Similarity Search in Large Databases

Course Organisation and Demo

Nikolaus Augsten

nikolaus.augsten@plus.ac.at
Department of Computer Science
University of Salzburg

database
research group

https://dbresearch.uni-salzburg.at

WS 2026/27

Version September 14, 2026

A Problem at Our Municipality of Bozen

- Given:
  - realty owners DB (name and address of the realty)
  - residents DB (name and residential address)
  - both DBs cover the same geographic area (the city of Bozen)

Owners (dataset A)

Peter	Gilmstrasse 1
Arturas	Gilmstrasse 3
Linas	Marieng. 1/A
Markus	Cimitero 4
Michael	Gilmstrasse 5
Igor	Friedensplatz 2/A/1
Andrej	Friedensplatz 3
Francesco	Untervigil 1
Johann	Cimitero 6/B
Igor	Friedensplatz 2/A/2
Nikolaus	Cimitero 6/A

Residents (dataset B)

Rosa	Siegesplatz 3/-/3
Dario	Friedhofplatz 4
Romans	Untervigli 1
Adriano	Mariengasse 1
Maria	Siegesplatz 3/-/2
Arturas	Hermann-von-Gilm-Str. 3/A
Peter	Hermann-von-Gilm-Str. 1
Markus	Siegesplatz 2/A
Juozas	Hermann-von-Gilm-Str. 3/B
Andrej	Siegesplatz 3/-/1
Luigi	Friedhofplatz 6
Anita	Herman-von-Gilm-Str. 6

- Query: Give me owner and resident for each apartment in Bozen!

Assumptions for the Solutions in this Course

- Large data volumes
  - cannot be done by hand
  - solution must be efficient
- Data-driven, not process-driven
  - Sometimes it is better to change the world, e.g., force people to adhere to coding conventions, instead of fixing the errors later.
  - We cannot change the world.
- No domain-specific solution (e.g., address standardization)
- No training phase (e.g., supervised learning)
- No expensive configuration (e.g., define dictionaries, rules)
- Tuning parameters (like weights) are OK

The Objects

- Many objects can be represented as sets:
  - text document as the set of its words
  - social network user as the set of group memberships or friends
  - sales as the set of product categories
  - user interaction as the set of visited links
- Strings are everywhere:
  - deduplicate product names retrieved by a wrapper
  - integration of customer records
  - cluster medical records by free text description
- Hierarchical data are represented as trees:
  - JSON or XML data
  - abstract syntax trees for code analysis
  - RNA secondary structures in biology

Outline

1. Course Organisation
2. Demo: Similarity Join on Residential Addresses
3. Conclusion

Outline

1 Course Organisation

2 Demo: Similarity Join on Residential Addresses

3 Conclusion

All Information about Lecture and Lab

https://dbresearch.uni-salzburg.at/teaching/ssdb/

Outline

1 Course Organisation

2 Demo: Similarity Join on Residential Addresses

3 Conclusion

Back to Our Initial Example

- Given:
  - realty owners DB (name and address of the realty)
  - residents DB (name and residential address)
  - both DBs cover the same geographic area (the city of Bozen/Italy)

Owners (dataset A)

Peter	Gilmstrasse 1
Arturas	Gilmstrasse 3
Linas	Marieng. 1/A
Markus	Cimitero 4
Michael	Gilmstrasse 5
Igor	Friedensplatz 2/A/1
Andrej	Friedensplatz 3
Francesco	Untervigil 1
Johann	Cimitero 6/B
Igor	Friedensplatz 2/A/2
Nikolaus	Cimitero 6/A

Residents (dataset B)

Rosa	Siegesplatz 3/-/3
Dario	Friedhofplatz 4
Romans	Untervigli 1
Adriano	Mariengasse 1
Maria	Siegesplatz 3/-/2
Arturas	Hermann-von-Gilm-Str. 3/A
Peter	Hermann-von-Gilm-Str. 1
Markus	Siegesplatz 2/A
Juozas	Hermann-von-Gilm-Str. 3/B
Andrej	Siegesplatz 3/-/1
Luigi	Friedhofplatz 6
Anita	Herman-von-Gilm-Str. 6

- Give me owner and resident for each apartment in Bozen!

Database Representation

Owners

A

strID	*name*	*num*	*entr*	*apt*
α_1	Gilmstrasse	1		
α_1	Gilmstrasse	3		
α_1	Gilmstrasse	5		
α_2	Friedensplatz	2	A	1
α_2	Friedensplatz	2	A	2
α_2	Friedensplatz	3		
α_3	Cimitero	4		
α_3	Cimitero	6	A	
α_3	Cimitero	6	B	
α_4	Untervigil	1		
α_5	Marieng.	1	A	

Residents

B

strID	*name*	*num*	*entr*	*apt*
β_2	Hermann-von-Gilm-Str.	1		
β_2	Hermann-von-Gilm-Str.	3	A	
β_2	Hermann-von-Gilm-Str.	3	B	
β_2	Hermann-von-Gilm-Str.	6		
β_3	Siegesplatz	2	A	
β_3	Siegesplatz	3	-	1
β_3	Siegesplatz	3	-	2
β_3	Siegesplatz	3	-	3
β_1	Friedhofplatz	4		
β_1	Friedhofplatz	6		
β_5	Untervigli	1		
β_4	Mariengasse	1		

String Similarity

- **Observation 1:** Some street names are similar.

dataset A	dataset B
`Gilmstrasse`	`Friedhofplatz`
`Friedensplatz`	`Hermann-von-Gilm-Str.`
`Cimitero`	`Siegesplatz`
`Untervigil`	`Mariengasse`
`Marieng.`	`Untervigli`

- We match:
  - `Untervigil` $\leftrightarrow$ `Untervigli`
  - `Marieng.` $\leftrightarrow$ `Mariengasse`
  - `Gilmstrasse` $\leftrightarrow$ `Hermann-von-Gilm-Str.`
- But what to do with the others?
  - `Friedensplatz` was renamed to `Siegesplatz`, but one database was not updated
  - `Cimitero` is the Italian name for `Friedhofplatz` (German name)
- Problem: `Friedensplatz` looks more like `Friedhofplatz` than like `Siegesplatz`!

Demo: String Similarity

- Street name tables:

strID	*name*
α_1	Gilmstrasse
α_2	Friedensplatz
α_3	Cimitero
α_4	Untervigil
α_5	Marieng.

strID	*name*
β_1	Friedhofplatz
β_2	Hermann-von-Gilm-Str.
β_3	Siegesplatz
β_4	Mariengasse
β_5	Untervigli

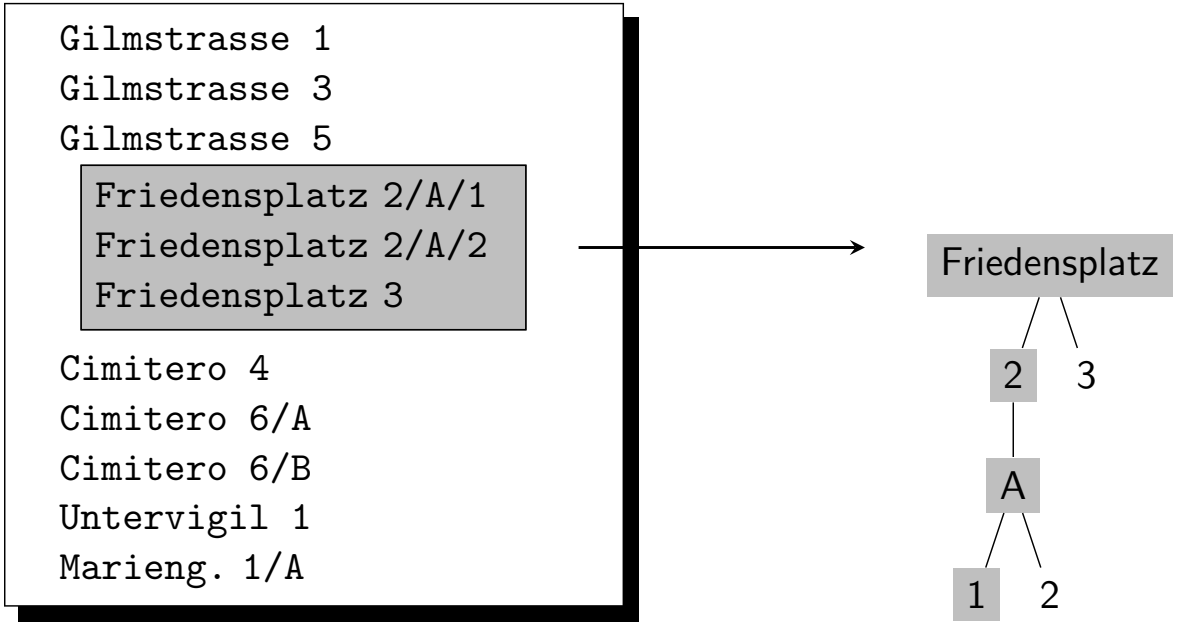
- Distance matrix for the q-gram distance between strings:

	β_1	β_2	β_3	β_4	β_5
α_1	1.0	0.8333	1.0	0.6923	1.0
α_2	0.3333	1.0	0.5714	0.9286	1.0
α_3	1.0	1.0	1.0	1.0	0.9091
α_4	1.0	0.9429	1.0	1.0	0.3333
α_5	0.92	0.9394	1.0	0.3913	1.0

- Matches with the global greedy algorithm:
  $\{(\alpha_1, \beta_2), (\alpha_2, \beta_1), (\alpha_3, \beta_3), (\alpha_4, \beta_5), (\alpha_5, \beta_4), \}$

Tree Similarity

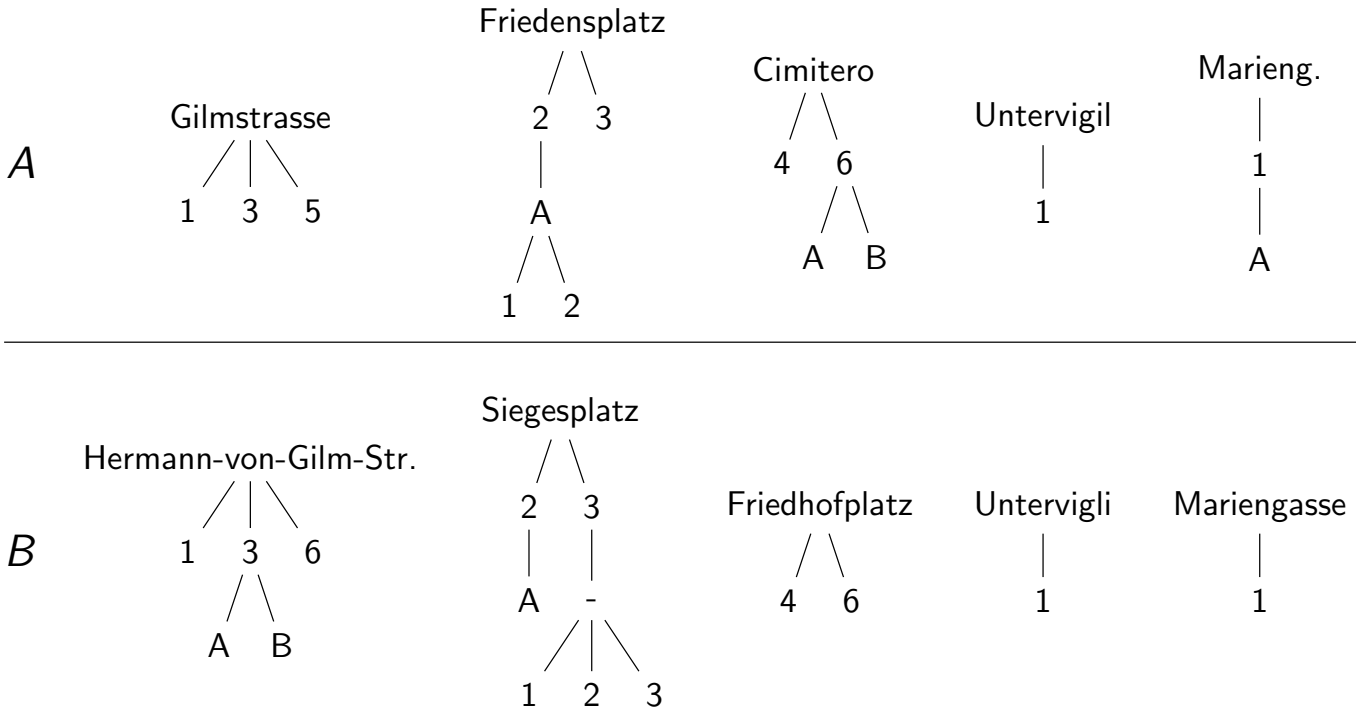
- **Observation 2**: Different streets have different addresses.
- Build *address tree*:



- Address is path from root to leaf.
- Example: Shaded path is the address `Friedensplatz 2/A/1` (house number `2`, entrance `A`, apartment `1`).

Tree Similarity

- Address trees of our example:



- Ignore root labels for distance computation.
- Trees of `Siegesplatz` and `Friedensplatz` are similar :-)
- Trees of `Cimitero` and `Friedhofplatz` are similar :-)
- But: `Untervigil` and `Mariengasse` have identical address trees in dataset B.

Demo: Tree Similarity

- Street name tables:

strID	*name*
α_1	Gilmstrasse
α_2	Friedensplatz
α_3	Cimitero
α_4	Untervigil
α_5	Marieng.

strID	*name*
β_1	Friedhofplatz
β_2	Hermann-von-Gilm-Str.
β_3	Siegesplatz
β_4	Mariengasse
β_5	Untervigli

- Distance matrix for the *pq*-gram distance between trees:

	β_1	β_2	β_3	β_4	β_5
α_1	1.0	0.7143	1.0	0.6667	0.6667
α_2	1.0	1.0	0.5758	1.0	1.0
α_3	0.4118	0.9167	1.0	1.0	1.0
α_4	1.0	0.7647	1.0	0.0	0.0
α_5	1.0	0.9	1.0	0.4545	0.4545

- Matches with the global greedy algorithm:
  $\{(\alpha_1, \beta_2), (\alpha_2, \beta_3), (\alpha_3, \beta_1), (\alpha_4, \beta_4), (\alpha_5, \beta_5)\}$

Combining String and Tree Distance

- Use strings *and* trees!
- String distance s, tree distance t
- Weight $\omega \in [0..1]$
  - $\omega = 0 \rightarrow$ only trees
  - $\omega = 1 \rightarrow$ only strings
- overall distance d (using weighted Euclidean distance):

$$d = \sqrt{\omega s^2 + (1 - \omega) t^2}$$

Demo: Combining String and Tree Distance

- Computed with $w = 0.5$ from string and tree matrices:

	β_1	β_2	β_3	β_4	β_5
α_1	1.0	0.7761	1.0	0.6796	0.8498
α_2	0.7454	1.0	0.5736	0.9649	1.0
α_3	0.7647	0.9592	1.0	1.0	0.9556
α_4	1.0	0.8584	1.0	0.7071	0.2357
α_5	0.9608	0.9199	1.0	0.4241	0.7767

- Matches with the global greedy algorithm:
  $\{(\alpha_4, \beta_5), (\alpha_5, \beta_4), (\alpha_2, \beta_3), (\alpha_3, \beta_1), (\alpha_1, \beta_2)\}$

- All matches are correct :-)

  `Gilmstrasse` ↔ `Hermann-von-Gilm-Str.`
  `Friedensplatz` ↔ `Siegesplatz`
  `Cimitero` ↔ `Friedhofplatz`
  `Untervigil` ↔ `Untervigli`
  `Marieng.` ↔ `Mariengasse`

`map_A_B`

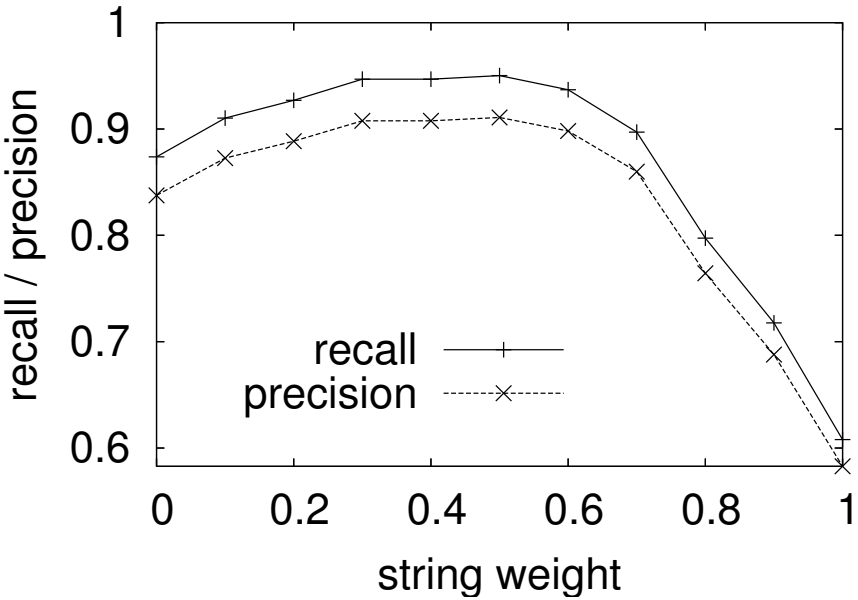
idA	*idB*
α_4	β_5
α_5	β_4
α_2	β_3
α_3	β_1
α_1	β_2

Experiments: Results for Real World Data

- Similarity join on three real databases:
  - electricity company (elec) – German street names, 45k addresses
  - registration office (reg) – Italian street names, 43k addresses
  - census database (cens) – German street names, 11k addresses
- Measure precision and recall
  - Precision: correctly computed matches to total number of computed matches
  - Recall: correctly computed matches to total number of correct matches

Experiments: Results for Real World Data

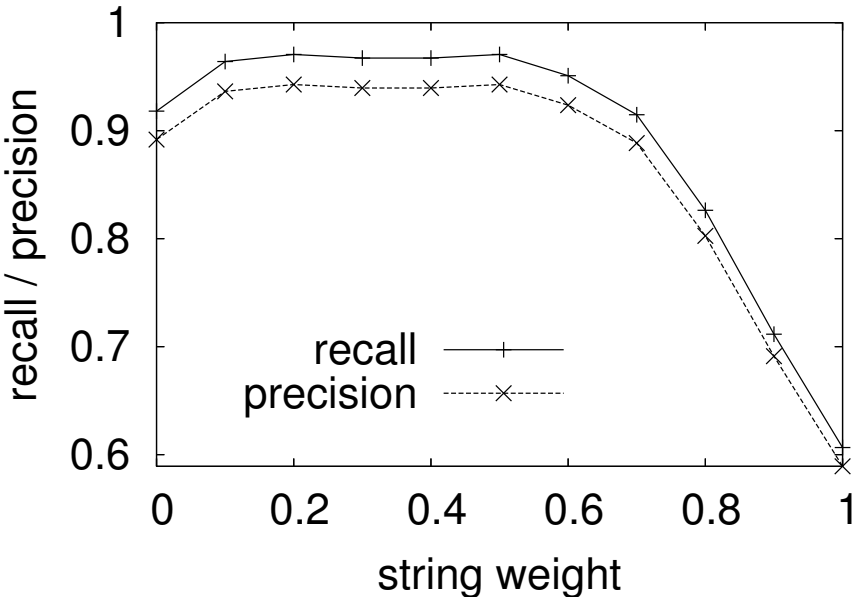
- Similarity join with global greedy matching
- String weight ω varies from 0 (only trees) to 1 (only strings)
- Measure precision and recall (high is good)



elec (German) $\leftrightarrow$ reg (Italian)

Experiments: Results for Real World Data

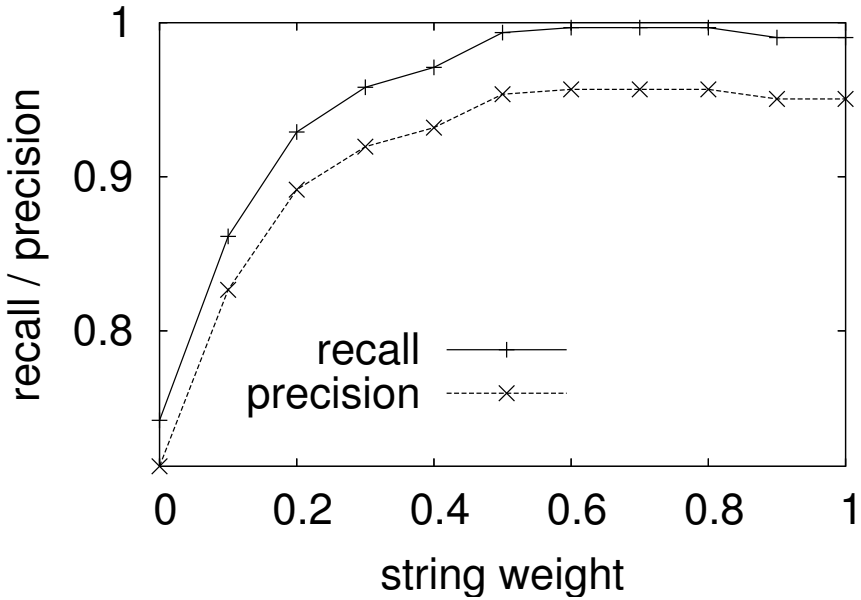
- Similarity join with global greedy matching
- String weight ω varies from 0 (only trees) to 1 (only strings)
- Measure precision and recall (high is good)



reg (Italian) $\leftrightarrow$ cens (German)

Experiments: Results for Real World Data

- Similarity join with global greedy matching
- String weight ω varies from 0 (only trees) to 1 (only strings)
- Measure precision and recall (high is good)



elec (German) $\leftrightarrow$ reg (German)

Experiments: Results for Real World Data

Summary of the experimental results:

- High string weight ω good for German-German, bad for German-Italian
- String weight $\omega = 0.5$ good for both German-German and German-Italian
- Precision and recall very high ($\omega = 0.5$):
  - more than 90% even for German-Italian
  - recall almost 100%, precision 95% for German-German ($\omega = 0.5$)

Outline

1 Course Organisation

2 Demo: Similarity Join on Residential Addresses

3 Conclusion

Conclusion

- Scope:
  - general, data-centric solutions
  - focus on efficiency
- Data types: sets, strings, trees
- Street matching demo:
  - leverage string and tree matching