possibilities

		A	u	s	t	r	i	a
	0	1	2	3	4	5	6	7
A	1	0	1	2	3	4	5	6
u	2	1	0	1	2	3	4	5
t	3	2	1	1	1	2	3	4
r	4	3	2	2	2	1	2	3
i	5	4	3	3	3	2	1	2
c	6	5	4	4	4	3	2	
h	7	6	5	5	5	4	3	
e	8	7	6	6	6	5	4	

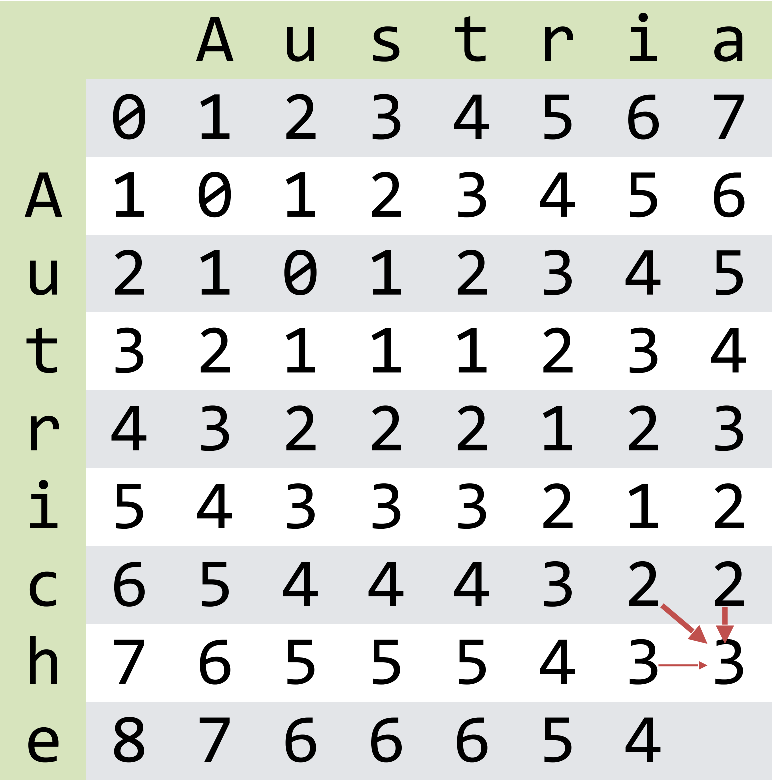
Computing the edit distance

- Create a score matrix to find the best string alignment
  - in each cell, choose minimum of 3 possibilities

		A	u	s	t	r	i	a
	0	1	2	3	4	5	6	7
A	1	0	1	2	3	4	5	6
u	2	1	0	1	2	3	4	5
t	3	2	1	1	1	2	3	4
r	4	3	2	2	2	1	2	3
i	5	4	3	3	3	2	1	2
c	6	5	4	4	4	3	2	2
h	7	6	5	5	5	4	3	
e	8	7	6	6	6	5	4	

Computing the edit distance

- Create a score matrix to find the best string alignment
  - in each cell, choose minimum of 3 possibilities



		A	u	s	t	r	i	a
	0	1	2	3	4	5	6	7
A	1	0	1	2	3	4	5	6
u	2	1	0	1	2	3	4	5
t	3	2	1	1	1	2	3	4
r	4	3	2	2	2	1	2	3
i	5	4	3	3	3	2	1	2
c	6	5	4	4	4	3	2	2
h	7	6	5	5	5	4	3	3
e	8	7	6	6	6	5	4	

Computing the edit distance

- Create a score matrix to find the best string alignment
  - in each cell, choose minimum of 3 possibilities

		A	u	s	t	r	i	a
	0	1	2	3	4	5	6	7
A	1	0	1	2	3	4	5	6
u	2	1	0	1	2	3	4	5
t	3	2	1	1	1	2	3	4
r	4	3	2	2	2	1	2	3
i	5	4	3	3	3	2	1	2
c	6	5	4	4	4	3	2	2
h	7	6	5	5	5	4	3	3
e	8	7	6	6	6	5	4	4

Computing the edit distance

- Create a score matrix to find the best string alignment
  – edit distance is 4

		A	u	s	t	r	i	a
	0	1	2	3	4	5	6	7
A	1	0	1	2	3	4	5	6
u	2	1	0	1	2	3	4	5
t	3	2	1	1	1	2	3	4
r	4	3	2	2	2	1	2	3
i	5	4	3	3	3	2	1	2
c	6	5	4	4	4	3	2	2
h	7	6	5	5	5	4	3	3
e	8	7	6	6	6	5	4	4

Computing the edit distance

- Create a score matrix to find the best string alignment
  - going backwards: there are 3 possible alignments

		A	u	s	t	r	i	a
	0	1	2	3	4	5	6	7
A	1	0	1	2	3	4	5	6
u	2	1	0	1	2	3	4	5
t	3	2	1	1	1	2	3	4
r	4	3	2	2	2	1	2	3
i	5	4	3	3	3	2	1	2
c	6	5	4	4	4	3	2	2
h	7	6	5	5	5	4	3	3
e	8	7	6	6	6	5	4	4

```
Austria␣␣
Au␣triche
```

Computing the edit distance

- Create a score matrix to find the best string alignment
  - going backwards: there are 3 possible alignments

		A	u	s	t	r	i	a
	0	1	2	3	4	5	6	7
A	1	0	1	2	3	4	5	6
u	2	1	0	1	2	3	4	5
t	3	2	1	1	1	2	3	4
r	4	3	2	2	2	1	2	3
i	5	4	3	3	3	2	1	2
c	6	5	4	4	4	3	2	2
h	7	6	5	5	5	4	3	3
e	8	7	6	6	6	5	4	4

```
Austria␣␣
Au␣triche

Austri␣a␣
Au␣triche
```

Computing the edit distance

- Create a score matrix to find the best string alignment
  - going backwards: there are 3 possible alignments

		A	u	s	t	r	i	a
	0	1	2	3	4	5	6	7
A	1	0	1	2	3	4	5	6
u	2	1	0	1	2	3	4	5
t	3	2	1	1	1	2	3	4
r	4	3	2	2	2	1	2	3
i	5	4	3	3	3	2	1	2
c	6	5	4	4	4	3	2	2
h	7	6	5	5	5	4	3	3
e	8	7	6	6	6	5	4	4

```
Austria␣␣
Au␣triche

Austri␣a␣
Au␣triche

Austri␣␣a
Au␣triche
```

Computing the edit distance

- Recurrence equation
  - for two strings $\boldsymbol{x} = x_1 x_2 \cdots x_n$ and $\boldsymbol{y} = y_1 y_2 \cdots y_m$

$$
d(i,j) = min \begin{cases} \begin{cases} d(i-1,j-1) & if \quad x_i = y_j \\ d(i-1,j-1)+1 & if \quad x_i \neq y_j \end{cases} & \text{(diagonally)} \\ d(i-1,j)+1 & \text{(vertically)} \\ d(i,j-1)+1 & \text{(horizontally)} \end{cases}
$$

Computing the edit distance

- Recurrence equation
  - for two strings $\boldsymbol{x} = x_1 x_2 \cdots x_n$ and $\boldsymbol{y} = y_1 y_2 \cdots y_m$

mismatch costs 1

$$d(i,j) = min\begin{cases} \begin{cases} d(i-1,j-1) & if \quad x_i = y_j \\ d(i-1,j-1)+1 & if \quad x_i \neq y_j \end{cases} \\ d(i-1,j)+1 \\ d(i,j-1)+1 \end{cases}$$

using a gap costs 1

Damerau-Levenshtein distance

- Similar to edit distance, but considers another type of change
    - mismatch
    - use gap
    - transposition between two adjacent characters

In the Levenshtein distance this would be 2 changes instead of 1

Damerau-Levenshtein distance

- Examples

	Austria Autriche	Ireland Ierland	Dinamarca Danimarca	Chipre Cypern
Levenshtein	4	2	2	4
Damerau-Levenshtein	4	1	2	4

Needleman-Wunsch measure

- Similar to edit distance but:
  - mismatch and gap have negative scores
  - match has zero or positive score
- Needleman-Wunsch is a similarity measure
  - the higher the score, the better
- Needleman-Wunsch is flexible
  - scores for match, mismatch and gap can be adjusted

Needleman-Wunsch measure

- Example

mismatch = -1 gap = -1 match = 0

		A	u	s	t	r	i	a
	0	-1	-2	-3	-4	-5	-6	-7
A	-1	0	-1	-2	-3	-4	-5	-6
u	-2	-1	0	-1	-2	-3	-4	-5
t	-3	-2	-1	-1	-1	-2	-3	-4
r	-4	-3	-2	-2	-2	-1	-2	-3
i	-5	-4	-3	-3	-3	-2	-1	-2
c	-6	-5	-4	-4	-4	-3	-2	-2
h	-7	-6	-5	-5	-5	-4	-3	-3
e	-8	-7	-6	-6	-6	-5	-4	-4

mismatch = -1 gap = -1 match = 2

		A	u	s	t	r	i	a
	0	-1	-2	-3	-4	-5	-6	-7
A	-1	2	1	0	-1	-2	-3	-4
u	-2	1	4	3	2	1	0	-1
t	-3	0	3	3	5	4	3	2
r	-4	-1	2	2	4	7	6	5
i	-5	-2	1	1	3	6	9	8
c	-6	-3	0	0	2	5	8	8
h	-7	-4	-1	-1	1	4	7	7
e	-8	-5	-2	-2	0	3	6	6

Needleman-Wunsch measure

- Scores can also be configured on a per-letter basis
  - e.g. match A-A with higher score than T-T
  - e.g. mismatch A-C with lower score than A-E
- Such scores can be configured in a **score matrix**

Needleman-Wunsch measure

- Example of score matrix
  - match between vowels (+2) and between consonants (+1)
  - mismatch between vowels (-2) and between consonants (-1)

	a	b	c	d	e	f	…
a	+2	-2	-2	-2	-1	-2	…
b	-2	+1	-1	-1	-2	-1	…
c	-2	-1	+1	-1	-2	-1	…
d	-2	-1	-1	+1	-2	-1	…
e	-1	-2	-2	-2	+2	-2	…
f	-2	-1	-1	-1	-2	+1	…
…	…	…	…	…	…	…	…

Needleman-Wunsch measure

- Recurrence equation
  - for two strings $\boldsymbol{x} = x_1 x_2 \cdots x_n$ and $\boldsymbol{y} = y_1 y_2 \cdots y_m$

from score matrix

$$s\left(i, j\right) = max \begin{cases} s\left(i-1, j-1\right) + c(x_i, y_j) \\ s\left(i-1, j\right) + c_g \\ s\left(i, j-1\right) + c_g \end{cases}$$

note: max

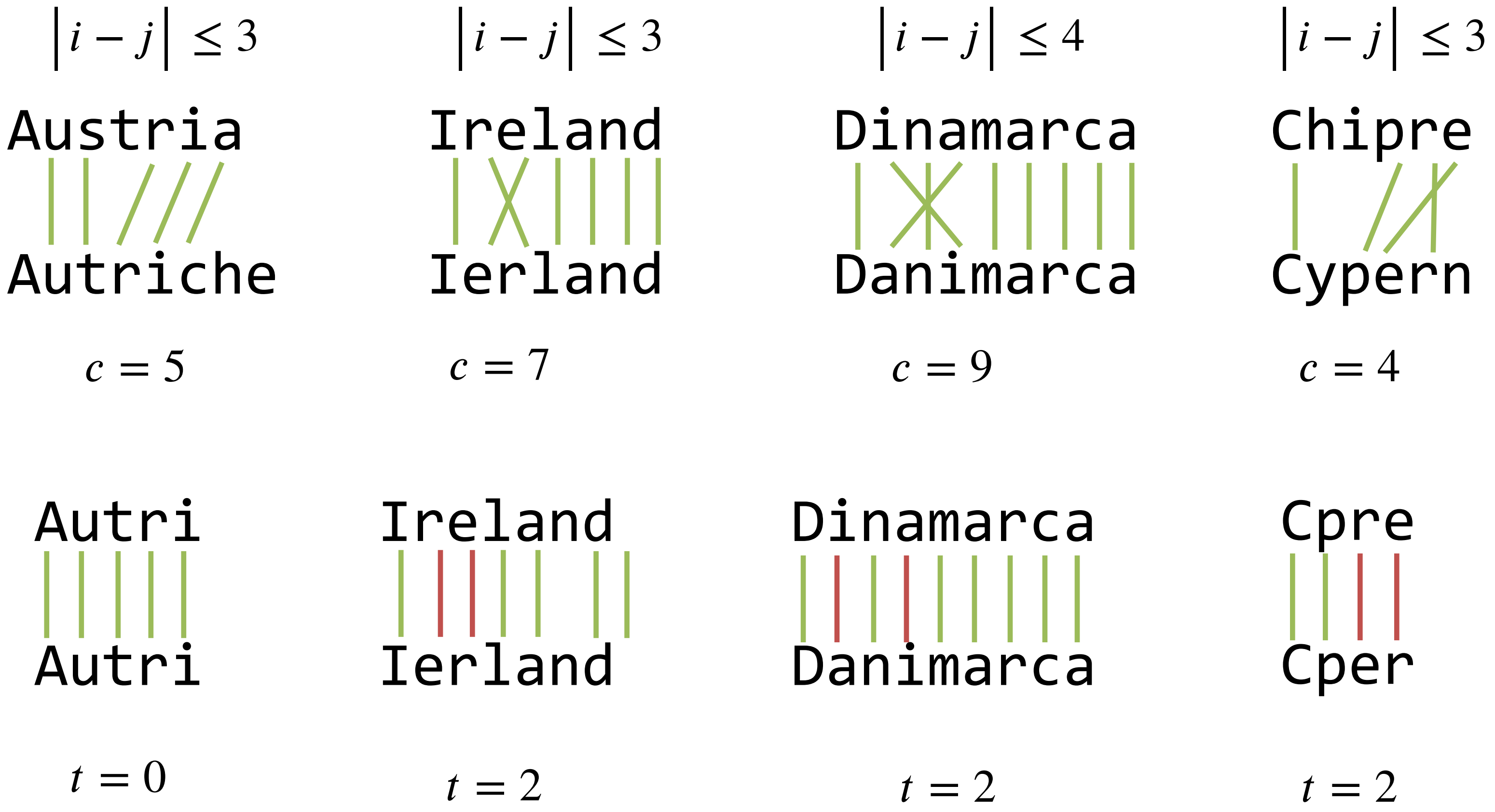
gap score (negative)

Jaro measure

- Used to compare short strings, such as first/last names
- Main focus on **common characters** and **transpositions**
  - **common character** means $x_i = y_j$ and
    $$\left|i - j\right| \leq \frac{min\left\{ |x|, \left|y\right| \right\}}{2}$$
    (i.e. x_i and y_j must be equal and not too distant)
  - compare sequence of common chars
    (should be equal unless there are transpositions)
  - each mismatch counts as one **transposition**

Jaro measure

- Examples



$|i-j| \le 3$ $|i-j| \le 3$ $|i-j| \le 4$ $|i-j| \le 3$

Austria Ireland Dinamarca Chipre

Autriche Ierland Danimarca Cypern

$c = 5$ $c = 7$ $c = 9$ $c = 4$

Autri Ireland Dinamarca Cpre

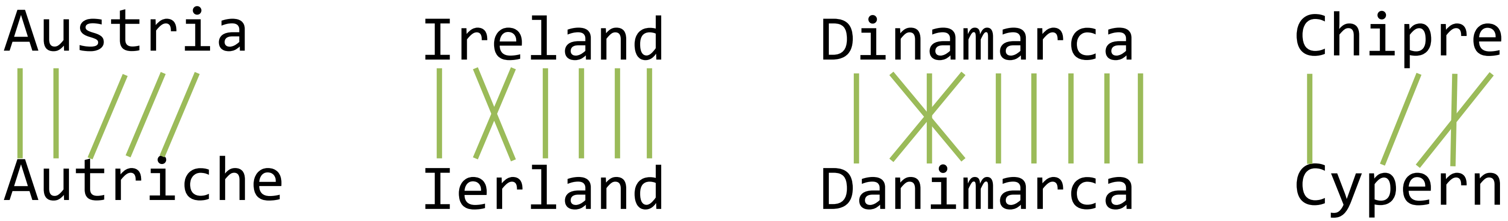
Autri Ierland Danimarca Cper

$t = 0$ $t = 2$ $t = 2$ $t = 2$

Jaro measure

- Formula

$$jaro(x,y) = \frac{1}{3}\left(\frac{c}{|x|} + \frac{c}{|y|} + \frac{c - \frac{t}{2}}{c}\right)$$



Austria	Ireland	Dinamarca	Chipre
Autriche	Ierland	Danimarca	Cypern
$c = 5$	$c = 7$	$c = 9$	$c = 4$
$t = 0$	$t = 2$	$t = 2$	$t = 2$
$jaro \approx 0.78$	$jaro \approx 0.95$	$jaro \approx 0.96$	$jaro \approx 0.69$

Jaro-Winkler measure

- A variant of Jaro for strings with a common prefix

```
Österreich
Österrike
```

$\underbrace{\texttt{Öster}}_{\text{prefix}}$

  - let PL be the prefix length
  - let PW be the prefix weight (typically 0.1)

$$jarowinkler(x, y) = (1 - PL * PW) * jaro(x, y) + PL * PW$$

Jaro-Winkler measure

- Examples

	Austria Autriche	Ireland Ierland	Dinamarca Danimarca	Chipre Cypern	Österreich Österrike
Jaro	0.78	0.95	0.96	0.69	0.85
Jaro-Winkler	0.82	0.96	0.97	0.72	0.91

Set Based Metrics

Jaccard measure

- This is a **set-based** measure based on ***n*-grams** (substrings with length n)
  - most common are **2-grams** or **bigrams**

```
Austria {#A, Au, us, st, tr, ri, ia, a#}

Autriche  {#A, Au, ut, tr, ri, ic, ch, he, e#}
```

In the two sets of bigrams above, there are bigrams in common

Jaccard measure

- Let B_x be the set of bigrams of string x
- Let B_y be the set of bigrams of string y

$$\boxed{jaccard(x, y) = \frac{\left|B_x \cap B_y\right|}{\left|B_x \cup B_y\right|}}$$

number of common bigrams
(no duplicates)

number of all bigrams
(no duplicates)

- set-based metric, and sets have no duplicates

Jaccard measure

- Example

Austria {#A, Au, us, st, tr, ri, ia, a#}

Autriche {#A, Au, ut, tr, ri, ic, ch, he, e#}

$$jaccard(x, y) = \frac{4}{13} \approx 0.31$$

Jaccard measure

- Example

```
Ireland   {#I, Ir, re, el, la, an, nd, d#}

Ierland   {#I, Ie, er, rl, la, an, nd, d#}
```

$$jaccard\left(x, y\right) = \frac{5}{11} \approx 0.45$$

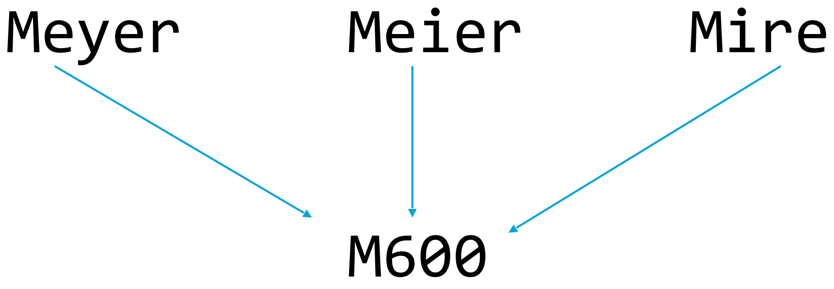
Phonetic Metrics

Phonetic similarity measures

- Match strings based on their sound
  - Soundex
  - Refined Soundex
  - etc.

Soundex measure

- This is a **phonetic** measure, used primarily to match surnames
  - Written in a different way, but sounding very similar
  - E.g. Meyer, Meier, Mire; Smith, Smithe, Smythe



Meyer

Meier

Mire

M600

Soundex measure

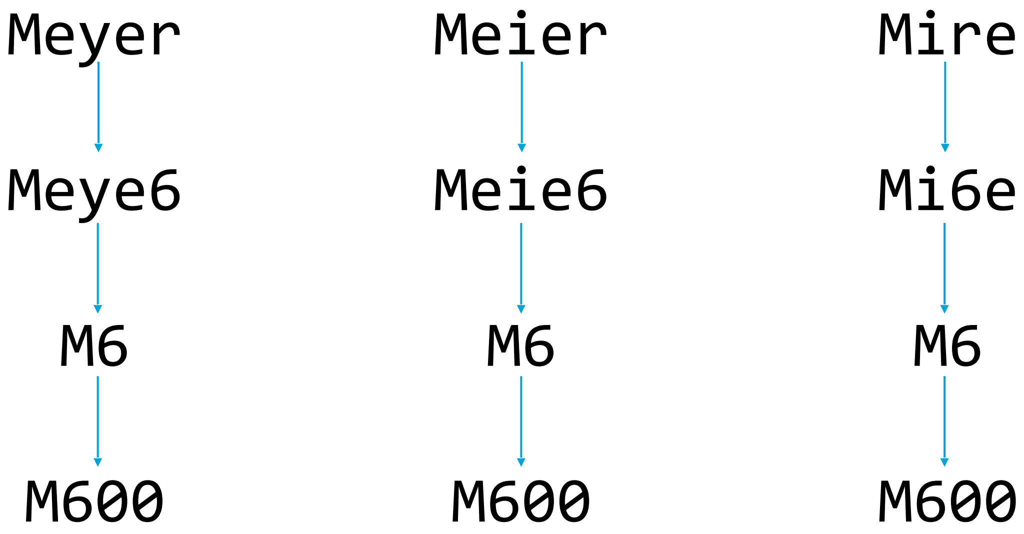
- The Soundex phonetic rules
  1. Keep the first letter
  2. Remove all occurrences of H and W
  3. Replace each letter with a digit (table below)
  4. Collapse any sequence of identical digits
  5. Drop all non-digit letters (except first one)
  6. Keep only four characters (pad with zero if needed)

B F P V	1
C G J K Q S X Z	2
D T	3
L	4
M N	5
R	6

note: not every letter is included

Soundex measure

- Examples



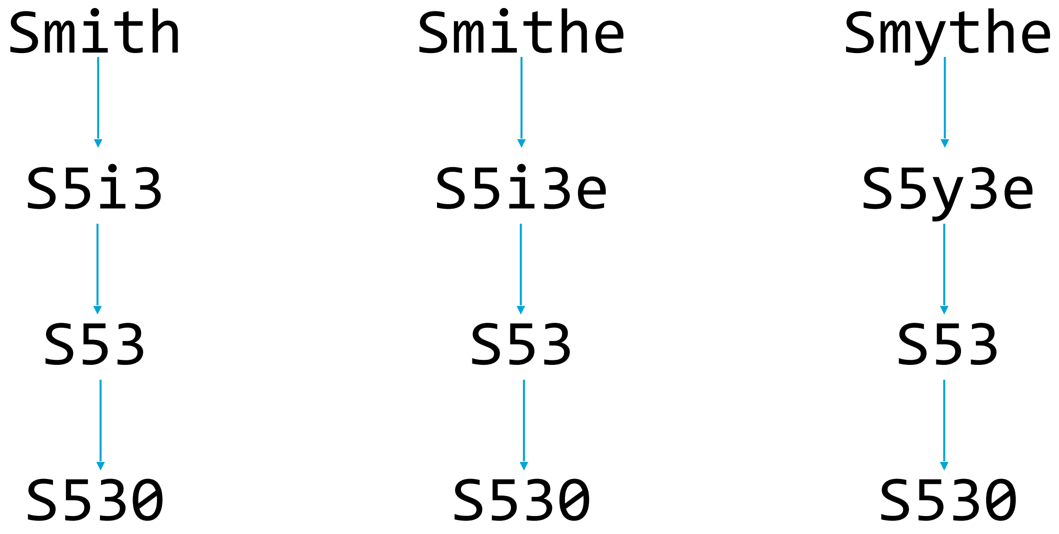
Meyer
Meye6
M6
M600

Meier
Meie6
M6
M600

Mire
Mi6e
M6
M600

Soundex measure

- Examples



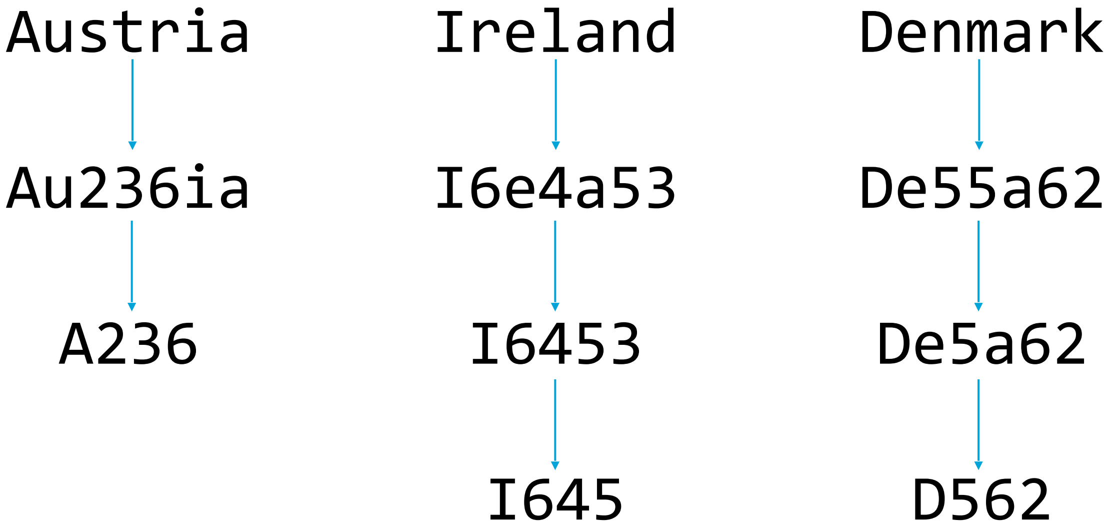
Smith → S5i3 → S53 → S530

Smithe → S5i3e → S53 → S530

Smythe → S5y3e → S53 → S530

Soundex measure

- Examples



Austria → Au236ia → A236

Ireland → I6e4a53 → I6453 → I645

Denmark → De55a62 → De5a62 → D562

Soundex measure

- Properties
  - It is tuned to a particular language (American English)
    - variants have been developed for other languages
  - Many false positives
    - Christopher (C623) vs. Christine (C623)
    - Ackermann (A265) vs. Azuron (A265)
  - Some false negatives
    - Christian (C623) vs. Kristian (K623)
    - Shultz (S432) vs. Shulz (S420)

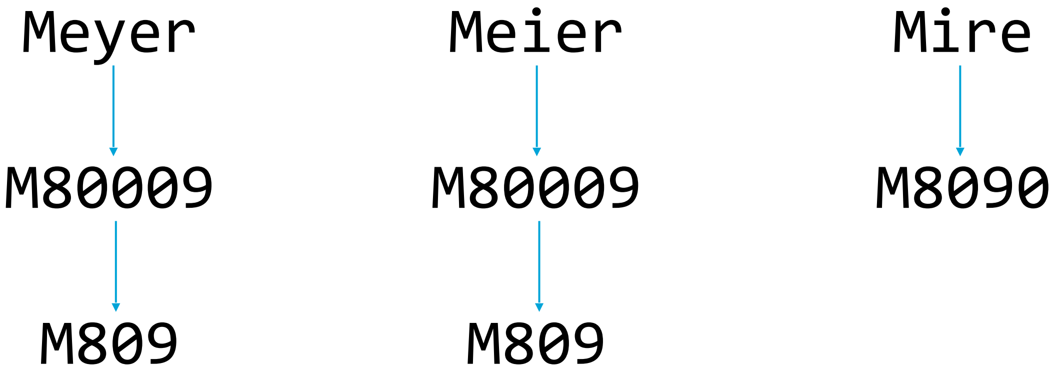
Refined Soundex measure

- Uses a different table (more groups)
- Has a group for vowels + H + W + Y
- No truncation to 4 characters
- First letter is kept and encoded as well

A E H I O U W Y	0
B P	1
F V	2
C K S	3
G J	4
Q X Z	5
D T	6
L	7
M N	8
R	9

Refined Soundex measure

- Examples



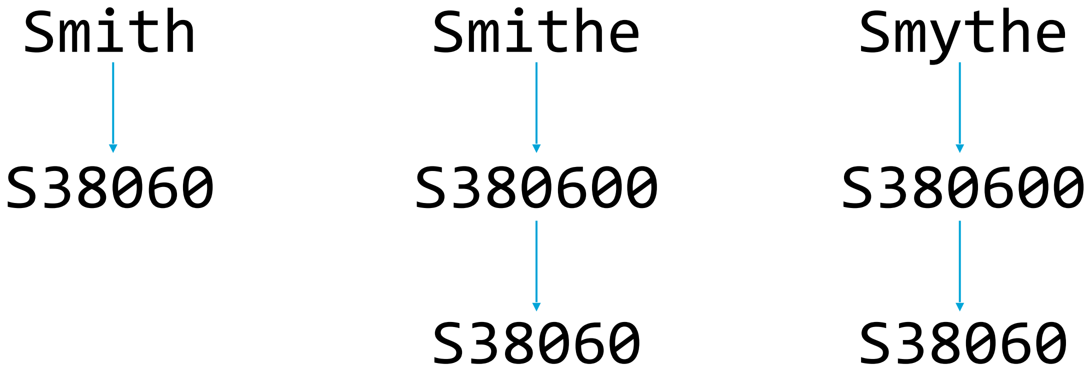
Meyer → M80009 → M809

Meier → M80009 → M809

Mire → M8090

Refined Soundex measure

- Examples



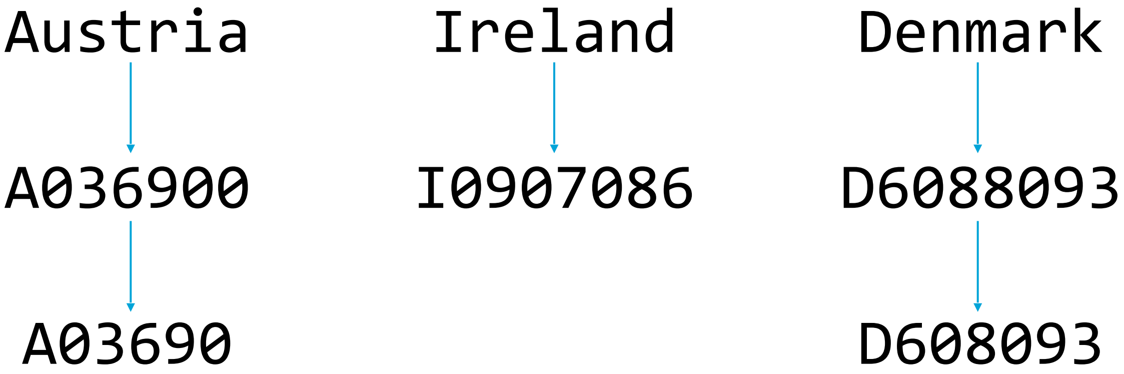
Smith → S38060

Smithe → S380600 → S38060

Smythe → S380600 → S38060

Refined Soundex measure

- Examples



Austria

A036900

A03690

Ireland

I0907086

Denmark

D6088093

D608093

Refined Soundex measure

- Properties
  - Less false positives
    - Christopher (C3090360109) vs. Christine (C309036080)
    - Ackermann (A0309808) vs. Azuron (A050908)

- Other variants and improvements
  - Algorithms with more complex rules
    - Metaphone, Double Metaphone, etc.