Hasso Plattner Institut
Digital Engineering · Universität Potsdam

Information Integration

Duplicate Detection

13.01.2020

Felix Naumann

Overview

1. Duplicate detection
2. Similarity measures
3. Algorithms
4. Data sets and evaluation
5. Data fusion

Felix Naumann
Information Integration
Winter 2019/20

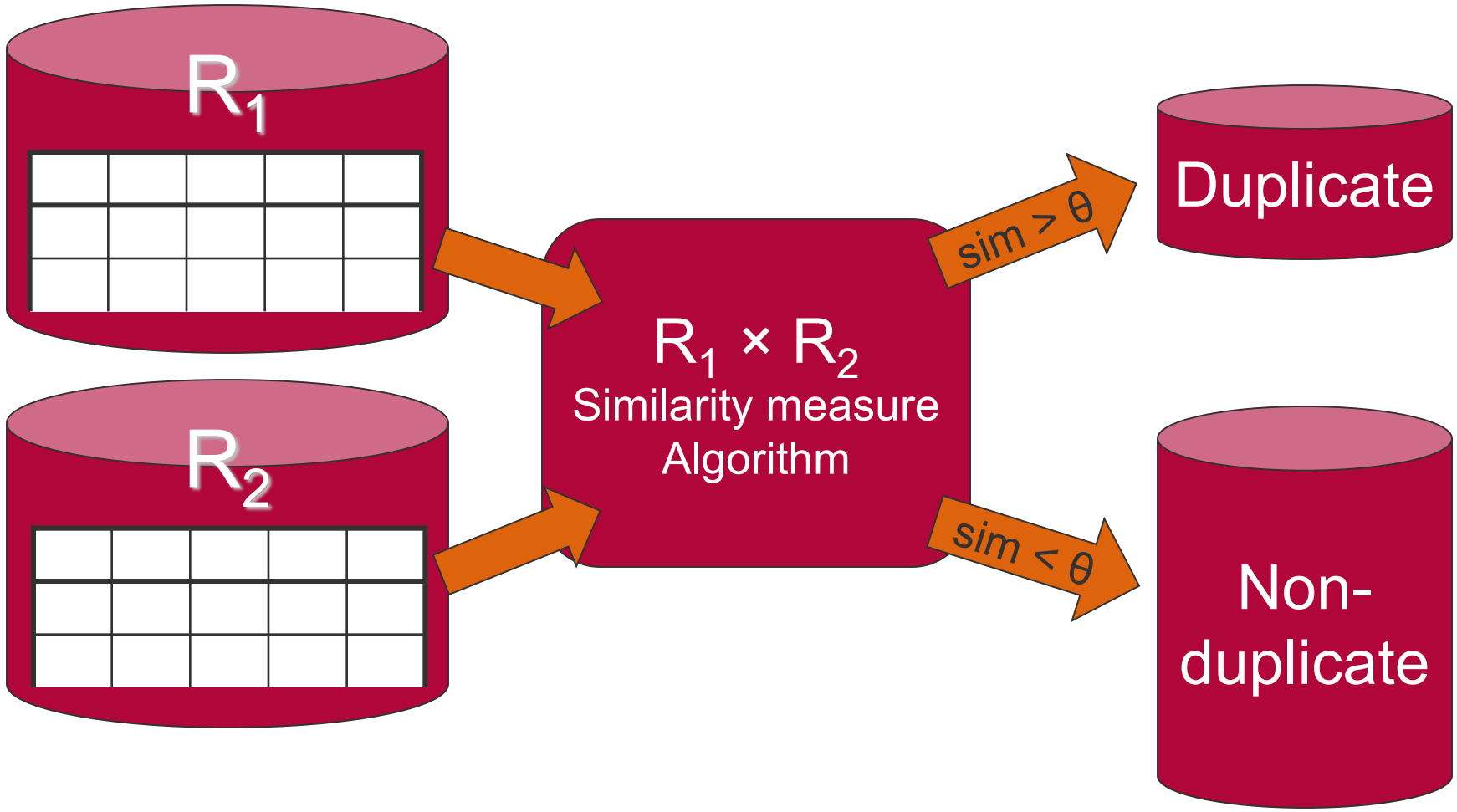
Duplikate / Dubletten

- Duplikate
  - Zwei Datensätze, die dasselbe Objekt repräsentieren
  - Person, Produkt, Ort, Rechnung…
  - Typisch: 10% aller Kundendatensätze sind Duplikate
  - Entstehung: Tippfehler, Datensilos, Betrug, …
- Probleme
  - Kunden unzufrieden
  - Mehrausgaben (Kataloge…)
  - Kreditrisiko

Duplicate Detection

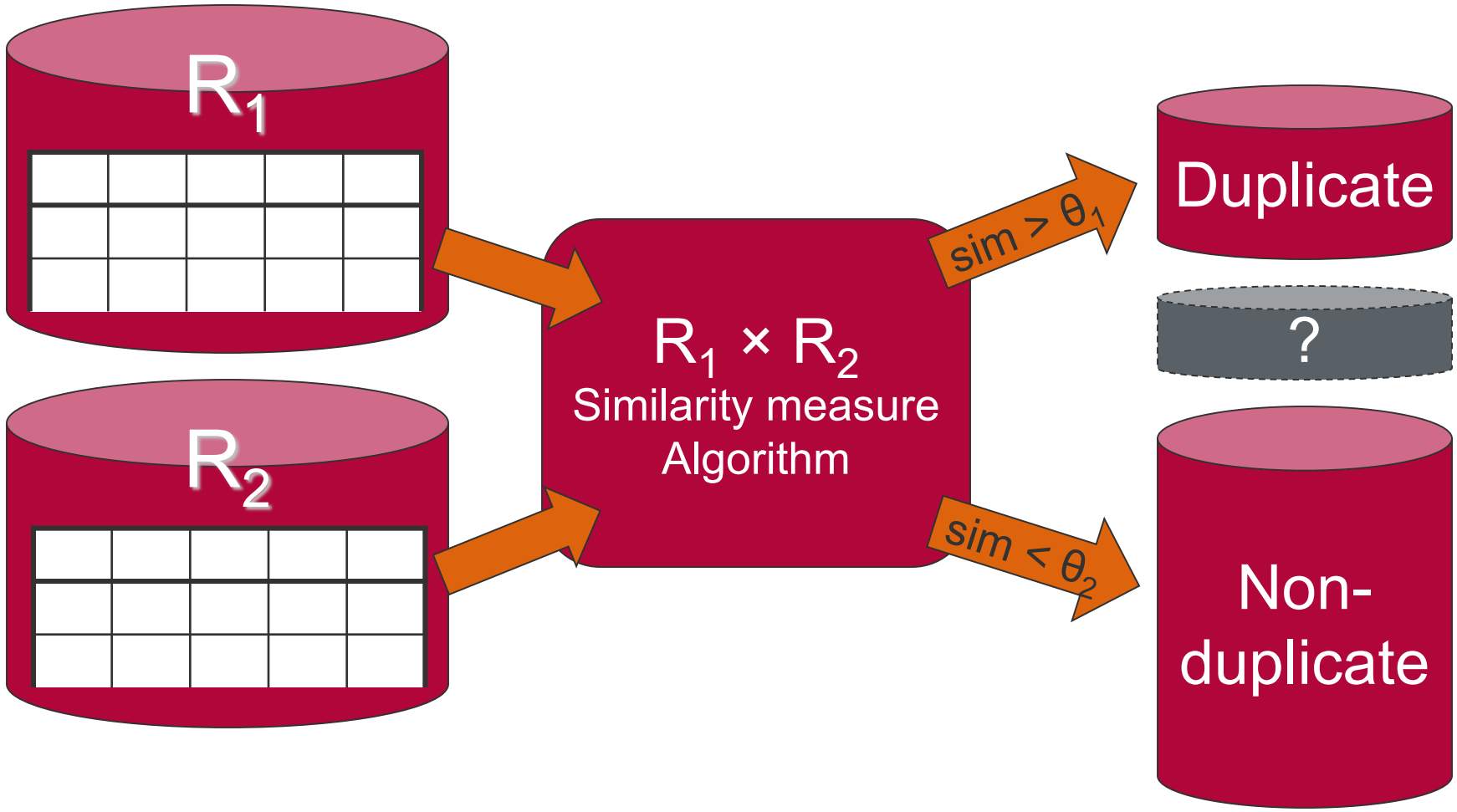
- Duplicate detection is the discovery of multiple representations of the same real-world object.
  - Problem 1: Representations are not identical.
    - *Fuzzy duplicates*
  - Solution: Similarity measures
    - Value- and record-comparisons
    - Domain-dependent or domain-independent
  - Problem 2: Data sets are large.
    - Quadratic complexity: Comparison of every pair of records.
  - Solution: Algorithms
    - E.g., avoid comparisons by partitioning.

Duplicate Detection



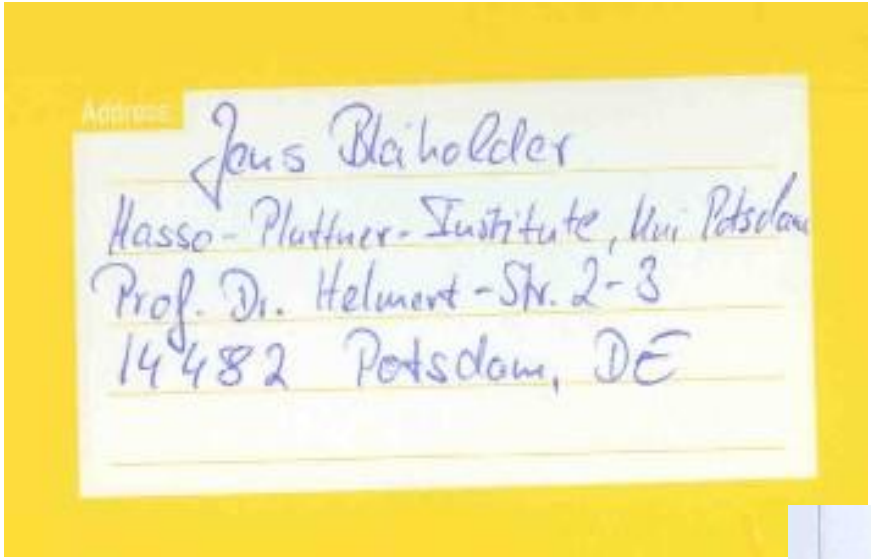
Felix Naumann
Information Integration
Winter 2019/20

Duplicate Detection



Felix Naumann
Information Integration
Winter 2019/20

Origins of duplicates



Original

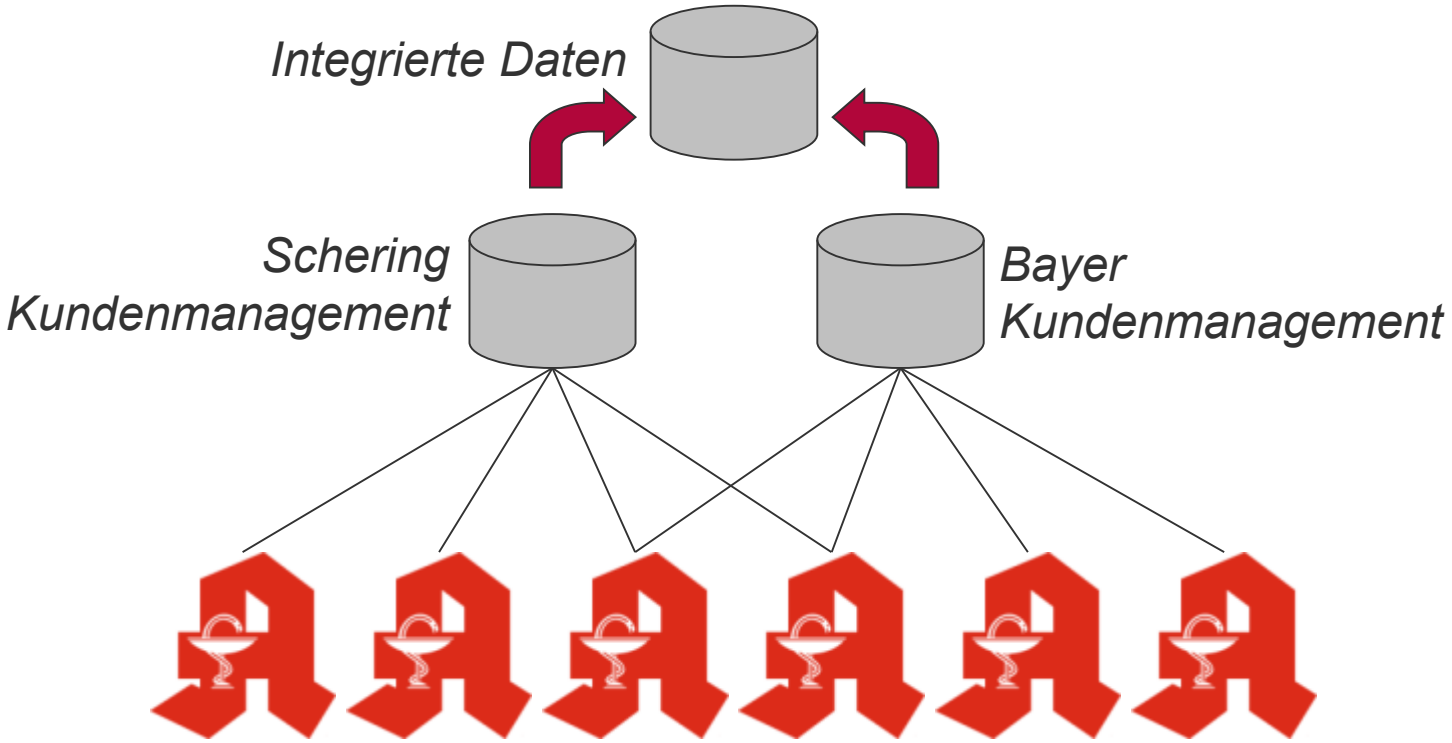
Scanned

Felix Naumann
Information Integration
Winter 2019/20

Origins of duplicates

Felix Naumann
Information Integration
Winter 2019/20

Origins of duplicates



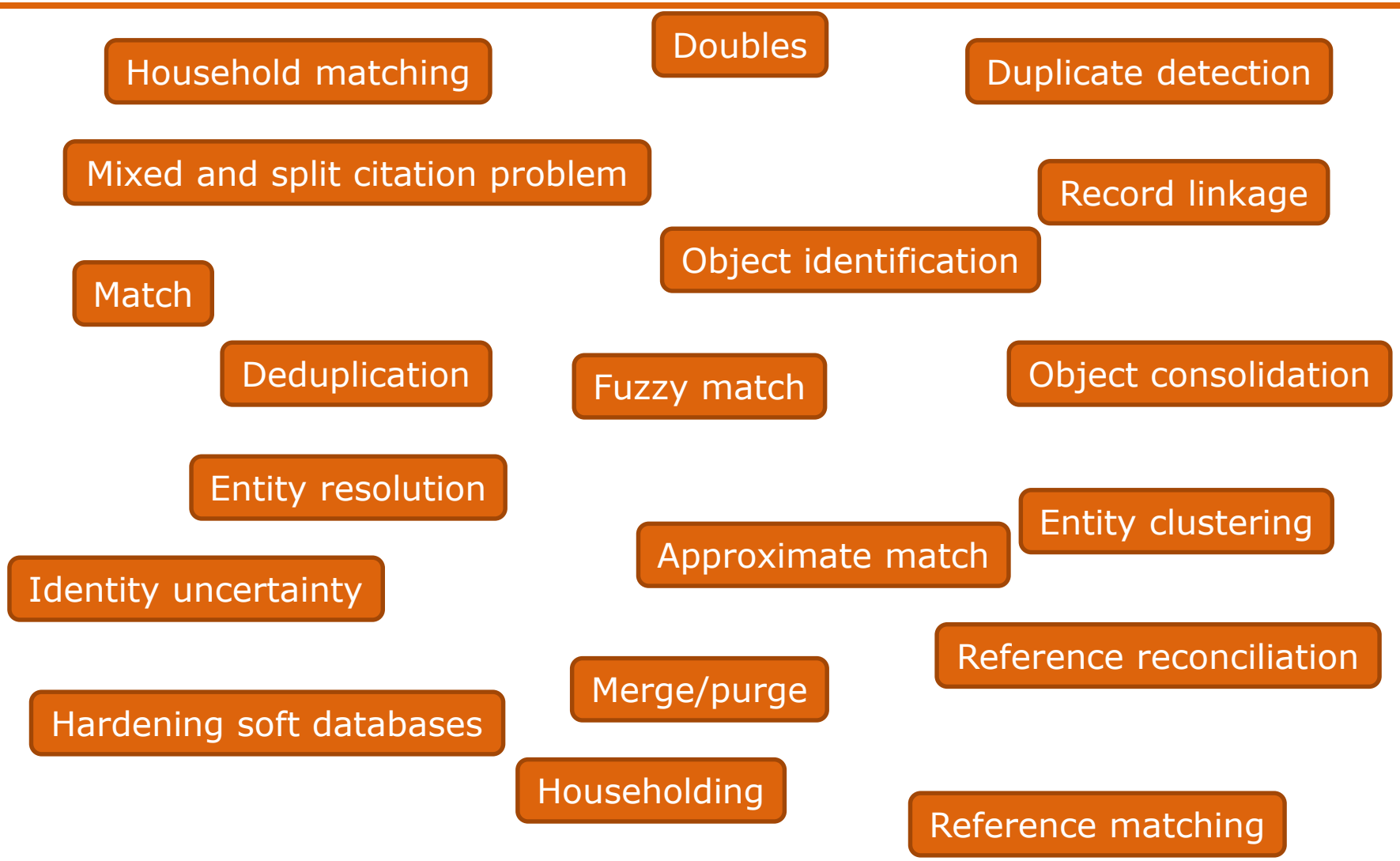
Felix Naumann
Information Integration
Winter 2019/20

Motivation

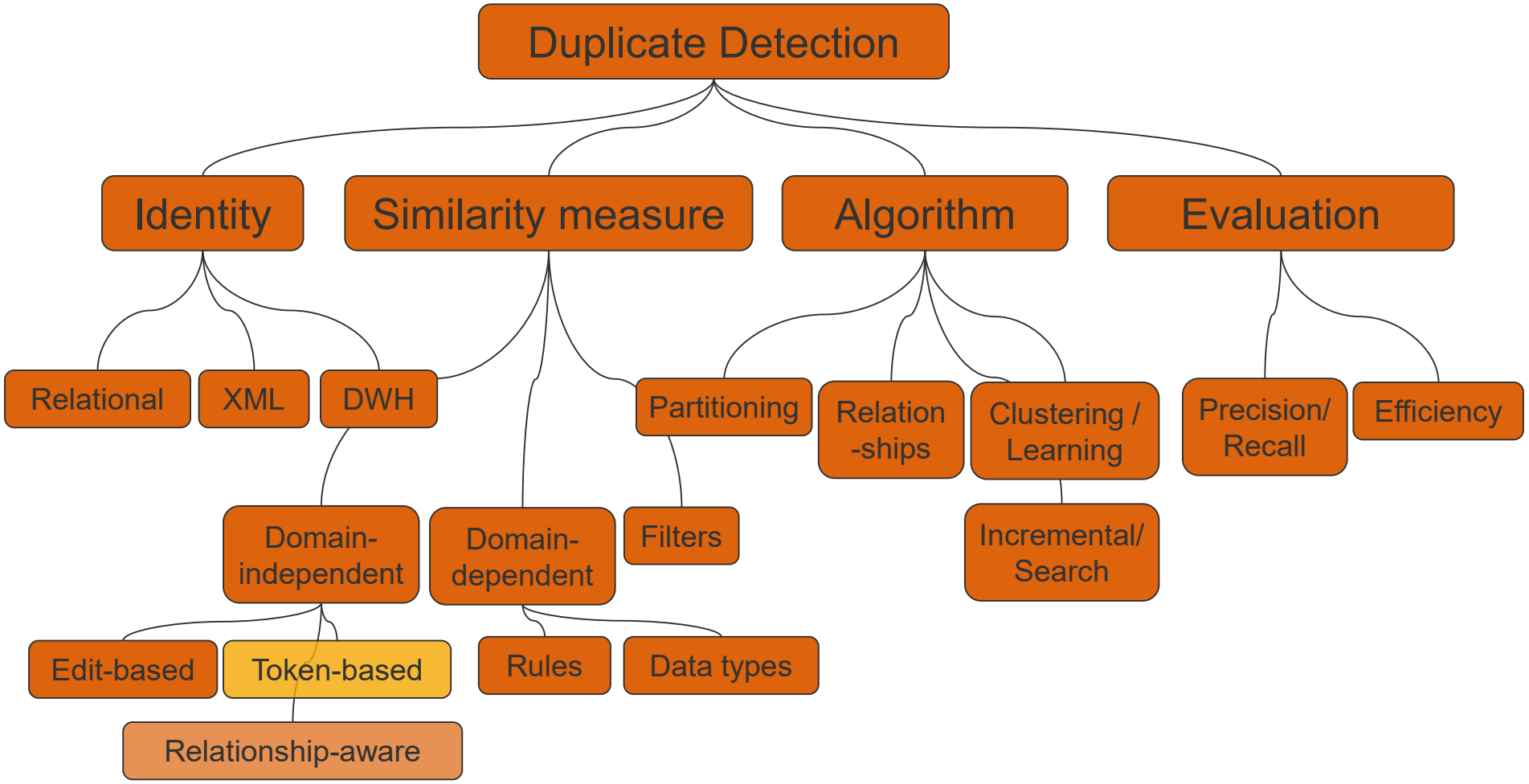
- Possible effects
  - Example: Portfolio Management Offers
  - Credit maximum not detected
  - Too high inventory levels
  - No quantity discount for multiple orders
  - Total revenue of preferred customers unknown
  - Multiple mailings of same catalog to same household
- General problems
  - Additional, unnecessary IT expenses
  - Low customer satisfaction
  - Potentials and dangers not detected
  - Poor quality financial data

Customer	Revenue
BMW	20.000
BaMoWe	5.000.000
Bayerische Motorenwerke	300.000
...	...

Ironically, “Duplicate Detection” has many Duplicates

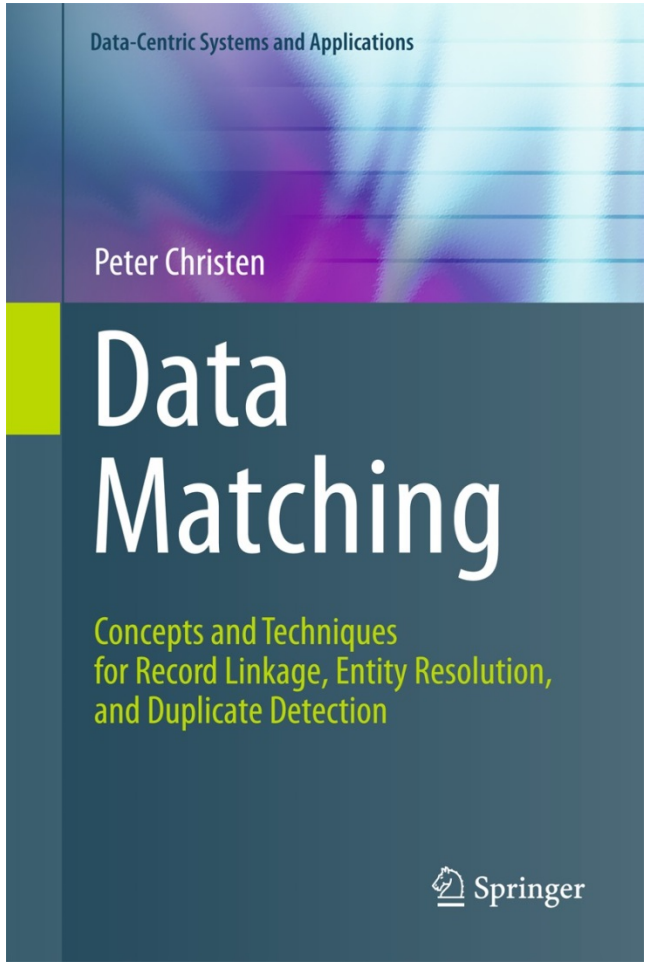


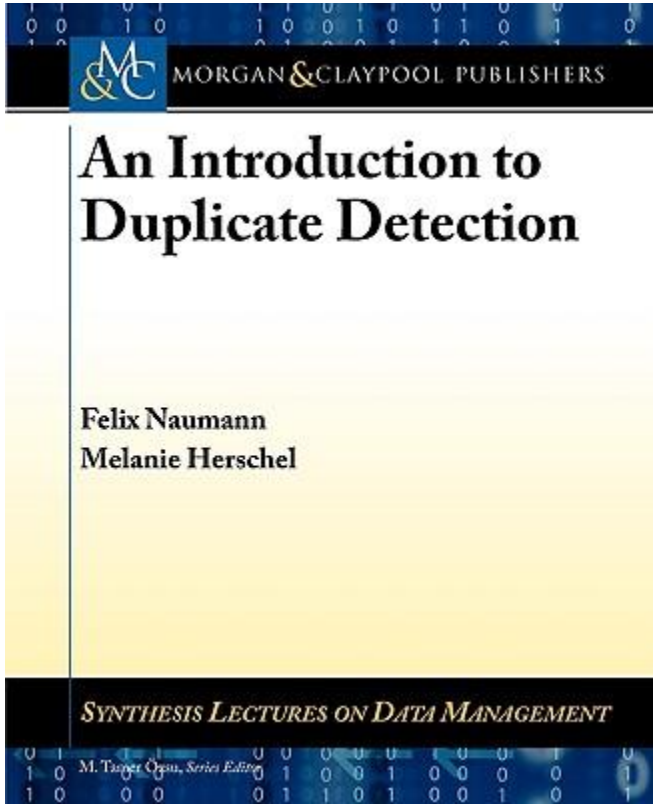
Duplicate Detection – Research



Felix Naumann
Information Integration
Winter 2019/20

Literature


Data-Centric Systems and Applications
Peter Christen
Data Matching
Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection
Springer


MORGAN & CLAYPOOL PUBLISHERS
An Introduction to Duplicate Detection
Felix Naumann
Melanie Herschel
SYNTHESIS LECTURES ON DATA MANAGEMENT

Overview

1. Duplicate detection
2. **Similarity measures**
3. Algorithms
4. Data sets and evaluation
5. Data fusion

Felix Naumann
Information Integration
Winter 2019/20

Ähnlichkeitsmaße

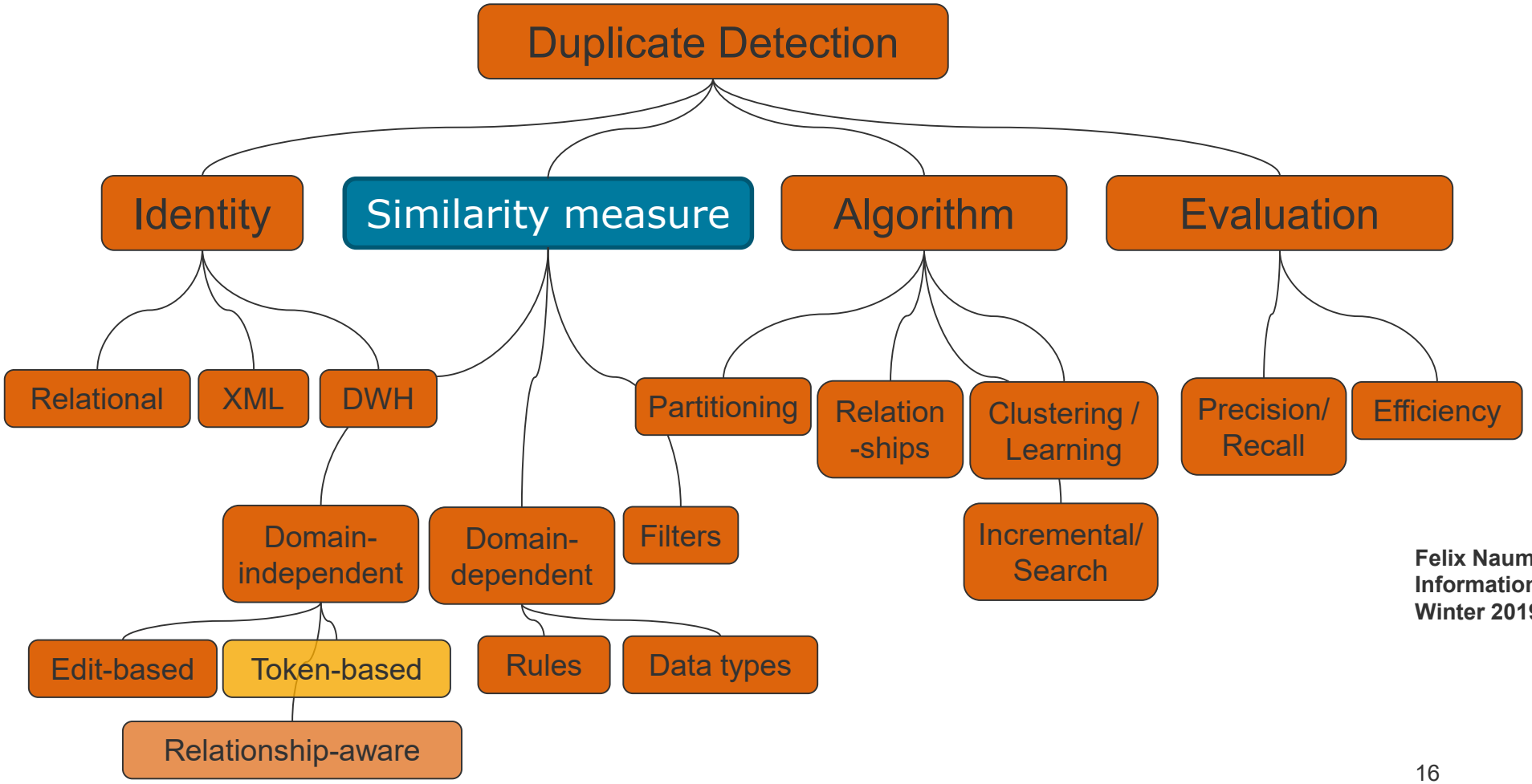
Vorname	Nachname	PLZ	Ort
Felix	Naumann	14482	Potsdam
Felix	Neumann	14440	Postdam
Peter	Müller	82049	Saarbrücken
Peter	Müller	82048	Saarbruecken

Vorname	Nachname	PLZ	Ort
↹⇨↙↓◣	↳⇨▶↘⇨↔↔	↔⇢⇢↕	↻↕◀▽⇦⇨↘
↹⇨↙↓◣	↳⇨▶↘⇨↔↔	↔⇢⇢⇞	↻↕▽◀⇦⇨↘
^↑↕↘⇨▽	↺▯↙↙⇨△	↔↕⇞↦	↺▯↔⇨↑⇨↔
^↑↕↘⇨▽	↺▯↙↙⇨△	↳↳↳↦	↺▶⇨↔⇨↑⇨↔

- Edit-Distanz
  - Minimale Anzahl an Edit-Operationen um ein Wort in das andere umzuwandeln
  - Naumann → Neumann: 1
  - Naumann → Müller: 7
- Häufigkeiten:

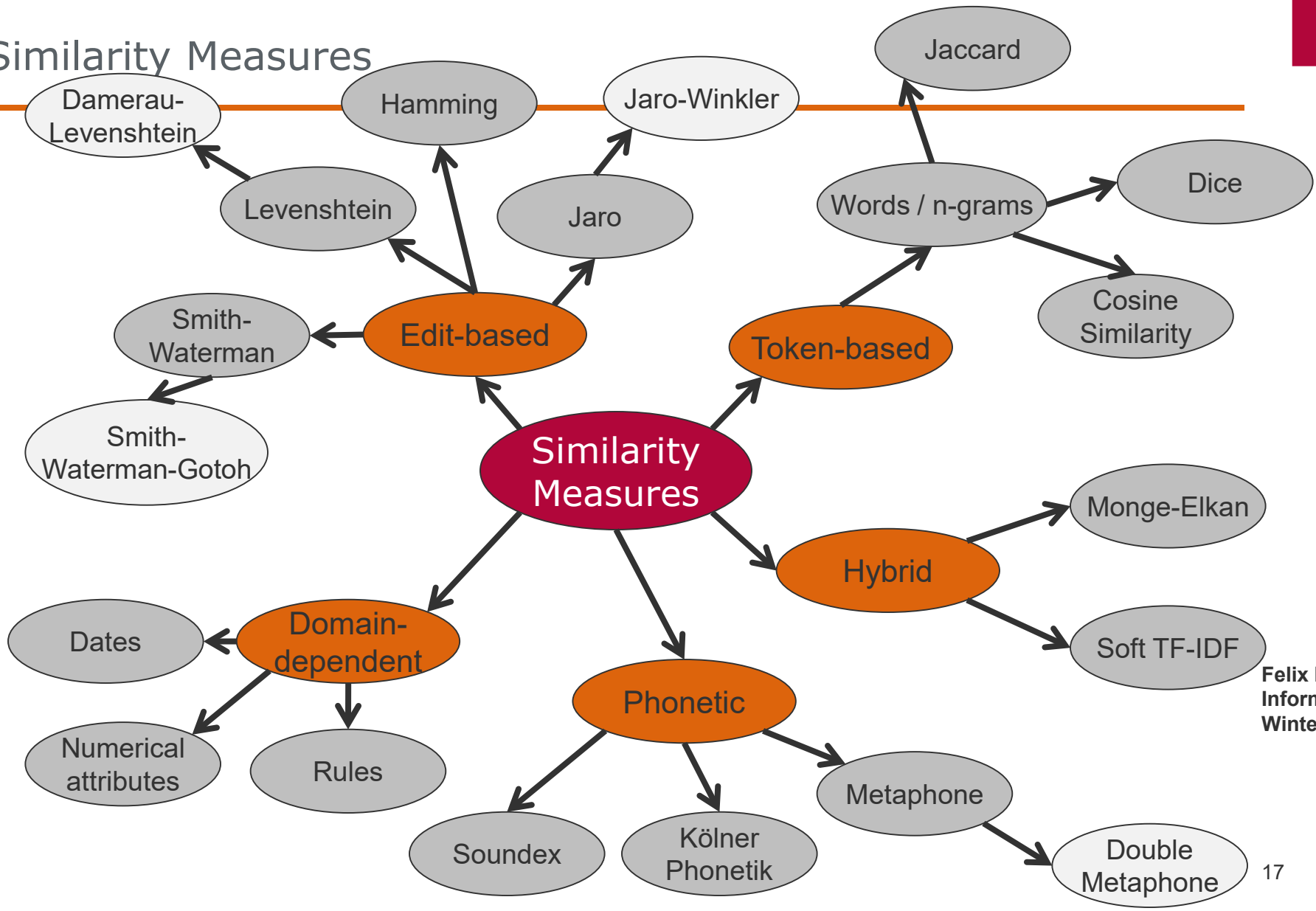
Felix Naumann
Information Integration
Winter 2019/20

Duplicate Detection – Research



Felix Naumann
Information Integration
Winter 2019/20

Overview Similarity Measures



Similarity measures

sim(x,y)

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

Normalized: sim(x,y) $\in$ [0,1]

- sim(x,y) = 1 for exact match
- sim(x,y) = 0 for „completely different" x and y.
- 0 < sim(x,y) < 1 for some approximate similarity

Distance function / distance metric

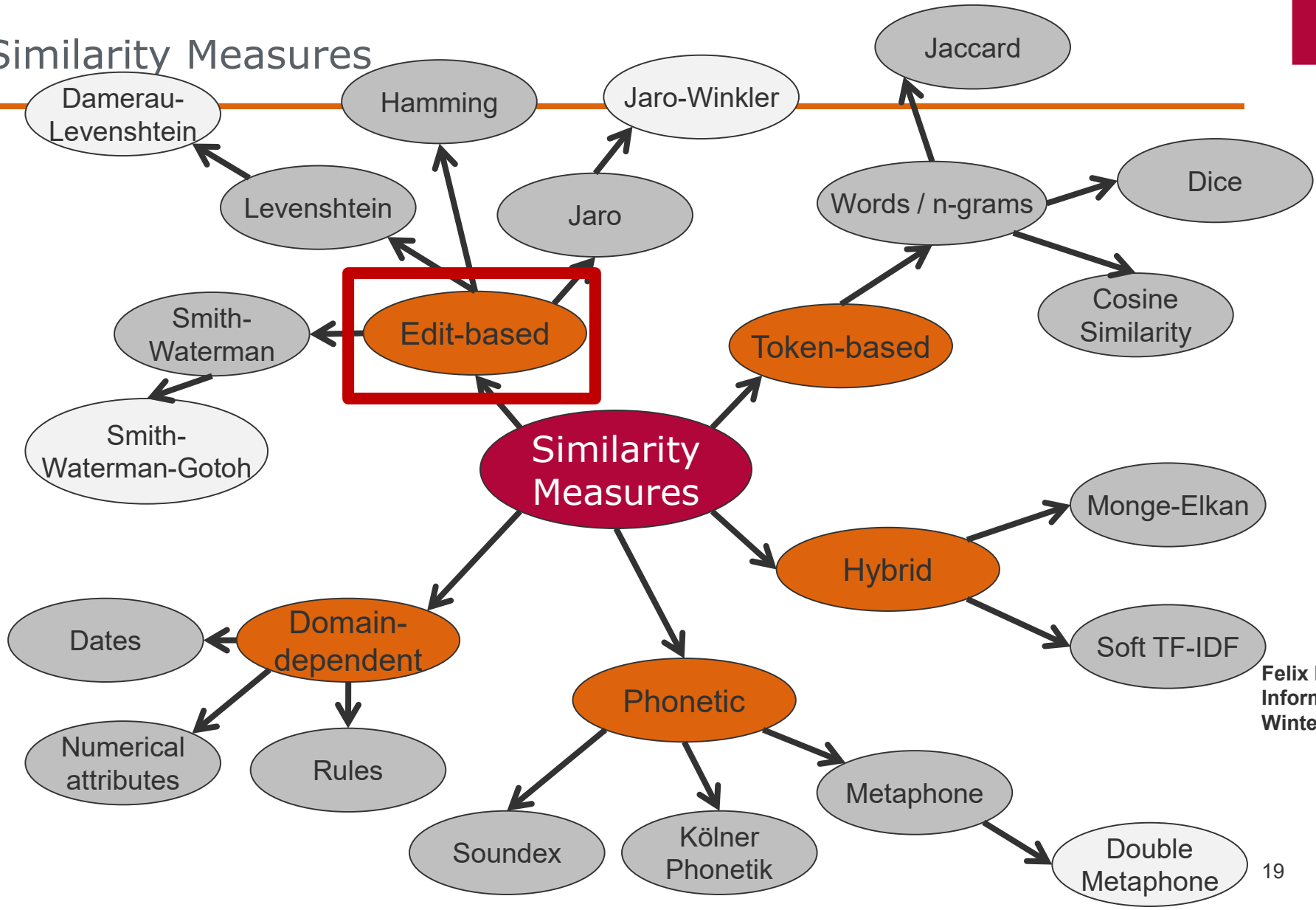
- Reflexive: dist(x,x) = 0
- Positive: dist(x,y) $\geq$ 0
- Symmetric: dist(x,y) = dist(y,x)
- Triangular inequation: dist(x,z) $\leq$ dist(x,y) + dist(y,z)

sim(x,y) = 1 – dist(x,y)

sim(x,y) = 1/dist(x,y)

Felix Naumann
Information Integration
Winter 2019/20

Overview Similarity Measures



Exact and truncated match

$$sim_{exact}(x, y) = \begin{cases} 1 \; if \; x = y \\ 0 \; if \; x \neq y \end{cases}$$

$$sim_{trunc\_beg}(x, y) = \begin{cases} 1 \; if \; x[1:k] = y[1:k] \\ 0 \; if \; x[1:k] \neq y[1:k] \end{cases}$$

$$sim_{trunc\_end}(x, y) = \begin{cases} 1 \; if \; x[k:n] = y[k:n] \\ 0 \; if \; x[k:n] \neq y[k:n] \end{cases}$$

$$sim_{encode}(x, y) = \begin{cases} 1 \; if \; encode(x) = encode(y) \\ 0 \; if \; encode(y) \neq encode(y) \end{cases}$$

- E.g., with a phonetic encoding

Hamming distance

- Number of positions in which two strings (of equal length) differ
  - Minimum number of *substitutions* required to change one string into the other
  - Minimum number of *errors* that could have transformed one string into the other.
- Used mostly for binary numbers and to measure communication errors.
  - Hamming distance = number of 1s in x XOR y.

- $dist_{hamming}$(peter,pedro) = 3

Edit distances

- Compare two strings based on individual characters
- Minimal number of edits required to transform one string into the other.
  - Edits: Insert, Delete, Replace (and Match)
  - Alternative: Smallest edit cost
  - Give different cost to different types of edits
  - Give different cost to different letters
- Naive approach: *editdistance*(Jones,Johnson)
  - DDDDDIIIIII = 12
  - But: Not minimal!

- Levenshtein distance: Basic form
  - Each edit has cost 1

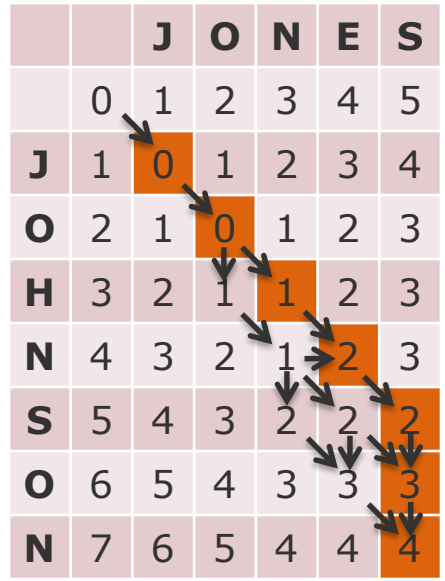
Levenshtein Distance

- Minimum number of character **insertions**, **deletions**, and **replacements** necessary to transform s_1 into s_2
- Compute transcript based on **dynamic programming** algorithm
  - Optimality principle: Best transcript of two substrings must be part of best overall solution
    1. Initialize matrix M of size $(|s_1|+1) \times (|s_2|+1)$
    2. Fill matrix: $M_{i,0} = i$ and $M_{0,j} = j$
    3. Recursion: $M_{i,j} = \begin{cases} M_{i-1,j-1} & if\ x[i] = y[j] \\ 1 + \min(M_{i-1,j}, M_{i,j-1}, M_{i-1,j-1}) & else \end{cases}$
    4. Distance: $LevenshteinDist(x, y) = M_{|x|,|y|}$
- Levenshtein Similarity: $sim_{Levenshtein}(x, y) = 1 - \frac{LevenshteinDist(x,y)}{\max(|x|,|y|)}$

Levenshtein Distance

		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					



$$M_{i,j} = \begin{cases} M_{i-1,j-1} & if\ x[i] = y[j] \\ 1 + \min\left(M_{i-1,j}, M_{i,j-1}, M_{i-1,j-1}\right) & else \end{cases}$$

Felix Naumann
Information Integration
Winter 2019/20

Levenshtein Distance – Example

$$sim_{Levenshtein} = 1 - \frac{LevenshteinDist}{\max(|s_1|, |s_2|)}$$

s_1	s_2	Levenshtein Distance	$sim_{Levenshtein}$
Jones	Johnson	4	0.43
Paul	Pual	2	0.5
Paul Jones	Jones, Paul	11	0

Levenshtein discussion

- Complexity
  - Time: $O(|x| \cdot |y|)$ (fill in matrix)
  - Space: $O(\min(|x|, |y|))$
    - Trick: Store only two rows of the matrix
- Some properties
  - $0 \leq LevenshteinDist(x, y) \leq \max(|x|, |y|)$
  - $| \, |x| - |y| \, | \leq LevenshteinDist(x, y)$
    - Often: Compare only strings with similar lengths
- Other cost models
  - Insert, delete cost 1.0 but replace 0.5
    - change in string length is punished, e.g. for zip codes
  - Character based: OCR (m $\simeq$ n, 1 $\simeq$ l) or keyboard (a $\simeq$ s) or brain (6 $\simeq$ 9) or biology (a $\simeq$ t)

Damerau–Levenshtein distance

Similar to Levenshtein distance, but additionally considers transposed characters

$M_{i,0} = i$ and $M_{0,j} = j$

$$M_{i,j} = \begin{cases} M_{i-1,j-1} & if x[i] = y[j] \\ 1 + min \begin{pmatrix} M_{i-1,j}, M_{i,j-1}, \\ M_{i-1,j-1}, \\ M_{i-2,j-2} \; if \; x[i] = y[j-1] \; and \; x[i-1] = y[j] \end{pmatrix} & else \end{cases}$$

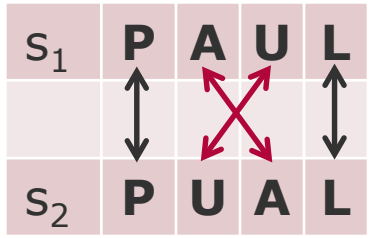
s_1	s_2	Levenshtein Distance	Damerau-Levenshtein Distance	$sim_{Damerau-Levenshtein}$
Jones	Johnson	4	4	0.43
Paul	Pual	2	1	0.75
Paul Jones	Jones, Paul	11	11	0

Jaro similarity

- Specifically designed for names at US Census Bureau
- Search for common characters
- m : number of matching characters
  - Search range matching characters: $\frac{\max(|x|,|y|)}{2} - 1$
- t : number of transpositions
- $sim_{jaro} = \frac{1}{3}\left(\frac{m}{|x|} + \frac{m}{|y|} + \frac{m-t}{m}\right)$

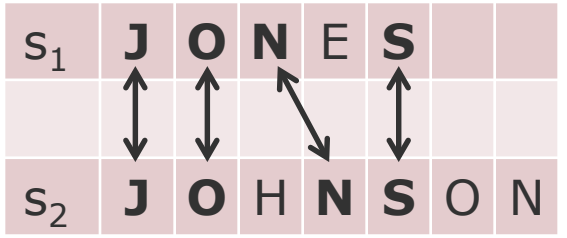
Jaro similarity – Example

$$sim_{jaro} = \frac{1}{3}\left(\frac{m}{|x|} + \frac{m}{|y|} + \frac{m-t}{m}\right)$$



$$m = 4 \qquad t = \frac{2}{2} = 1$$

$$sim_{jaro} = \frac{1}{3} \cdot \left(\frac{4}{4} + \frac{4}{4} + \frac{4-1}{4}\right) \approx 0.92$$



$$m = 4 \qquad t = \frac{0}{2} = 0$$

$$sim_{jaro} = \frac{1}{3} \cdot \left(\frac{4}{5} + \frac{4}{7} + \frac{4-0}{4}\right) \approx 0.79$$

Winkler similarity

- Intuition 1: Similarity of first few letters is most important.
- Let p be the length of the common prefix of x and y.
- $sim_{winkler}(x,y) = sim_{jaro}(x,y) + (1 - sim_{jaro}(x,y))\frac{p}{10}$
  - = 1 if common prefix is ≥ 10

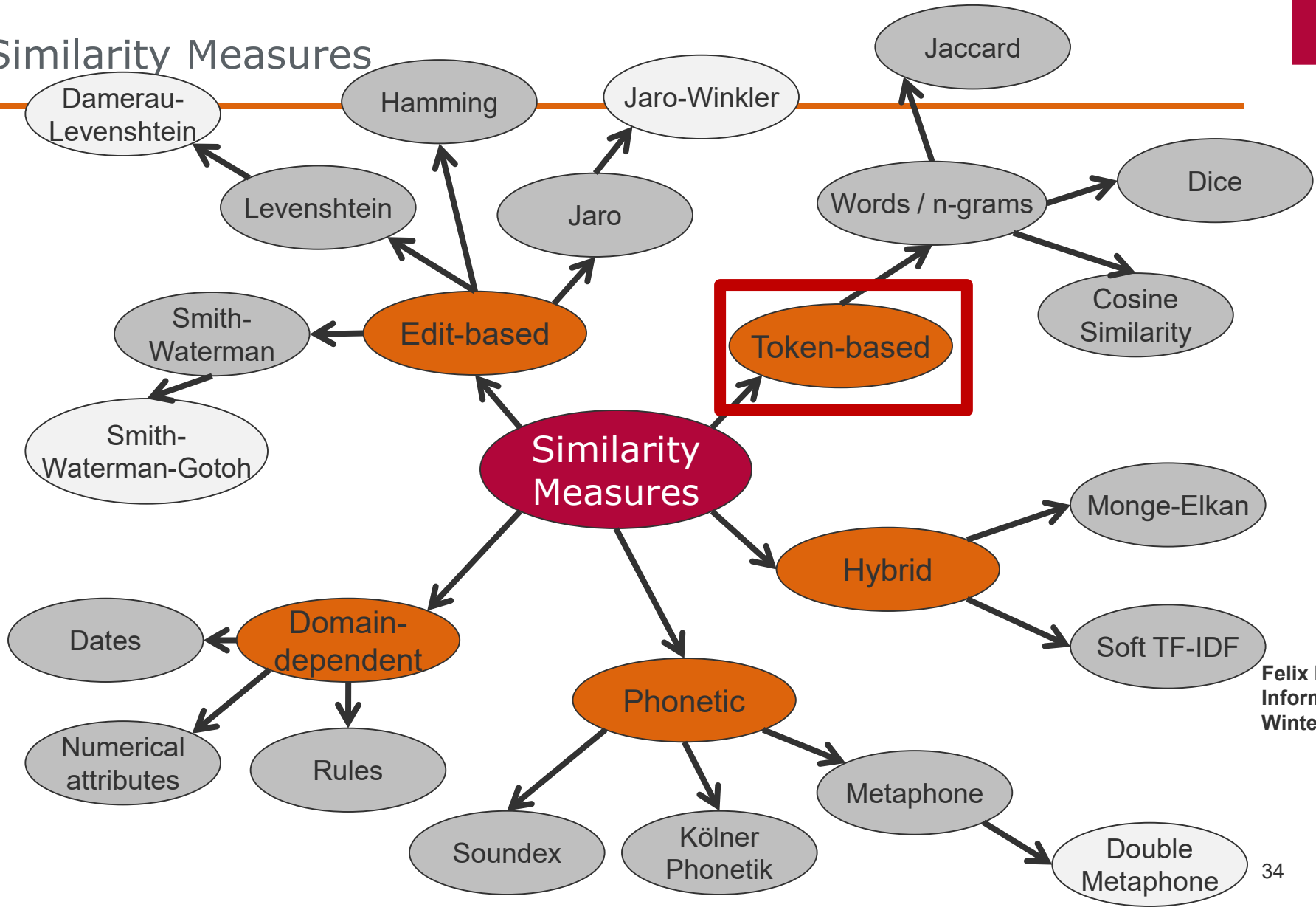
- Intuition 2: Longer strings with even more common letters
- $sim_{winkler\_long}(x,y) = sim_{winkler}(x,y) + (1 - sim_{winkler}(x,y))\frac{c-(p+1)}{|x|+|y|-2(p-1)}$
  - Where c is overall number of common letters and p is common prefix
  - Apply only if
    - Long strings (both at least 5 letters): $\min(|x|,|y|) \geq 5$
    - At least two additional common letters besides prefix: $c - p \geq 2$
    - At least half remaining letters of shorter string are in common: $c - p \geq \frac{\min(|x|,|y|)-p}{2}$

Felix Naumann
Information Integration
Winter 2019/20

Comparison

String 1	String 2	c	t	p	c_{sim}	sim_{jaro}	$sim_{winkler}$	$sim_{winkler\_long}$
shackleford	**shackelford**	11	1	4	0	0.9697	0.9818	0.9886
nichleson	**nichulson**	8	0	4	0.3	0.9259	0.9556	0.9667
jones	**johnson**	4	0	2	0.3	0.7905	0.8324	0.8491
massey	**massie**	5	0	4	0.3	0.8889	0.9333	–
jeraldine	**geraldine**	8	0	0	0.3	0.9259	0.9259	0.9519
michelle	**michael**	6	0	4	0.3	0.8690	0.9214	0.9302

Overview Similarity Measures

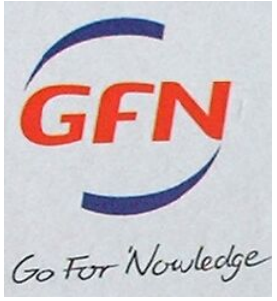


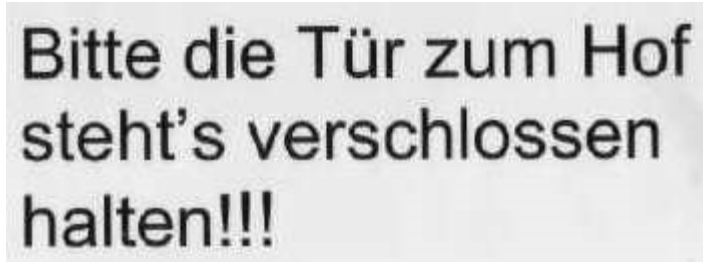
Tokenization

- Forming words from sequence of characters
- General idea: Separate string into tokens using some separator
  - Space, hyphen, punctuation, special characters
  - Usually also convert to lower-case
- Problems
  - Both hyphenated and non-hyphenated forms of many words are common
    - Sometimes hyphen is *not needed: e-bay, wal-mart, active-x, cd-rom, t-shirts*
    - Sometimes hyphens *should be considered* either as part of the word or a word separator: *winston-salem, mazda rx-7, e-cards, pre-diabetes, t-mobile, spanish-speaking*
  - Numbers can be important, including decimals
    - *nokia 3250, top 10 courses, united 93, quicktime 6.5 pro, 92.3 the beat, 288358*
  - Periods can occur in numbers, abbreviations, URLs, ends of sentences, and other situations
    - *I.B.M., Ph.D., cs.umass.edu, F.E.A.R.*
  - Apostrophes can be a part of a word, a part of a possessive, or just a mistake
    - *rosie o'donnell, can't, don't, 80's, 1890's, men's straw hats, master's degree, england's ten largest cities, shriner's*

Die Kapostropheum-Gruselgalerie – Kategorie „Völlig willenlos“

http://www.apostroph.de/





Felix Naumann
Information Integration
Winter 2019/20

n-grams (aka q-grams)

- Split string into short substrings of length *n*.
  - Sliding window over string
  - *n*=2: Bigrams
  - *n*=3: Trigrams
  - Variation: Pad with *n* − 1 special characters
    - Emphasizes beginning and end of string
  - Variation: Include positional information to weight similarities
- Number of *n*-grams = $|x| - n + 1$
- Count how many n-grams are common in both strings

String	Bigrams	Padded bigrams	Positional bigrams	Trigrams
gail	ga, ai, il	⊙g, ga, ai, il, l⊗	(ga,1), (ai,2), (il,3)	gai, ail
gayle	ga, ay, yl, le	⊙g, ga, ay, yl, le, e⊗	(ga,1), (ay,2), (yl,3), (le,4)	gay, ayl, yle
peter	pe, et, te, er	⊙p, pe, et, te, er, r⊗	(pe,1), (et,2), (te,3), (er,4)	pet, ete, ter
pedro	pe, ed, dr, ro	⊙p, pe, ed, dr, ro, o⊗	(pe,1), (ed,2), (dr,3), (ro,4)	ped, edr, dro

Token-based Similarity Measures

- Token similarity
  - Overlap coefficient: $sim_{overlap}(x,y) = \frac{|tok(x) \cap tok(y)|}{\min(|tok(x)|,|tok(y)|)}$
  - Jaccard coefficient:

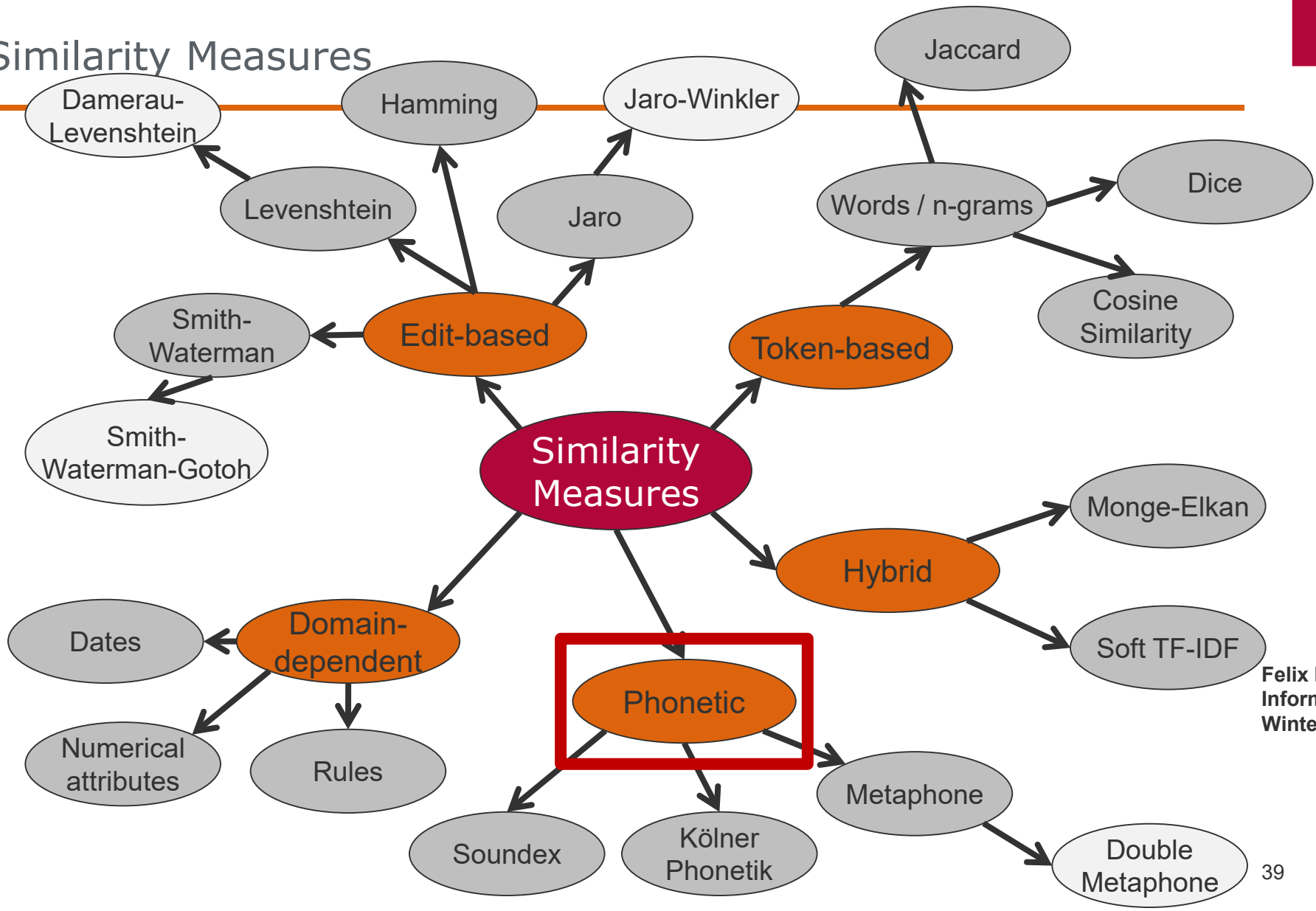
$$sim_{jaccard}(x,y) = \frac{|tok(x) \cap tok(y)|}{|tok(x)| + |tok(y)| - |tok(x) \cap tok(y)|} = \frac{|tok(x) \cap tok(y)|}{|tok(x) \cup tok(y)|}$$

  - Dice's coefficient: $sim_{dice}(x,y) = \frac{2 \cdot |tok(x) \cap tok(y)|}{|tok(x)| + |tok(y)|}$
- Tokens („Paul Jones“)
  - Words / Terms („Paul“ „Jones“)
  - Padded n-grams (_P, Pa, au, ul, l_, _J, Jo, on, ne, es, s_)

Mengenoperationen: Multimengensemantik

s_1	s_2	Jaccard	Dice
Jones	Johnson	0.17	0.29
Paul	Pual	0.33	0.40
Paul Jones	Jones, Paul	0.77	0.87

Overview Similarity Measures



Soundex

- Soundex codes a last name based on the way a last name sounds
  1. Retain first letter of the name and drop all other occurrences of A, E, H, I, O, U, W, Y
  2. Replace consonants with digits
  3. Two adjacent letters with the same number are coded as a single number
  4. Continue until you have one letter and three numbers. If you run out of letters, pad with 0s.
- If a surname has a prefix, such as Van, Con, De, Di, La, or Le, code both with and without the prefix

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

- Example
  - PAUL: P400
  - PUAL: P400
  - JONES: J520
  - JOHNSON: J525

Jenkins, Jansen, Jameson

Soundex on WolframAlpha

Kölner Phonetik

- Like Soundex, but specialized for German last names
- Letters get different codes based on the context
- Code length is not restricted
- Multiple occurrences of the same code and „0“ are removed

- Example
  - PAUL: 15
  - PUAL: 15
  - JONES: 68
  - JOHNSON: 686

Letter	Context	Code
A, E, I, J, O, U, Y		0
H		-
B		1
P	not before H	
D, T	not before C, S, Z	2
F, V, W		3
P	before H	
G, K, Q		4
C	in the initial sound before A, H, K, L, O, Q, R, U, X	
	before A, H, K, O, Q, U, X but not after S, Z	
X	not after C, K, Q	48
L		5
M, N		6
R		7
S, Z		8
C	after S, Z	
	in the initial sound, but not before A, H, K, L, O, Q, R, U, X	
	not before A, H, K, O, Q, U, X	
D, T	before C, S, Z	
X	after C, K, Q	

Metaphone

- Improves on the Soundex algorithm
  - Knows variations and inconsistencies in English spelling and pronunciation

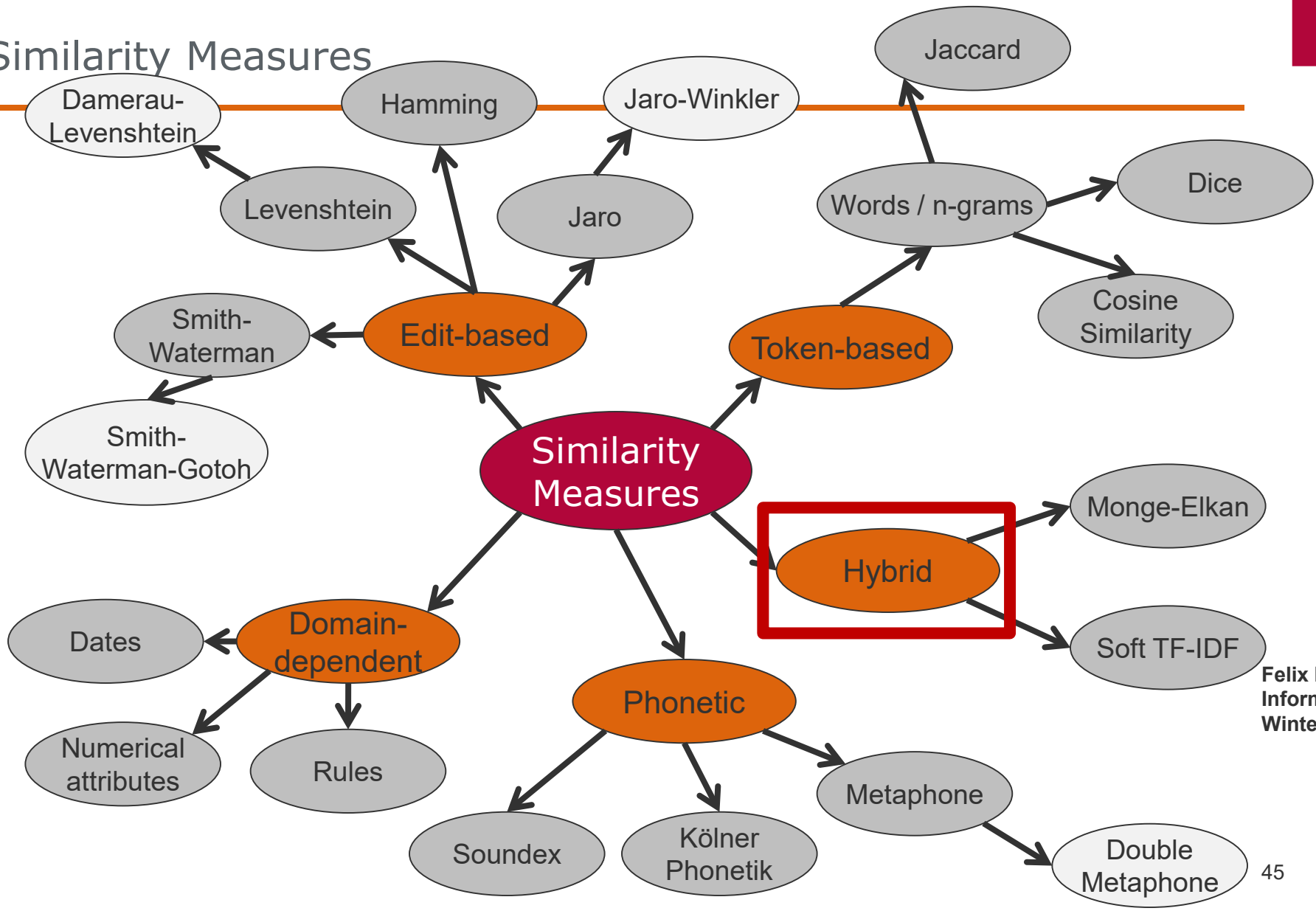
- Further improvements
  - Double Metaphone
    - Includes other languages: Slavic, Germanic, Celtic, Greek, French, Italian, Spanish, Chinese
    - Accuracy 89%
  - Metaphone 3
    - Accuracy over 99% (says author)

Original Metaphone Algorithm

- 16 consonant symbols 0BFHJKLMNPRSTWXY
  - '0' represents "th", 'X' represents "sh" or "ch"

1. Drop duplicate adjacent letters, except for C.
2. If the word begins with 'KN', 'GN', 'PN', 'AE', 'WR', drop the first letter.
3. Drop 'B' if after 'M' at the end of the word.
4. 'C' transforms to 'X' if followed by 'IA' or 'H' (unless in latter case, it is part of '-SCH-', in which case it transforms to 'K'). 'C' transforms to 'S' if followed by 'I', 'E', or 'Y'. Otherwise, 'C' transforms to 'K'.
5. 'D' transforms to 'J' if followed by 'GE', 'GY', or 'GI'. Otherwise, 'D' transforms to 'T'.
6. Drop 'G' if followed by 'H' and 'H' is not at the end or before a vowel. Drop 'G' if followed by 'N' or 'NED' and is at the end.
7. 'G' transforms to 'J' if before 'I', 'E', or 'Y', and it is not in 'GG'. Otherwise, 'G' transforms to 'K'.
8. Drop 'H' if after vowel and not before a vowel.
9. 'CK' transforms to 'K'.
10. 'PH' transforms to 'F'.
11. 'Q' transforms to 'K'.
12. 'S' transforms to 'X' if followed by 'H', 'IO', or 'IA'.
13. 'T' transforms to 'X' if followed by 'IA' or 'IO'. 'TH' transforms to '0'. Drop 'T' if followed by 'CH'.
14. 'V' transforms to 'F'.
15. 'WH' transforms to 'W' if at the beginning. Drop 'W' if not followed by a vowel.
16. 'X' transforms to 'S' if at the beginning. Otherwise, 'X' transforms to 'KS'.
17. Drop 'Y' if not followed by a vowel.
18. 'Z' transforms to 'S'.
19. Drop all vowels unless it is the beginning

Overview Similarity Measures



Monge-Elkan

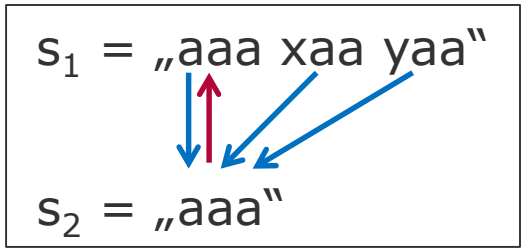
- Hybrid: Token-based and internal similarity function for tokens
  - Find best match for each token

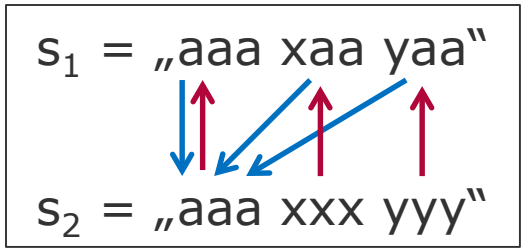
- $sim_{MongeElkan}(x, y) = \frac{1}{|x|}\sum_{i=1}^{|x|} \max_{j=1,|y|} sim'(x[i], y[j])$
  - $|x|$ is number of tokens in x
  - *sim'* is internal similarity function (e.g., Levenshtein)
- If strings contain just one token each
  - $sim_{MongeElkan}(x, y) = sim'(x, y)$

- Complexity: Quadratic in number of tokens

Monge-Elkan – Example

- $sim_{MongeElkan}(x,y) = \frac{1}{|x|}\sum_{i=1}^{|x|} \max_{j=1,|y|} sim'(x[i],y[j])$
- Peter Christen vs. Christian Pedro
  - sim_{jaro}(peter, christian) = 0.3741
  - sim_{jaro}(peter, pedro) = 0.7333
  - sim_{jaro}(christen, christian) = 0.8843
  - sim_{jaro}(christen, pedro) = 0.4417
- $sim_{MongeElkan}('peter\ christen','christian\ pedro') = \frac{1}{2}(0.7333 + 0.8843) = 0.8088$
- $sim_{MongeElkan}(x,y) \neq sim_{MongeElkan}(y,x)$





- Monge-Elkan is **not** symmetric:

Extended Jaccard Similarity

- $sim_{jaccard}(x,y) = \frac{|tok(x) \cap tok(y)|}{|tok(x)|+|tok(y)|-|tok(x) \cap tok(y)|} = \frac{|tok(x) \cap tok(y)|}{|tok(x) \cup tok(y)|}$

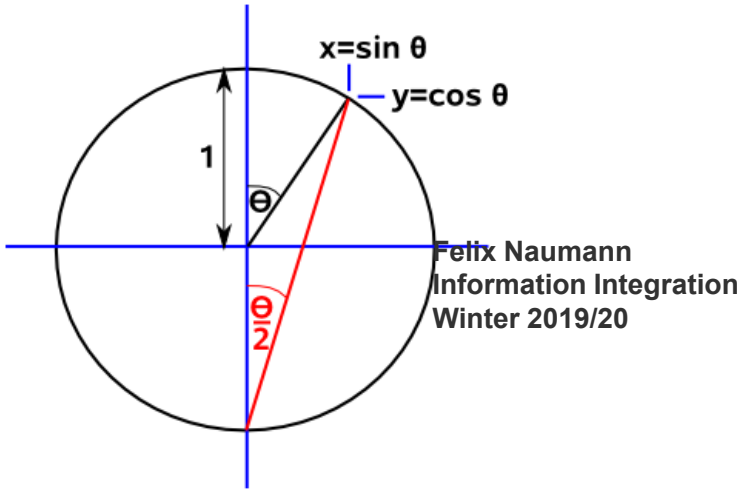
- If strings contain multiple words, choose words as tokens.
- Use internal similarity function to calculate similarity between all pairs of tokens.
  - Shared tokens: $S = \{(x_i, y_j) | \, x_i \in tok(x) \wedge y_j \in tok(y) : sim'(x_i, y_j) \geq \theta\}$
  - Unique tokens: $U_{tok(x)} = \{x_i | x_i \in tok(x) \wedge y_j \in tok(y) \wedge (x_i, y_j) \notin S\}$

- $sim_{jaccard\_ext}(x,y) = \frac{|S|}{|S|+|U_{tok(x)}|+|U_{tok(y)}|}$

Vector Space Model

- Each document ranked by distance between points representing query and document
- Popular measure: Cosine similarity
  - Cosine of angle between document and query vectors
  - Normalized dot-product

$$Cosine(D_i, Q) = \frac{\sum_{j=1}^{t} d_{ij} \cdot q_j}{\sqrt{\sum_{j=1}^{t} {d_{ij}}^2 \cdot \sum_{j=1}^{t} {q_j}^2}}$$



http://www.euclideanspace.com/math/geometry/trig/derived/index.htm

Similarity Calculation – Example

- Consider three documents D_1, D_2, D_3 and query Q
  - D_1 = (0.5, 0.8, 0.3), D_2 = (0.9, 0.4, 0.2), D_3 = (0, 0.9, 0.1)
  - Q = (1.5, 1.0, 0)
- Vector space model reflects some term weights and number of matching terms (in contrast to Boolean retrieval)

$$Cosine(D_1, Q) = \frac{(0.5 \times 1.5) + (0.8 \times 1.0)}{\sqrt{(0.5^2 + 0.8^2 + 0.3^2)(1.5^2 + 1.0^2)}}$$

$$= \frac{1.55}{\sqrt{(0.98 \times 3.25)}} = 0.87$$

$$Cosine(D_2, Q) = \frac{(0.9 \times 1.5) + (0.4 \times 1.0)}{\sqrt{(0.9^2 + 0.4^2 + 0.2^2)(1.5^2 + 1.0^2)}}$$

$$= \frac{1.75}{\sqrt{(1.01 \times 3.25)}} = 0.97$$

- But: How to assign

Cosine(D_3,Q) = 0.55

Term Weights – *tf.idf*

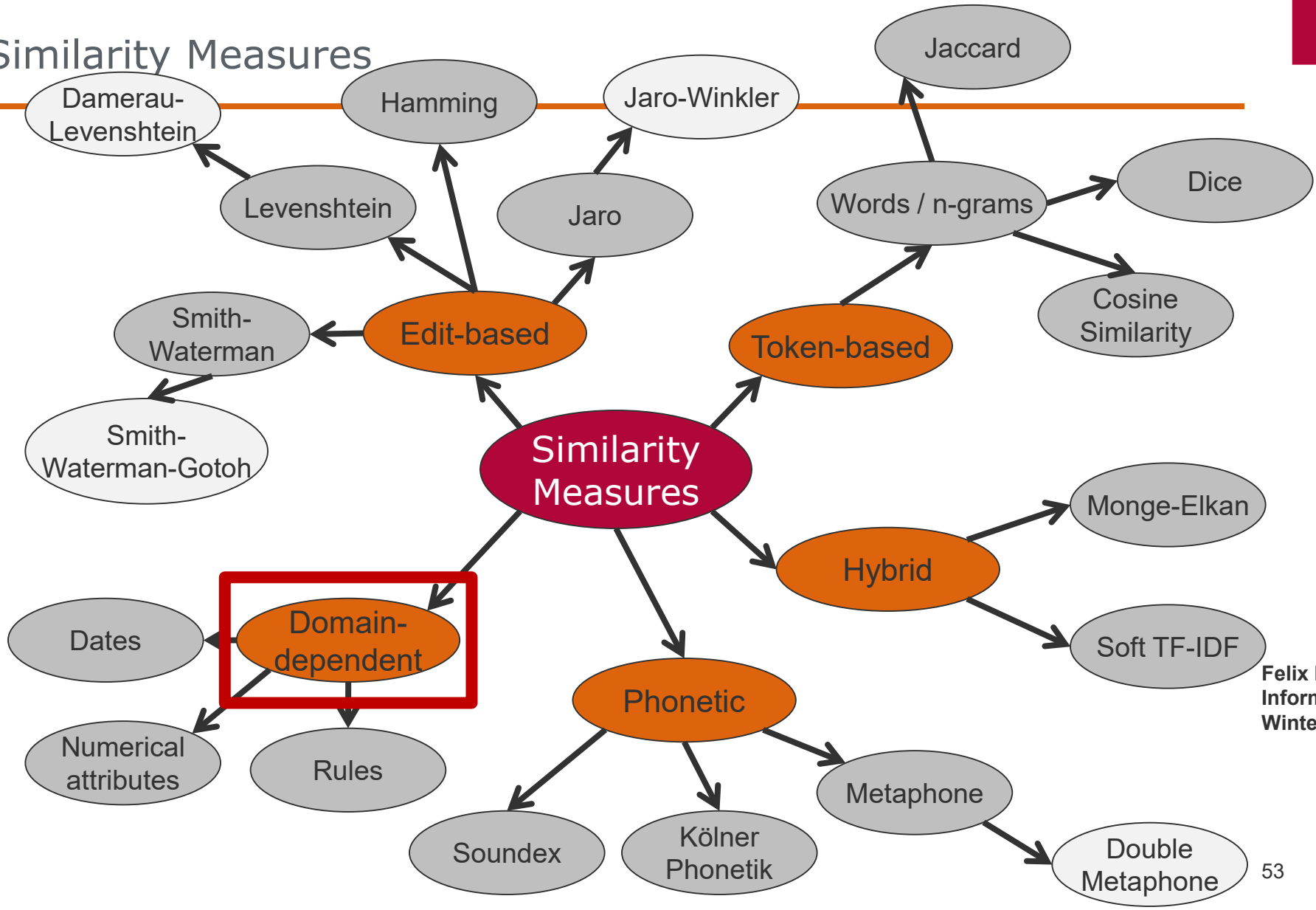
- Term frequency weight *tf* measures importance of term k in document *i*: $tf_{ik} = \frac{f_{ik}}{\sum_{j=1}^{t} f_{ij}}$
  - *log(f_{ik})* to reduce this impact of frequent words
- Inverse document frequency *idf* measures importance in collection: $idf_k = log \frac{N}{n_k}$
  - Reflects "amount of information" carried by term
- *tfidf* by multiplying *tf* and *idf* with some heuristic modifications

Felix Naumann
Information Integration
Winter 2019/20

SoftTFIDF

- Apply idea to records or values: Much shorter than documents
  - $CLOSE(\theta, tok(x), tok(y))$ is set of tokens from x that have at least one sufficiently similar token in y.
- $sim_{softtfidf}(x, y) = \sum_{t \in CLOSE(\theta, tok(x), tok(y))} V\big(t, tok(x)\big) \cdot V(t, tok(y)) \cdot N(t, tok(y))$
  - $V\big(t, tok(x)\big)$ is TFIDF weight of token t in all tokens of x
  - $N\big(t, tok(y)\big) = \max(\{sim'\big(t, y_j\big) | y_j \in tok(y)\})$
    - Similarity of best matching token
- Soft: Tokens are considered a partial match if they get a good score using an internal similarity measure (CLOSE).
- Problem: Weights are calculated over entire database
  - Scan all data
  - Store weight for each unique token

Overview Similarity Measures



Numerical comparison

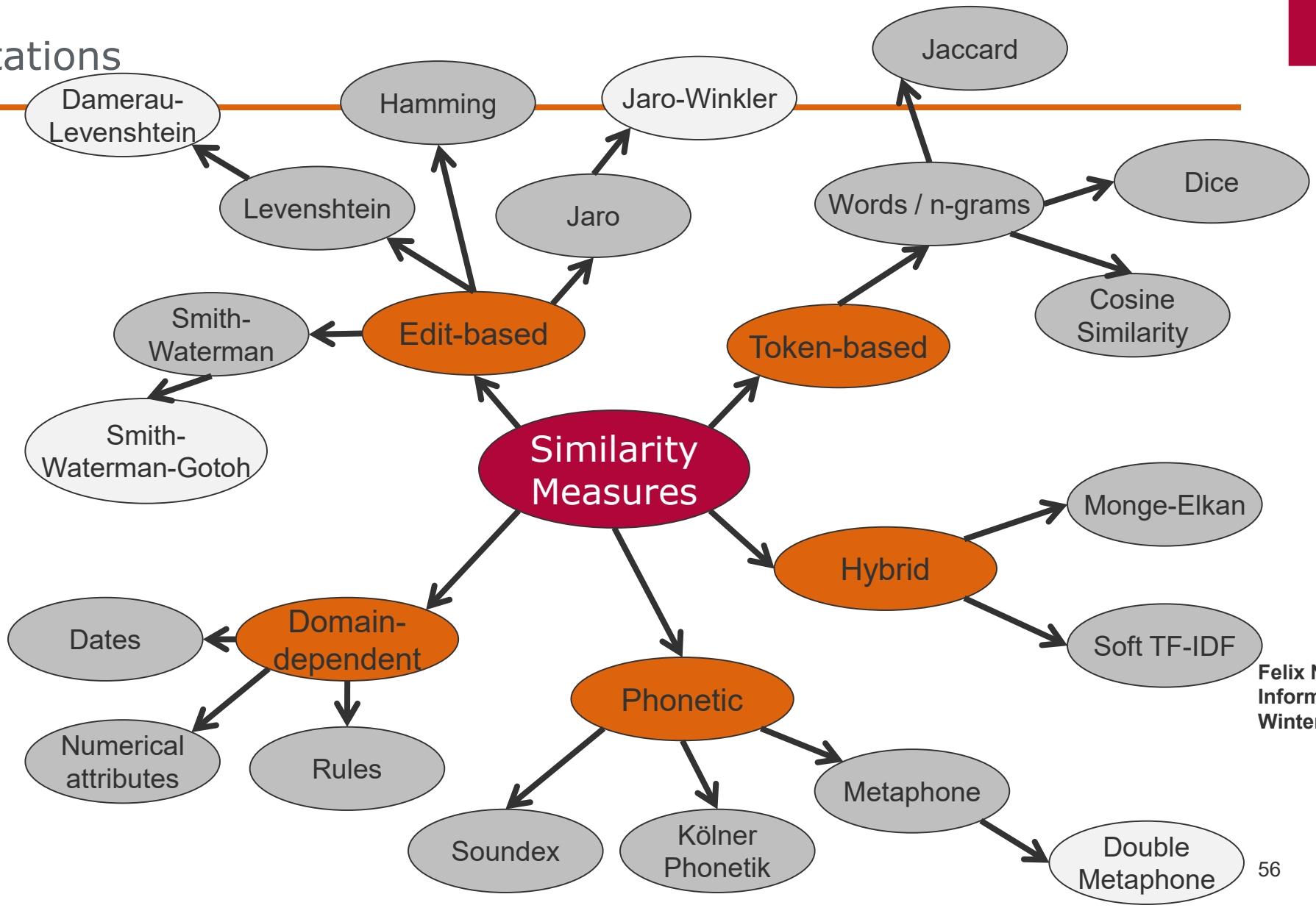
- $sim_{num\_abs}(n,m) = \begin{cases} 1 - \left(\frac{|n-m|}{d_{max}}\right) & if\ |n-m| < d_{max} \\ 0 & else \end{cases}$
  - Linear extrapolation between 0 and d_{max}
- Example:
  - d_{max} = \$1,000
  - $sim_{num\_abs}(2{,}000,\ 2{,}500) = 1 - \frac{500}{1{,}000} = 0.5$
  - $sim_{num\_abs}(200{,}000,\ 200{,}500) = 1 - \frac{500}{1{,}000} = 0.5$
- $sim_{num\_perc}(n,m) = \begin{cases} 1 - \left(\frac{pc}{pc_{max}}\right) & if\ pc < pc_{max} \\ 0 & else \end{cases}$
  - $pc = \frac{|n-m|}{\max(|n|,|m|)} \cdot 100$ is percentage difference
  - pc_{max} = 33%
  - $sim_{num\_perc}(2{,}000,\ 2{,}500) = 1 - \frac{20}{33} = 0.394$ because $pc = \frac{|2{,}000-2{,}500|}{2{,}500} \cdot 100 = 20$
  - $sim_{num\_perc}(200{,}000,\ 200{,}500) = 1 - \frac{0{,}25}{33} = 0.993$ because $pc = \frac{500}{200{,}500} \cdot 100 = 0{,}25\%$

Felix Naumann
Information Integration
Winter 2019/20

Time and space comparisons

- Calculate difference in days and use sim_{num_abs}
- Special cases
  - Swapped day and month and both ≤ 12: Return some fixed similarity, e.g. 0.5
  - Single error in month could exceed d_{max}: Return some fixed similarity, e.g., 0.75
- Dates of birth can be converted to age (num days or num years)
  - Then apply numerical measures
  - $$sim_{age\_perc}(n,m) = \begin{cases} 1 - \left(\frac{apc}{apc_{max}}\right) & if\ apc < apc_{max} \\ 0 & else \end{cases}$$
  - $apc = \frac{|n-m|}{\max(|n|,|m|)} \cdot 100$ is percentage difference
- Geographical data: Compute distance based on some projection

Implementations



Similarity function packages

- SecondString
  - Java classes
  - All basic string comparisons
  - MongeElkan, SoftTFIDF
  - Similarity learner
  - http://sourceforge.net/projects/secondstring/
- SimMetrics
  - Java package
  - All basic string comparisons
  - Long sequences: Needleman-Wunsch, Smith-Waterman, Smith-Waterman-Gotoh
  - http://sourceforge.net/projects/simmetrics/
- Geographiclib for geographic similarity
  - http://geographiclib.sourceforge.net

Overall Similarity Measures

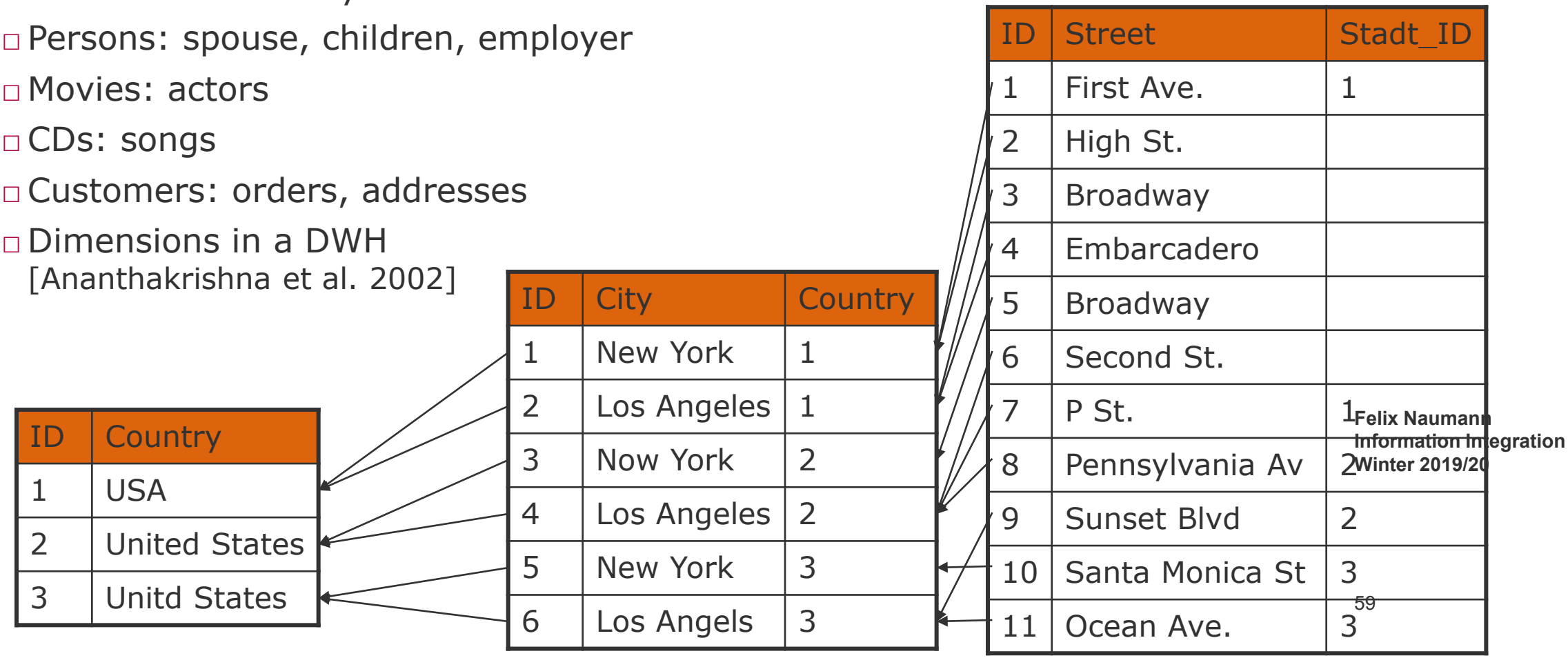
- Weighted average
  - Determine similarity for each attribute (typically no more than 10)
    - Each in [0, 1]
  - Determine weights per attribute
    - Sum of weights = 1
  - Calculate weighted average (also in [0, 1])
  - Compare with threshold

Useful java packages
- SecondString (http://secondstring.sourceforge.net/)
- SimMetrics (https://github.com/Simmetrics/simmetrics)

- Rules
  - [Hernandez Stolfo 1998], [Lee et al. 2000]
  - **Given two records, r1 and r2.**
    IF last name of r1 = last name of r2,
    AND first names differ slightly,
    AND address of r1 = address of r2
    THEN r1 is equivalent to r2.

Relationship-aware Similarity Measures

- Idea: Not only values of the records, but values of related records are relevant for similarity.
  - Persons: spouse, children, employer
  - Movies: actors
  - CDs: songs
  - Customers: orders, addresses
  - Dimensions in a DWH
    [Ananthakrishna et al. 2002]

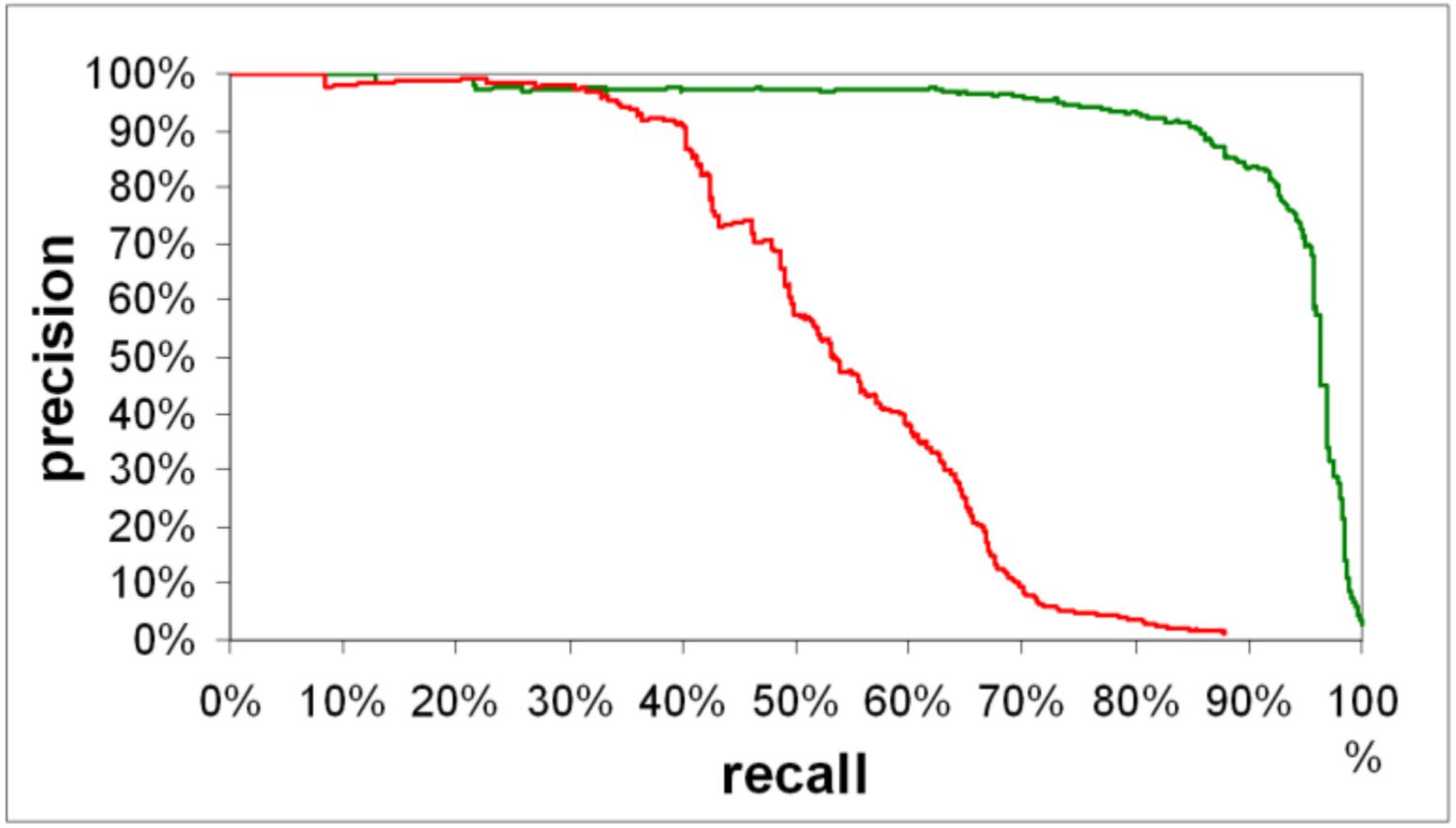


ID	Country
1	USA
2	United States
3	Unitd States

ID	City	Country
1	New York	1
2	Los Angeles	1
3	Now York	2
4	Los Angeles	2
5	New York	3
6	Los Angels	3

ID	Street	Stadt_ID
1	First Ave.	1
2	High St.	
3	Broadway	
4	Embarcadero	
5	Broadway	
6	Second St.	
7	P St.	1
8	Pennsylvania Av	2
9	Sunset Blvd	2
10	Santa Monica St	3
11	Ocean Ave.	3

Relationship-aware Similarity Measures – Evaluation



Felix Naumann
Information Integration
Winter 2019/20

On choosing thresholds

- Notoriously difficult problem
- Calculate similarity for all pairs of records
  - Or for a carefully chosen subset with at least many duplicates
- Sort pairs descendingly by their similarity
- Quickly browse list top to bottom to find first few non-duplicates
- Carefully continue to browse until you find more non-duplicates than duplicates
- That similarity is a good threshold

Overview

1. Duplicate detection
2. Similarity measures
3. **Algorithms**
4. Data sets and evaluation
5. Data fusion

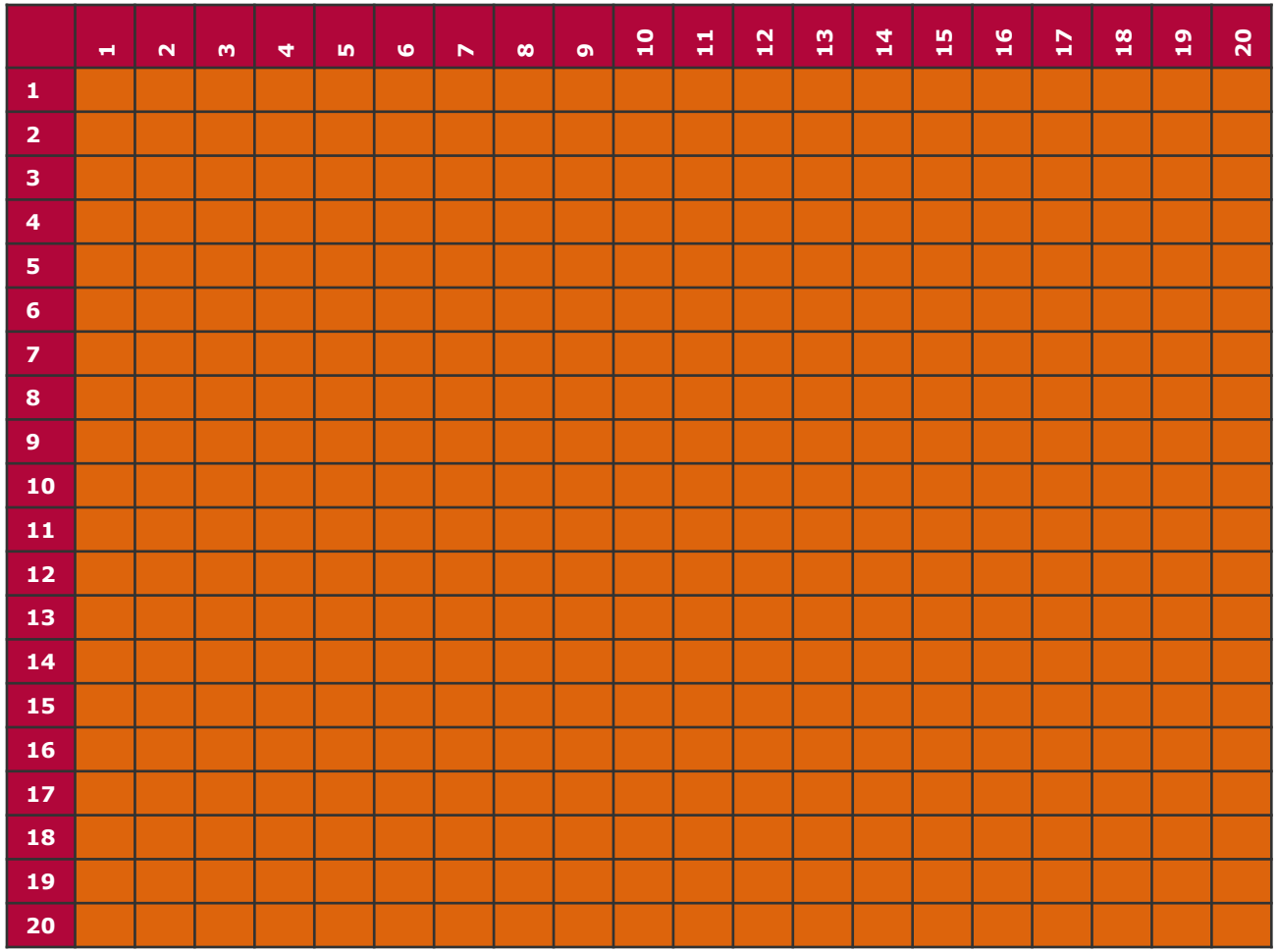
Felix Naumann
Information Integration
Winter 2019/20

Record Pairs as Matrix

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1																				
2																				
3																				
4																				
5																				
6																				
7																				
8																				
9																				
10																				
11																				
12																				
13																				
14																				
15																				
16																				
17																				
18																				
19																				
20																				

Felix Naumann
Information Integration
Winter 2019/20

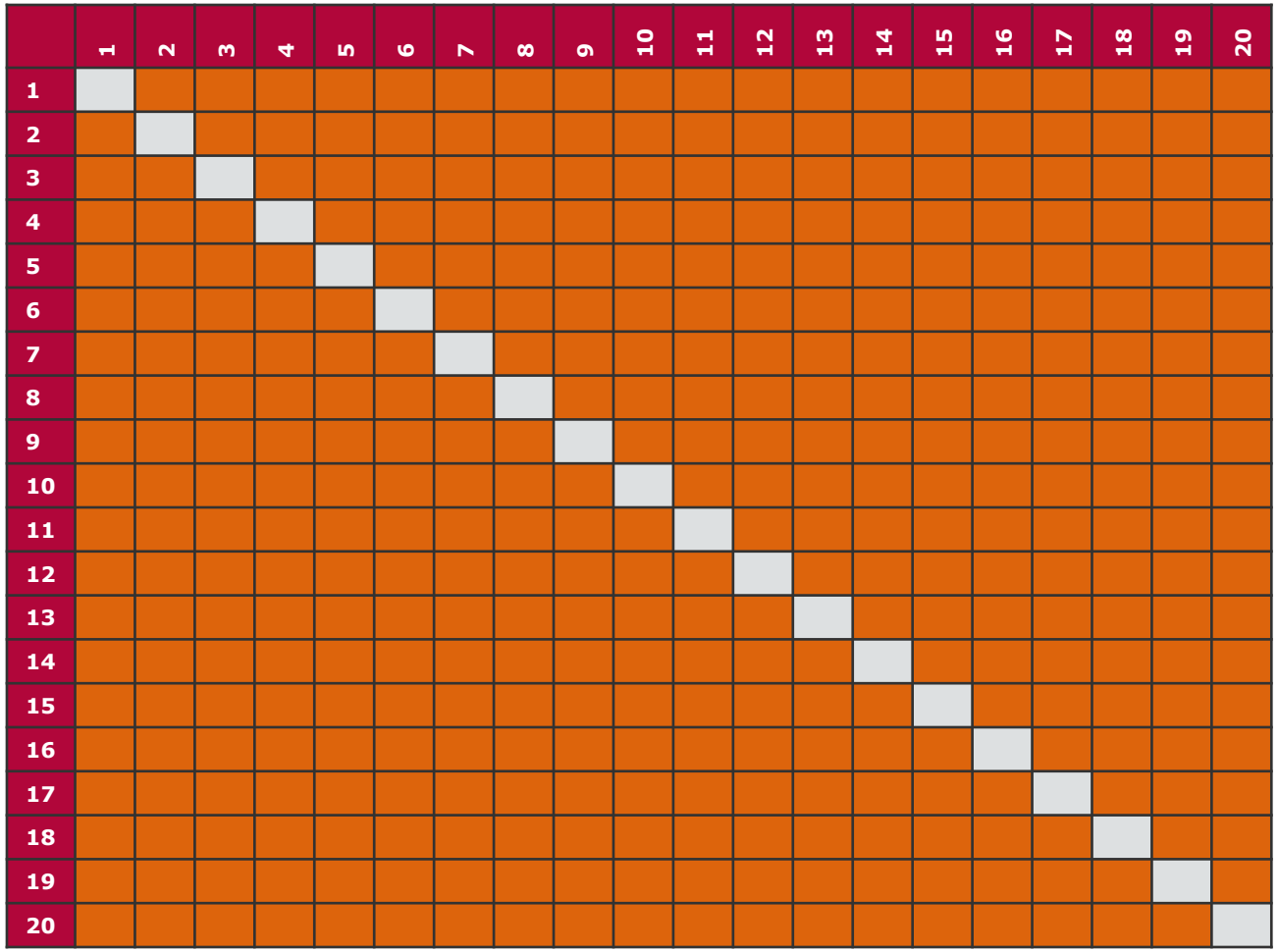
Number of comparisons: All pairs



400 comparisons

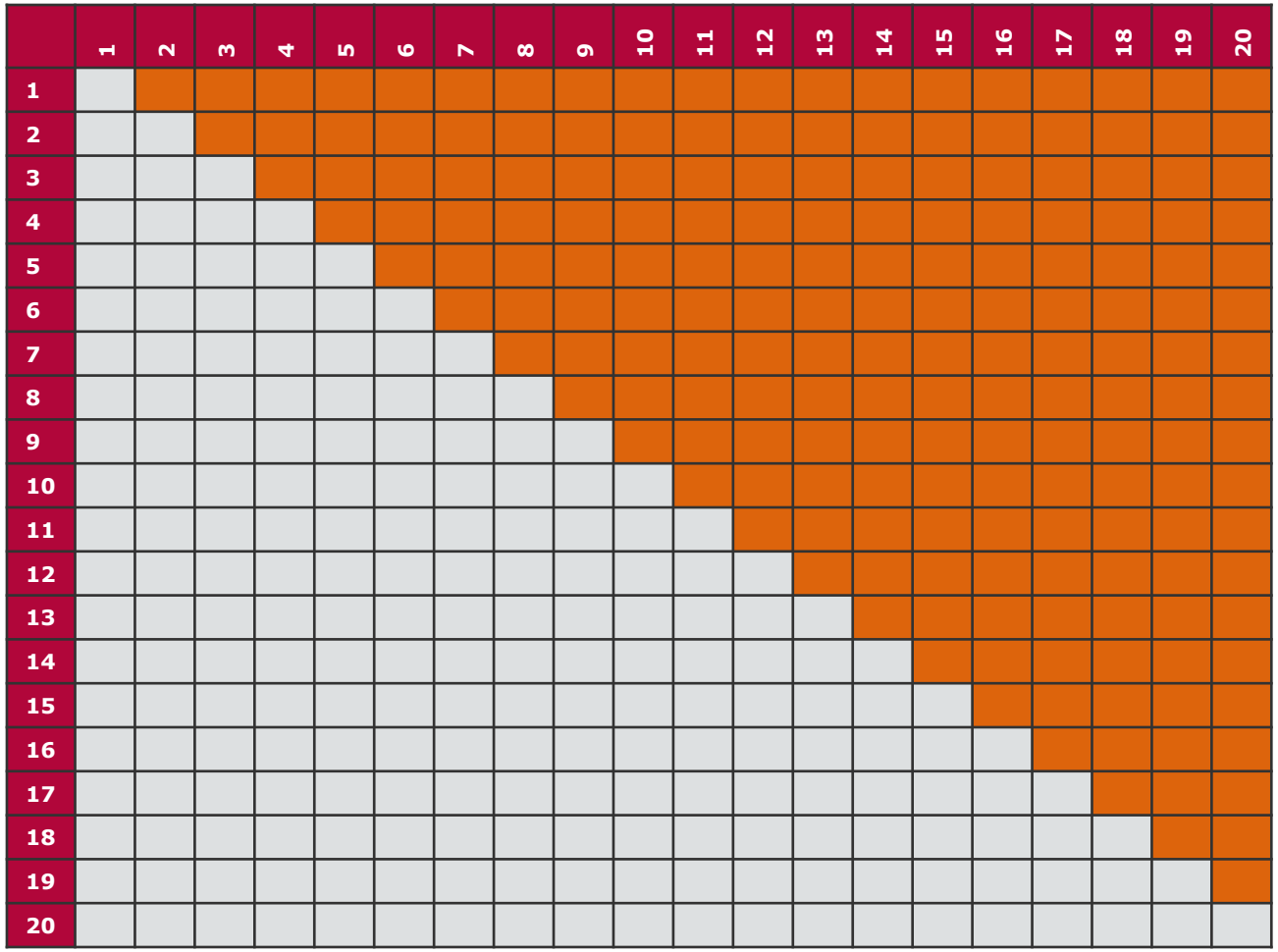
Felix Naumann
Information Integration
Winter 2019/20

Reflexivity of Similarity



380 comparisons

Symmetry of Similarity



190 comparisons

Felix Naumann
Information Integration
Winter 2019/20

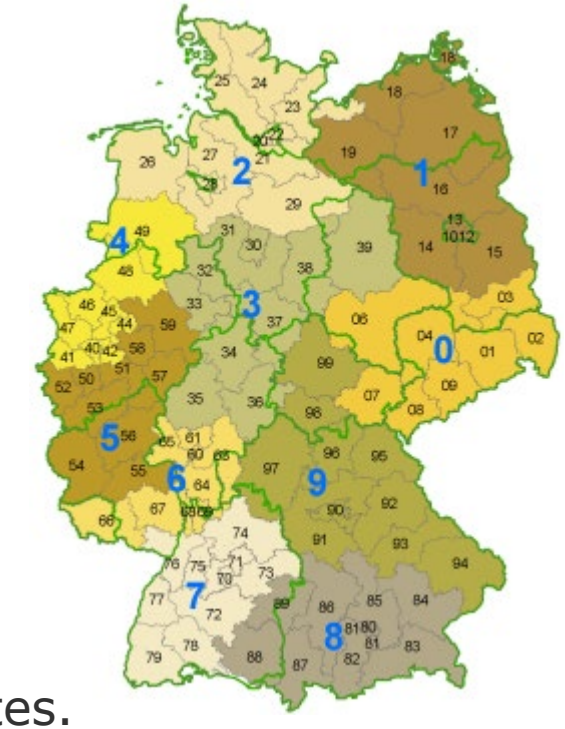
Complexity

- Problem: Too many comparisons!
  - 10.000 customers
    => 49.995.000 comparisons
    - $(n^2 - n) / 2$
    - Each comparison is already expensive.

- Idea: Avoid comparisons…
  - … by filtering out individual records.
  - … by partitioning the records and comparing only within a partition.

Partitioning / Blocking

- Partition the records (horizontally) and compare pairs of records only within a partition.
  - Partitioning by first two zip-digits
    - Ca. 100 partitions in Germany
    - Ca. 100 customers per partition
    - => 495.000 comparisons
  - Partition by first letter of surname
  - …
- Idea: Partition multiple times by different criteria.
  - Then apply transitive closure on discovered duplicates.



Source: wikipedia.de

Felix Naumann
Information Integration
Winter 2019/20

The problem with transitivity

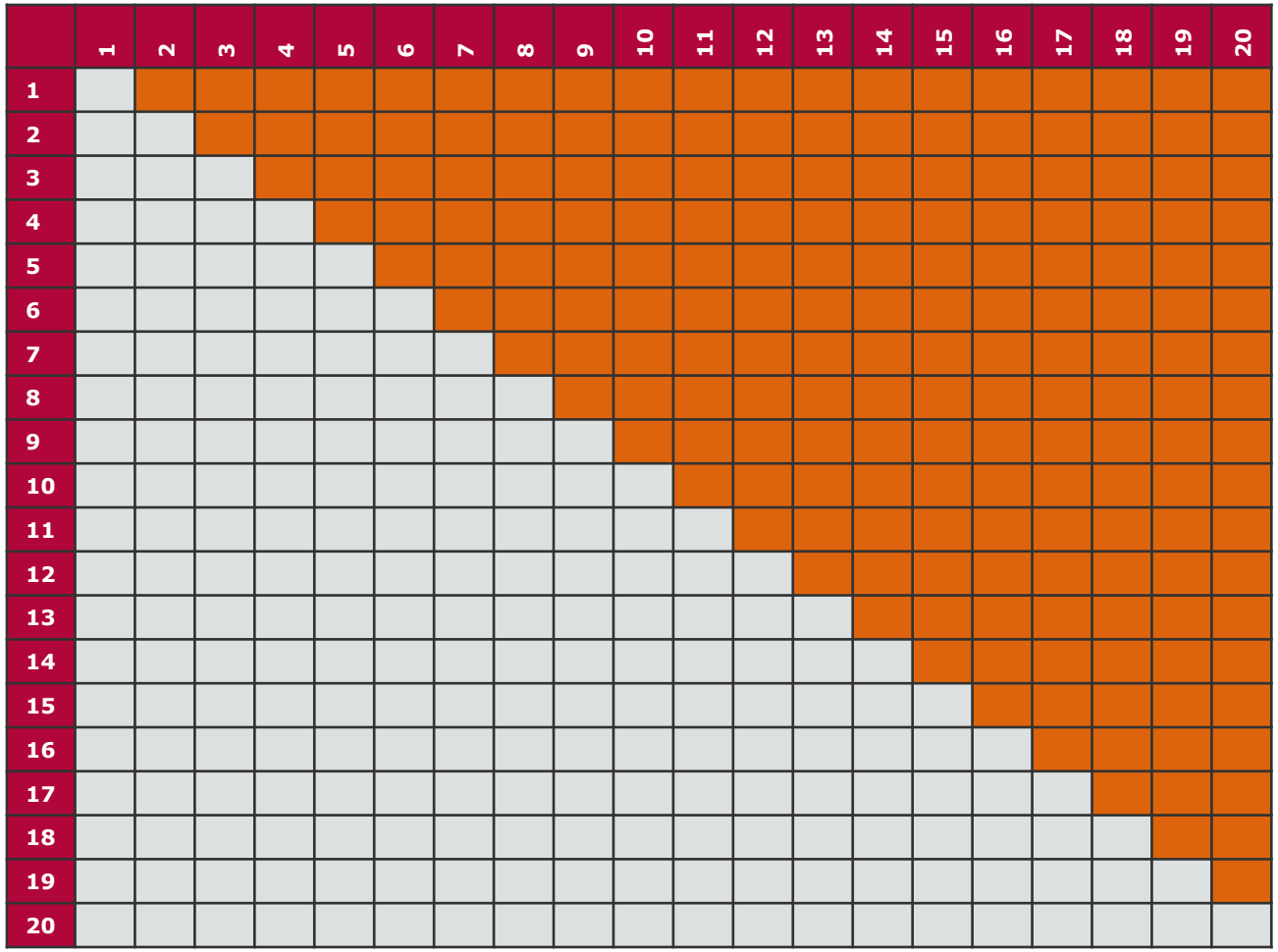
- Duplicity is
  - Reflexive: $A \sim A$
  - Symmetric: If $A \sim B$ then $B \sim A$
  - Transitive: $A \sim B$ and $B \sim C$ then $A \sim C$
- Use transitive closure to find more duplicates

- Similarity measures are usually
  - Reflexive: $sim(A,A) = 1$
  - Symmetic: $sim(A,B) = sim(B,A)$
  - But: If $sim(A,B) < \vartheta$ and $sim(B,C) < \vartheta \not\Rightarrow sim(A,C) < \vartheta$

- Solution 1: Accept incohesive clusters
- Solution 2: Break clusters along weakest similarities

Felix Naumann
Information Integration
Winter 2019/20

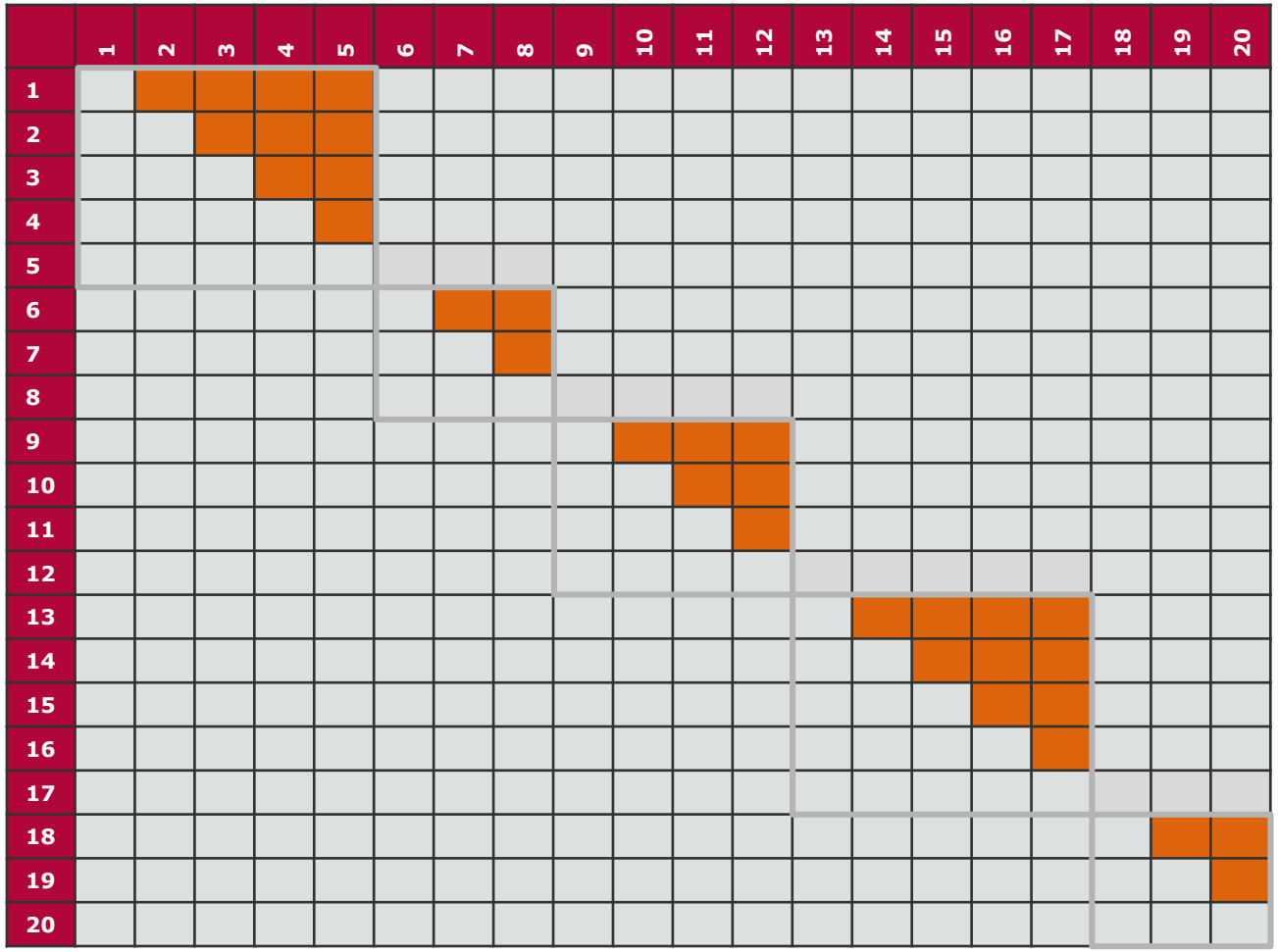
Records sorted by ZIP



190 comparisons

Felix Naumann
Information Integration
Winter 2019/20

Blocking by ZIP



32 comparisons

Felix Naumann
Information Integration
Winter 2019/20

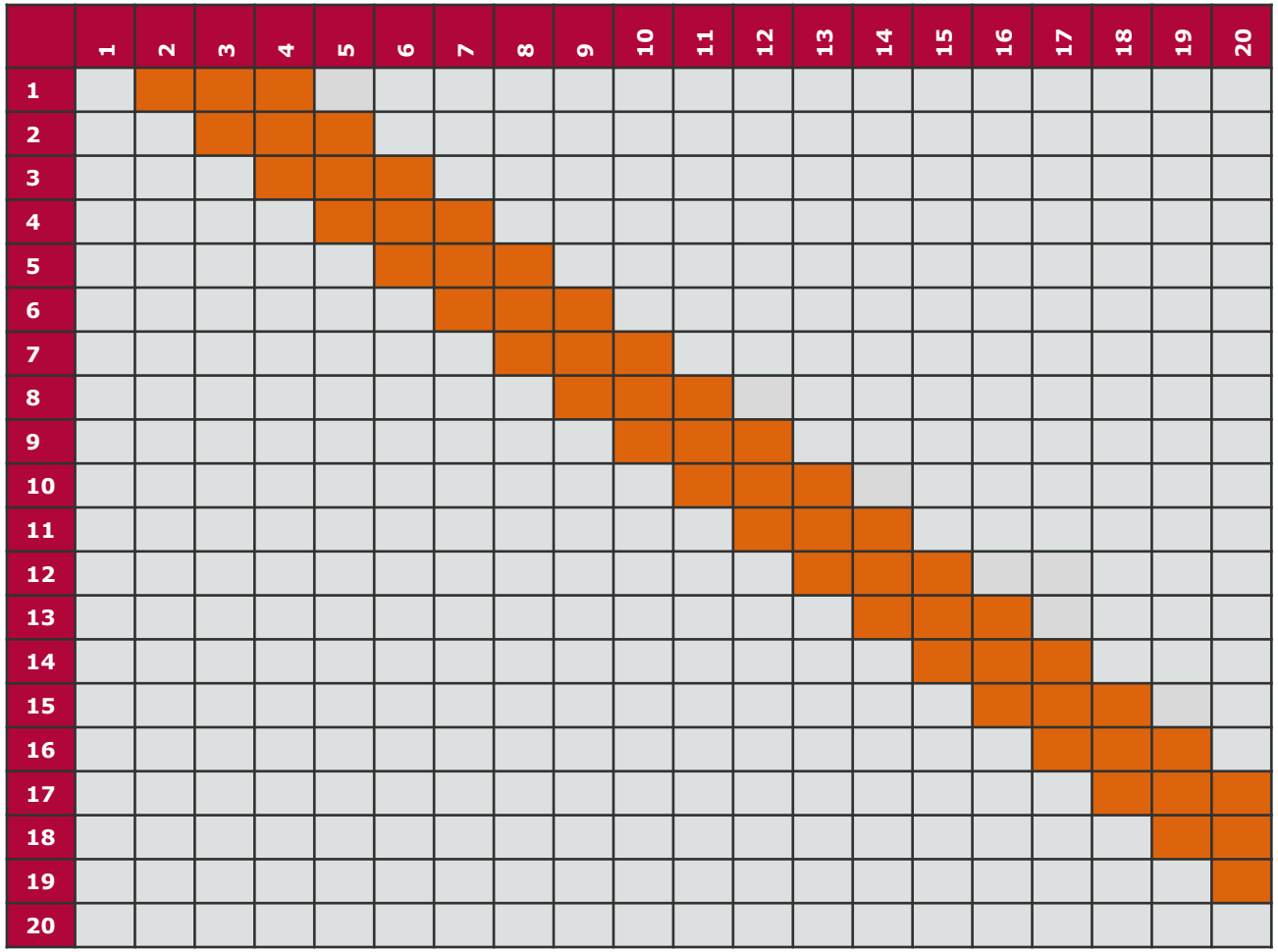
Sorted Neighborhood [Hernandez Stolfo 1998]

- Idea
  - Sort tuples so that similar tuples are close to each other.
  - Only compare tuples within a small neighborhood (window).

1. Generate key
   - E.g.: SSN+"first 3 letters of name" + ...
2. Sort by key
   - Similar tuples end up close to each other.
3. Slide window over sorted tuples
   - Compare all pairs of tuples within window.

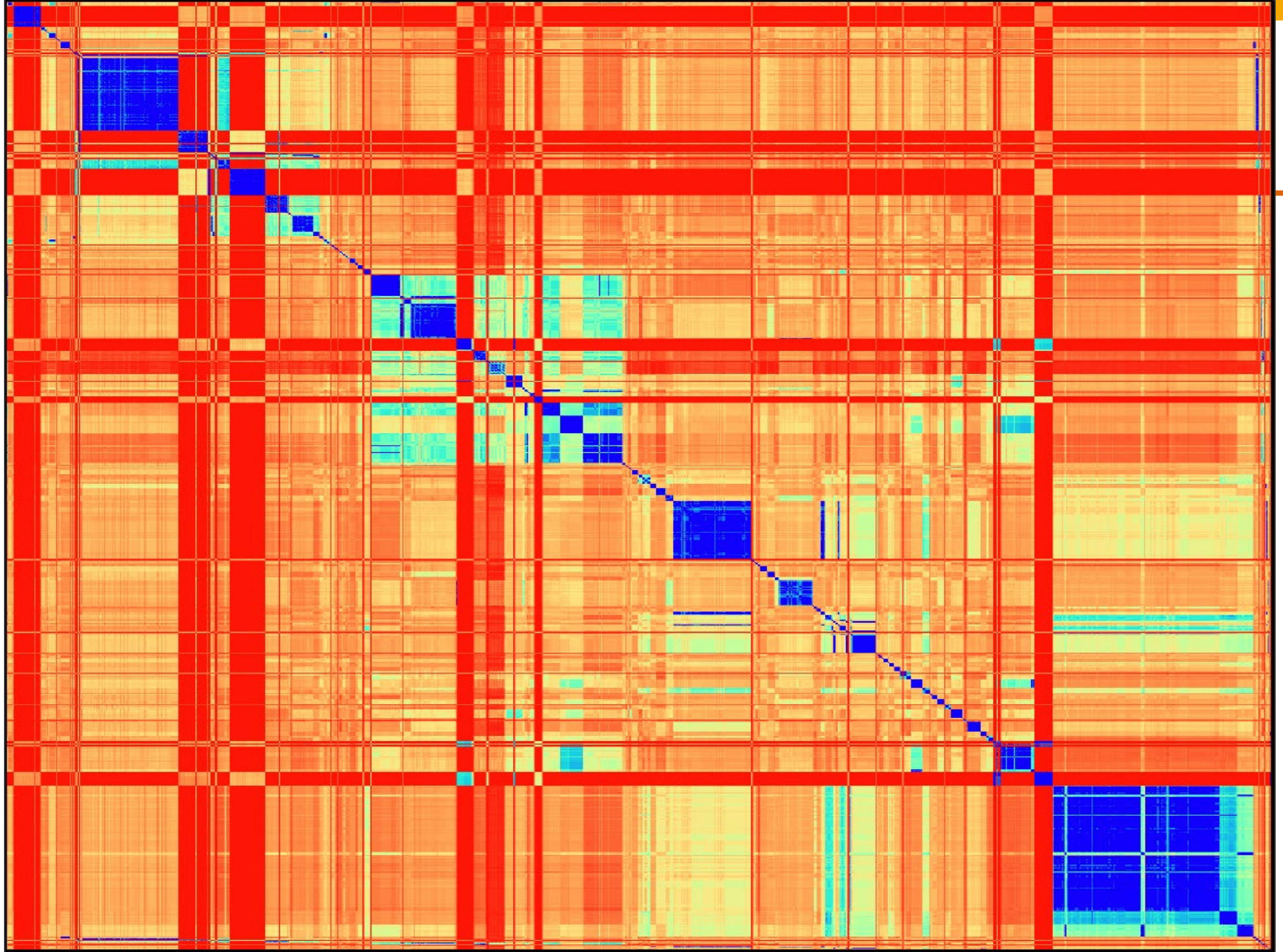
- Problems
  - Choice of key
  - Choice of window size
- Complexity: At least 3 passes over data
  - Sorting!

SNM by ZIP (window size 4)



54 comparisons

Felix Naumann
Information Integration
Winter 2019/20



Felix Naumann
Information Integration
Winter 2019/20

Overview

1. Duplicate detection
2. Similarity measures
3. Algorithms
4. **Data sets and evaluation**
5. Data fusion

Felix Naumann
Information Integration
Winter 2019/20

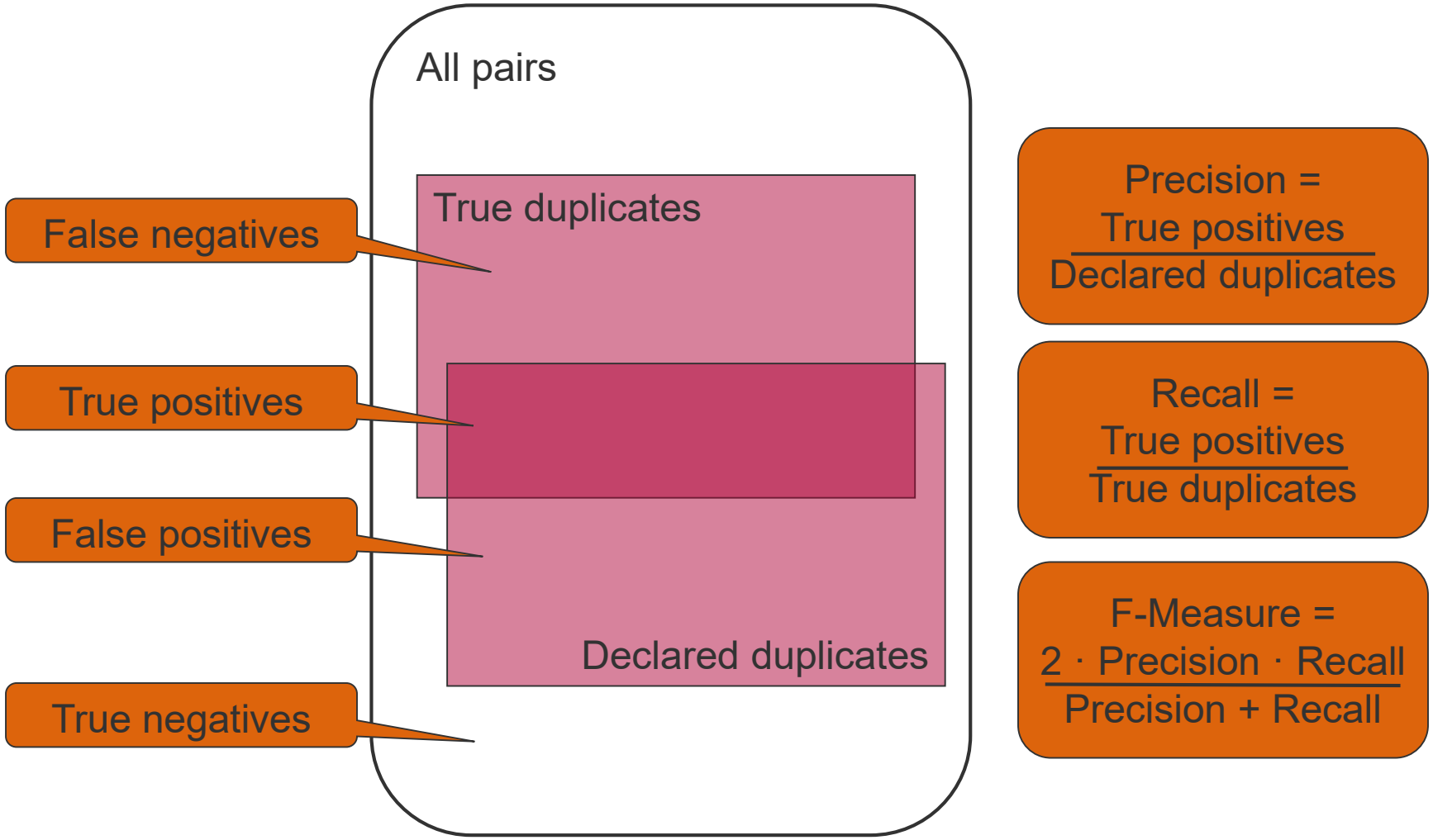
Precision & Recall

- True positives (TP): Correctly declared duplicates
- False positives (FP): Incorrectly declared duplicates
- True negatives (TN): Correctly avoided pairs
- False negatives (FN): Missed duplicates

- Precision = TP / (TP + FP)
  - = TP / declared dups
  - Proportion of found matches that are correct
  - Correctness
- Recall = TP / (TP + FN)
  - = TP / all dups
  - Proportion of correct matches that are found
  - Completeness

Felix Naumann
Information Integration
Winter 2019/20

Precision & Recall



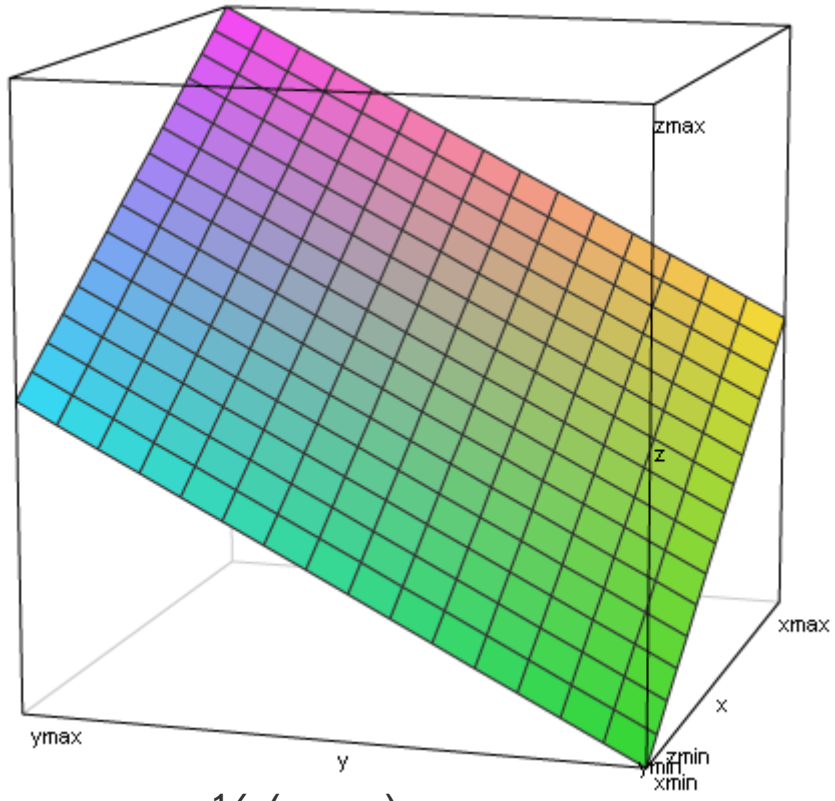
$$\text{Precision} = \frac{\text{True positives}}{\text{Declared duplicates}}$$

$$\text{Recall} = \frac{\text{True positives}}{\text{True duplicates}}$$

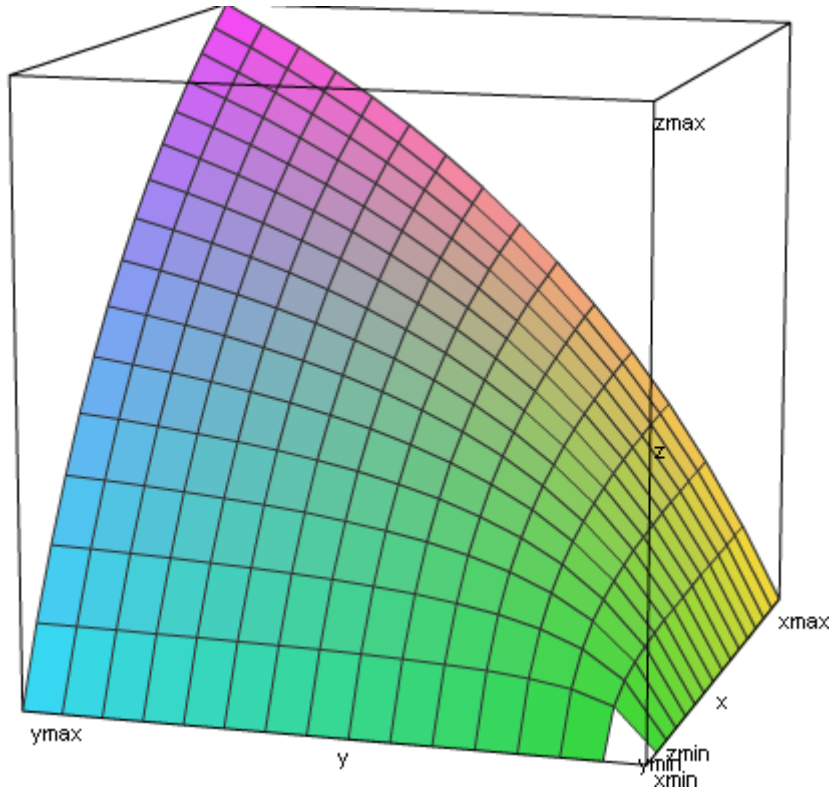
$$\text{F-Measure} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Felix Naumann
Information Integration
Winter 2019/20

Arithmetic mean („Average“) vs. Harmonic mean („F-Measure“)



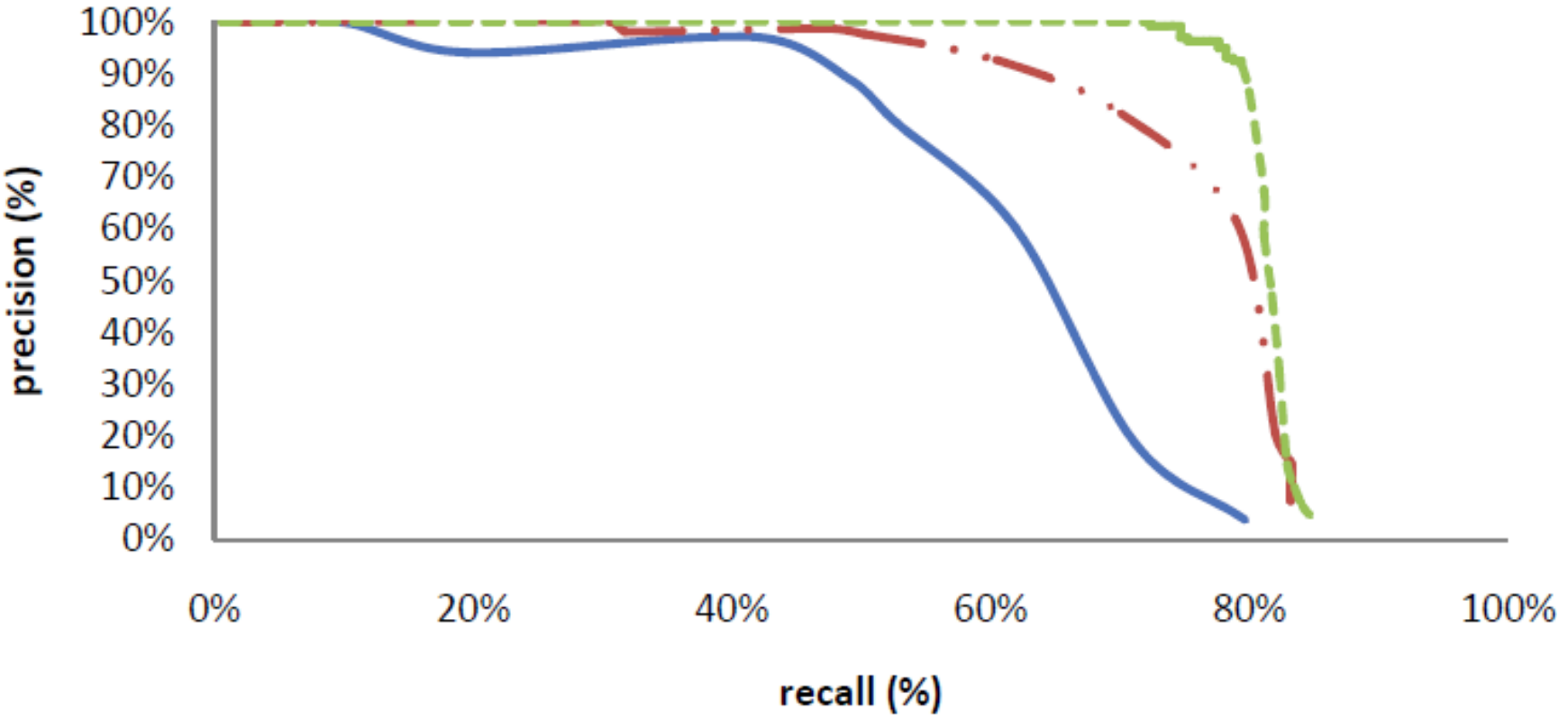
$z = ½ (x + y)$



$z = 2 (x \cdot y) / (x + y)$

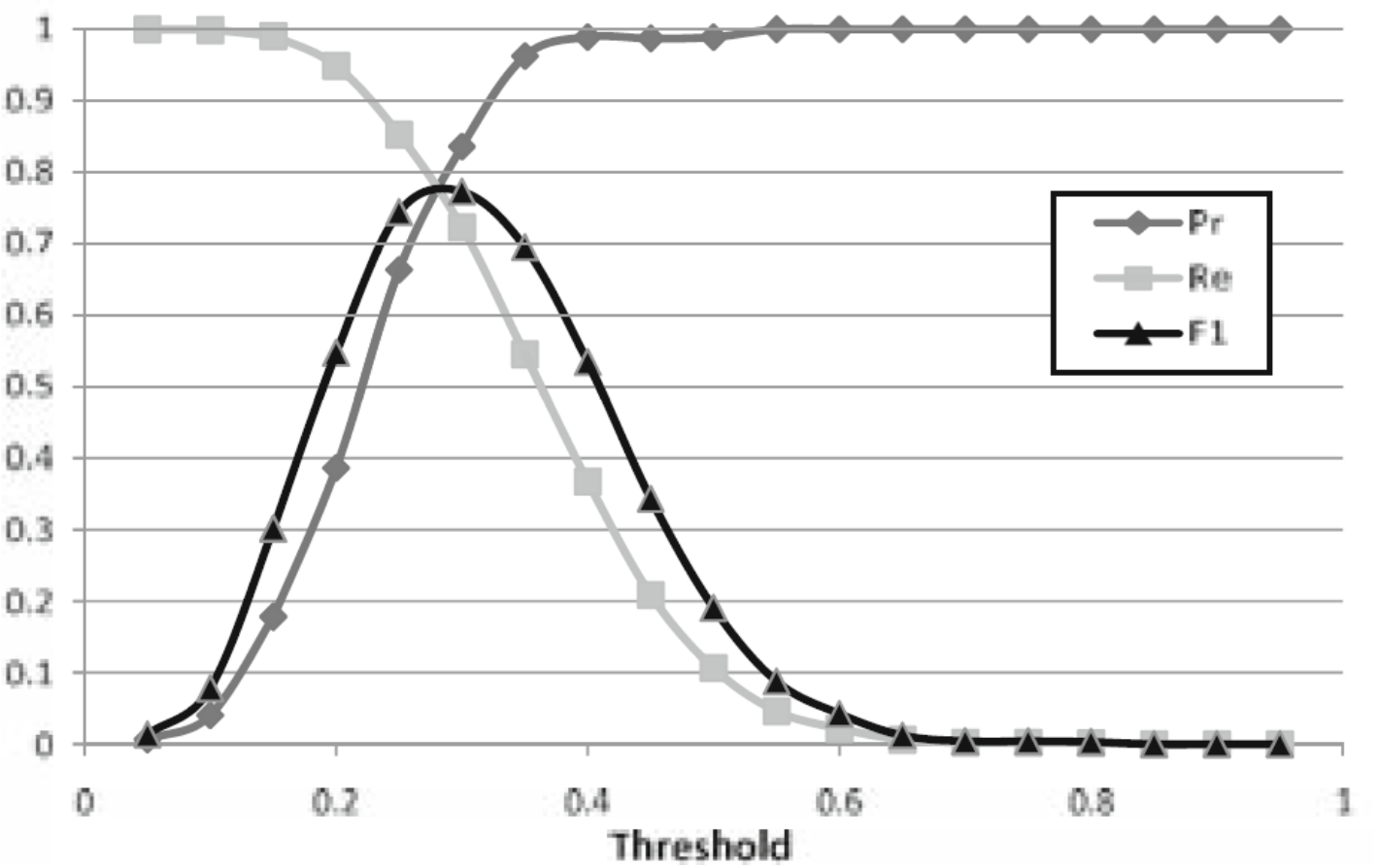
Felix Naumann
Information Integration
Winter 2019/20

Recall-precision-diagram



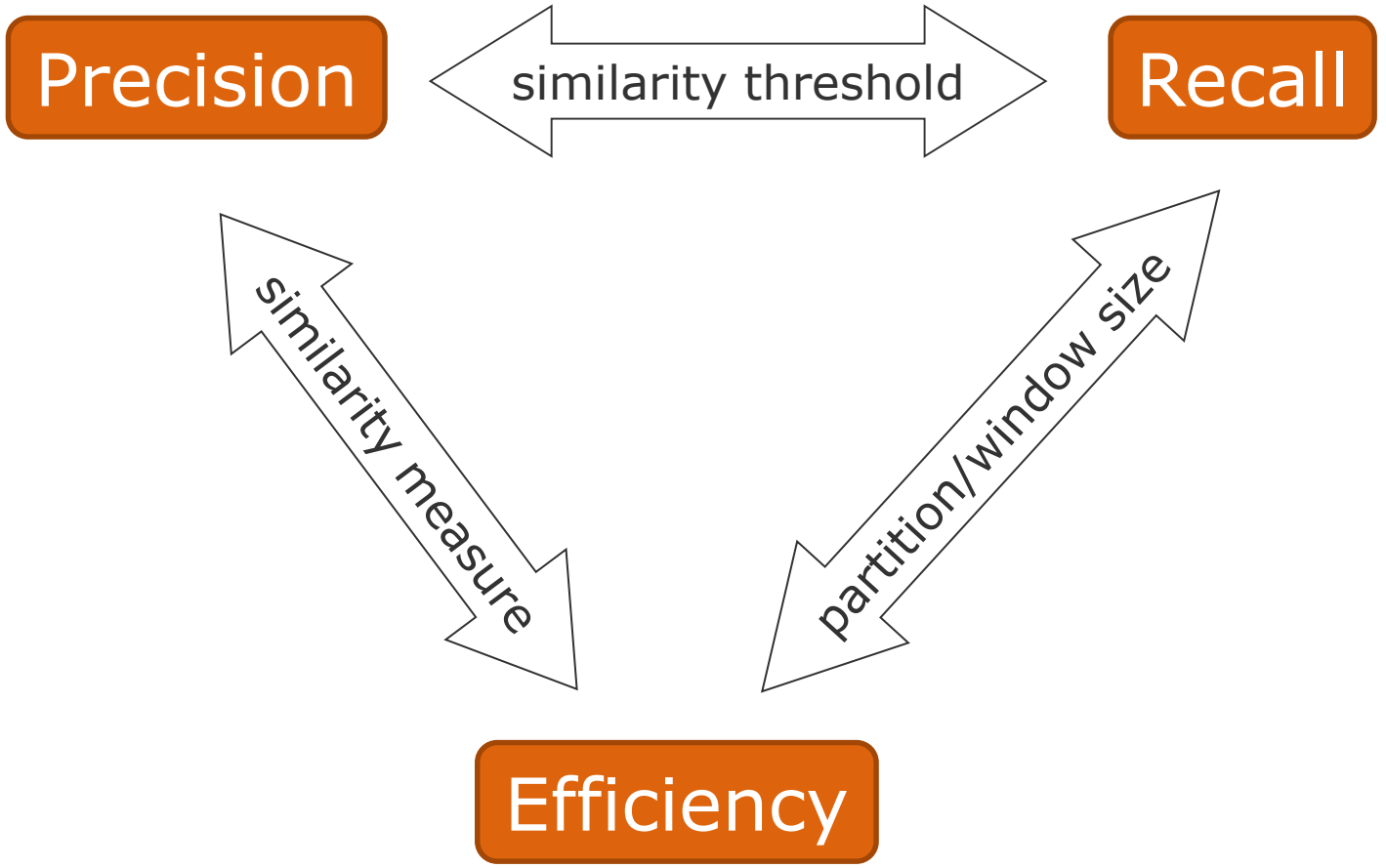
Felix Naumann
Information Integration
Winter 2019/20

F1 Graph



From: Creating probabilistic databases from duplicated data
Oktie Hassanzadeh · Renée J. Miller (VLDBJ)

Evaluating Duplicate Detection

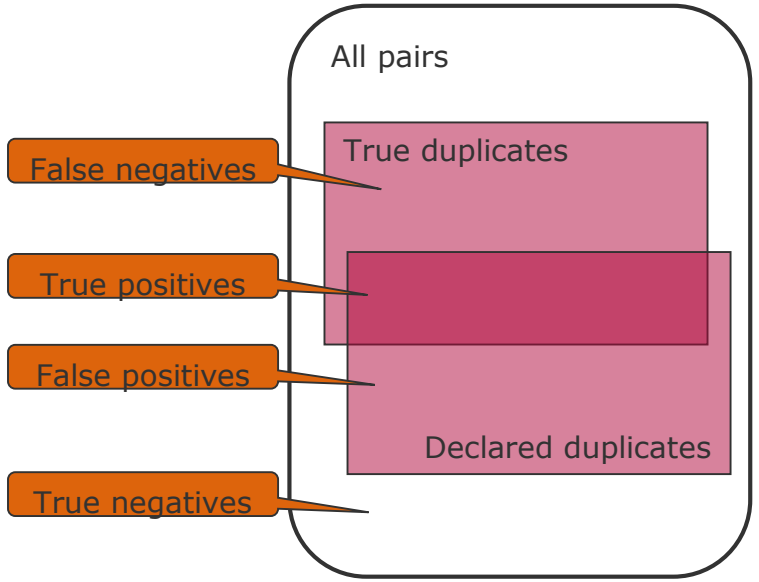


Felix Naumann
Information Integration
Winter 2019/20

Other effectiveness measures

- Accuracy
  - (TP + TN) / (TP + FP + TN + FN)
  - Used for balanced classes
  - For duplicate detection, TN usually dominates overall result

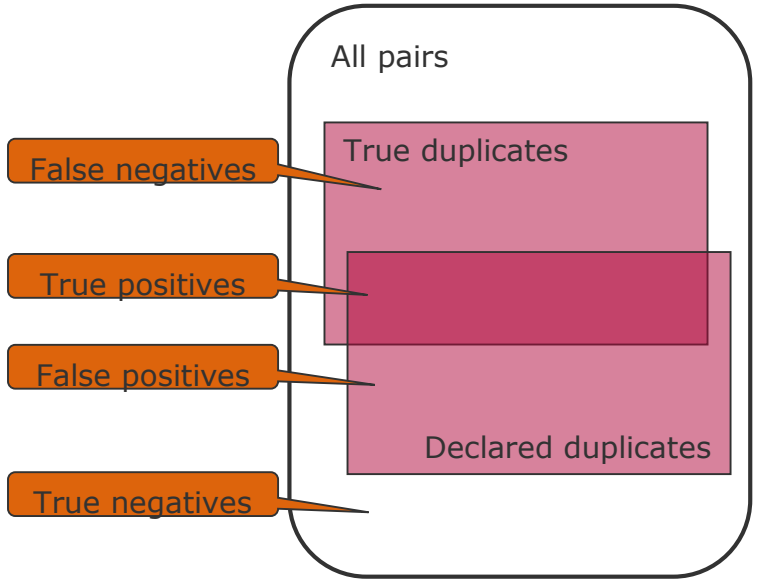
- Specificity
  - TN / (TN + FP)
  - = TN / true non-matches
  - Again: TN dominates overall result



Complexity measures

- Problem: Measure not only quality of similarity measure, but also that of algorithm
  - Solution 1: Runtime measurements
    - But: Different hardware, difficult repeatability
  - Solution 2: Measure how well/poor algorithms filter candidates

- Reduction ratio
  - 1 – ((TP+TN)/(FN+TP+FP+TN))
    = 1 – accuracy
- Pairs completeness
  - TP / (FN + TP) = recall
- Pairs quality
  - TP / (TP + TN)



Data sets to evaluate deduplication

- Requirements
  - Real-world application
  - Interestingly large
  - Interestingly dirty
  - Gold standard
  - Publicly available
  - (Relational)
  - (Understandable)

- Hardly available
  - Privacy issues
  - Security issues
  - Embarrasment
  - Data fiefdoms

Small datasets with gold standard

- CORA
  - 1878 bibliographic references in XML format
  - http://www.hpi.uni-potsdam.de/naumann/projekte/repeatability/datasets/cora_dataset.html
- DBLP
  - 50,000 bibliographic references in XML format
  - http://www.hpi.uni-potsdam.de/naumann/projekte/repeatability/datasets/dblp_dataset.html
- Restaurants
  - 864 restaurants with 112 duplicates
  - http://www.cs.utexas.edu/users/ml/riddle/data.html
- Whirl datasets
  - 11 smaller datasets with a single string attribute
  - http://www.cs.purdue.edu/commugrate/data/whirl/match/

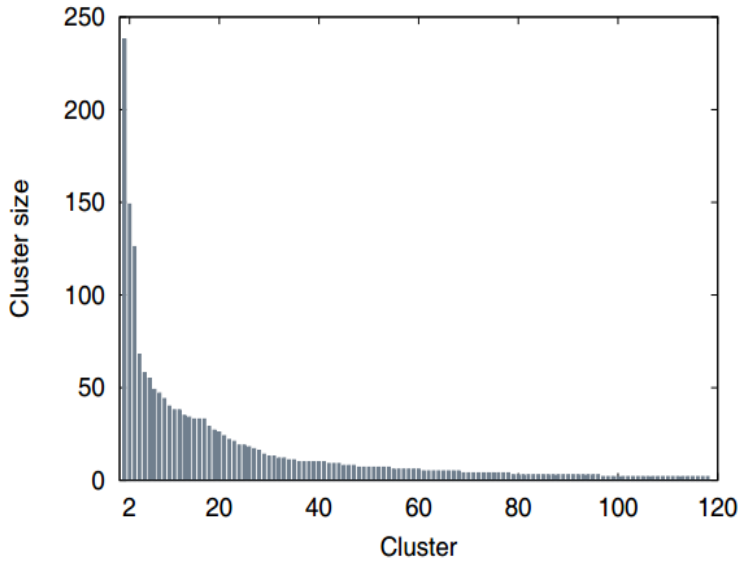


Fig. 2. Number of records per cluster in the Cora data set.

Large datasets without gold standard

- Places
  - 1.4 million POIs from Facebook, Gowalla, Foursquare
- WheelMap
  - 120,000 places/things in Germany
- FreeDB
  - 1.9 million CDs, dirty, some duplicate clusters quite large
  - original: http://www.freedb.org/en/download__database.10.html
  - derived: http://www.hpi.uni-potsdam.de/naumann/projekte/repeatability/datasets/cd_datasets.html
- CITESEERX
  - 1.3 million publications in CSV
  - http://asterix.ics.uci.edu/data/csx.raw.txt.gz

Felix Naumann
Information Integration
Winter 2019/20

http://hpi.de/naumann/projects/repeatability/datasets.html

Data generation

- For lack of gold standard: create one
- Data base
  - Real-world data sets (without or without enough duplicates)
  - Real-world values (from dictionaries)
  - Synthetic strings
- Data corruption: Duplicate and modify some percentage of tuples
  - Duplication: Cluster sizes?
  - Data values
    - Insert/remove/transpose/change certain letters
    - Delete values
    - Swap values (within tuple, from dictionary, across tuples)
- General suspicion: Similarity measure and candidate selction is geared towards known types of errors.

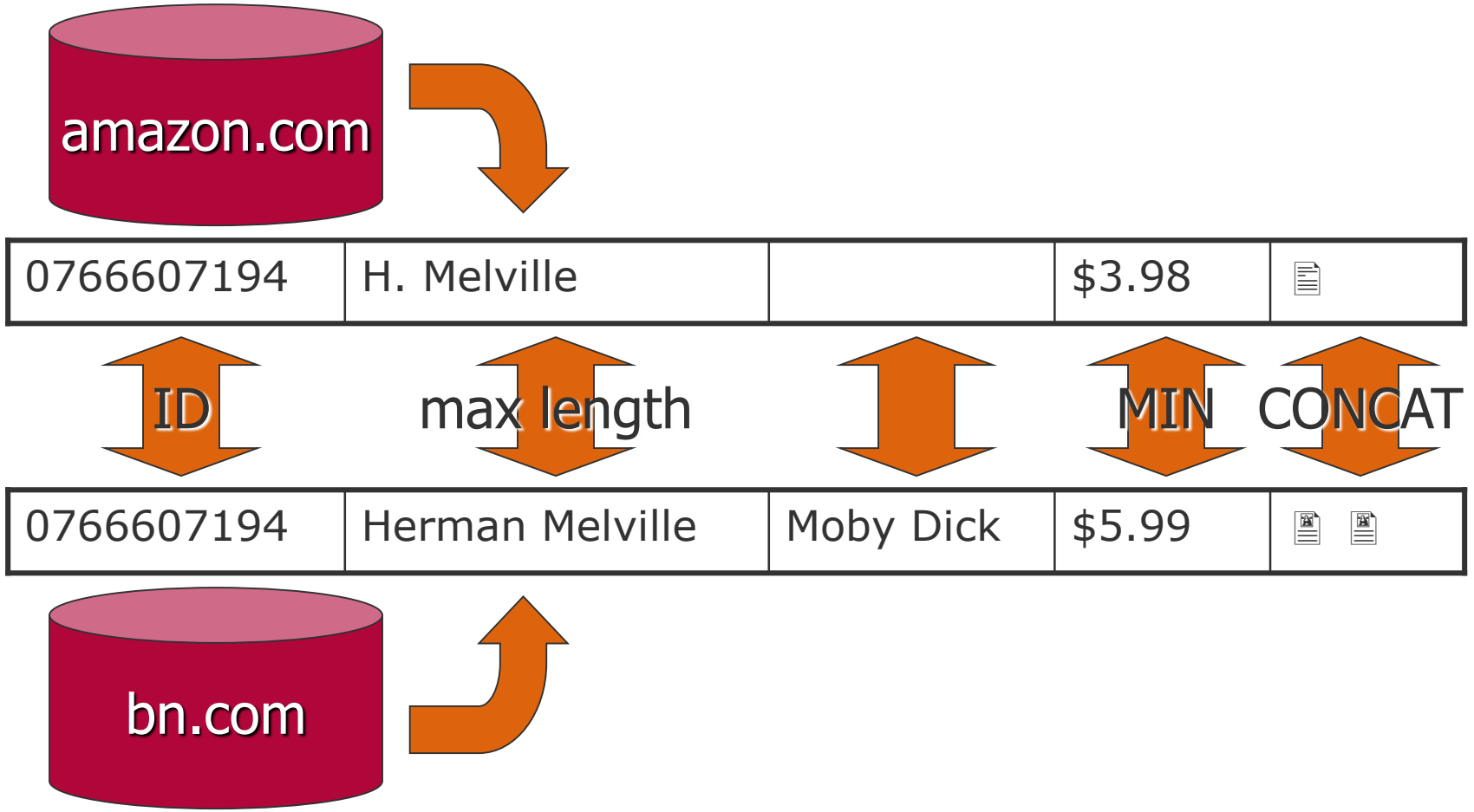
Data generators

- UIS Database Generator
  - Generates a list of randomly perturbed names and US mailing addresses.
  - Written by Mauricio Hernández.
  - http://www.cs.utexas.edu/users/ml/riddle/data/dbgen.tar.gz
- FEBRL-Generator
  - Part of a cleansing suite
  - Dictionaries with frequencies
  - http://sourceforge.net/projects/febrl/
- Dirty XML Generator
  - http://www.hpi.uni-potsdam.de/naumann/projekte/completed_projects/dirtyxml.html

Overview

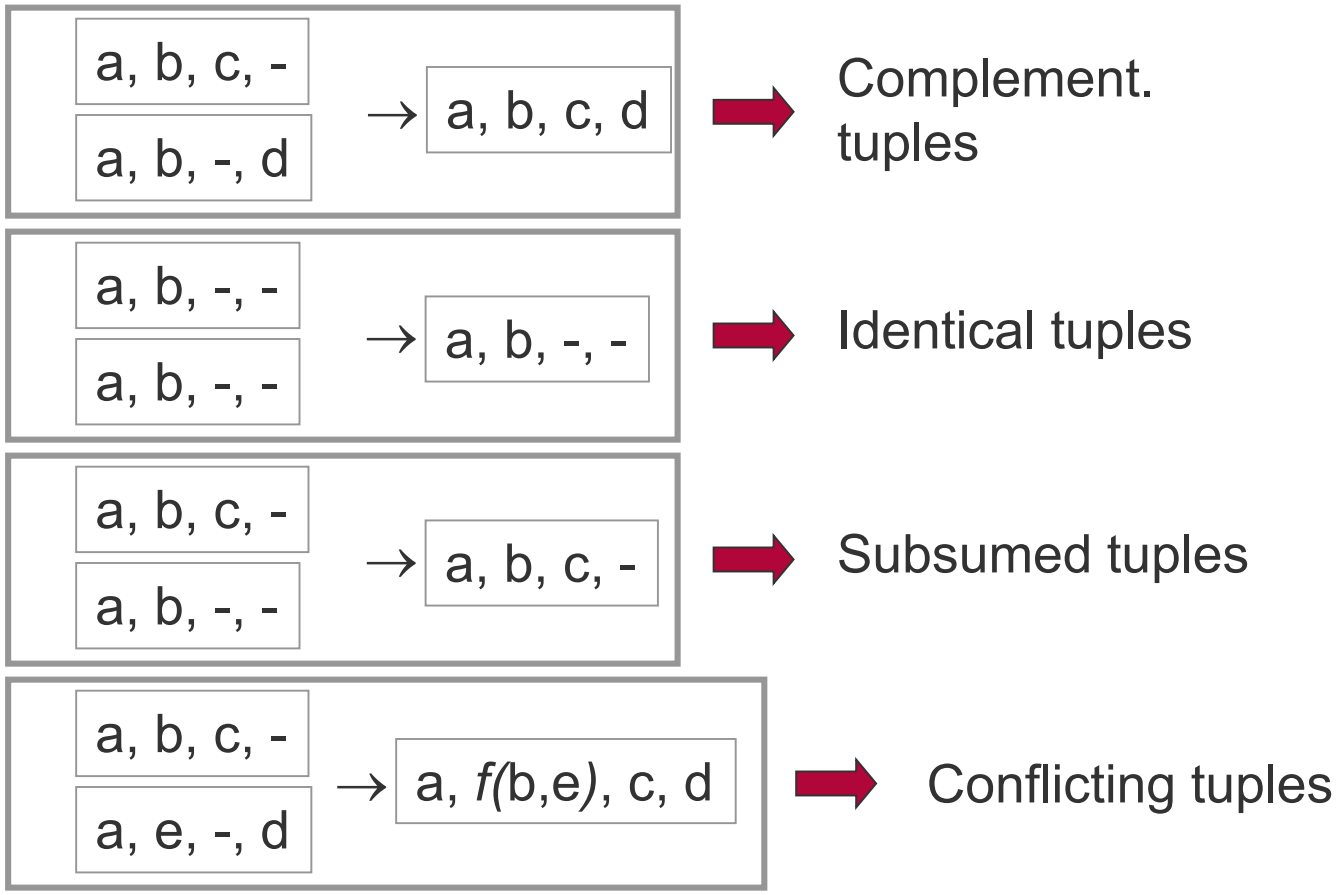
1. Duplicate detection
2. Similarity measures
3. Algorithms
4. Data sets and evaluation
5. **Data fusion**

Data Fusion



Felix Naumann
Information Integration
Winter 2019/20

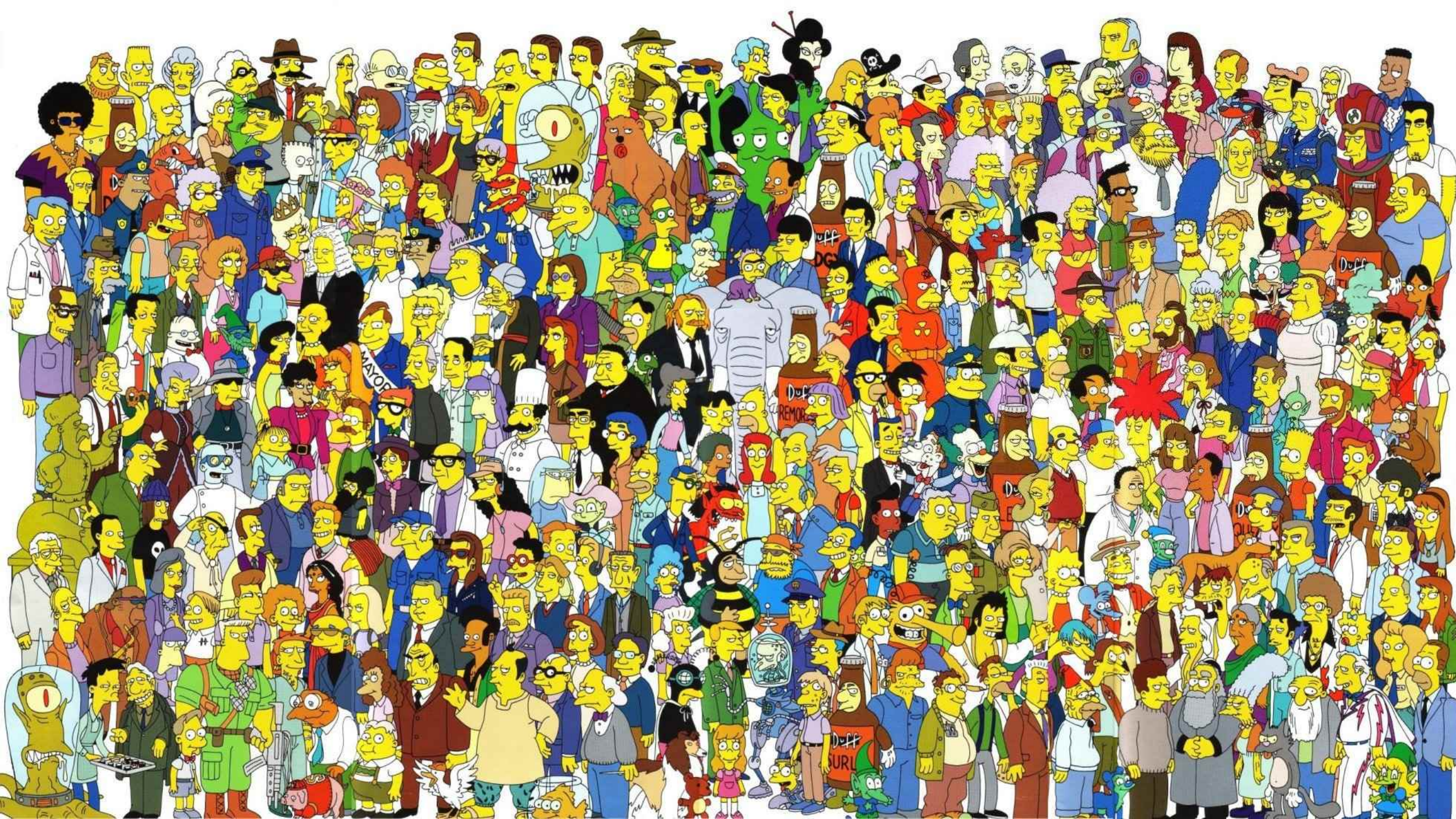
“Proper” Data Fusion



Felix Naumann
Information Integration
Winter 2019/20

Conflict Resolution Functions

Min, Max, Sum, Count, Avg, StdDev	Standard aggregation
Random	Random choice
First, Last	Choose first/last value; depends on order
Longest, Shortest	Choose longest/shortest value
Choose(source)	Choose value froma particular source
ChooseDepending(col, val)	Choose depending on val in other column col
Vote	Majority decision
Coalesce	Choose first non-null value
Group, Concat	Group or concatenate all values
MostRecent	Choose most recent (up-to-date) value
MostAbstract, MostSpecific	Use a taxonomy / ontology
….	….


MAYOR
Duff
REMOR
Duff
SURL
Duff

References

- [Ananthakrishna et al. 2002] R. Ananthakrishna, S. Chaudhuri, V. Ganti: Eliminating Fuzzy Duplicates in Data Warehouses, Proc. of the 28th VLDB Conference, Hong Kong, China, pages 586-597, 2002.
- [Dey et al. 2002] D. Dey, S. Sarkar, P. De: A Distance-based Approach to Entity Reconciliation in Heterogeneous Databases, IEEE Transactions on Knowledge and Data Engineering (TKDE) 14(3): 567-582, 2002.
- [Cohen03] William W. Cohen, Pradeep D. Ravikumar, Stephen E. Fienberg: A Comparison of String Distance Metrics for Name-Matching Tasks. IIWeb 2003: 73-78
- [Gravano et al. 2001] L. Gravano, P. Ipeirotis, H. Jagadish, N. Koudas, S. Muthukrishnan, D. Srivastava: Approximate String Joins in a Database (Almost) for Free, Proc. of the 27th VLDB Conference, Roma, Italy, pages 491-500, 2001.
- [Hernandez Stolfo 1995] M. Hernandez, S. Stolfo: The Merge/Purge Problem for Large Databases, Proc. ACM SIGMOD Conference 1995, San Jose, USA, pages 127-138, 1995.
- [Hernandez Stolfo 1998] M. Hernandez, S. Stolfo: Real-world Data is Dirty: Data Cleansing and the Merge/Purge, Journal of Data Mining and Knowledge Discovery, 2(1):9-37, 1998.
- [Jaro 1989] M. Jaro: Advances in Record Linkage Methodology as Applied to Matching the 1985 Census of Tampa, Florida, Journal of the American Statistical Association 84(406):414-420, 1989.
- [Low et al. 2001] W. Low, M. Lee, T. Ling: A Knowledge-based Approach for Duplicate Elimination in Data Cleaning, Information Systems 26(8):585-606, 2001.
- [Monge 2000] A. Monge: Matching Algorithms within a Duplicate Detection System, IEEE Data Engineering Bulletin, 23(4):14-20, 2000.
- [Navarro 2001] G. Navarro: A Guided Tour of Approximate String Matching, ACM Computing Surveys 31(1):31-88, 2001.
- [Weis Naumann 2005] Melanie Weis, Felix Naumann: DogmatiX Tracks down Duplicates in XML. In Proc. of the ACM Conference on Management of Data (SIGMOD) 2005.
- [GL94] Outerjoins as Disjunctions, Cesar A. Galindo-Legaria, SIGMOD 1994 conference
- [Cod79] E. F. Codd: Extending the Database Relational Model to Capture More Meaning. TODS 4(4): 397-434 (1979)
- [Ull89] Jeffrey D. Ullman: Principles of Database and Knowledge-Base Systems, Volume II. Computer Science Press 1989