Lecture 4

Entity Resolution

Are we the same?

Data Cleaning Course

VLDB 2017 Summer School, Beijing

Also known as

Duplicate detection	Record linkage
Match	Object identification
Fuzzy match	Deduplication
Object consolidation	Identity uncertainty
Entity clustering	Reference reconciliation
Approximate match	Merge/purge
Reference matching	....

Ironically, "Duplicate Detection" has many duplicates...

Definition

Duplicate detection is the discovery of **multiple representations** of the **same real-world object**

Entity Resolution

Problem 1

Representations are not identical.

Solution:

- Similarity measures
- Value- and record-comparisons
- Domain-dependent or domain-independent

Problem 2

Data sets are large.
Quadratic complexity: Comparison of every pair of records.

Solution:

- Algorithms that avoid all comparisons
- Partitioning
- Hash-based

Entity Resolution

Problem 3

Interaction between objects

Solution:

- Constraint-based reasoning

Outline

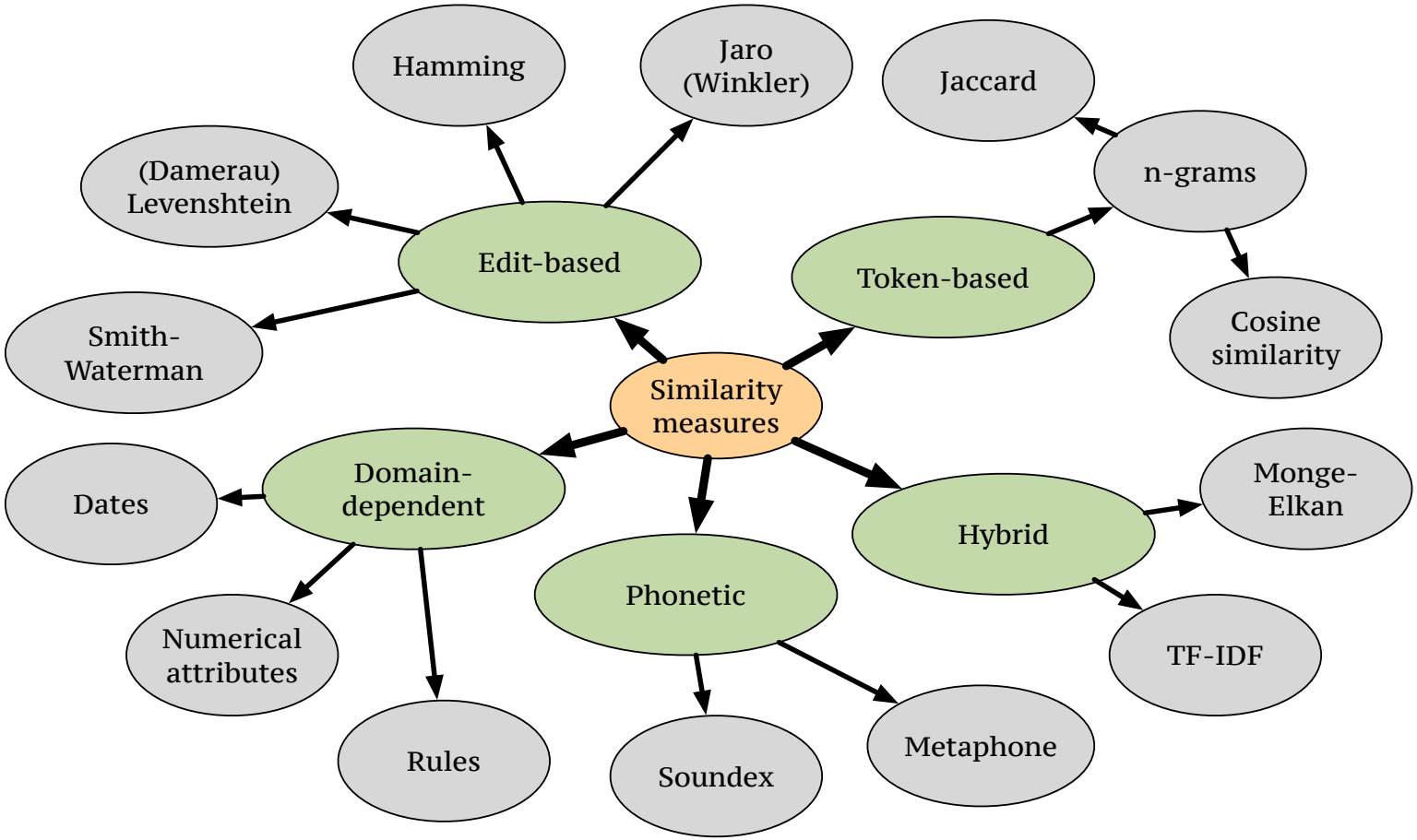
1. **Similarity Measures**
2. Three (constraint-based) ER methods
3. Conclusions

Similarity measures

The very first step in the Entity Resolution process to identify **when to objects are similar**.

At the basis of this lie **similarity measures**.

Overview of similarity measures



What is a similarity measure?

Denote by $\text{sim}(x, y)$ the **similarity** between objects x and y

- x and y can be strings, numbers, tuples, objects, images, ...

Normalized when $\text{sim}(x, y) \in [0, 1]$:

- $\text{sim}(x, y) = 1$ for exact match
- $\text{sim}(x, y) = 0$ for "completely different" x and y.
- $0 < \text{sim}(x, y) < 1$ for some approximate similarity.

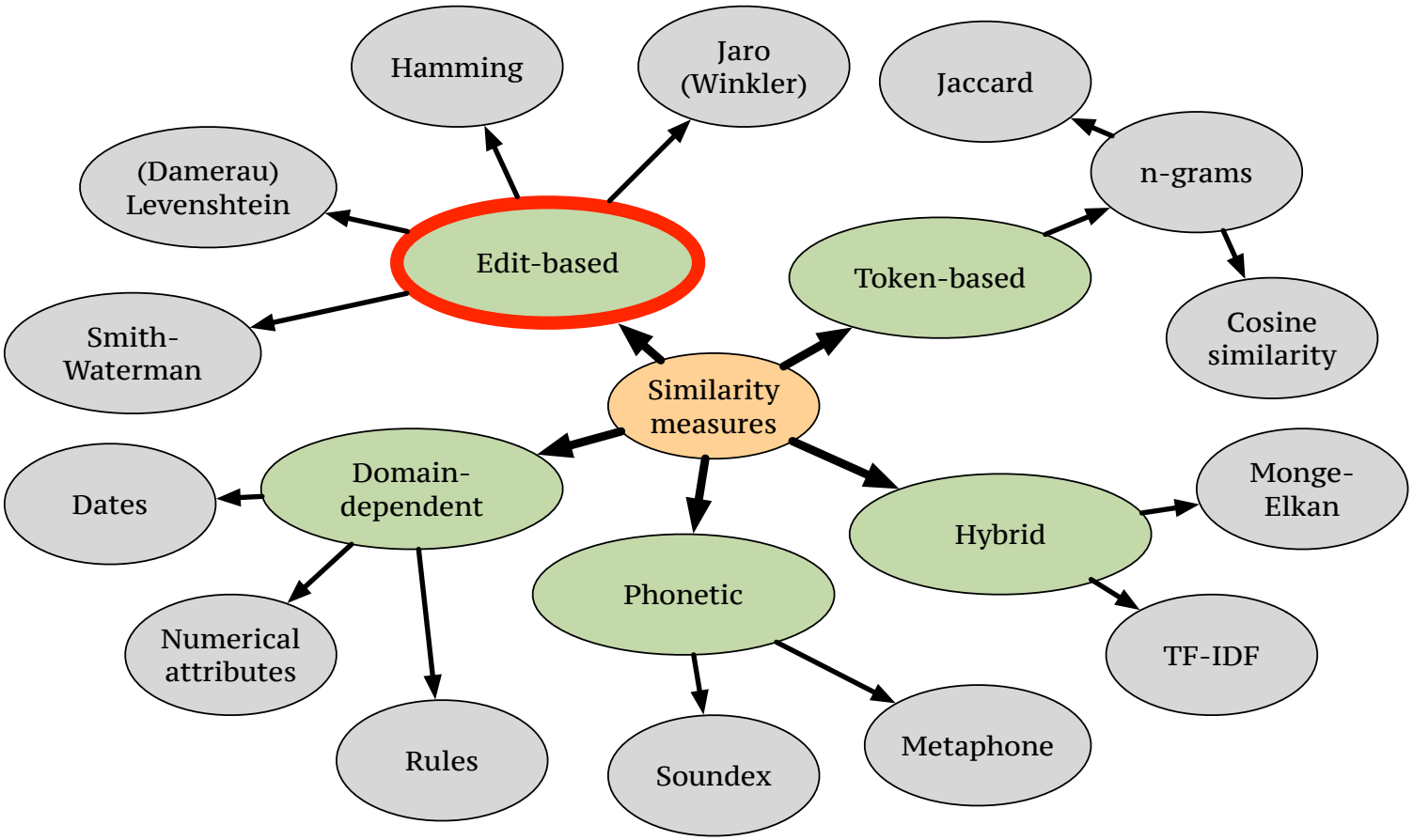
Example

Distance based Often used

$$\text{sim}(x, y) = 1 - \text{dist}(x, y) \text{ or } \text{sim}(x, y) = \frac{1}{\text{dist}(x, y)},$$

for distance function $\text{dist}(x, y)$. [1]

[1] Reflexive: $\text{dist}(x, x) = 0$, Positive: $\text{dist}(x, y) \geq 0$, Symmetric: dist(x,y) = dist(y,x), Triangular inequality: $\text{dist}(x, z) \leq \text{dist}(x, y) + \text{dist}(y, z)$



Hamming distance

Definition

- Number of **positions** in which two strings (of equal length) differ; or
- Minimum number of **substitutions** required to change one string into the other; or
- Minimum number of **errors** that could have transformed one string into the other.

$\Rightarrow$ Used mostly for binary numbers and to measure communication errors.

Example

- Hamming distance = number of 1's in x XOR y.
- $\text{dist}_{\text{hamming}}(\text{peter},\text{pedro}) = 3$.

Edit distances

Compare two strings based on individual characters.

Definition

- Minimal number of **edits** required to transform one string into the other.
- **Edits:** Insert, Delete, Replace (and Match)
- Give different cost to different types of edits
- Give different cost to different letters

Non-minimal edit cost

Consider

$$\text{dist}_{\text{edit}}(\text{Jones}, \text{Johnson})$$

Delete "Jones", then insert "Johnson"

DDDDDIIIIIII = 12 edits.

Levenshtein Distance

Definition

Minimum number of **character insertions, deletions, and replacements** necessary to transform s_1 into s_2. (edit distance, unit cost for each edit).

Is computed using **dynamic programming**: Optimality principle: Best transcript of two substrings must be part of best overall solution

Levenshtein

1 Initalize matrix M of size $(|s_1|+1) \times (|s_2|+1)$

2 Fill matrix $M[i,0] = i$ and $M[j,0] = j$.

3 Recursion

$$M[i,j] = \begin{cases} M[i,j] & \text{if } s_1[i] = s_2[j] \\ 1 + \min\{M[i-1,j], M[i,j-1], M[i-1,j-1]\} & \text{otherwise} \end{cases}$$

4 **return** $M[|s_1|, |s_2|]$

Levenshtein Distance

$$M[i,j] = \begin{cases} M[i,j] & \text{if } s_1[i] = s_2[j] \\ 1 + \mathbf{min}\{M[i-1,j], M[i,j-1], M[i-1,j-1]\} & \text{otherwise} \end{cases}$$

		J	O	N	E	S
	0	1	2	3	4	5
J	1					
O	2					
H	3					
N	4					
S	5					
O	6					
N	7					

		J	O	N	E	S
	0	1	2	3	4	5
J	1	0	1	2	3	4
O	2					
H	3					
N	4					
S	5					
O	6					
N	7					

		J	O	N	E	S
	0	1	2	3	4	5
J	1	0	1	2	3	4
O	2	1	0	1	2	3
H	3	2	1	1	2	3
N	4	3	2	1	2	3
S	5	4	3	2	2	2
O	6	5	4	3	3	3
N	7	6	5	4	4	4

Levenshtein similarity

Definition

$$\text{sim}_{\text{Levenshtein}}(s_1, s_2) = 1 - \frac{\text{dist}_{\text{Levenshtein}}(s_1, s_2)}{\max\{|s_1|, |s_2|\}}$$

Example

s_1	s_2	distance	similarity
Jones	Johnson	4	0.43
Paul	Pual	2	0.5
Paul Jones	Jones, Paul	11	0

Jaro Similarity

Specifically tailored towards **sharing of characters**:

Definition

Let m be the number of **matching characters** in s_1 and s_2:

- two characters x and y are matching if they are the *same* and not farther apart than

$$\lfloor \frac{\max\{|s_1|, |s_2|\}}{2} \rfloor - 1$$

.

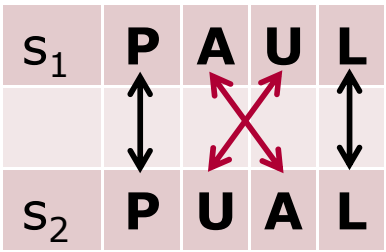
Let t be the number of matches that appear in a different order in s_1 and s_2.

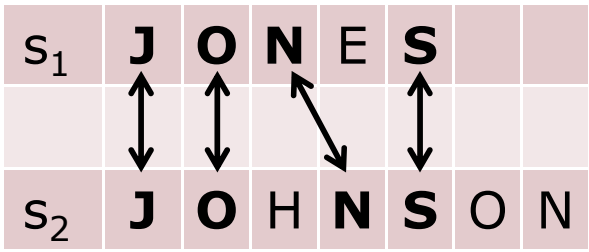
Then,

$$\text{sim}_{\text{Jaro}} = \frac{1}{3}\left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m - t/2}{m}\right).$$

Jaro similarity: Example

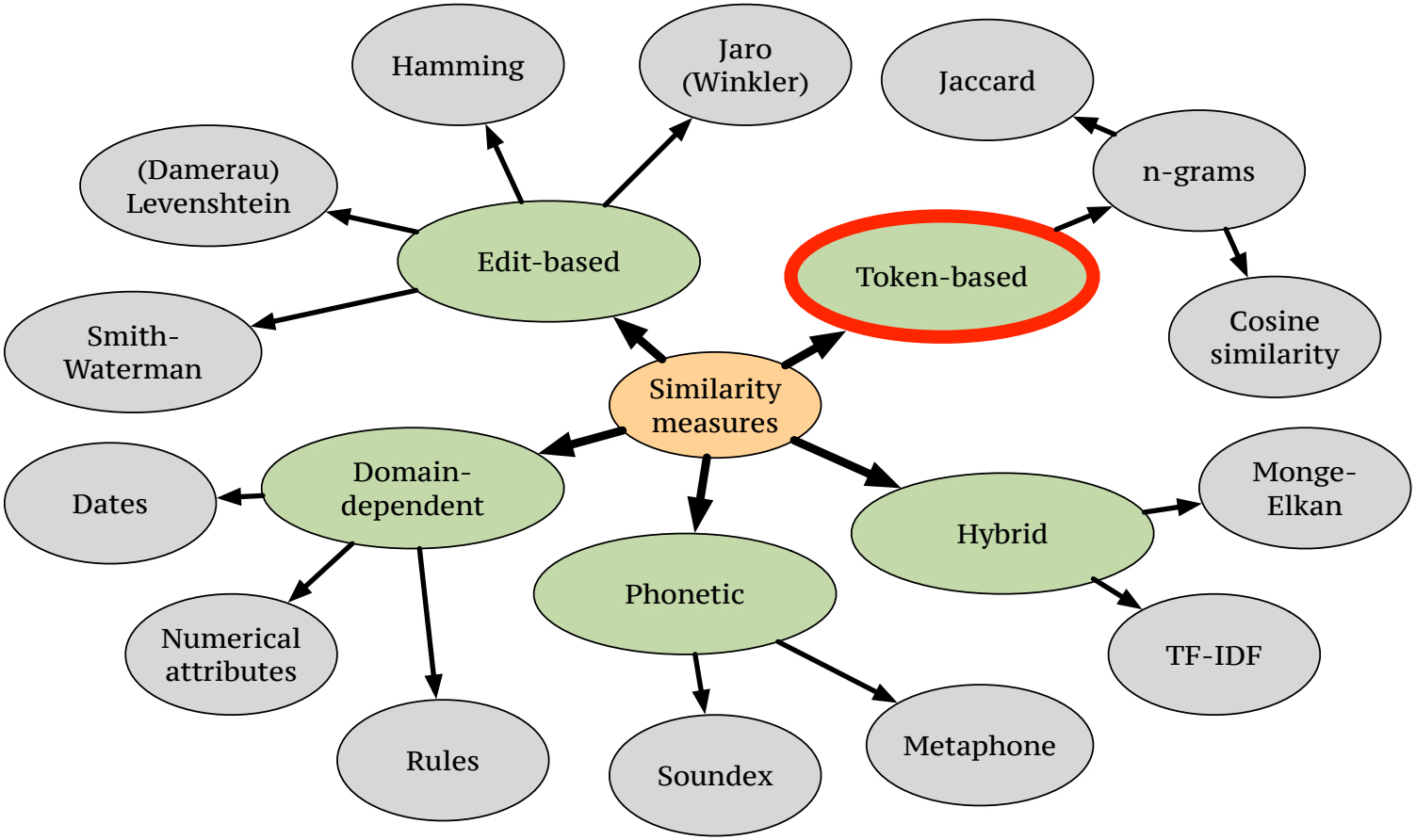
$$\text{sim}_{\text{Jaro}} = \frac{1}{3}\left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m - t/2}{m}\right).$$





$m = 4, \quad t = 2/2 = 1$
$\text{sim}_{\text{Jaro}} = \frac{1}{3}\left(\frac{4}{4} + \frac{4}{4} + \frac{4-1}{4}\right) \approx 0.92$

$m = 4,\ t = 0/2 = 0$
$\text{sim}_{\text{Jaro}} = \frac{1}{3}\left(\frac{4}{5} + \frac{4}{7} + \frac{4-0}{4}\right) \approx 0.79$



N-grams

To translate words and text into a **set of small pieces**, then use similarity function between sets.

Definition

For texts, a k-gram is a **consecutive set of k words**.
Sometimes, a k-gram also means just set of substrings of size k.

Example

Consider four documents:

D_1 : I am Sam D_3 : I do not like green eggs and ham
D_2 : Sam I am D_4 : I do not like them, Sam I am.

1-grams of all documents: { I, am, Sam, do, not, like, eggs, and, ham, green, then}}

2-grams {{ {I, am}, {am, Sam}, {Sam, I}, {I do}, {do not}, {not like},{like green}, {green eggs}, {eggs and},{and ham}, {like them}, {them Sam}, {Sam I}}

Jaccard Similarity

Definition

Given two sets A and B:

$$\text{sim}_{\text{Jaccard}}(A, B) = \frac{|A \cap B|}{|A \cup B|}.$$

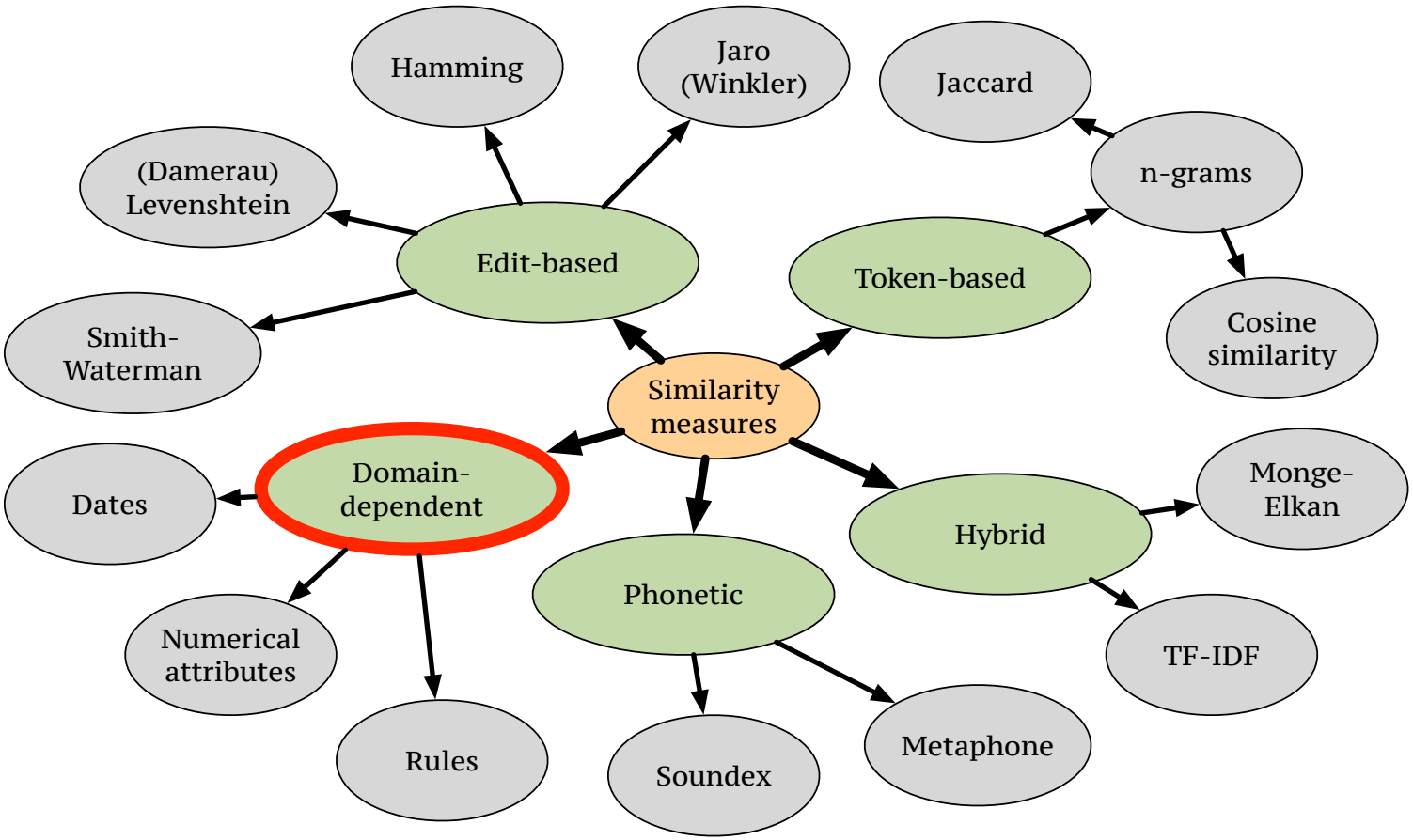
Example

When applied to 2-grams of D_1 and D_2:

$$D_1 := A = \{\{Iam\}, \{amSam\}$$
$$D_2 := B = \{\{SamI\}, \{Iam\}\}$$

Then,

$$\text{sim}_{\text{Jaccard}}(D_1, D_2) = \frac{|A \cap B|}{|A \cup B|} = 1/3 \approx 0.333$$



Numerical domains

- Normalized absolute distance:

$$\text{sim}_{\text{normabs}}(n, m) = \begin{cases} 1 - \left(\frac{|n-m|}{d_{\max}}\right) & \text{if } |n-m| \leq d_{\max} \\ 0 & \text{otherwise.} \end{cases}$$

Example

If $d_{\max} = \$1,000$. Then
$\text{sim}_{\text{normabs}}(\$2000, \$2500) = 1 - 1/2 = 1/2$. Also
$\text{sim}_{\text{normabs}}(\$200\,000, \$200\,500) = 1 - 1/2 = 1/2$

- Percentage:

$$\text{sim}_{\text{perc}}(n, m) = \begin{cases} 1 - \left(100\frac{|n-m|}{\max\{|n|,|m|\}p_{\max}}\right) & \text{if } 100\frac{|n-m|}{\max\{|n|,|m|\}} \leq p_{\max} \\ 0 & \text{otherwise.} \end{cases}$$

Example

If $p_{\max} = 33\%$. Then
$\text{sim}_{\text{perc}}(\$2000, \$2500) = 1 - 20/33 \approx 0.394$. Now,
$\text{sim}_{\text{perc}}(\$200\,000, \$200\,500) = 1 - 0,25/33 \approx 0.993$.

Time and geo domains

- Compute difference in dates in terms of **number of days**, then apply similarity measure on numerical domain.
- Dates of birth can also be converted to **age**, again using measure on numerical domain.
- Geographical location: Map it again to a number (using some **geographical projection**); or use **distance measures** and derived similarity measure.

Many more

There are many more similarity measures ...

See e.g., Tutorial [Record Linkage: Similarity Measures and Algorithms Nick Koudas, Sunita Sarawagi, Divesh Srivastava, SIGMOD 2006.]

In the following, I simply use "$\asymp$" to denote some similarity function...

Outline

1. Similarity Measures
2. **Three (constraint-based) ER methods**
   - Swoosh
   - Matching dependencies
   - Dedudaplog
3. Conclusions

Outline

1. Similarity Measures
2. **Three (constraint-based) ER methods**
   - **Swoosh**
   - Matching dependencies
   - Dedudaplog
3. Conclusions

Swoosh: A Generic Approach to Entity Resolution

- Developed in Stanford [Benjelloun, Omar and Garcia-Molina, Hector and Menestrina, David and Su, Qi and Whang, Steven Euijong and Widom, Jennifer (2008) Swoosh: a generic approach to entity resolution. The VLDB Journal]
- Very **generic** approach to ER:
  - functions for comparing and merging records as **black-boxes**
  - you can implement them however you want.
- Whenever these functions satisfy **certain properties**, however, you will end up with an **efficient ER algorithm.**

Swoosh: Inuitive Example

Example

tuple id	Name	Phone	E-mail
1	John Doe	235-2635	jdoe@email.com
2	J. Doe	234-4358	
3	John D.	234-4358	jdoe@email.com

Matching rule:

$$\text{Match}(i,j) \leftarrow t_i[\text{Name}] \asymp t_j[\text{Name}] \vee ((t_i[\text{Phone}] = t_j[\text{Phone}]) \wedge (t_i[\text{E-mail}] = t_j[\text{E-mail}]))$$

1. Tuples 1 and 2 match.
2. Merge tuples 1 and 2: New tuple 4:

4	John Doe	{235-2635, 234-4358}	jdoe@email.com

3. tuples 3 and 4 match. Merge.
4. Repeat.

Exhaustive procedure.

Swoosh: Inuitive Example

Example

tuple id	Name	Phone	E-mail
1	John Doe	235-2635	jdoe@email.com
2	J. Doe	234-4358	
3	John D.	234-4358	jdoe@email.com

Matching rule:

$$\text{Match}(i,j) \leftarrow \quad t_i[\text{Name}] \asymp t_j[\text{Name}]$$
$$\vee \left((t_i[\text{Phone}] = t_j[\text{Phone}]) \wedge (t_i[\text{E-mail}] = t_j[\text{E-mail}])\right)$$

1. Tuples 1 and 2 match.
2. Merge tuples 1 and 2: New tuple 4:

4	John Doe	{235-2635, 234-4358}	jdoe@email.com

3. tuples 3 and 4 match. Merge.
4. Repeat.

Exhaustive procedure.

Swoosh: Inuitive Example

Example

tuple id	Name	Phone	E-mail
1	John Doe	235-2635	jdoe@email.com
2	J. Doe	234-4358	
3	John D.	234-4358	jdoe@email.com

Matching rule:

$$\text{Match}(i,j) \leftarrow \quad t_i[\text{Name}] \asymp t_j[\text{Name}]$$
$$\vee \left((t_i[\text{Phone}] = t_j[\text{Phone}]) \wedge (t_i[\text{E-mail}] = t_j[\text{E-mail}])\right)$$

1. Tuples 1 and 2 match.
2. Merge tuples 1 and 2: New tuple 4:

4	John Doe	{235-2635, 234-4358}	jdoe@email.com

3. tuples 3 and 4 match. Merge.
4. Repeat.

Exhaustive procedure.

Swoosh: Inuitive Example

Example

tuple id	Name	Phone	E-mail
1	John Doe	235-2635	jdoe@email.com
2	J. Doe	234-4358	
3	John D.	234-4358	jdoe@email.com

Matching rule:

$$\text{Match}(i,j) \leftarrow \quad t_i[\text{Name}] \asymp t_j[\text{Name}]$$
$$\vee \left((t_i[\text{Phone}] = t_j[\text{Phone}]) \wedge (t_i[\text{E-mail}] = t_j[\text{E-mail}])\right)$$

1. Tuples 1 and 2 match.
2. Merge tuples 1 and 2: New tuple 4:

4	John Doe	{235-2635, 234-4358}	jdoe@email.com

3. tuples 3 and 4 match. Merge.
4. Repeat.

Exhaustive procedure.

Notation

Match function

Is a boolean function $\mu : \mathcal{D} \times \mathcal{D} \to \{\bot, \top\}$.

- $\bot$ stands for false; $\top$ for true
- $\mu(s, t)$ if and only if s and t are the same
- E.g., $\mu(s, t) = \top$ iff $\text{sim}(s, t) \geq \theta$.
- The match function is a black box

Merge function

Merge of s and t is denoted by $\mathfrak{m}(s, t)$

- Only defined for matching records
- The merge function is also a black box

Merge closure

Closure of database under merge function $\mathfrak{m}$

Let $\mathcal{D}$ be a database instance. Then the **merge closure** of $\mathcal{D}$, denoted by $\mathcal{D}^\star$ is the smallest set of tuples such that

- $\mathcal{D} \subseteq \mathcal{D}^\star$; and
- for any $s, t \in \mathcal{D}^\star$, $\mathfrak{m}(s, t) \in \mathcal{D}^\star$.

The closure is the result of exhaustively applying the merge operation.

Properties

- Closure is unique :-)
- Can be infinite :-(

Not realistic

The closure will not be very practical ...

Domination: Reducing the Closure...

Domination

A tuple s is **dominated** by tuple t if

- $\mu(s, t) = \top$ (they match); and
- $s \preceq t$ (t holds more information that s)

Here, $\preceq$ is any **partial order** on tuples:

- $\preceq$ is reflexive, transitive, and anti-symmetric
- Application/domain specific.

Example

We could assume that $t_1 \preceq t_4$ and $t_1 \preceq t_4$

tuple id	Name	Phone	E-mail
1	John Doe	235-2635	jdoe@email.com
2	J. Doe	234-4358	
3	John D.	234-4358	jdoe@email.com
4	John Doe	{235-2635, 234-4358}	jdoe@email.com

so that

$\mathfrak{m}(t_1, t_2)$ contains more information and dominates t_1 and t_2.

Instance Domination

We can lift domination between tuples to **domination on instances:**

Definition

Instance $\mathcal{D}'$ **dominates** instance $\mathcal{D}$ if every tuple in $\mathcal{D}$ is dominated by a tuple in $\mathcal{D}'$.

Note that instance domination is

- reflexive, transitive
- not antisymmetric. Why? $t_1 \preceq t_4$,then $t_4 \preceq \{t_1, t_4\}$ and $\{t_4, t_1\} \preceq t_4$.

Example

Assuming that $t_1 \preceq t_4$ and $t_1 \preceq t_4$

tuple id	Name	Phone	E-mail
1	John Doe	235-2635	jdoe@email.com
2	J. Doe	234-4358	
3	John D.	234-4358	jdoe@email.com
4	John Doe	{235-2635, 234-4358}	jdoe@email.com

dominates the original instance.

Instance Domination

We can lift domination between tuples to **domination on instances:**

Definition

Instance $\mathcal{D}'$ **dominates** instance $\mathcal{D}$ if every tuple in $\mathcal{D}$ is dominated by a tuple in $\mathcal{D}'$.

Note that instance domination is

- reflexive, transitive
- not antisymmetric. Why? $t_1 \preceq t_4$,then $t_4 \preceq \{t_1, t_4\}$ and $\{t_4, t_1\} \preceq t_4$.

Example

Assuming that $t_1 \preceq t_4$ and $t_1 \preceq t_4$

tuple id	Name	Phone	E-mail
1	John Doe	235-2635	jdoe@email.com
2	J. Doe	234-4358	
3	John D.	234-4358	jdoe@email.com
4	John Doe	{235-2635, 234-4358}	jdoe@email.com

dominates the original instance.

Instance Domination

We can lift domination between tuples to **domination on instances:**

Definition

Instance $\mathcal{D}'$ **dominates** instance $\mathcal{D}$ if every tuple in $\mathcal{D}$ is dominated by a tuple in $\mathcal{D}'$.

Note that instance domination is

- reflexive, transitive
- not antisymmetric. Why? $t_1 \preceq t_4$,then $t_4 \preceq \{t_1, t_4\}$ and $\{t_4, t_1\} \preceq t_4$.

Example

Assuming that $t_1 \preceq t_4$ and $t_1 \preceq t_4$

tuple id	Name	Phone	E-mail
1	John Doe	235-2635	jdoe@email.com
2	J. Doe	234-4358	
3	John D.	234-4358	jdoe@email.com
4	John Doe	{235-2635, 234-4358}	jdoe@email.com

dominates the original instance.

Entity resolution according to Swoosh

Definition

Given an instance $\mathcal{D}$, an **entity resolution of** $\mathcal{D}$, denoted by $\text{ER}(\mathcal{D})$ is a set of tuples such that

- $\text{ER}(\mathcal{D}) \subseteq \mathcal{D}^*$ (should be in $\mathcal{D}$'s merge closure)
- $\text{ER}(\mathcal{D})$ dominates $\mathcal{D}^\star$ (it carries at least as much information as the merge closure)
- It is the minimal set of tuples satisfying the previous two conditions.

The hope is that dominance ensures that $\text{ER}(\mathcal{D})$ is a finite set.

Assumptions on merge and match function will need to be made to ensure finiteness.

ICAR properties

Idempotence:

- for any tuple t, $\mu(t, t) = \top$ and $\mathfrak{m}(t, t) = t$.
- A record always matches itself, and merging it with itself still yields the same record.

Commutativity:

- for any tuples s and t, $\mu(s, t) = \mu(t, s)$ and if $\mu(s, t) = \top$ then $\mathfrak{m}(s, t) = \mathfrak{m}(t, s)$.
- Direction of match and merge is irrelevant

Associativity:

- for any tuples s, t and u such $\mathfrak{m}(\mathfrak{m}(s, t), u)$ and $\mathfrak{m}(s, \mathfrak{m}(t, u))$ exist, then

$$\mathfrak{m}(\mathfrak{m}(s, t), u) = \mathfrak{m}(s, \mathfrak{m}(t, u)).$$

- Order of merge is irrelevant.

Representativity:

- for any tuple $u = \mathfrak{m}(s, t)$, if $\mu(v, s) = \top$ then also $\mu(v, u)$.
- Merging does not lose matches.

Merge domination

When the match and merge functions satisfy the ICAR properties, there is a natural domination order.

Merge domination

Given two tuples s and t we say that s is **merge dominated** by t, denoted $s \leq t$, if

- $\mu(s, t) = \top$; and
- $\mathfrak{m}(s, t) = t$.

It just means that s does **not add information and can be replaced by t.**

Monotonicity

Properties of merge domination

For any tuples s and t that match, it holds that

$$s \leq \mathfrak{m}(s,t) \text{ and } t \leq \mathfrak{m}(s,t).$$

- Merge record always dominates the records it was derived from

If $s \leq t$ and s matches u then also t matches u.

- Match function is monotonic

If $s \leq t$ and s matches u, then $\mathfrak{m}(s,u) \leq \mathfrak{m}(t,u)$.

- Merge function is monotonic

If $s \leq u$ and $t \leq u$ and s and t match, then $\mathfrak{m}(s,t) \leq u$.

- Merge is "smallest" dominating tuple.

Swoosh Guarantees

If ICAR properties are satisfied then

1. ER process is guaranteed to be finite
2. Records can be matched and merged in any order
3. Dominated records can be discarded anytime

That's what we wanted!

R-Swoosh Algorithm

R-Swoosh

Function: `r-swoosh`$(\mathcal{D})$

return ER$(\mathcal{D})$.

$\mathcal{D}' : \emptyset$
while $\mathcal{D}' \neq \emptyset$ **do**
  t_{current}:=a tuple in $\mathcal{D}'$
  Remove t_{current} from $\mathcal{D}'$
  t_{buddy}:=null
  for $t' \in$ ER$(\mathcal{D})$ **do**
    if $\mu(t', t_{\text{current}}) = \top$ **then**
      /*Recall that μ can be based on matching rules!*/
      $t_{\text{buddy}} = t'$ and **ExitFor**
  if $t_{\text{buddy}} =$ null **then**
    Add t_{current} to ER$(\mathcal{D})$
  else
  Add $\mathfrak{m}(t_{\text{current}}, t_{\text{buddy}})$ to ER$(\mathcal{D})$
  Remove t_{buddy} from ER$(\mathcal{D})$.
return ER$(\mathcal{D})$.

Swoosh - Conclusion

Very generic approach

Some optimizations and variants

- Smart ordering reduces comparisons
- F-swoosh: Uses hashing techniques on features
- Incremental F-Swoosh
- D-Swoosh: distributed ER

Please check Stanford Entity Resolution Framework for more information: `http://infolab.stanford.edu/serf/`

Outline

1. Similarity Measures
2. **Three (constraint-based) ER methods**
   - Swoosh
   - **Matching dependencies**
   - Dedudaplog
3. Conclusions

Matching dependencies

- We have seen matching dependencies in the first lecture
- Introduced in a series of papers:
  - [Wenfei Fan, Shuai Ma, Nan Tang, Wenyuan Yu: Interaction between Record Matching and Data Repairing.. J. Data and Information Quality 4(4): 16:1-16:38 (2014)
  - [Wenfei Fan, Hong Gao, Xibei Jia, Jianzhong Li, Shuai Ma: Dynamic constraints for record matching. VLDB J. 20(4): 495-520 (2011)]
  - [Wenfei Fan, Xibei Jia, Jianzhong Li, Shuai Ma: Reasoning about Record Matching Rules. PVLDB 2(1): 407-418 (2009)]
- Semantics of matching dependencies further explored by Bertossi et al [Leopoldo E. Bertossi, Solmaz Kolahi, Laks V. S. Lakshmanan: Data Cleaning and Query Answering with Matching Dependencies and Matching Functions. Theory Comput. Syst. 52(3): 441-482 (2013)]

Matching dependencies

- Matching dependencies naturally fit in the Swoosh approach (as the merge and match function black boxes)
- When used for ER, they also can be equipped with a **chase semantics**.
  - We have seen examples of the chase in the previous lecture.

Matching dependencies: Example

MD

"The similarities of phone and address indicate that the tuples refer to the same person, and the names should be matched."

Consider table P:

Name	Phn	Addr
John Smith	723-9583	10-43 Oak St.
J. Smith	(750) 723-9583	43 Oak St. Ap. 10

Here, 723-9583$\asymp$ (750) 723-9583 and 10-43 Oak St. $\asymp$ 43 Oak St. Ap. 10.
A matching dependency capturing this cleaning policy:

$$P[\text{Phn}] \asymp P[\text{Phn}] \wedge P[\text{Addr}] \asymp P[\text{Addr}] \rightarrow P[\text{Name}] \equiv P[\text{Name}]$$

Matching dependencies

MDs are rules of the form

$$\bigwedge_{i,j} R[A_i] \asymp_{i,j} S[B_j] \rightarrow \bigwedge_{k,\ell} R[D_k] \equiv S[E_\ell].$$

The left side captures a similarity condition on pairs of tuples, in relations R and S Abbreviation: $R[\bar{A}] \asymp S[\bar{B}] \rightarrow R[\bar{D}] \equiv S[\bar{E}]$.

Static interpretation:

- If antecedent is true for a pair of tuples, then the values $R[D_k]$ and $S[E_\ell]$ should be the same

Dynamic interpretation:

- Those values on the RHS should be updated to some **(unspecified)** common value

Ingredients

To make sure that the MDs know how to fix the RHS, we can fit it into Swoosh:

- A set Σ of MDs
- for every attribute A with domain $\mathrm{Dom}(A)$:
  - a similarity relation $\asymp_A \subseteq \mathrm{Dom}(A) \times \mathrm{Dom}(A)$
  - a merge function $\mathfrak{m}_A : \mathrm{Dom}(A) \times \mathrm{Dom}(A) \to \mathrm{Dom}(A)$ which idempotent, commutative, and associative.

Enforcing MDs

MD Chase step

- Given, a pair of instances $\mathcal{D}$ and $\mathcal{D}'$
- MD $\varphi = R_1[X_1] \asymp R_2[X_2] \rightarrow R_1[A_1] \equiv R_2[A_2]$
- A pair of tuples s and t in $\mathcal{D}$ such that $s[X_1] \asymp t[X_2]$ but $s[A_1] = a_1 \neq s[A_2] = a_2$
- Then, $\mathcal{D} \Rightarrow_{\varphi,s,t} \mathcal{D}'$ if $\mathcal{D}'$ is the same as $\mathcal{D}$ except that

$$s[A_1] = t[A_2] = \mathfrak{m}(a_1, a_2).$$

Clean Instance

A clean instance $\mathcal{D}'$ is the result of exhaustively applying MD chase steps:

$$\mathcal{D} = \mathcal{D}_0 \Rightarrow_{\varphi_1,s_1,t_1} \mathcal{D}_1 \Rightarrow_{\varphi_2,s_2,t_2} \mathcal{D}_2 \Rightarrow_{\varphi_1,s_3,t_3} \cdots \Rightarrow_{\varphi_k,s_k,t_k} \mathcal{D}'$$

and no rule can be applied anymore to $\mathcal{D}'$.

Guarantees of the Chase Process

Only ICA on matching function is required.
The process terminates after a finite number of steps, resulting in a clean instance.

If in addition $a \asymp a'$ implies that $a \asymp \mathfrak{m}(a, a')$ then

ICA assumptions

The process terminates after a finite number of steps, resulting in a **unique clean instance.**

ER with Matching Dependencies

- Implements black box of match in Swoosh in a **declarative way**
- Only conditions (ICA) on the merge function $\mathfrak{m}$ is required to guarantee a unique solution.
- This leads to a **very flexible** approach.

Outline

1. Similarity Measures
2. **Three (constraint-based) ER methods**
   - Swoosh
   - Matching dependencies
   - **Dedudaplog**
3. Conclusions

Logic + Clustering

We next consider an approach that relates

Constraints (logic) + Clustering

It uses a **completely different approach** to do ER with constraints....

> [Arvind Arasu, Christopher Ré, Dan Suciu: Large-Scale Deduplication with Constraints Using Dedupalog. ICDE 2009: 952-963]

Example

Consider `wrote(id, pos, author)` table:

id	pos	Authors
1	1	A. Gionis
1	2	H. Manilla
1	3	P. Tsaparas
2	1	A. Gionis
3	1	L. Bhattacharya
4	2	L. Getoor

and `PaperRefs(id,title,venue, year)` table

id	title	venue	year
1	Cluster Aggregation	ICDE	2005
2	Clustering Aggregations	Conference on Data Eng	2005
3	Collective ER	Data Eng Bull.	2007
4	Collective ER	Data Engineering	2007

Example

First step: Identify Entity Reference Tables

```
Author!(id,pos) ← wrote(id, pos,_)
  Publisher!(p) ← PaperRefs(_,_,p, _)
     Paper!(id) ← PaperRefs(id,_,_, _)
```

These list on which attributes we may want to do ER.

Second step: Associate binary (clustering) relations

```
Author*(id,pos,id',pos')
Publisher*(p,p')
Paper*(id,id)
```

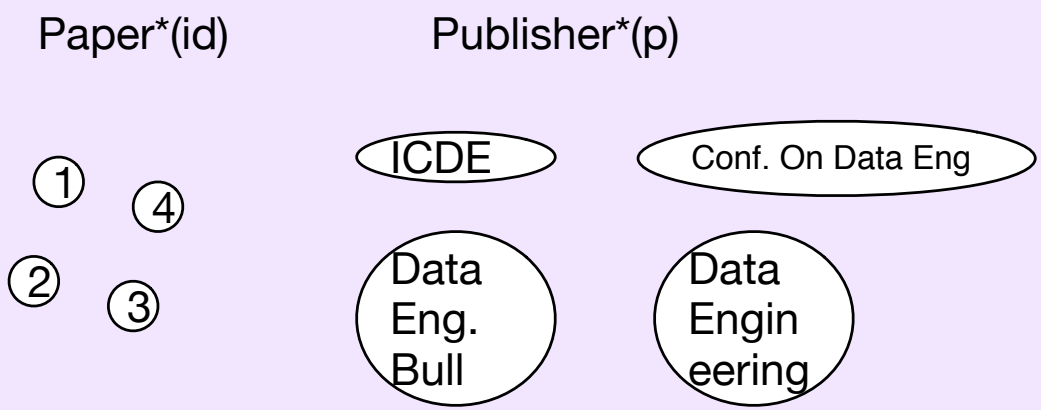
These list pairs of objects that may be the same. Dedupalog will find a **clustering** of these objects.

Linking things together

Step 3: Use constraints

Relate the tables with the clustering relations using rules (constraints)

id	title	venue	year
1	Cluster Aggregation	ICDE	2005
2	Clustering Aggregations	Conference on Data Eng	2005
3	Collective ER	Data Eng Bull.	2007
4	Collective ER	Data Engineering	2007



Soft-complete rules

Example

"papers with similar titles are likely duplicates"

$$\begin{aligned}\texttt{Paper}*(id, id') \leftrightarrow \quad & \texttt{PaperRefs}(id, t, \_, \_, \_), \\ & \texttt{PaperRefs}(id', t', \_, \_, \_), \\ & \texttt{TitleSimilar}(t, t')\end{aligned}$$

- Paper references whose titles appear in `TitleSimilar` are likely to be clustered together.
- Paper references whose titles do not appear in `TitleSimilar` are not likely to be clustered together.

Soft-incomplete rules

Example

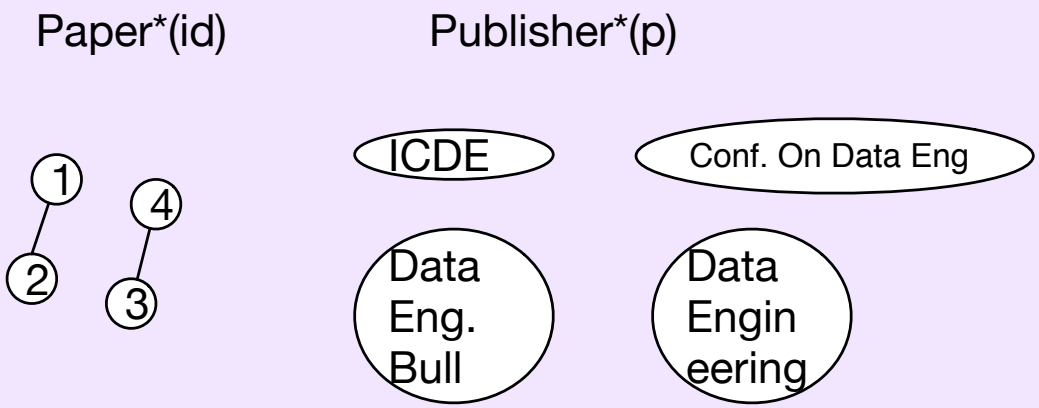
"papers with very similar titles are likely duplicates"

$$\texttt{Paper}*(id, id') \leftarrow \texttt{PaperRefs}(id, t, \_, \_, \_), \texttt{PaperRefs}(id', t', \_, \_, \_), \texttt{TitleVerySimilar}(t, t')$$

- Paper references whose titles appear in `TitleVerySimilar` are likely to be clustered together.
- This rule says nothing about paper references whose titles do **not appear** in `TitleVerySimilar`.

Effect of rules

id	title	venue	year
1	Cluster Aggregation	ICDE	2005
2	Clustering Aggregations	Conference on Data Eng	2005
3	Collective ER	Data Eng Bull.	2007
4	Collective ER	Data Engineering	2007



Hard rules

Example

"the publisher references listed in `PublisherEQ` *must be clustered together"*

"the publisher references in `PublisherNEQ` *must not be clustered together"*

$$\texttt{Publisher}*(x, y) \Leftarrow \quad \texttt{PublisherEq}(x, y)$$
$$\neg\texttt{Publisher}*(x, y) \Leftarrow \quad \texttt{PublisherNEq}(x, y)$$

First rule indicates a "must link", the second one a "cannot link".

- Hard rules must be satisfied in any legal clustering

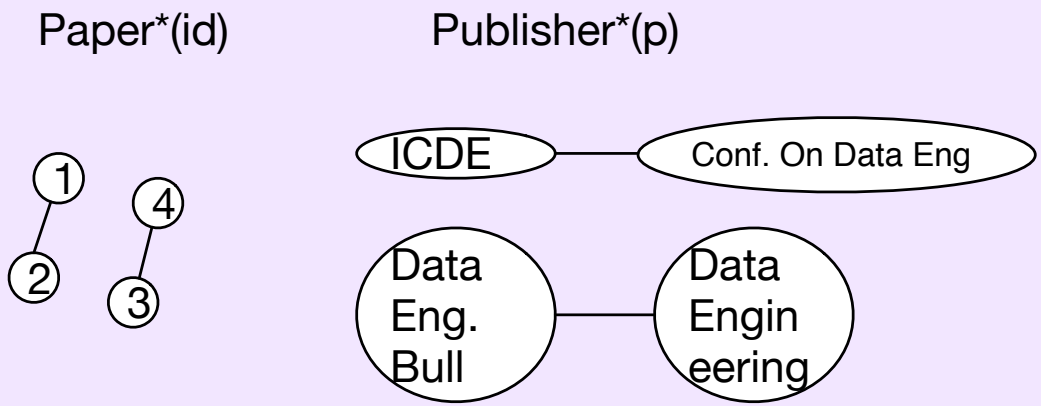
Complex Hard rules

Example

"whenever we cluster two papers, we must also cluster the publishers of those papers"

$$\texttt{Publisher*}(x, y) \Leftarrow \texttt{Publishes}(x, p_1)$$
$$\texttt{Publishes}(y, p_2), \texttt{Paper*}(p_1, p_2).$$

id	title	venue	year
1	Cluster Aggregation	ICDE	2005
2	Clustering Aggregations	Conference on Data Eng	2005
3	Collective ER	Data Eng Bull.	2007
4	Collective ER	Data Engineering	2007



Complex Negative rules

Example

"two distinct author references on a single paper cannot be the same person"

$$\neg \texttt{Author}*(x, i, y, j) \Leftarrow \quad \texttt{Wrote}(p, x, i), \texttt{Wrote}(p, y, j), i \neq j$$

Recursive rules

Example

"Authors that do not share common coauthors are unlikely to be duplicates"

$$\neg\texttt{Author}*(x, i, y, j) \leftarrow \neg\big(\texttt{Wrote}(x, i, \_), \texttt{Wrote}(y, j\ \_),$$
$$\texttt{Wrote}(x, p, \_), \texttt{Wrote}(y, p', \_),$$
$$\texttt{Author}*(x, p, y, p')\big).$$

Clustering

Finding the best clustering

Given the entity reference tables $\mathcal{D}$ and dedupalog program Γ. find the clustering C of $\mathcal{D}$ such that

- $C \models \Gamma_{\text{hard}}$; and
- the cost

$$\text{Cost}(C, \Gamma) = \sum_{\gamma \in \Gamma_{\text{soft}}} \text{Cost}(C, \gamma)$$

  is minimal.

Here, $\text{Cost}(C, \gamma)$ is the number of pairs in the clustering that γ is not satisfied on C.

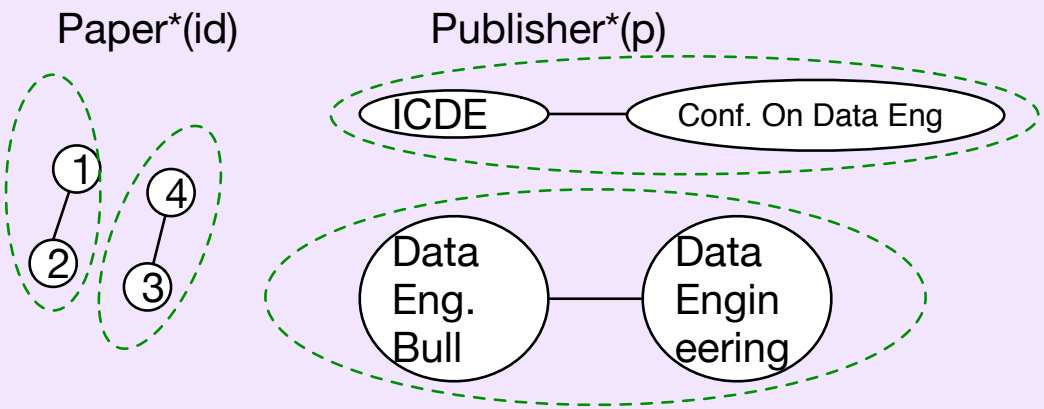
Complexity

NP-complete to decide whether there is a clustering below a certain threshold.

Example

Example

id	title	venue	year
1	Cluster Aggregation	ICDE	2005
2	Clustering Aggregations	Conference on Data Eng	2005
3	Collective ER	Data Eng Bull.	2007
4	Collective ER	Data Engineering	2007



Perfect clustering

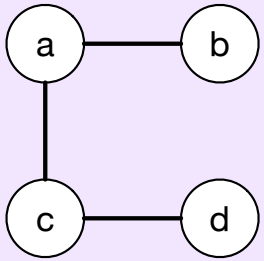
Another Example

Example

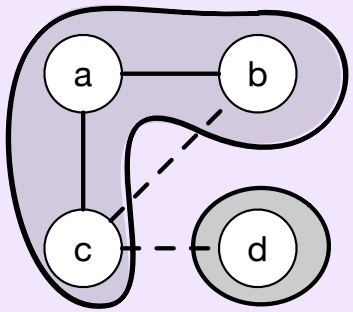
Consider

$$\mathrm{R}*(x, y) \leftrightarrow E(x, y)$$

where E is the graph shown below.



E



C

Entity reference table E!:

a
b
c
d

Cost of clustering C is **two**:

- d should belong the same cluster as c
- c should not belong to the same cluster as b.

Correlation clustering

Definition

Given an undirected graph $G = (V, E)$ with edge labels $\{+, -\}$.

- A **correlation clustering** $\mathcal{C}$ is a partitioning of the vertices in V.
- A **false positive edge** is a $-$-labeled edge (v, w) such that u and v are clustered together in $\mathcal{C}$.
- A **false negative edges** is a $+$-labeled edge (v, w) such that u and v are not clustered together in $\mathcal{C}$
- The **cost** of $\mathcal{C}$ is defined as,

$$\text{cost}(\mathcal{C}, G) = |\text{false positive edges}| + |\text{false negative edges}|$$

Problem

Find the correlation clustering of smallest cost.

Label assignment

Idea:

With each dedupalog rule $\gamma \in \Gamma$, associate **counting rules** γ_c.

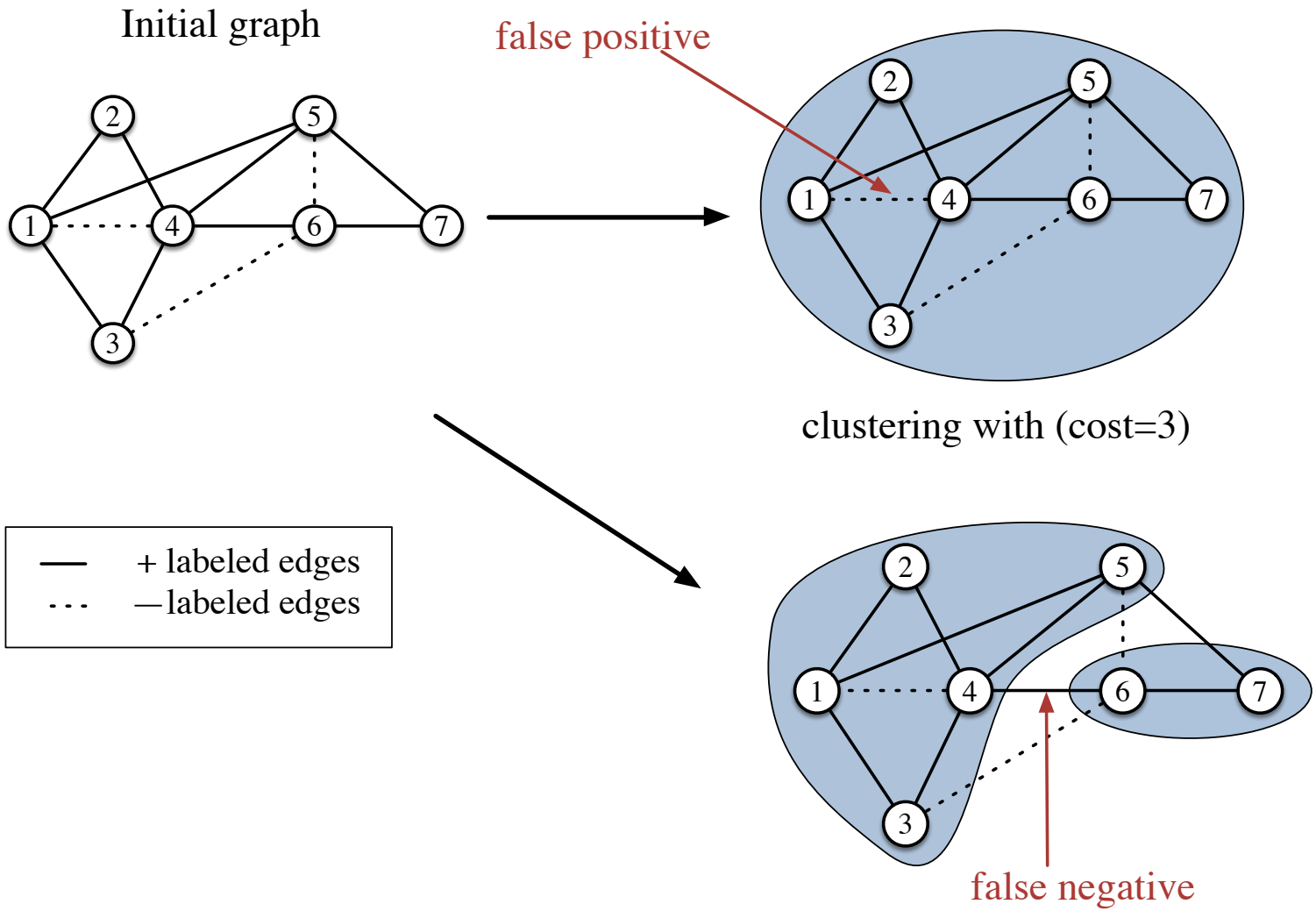
- Each pair of objects may**retrieve** $+$ **or** $-$ **from** γ_c

Use **majority voting** to decide final label of pair of objects:

- If a pair received more $+$-labels than $-$-labels: Final label is "$+$"
- If a pair received more $-$-labels than $+$-labels: Final label is "$-$"

In this way, a graphs are obtained from $\mathcal{D}$ and Γ that are given as **input to correlation clustering problem.**
(things are bit more complicated in the presence of recursive rules)

Correlation Clustering - Example



Correlation Clustering algorithm

Problem is NP-hard in general $\Rightarrow$ Approximation

Naive Algorithm

Function: `region-growing` (G)

return A clustering $\mathcal{C}$

Solve a Linear Program (LP) for Correlation Clustering
Let $w(e_i)$ be the (fractional weight of edge e_i
Select a vertex v.
Neighborhoud $V = \{v\}$
while $G \neq \emptyset$ **do**
    while condition is not met **do**
        Keep adding neighbours of V to V
    return V
    Let Δ be all vertices and edges adjacent to V from G
    Let $G := G \setminus \Delta$.

Integer vs Linear Program formulation

The correlation clustering can be described as solving an **integer program**:

$$\begin{array}{ll} \text{minimize} & \sum_{e\in E^-}(1-xe)+\sum_{e\in E^+}x_e \\ \text{subject to} & x_e \in \{0,1\} \\ & x_{uv}+x_{vw}\geq x_{uw} \\ , & x_{uv}=x_{vu}. \end{array}$$

NP-hard to solve. Instead solve **linear relaxation:**

$$\begin{array}{ll} \text{minimize} & \sum_{e\in E^-}(1-xe)+\sum_{e\in E^+}x_e \\ \text{subject to} & x_e \in [0,1] \\ & x_{uv}+x_{vw}\geq x_{uw} \\ , & x_{uv}=x_{vu}. \end{array}$$

PTIME to solve and $SOL_{lp} \leq SOL_{ip}$.

The weights of the solution of the linear relaxation are used to measure how large a region can grow.

Guarantee of Region growing algorithm

The clustering $\mathcal{C}$ returned by region growing is at most a factor $O(\log(E))$ from the optimal solution.

Not a heuristic, but a true **approximation algorithm.**

Comparison of various correlation clustering algorithm [Elsner et al, ILP-NLP'09]

See also tutorial at KDD: Correlation Clustering: from Theory to Practice Francesco Bonchi, David Garcia-Soriano, Edo Liberty, KDD 2014

Summary Dedupalog

- Declarative way of doing ER by means of clustering
- This is not the only way one could get clusterings.
- It is open whether other clustering techniques may give better results.

To conclude,

- Only scratched the surface of ER techniques
- Focus mainly on constraint-based approaches
- See VLDB 2012 Tutorial for other techniques [Entity Resolution: Tutorial, by Lise Getoor, Ashwin Machanavajjhala]