in the process of retrieving data with global and local schema we do need entity linkage

Data Wrangling and Data Analysis
Heterogeneous Data Integration (Part B)
Entity Linkage

Hakim Qahtan

Slides prepared by

Prof. Yannis Velegrakis

Department of Information and Computing Sciences

Utrecht University

I won a trip to L.A.

That is
Lekanopedio **A**ttikis

Not the famous **L**os **A**ngeles !!

How many names, descriptions are used for the same real-world "entity"?

(c) Papadakis et al.

How many names, descriptions are used for the same real-world “entity”?

London 런던 ලන්ඩන් लंडन लंदन લંડન ለንደን ロンドン
লন্ডন ลอนดอน இலண்டன் ლონდონი Llundain
Londain Londe Londen Londen Londen Londinium
London Londona Londonas Londoni Londono Londra
Londres Londrez Londyn Lontoo Loundres Luân Đôn
Lunden Lundúnir Lunnainn Lunnon لندن لندن لندن لوندون
לאנדאן לונדון Λονδίνο Лёндан Лондан Лондон Лондон
Лондон Լոնդոն 伦敦 …

How many names, descriptions are used for the same real-world “entity”?

London 런던 ලන්ඩන් लंडन लंदन લંડન ለንደን ロンドン লন্ডন ลอนดอน இலண்டன் ლონდონი Llundain Londain Londe Londen Londen Londen Londinium London Londona Londonas Londoni Londono Londra Londres Londrez Londyn Lontoo Loundres Luân Đôn Lunden Lundúnir Lunnainn Lunnon لندن لندن لندن لوندون לאנדאן לונדון Λονδίνο Лёндан Лондан Лондон Лондон Лондон Լոնդոն 伦敦 …

capital of UK, host city of the IV Olympic Games, host city of the XIV Olympic Games, future host of the XXX Olympic Games, city of the Westminster Abbey, city of the London Eye, the city described by Charles Dickens in his novels, …

all describes the same city. So not just the different languages but also different representations

How many names, descriptions are used for the same real-world "entity"?

London 런던 ലണ്ടൻ लंडन लंदन લંડન ለንደን ロンドン
লন্ডন ลอนดอน இலண்டன் ლონდონი Llundain
Londain Londe Londen Londen Londen Londinium
London Londona Londonas Londoni Londono Londra
Londres Londrez Londyn Lontoo Loundres Luân Đôn
Lunden Lundúnir Lunnainn Lunnon لندن لندن لندن لوندون
לונדון לאנדאן Λονδίνο Лёндан Лондан Лондон Лондон
Лондон Լոնդոն 伦敦 …

capital of UK, host city of the IV Olympic Games, host city of the XIV Olympic Games, future host of the XXX Olympic Games, city of the Westminster Abbey, city of the London Eye, the city described by Charles Dickens in his novels, …

http://sws.geonames.org/2643743/
http://en.wikipedia.org/wiki/London
http://dbpedia.org/resource/Category:London
…

… or …

How many "entities" have the same name?

- London, KY
- London, Laurel, KY
- London, OH
- London, Madison, OH
- London, AR
- London, Pope, AR
- London, TX
- London, Kimble, TX
- London, MO
- London, MO
- London, London, MI
- London, London, Monroe, MI
- London, Uninc Conecuh County, AL
- London, Uninc Conecuh County, Conecuh, AL
- London, Uninc Shelby County, IN
- London, Uninc Shelby County, Shelby, IN
- London, Deerfield, WI
- London, Deerfield, Dane, WI
- London, Uninc Freeborn County, MN
- …

… or …

names, different cities and different meanings

How many "entities" have the same name?

- London, KY
- London, Laurel, KY
- London, OH
- London, Madison, OH
- London, AR
- London, Pope, AR
- London, TX
- London, Kimble, TX
- London, MO
- London, MO
- London, London, MI
- London, London, Monroe, MI
- London, Uninc Conecuh County, AL
- London, Uninc Conecuh County, Conecuh, AL
- London, Uninc Shelby County, IN
- London, Uninc Shelby County, Shelby, IN
- London, Deerfield, WI
- London, Deerfield, Dane, WI
- London, Uninc Freeborn County, MN
- …

- London, Jack
  2612 Almes Dr
  Montgomery, AL
  (334) 272-7005
- London, Jack R
  2511 Winchester Rd
  Montgomery, AL 36106-3327
  (334) 272-7005
- London, Jack
  1222 Whitetail Trl
  Van Buren, AR 72956-7368
  (479) 474-4136
- London, Jack
  7400 Vista Del Mar Ave
  La Jolla, CA 92037-4954
  (858) 456-1850
- …

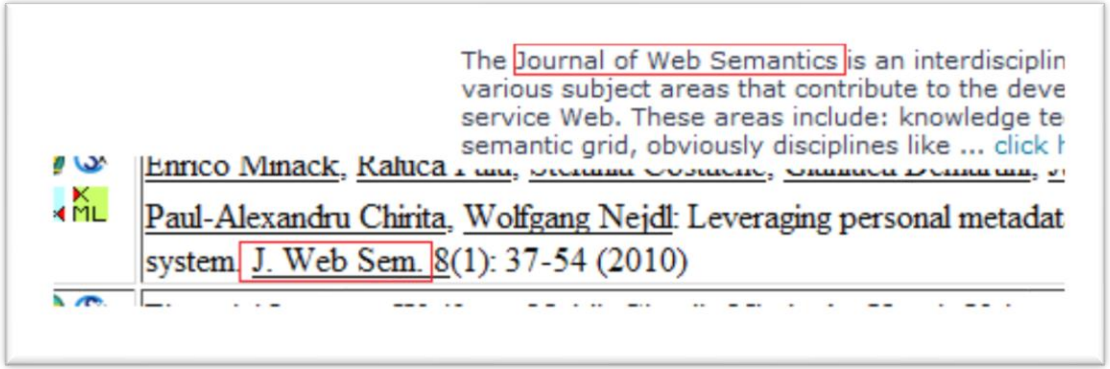
Reasons of Different Descriptions

- Text variations:
  - Misspellings
  - Acronyms
  - Transformations
  - Abbreviations
  - etc.

international conference on data engineering

Welcome to ICDE 2011

The IEEE International Conference on Data Engineering
results and advanced data-intensive applications and dis
The mission of the conference is to share research soluti
identify new issues and directions for future research anc



The Journal of Web Semantics is an interdisciplin
various subject areas that contribute to the deve
service Web. These areas include: knowledge te
semantic grid, obviously disciplines like ... click

Enrico Minack, Raluca

Paul-Alexandru Chirita, Wolfgang Nejdl: Leveraging personal metadat
system. J. Web Sem. 8(1): 37-54 (2010)

Reasons for Different Descriptions

- Text variations
- Local knowledge:
  - Each source uses different formats

    e.g., person from publication vs. person from email
  - Lack of global coordination for identifier assignment

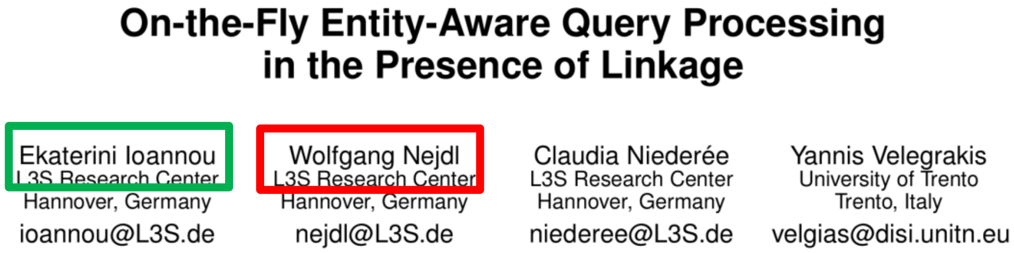
different variations referring to the same entity - different representations

to You

Prof. Nejdl, ris Zieseniß

E. Ioannou <ioannou@L3S.de> wrote:

Hello Prof. Nejdl,



On-the-Fly Entity-Aware Query Processing in the Presence of Linkage

Ekaterini Ioannou
L3S Research Center
Hannover, Germany
ioannou@L3S.de

Wolfgang Nejdl
L3S Research Center
Hannover, Germany
nejdl@L3S.de

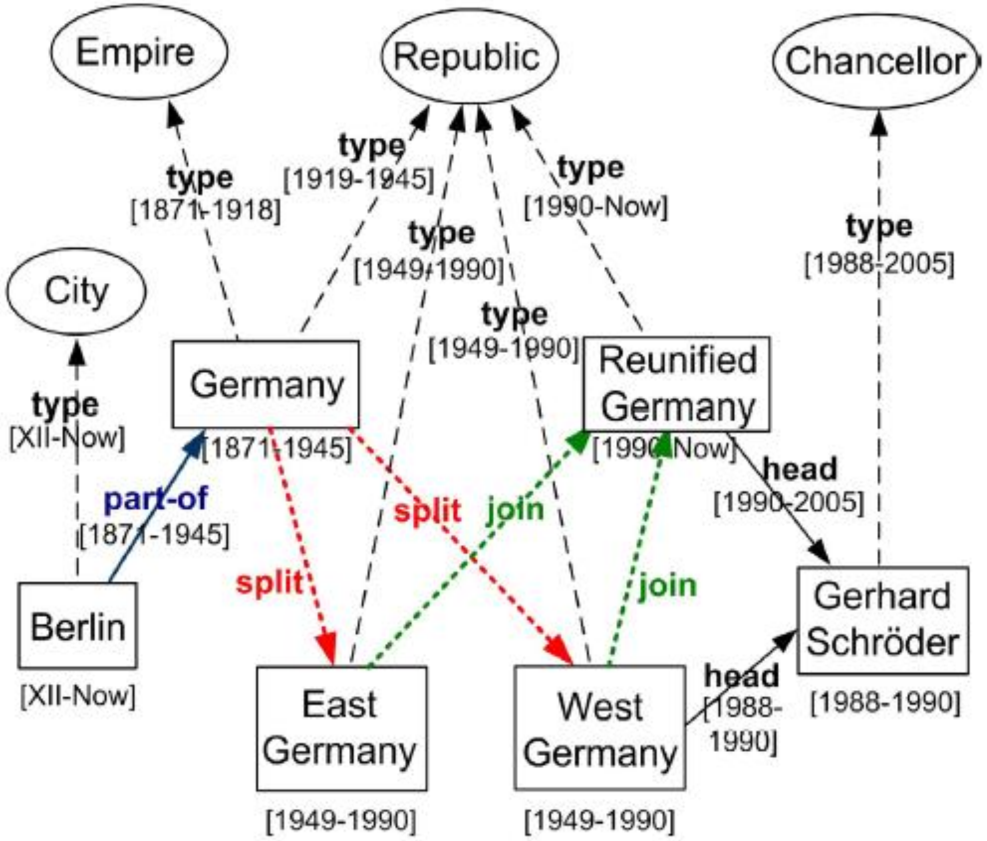
Claudia Niederée
L3S Research Center
Hannover, Germany
niederee@L3S.de

Yannis Velegrakis
University of Trento
Trento, Italy
velgias@disi.unitn.eu

Reasons for Different Descriptions

[from [VelegrakisBM09]]

- Text variations
- Local knowledge
- Evolving nature of data:
  - Entity alternative names
  - appearing in time
  - Updates in entity data



description of entity has been changed over time

Jacqueline Lee Bouvier

Reasons for Different Descriptions

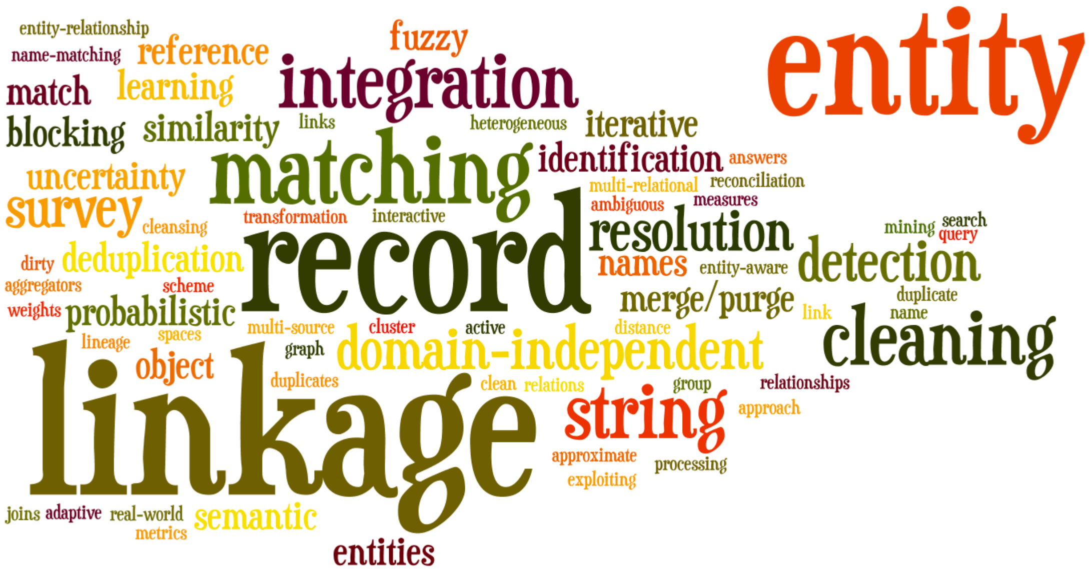
- Text variations
- Local knowledge
- Evolving nature of data
- New functionality:
  - Import data collections from various applications
    - e.g., Wikipedia data used in Freebase

Entity Resolution

trying to find the records that can represent the same entity to remove/merge them into one record

[Dong et al., Book 2015] [Elmagarmid et al., TKDE 2007] :

identify the different structures/records that model the same real-world object.



Why it is useful

example:
linking records to remove duplicates

- Improves data quality and integrity
- Fosters re-use of existing data sources
- Optimize space

Application areas:

Linked Data, Social Networks, census data, price comparison portals

good examples for entity linkage

Challenges for ER

- Variety – Semantic
  - Semi-structured data → unprecedented levels of heterogeneity
  - Numerous entity types & vocabularies
  - LOD Cloud*: ~50,000 predicates, ~12,000 vocabularies

different structures- structures/unstructured doesn't allow to combine the information

Outline

1. **Atomic similarity methods**

   working on specific single strings/values

Atomic String Similarity

single value

- Examples of targeting cases:
  - Publication authors: "John D. Smith" vs. "J. D. Smith"
  - Journal names: "Transactions on Knowledge and Data Engineering" vs. "Trans. Knowl. Data Eng."

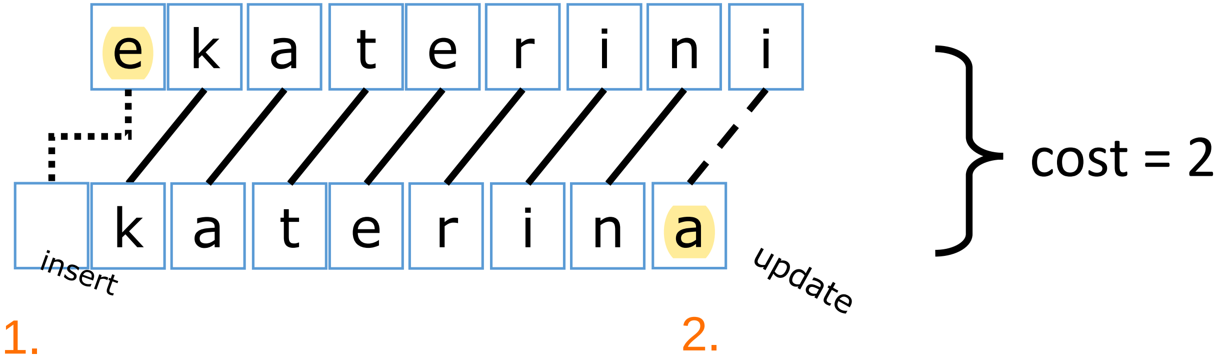
what level of similarity between the names

would be a distance of 7 for TRANSACTION

Edit Distance: very common technique

- Number of operations to convert from 1st to 2nd string
- Operations in Levenstein distance [Lev66]

→ delete, insert, and update a character with cost 1



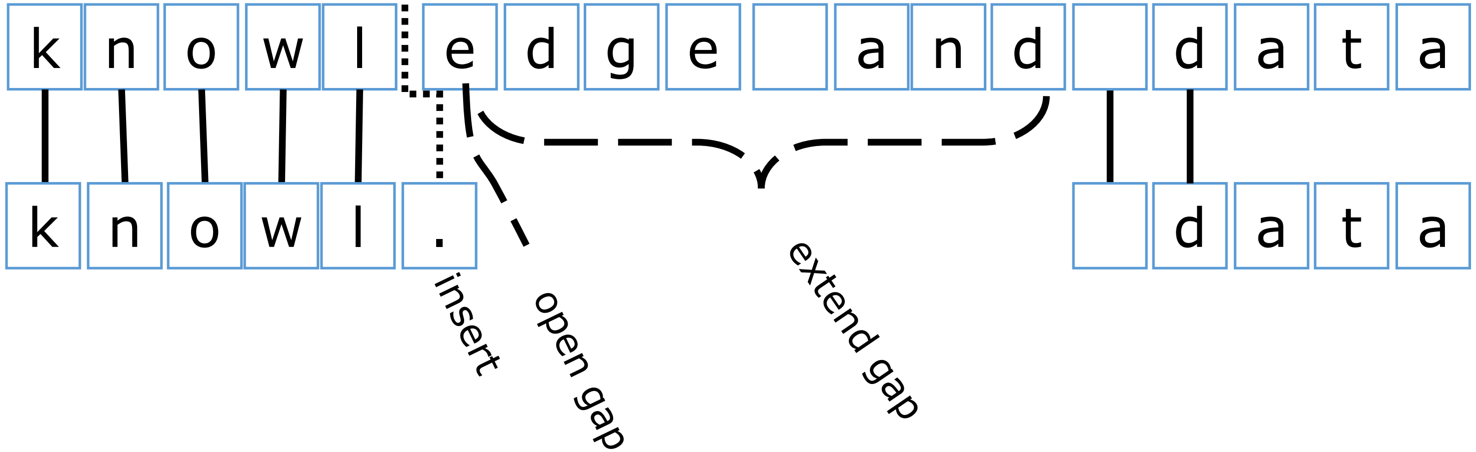
how many edits do we have to do to get from the first to the second string

Atomic String Similarity

- Gap Distance:
  - Overcome limitation of edit distance with shortened strings
  - Considers two extra operations [Nav01]

→ open gap, and extend gap (with small cost)

shorten the string distance metric



dot=insertion then gap until space that is matched with the first srting

cost = 1 + o + 8e

1. insertion+ open gap+8e

Atomic String Similarity

number of letters in common

- Jaro similarity [Jar89]:
  - Small strings, e.g., first and last names

C letters in common

T transposition

just the formular

$$JaroSim(S_1, S_2) = \frac{1}{3}\left(\frac{C}{|S_1|} + \frac{C}{|S_2|} + \frac{C - T}{C}\right)$$

|length of string|

C → common characters in S_1 and S_2

T →transpositions/2 (The number of matching (but different sequence order) characters divided by 2 defines the number oftranspositions.)

Example: “DEIS”vs. “DESI”

C=4, T=2/2,

JaroSim=$\frac{1}{3}\left(\frac{4}{4} + \frac{4}{4} + \frac{4-1}{4}\right) = 0.9167$

count number of letters in common= all of them

IS not in the same order

Atomic String Similarity

length= number of prefix letters common in the string (initial characters of the word)

- Jaro-Winkler similarity [Win99]:
  - Extension that gives higher weight to matching prefix
  - Increasing it’s applicability to names
  - $J_w(S_1, S_2) = JaroSim + P * L * (1 - JaroSim)$
  - P is a scaling factor (0.1 by default)
  - L is the length of the common prefix up to maximum 4
  - Example: Compute $J_w(arnab, aranb)$
    - $JaroSim(arnab, aranb) = \frac{1}{3}\left(\frac{5}{5} + \frac{5}{5} + \frac{4}{5}\right) = 0.933$
    - $J_w(arnab, aranb) = 0.933 + 0.1 * 2 * (1 - 0.933) = 0.9466$

only look at the first 4 characters; they are more likely to be in common

all letters are in common

letters in common

"na"has been swapped dividing trans. over 2 it will be 1 so 4/5 because C=5 and T=1 --> 5-1=4 -->4/5

JaroSim

L=2 because those are the ones in common AR

Atomic String Similarity

if it's the same sound then it'll get the same string

e.g. saying your name and another person writes it down differently

- Soundex: we won't apply it
  - Converts each word into a phonetic encoding by assigning the same code to the string parts that sound the same
  - Similarity is computed between the corresponding phonetic encodings
  - Examples: Meier and Mayer or Smith, Smyth, Smithe, Smeth, Smeeth

- Remarks:
  - Surveys: [CRF03], [Win06]
  - Existing API with these methods:
    - SecondString: http://secondstring.sourceforge.net/
    - SimMetrics: http://www.dcs.shef.ac.uk/~sam/simmetrics.html

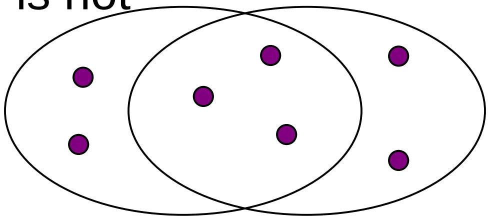
Outline

1. Atomic similarity methods
2. **Similarity methods for sets**

instead of similarity between strings/single values
similarity between bigger sets of values

Similarity methods for sets

example is not related



- Jaccard distance/similarity
  - The Jaccard similarity of two sets is the size of their intersection divided by the size of their union:
  - $$sim(C_1, C_2) = |C_1 \cap C_2| / |C_1 \cup C_2|$$

3 in intersection
8 in union
Jaccard similarity= 3/8
Jaccard distance = 5/8
Jaccard bag Similarity = 6/10

similarity

  - Jaccard distance: $d(C_1, C_2) = 1 - |C_1 \cap C_2| / |C_1 \cup C_2|$ 1-3/7
- Similarity between {a, b, c, d} and {a, b, e, f} = 2/6 = 1/3

- Jaccard bag similarity counts the repetition of the elements
- The similarity between $\{a,\overset{3}{a},a,b\}$ and $\{a,\overset{3}{a},b,b,c\}$ = 3/9 = 1/3

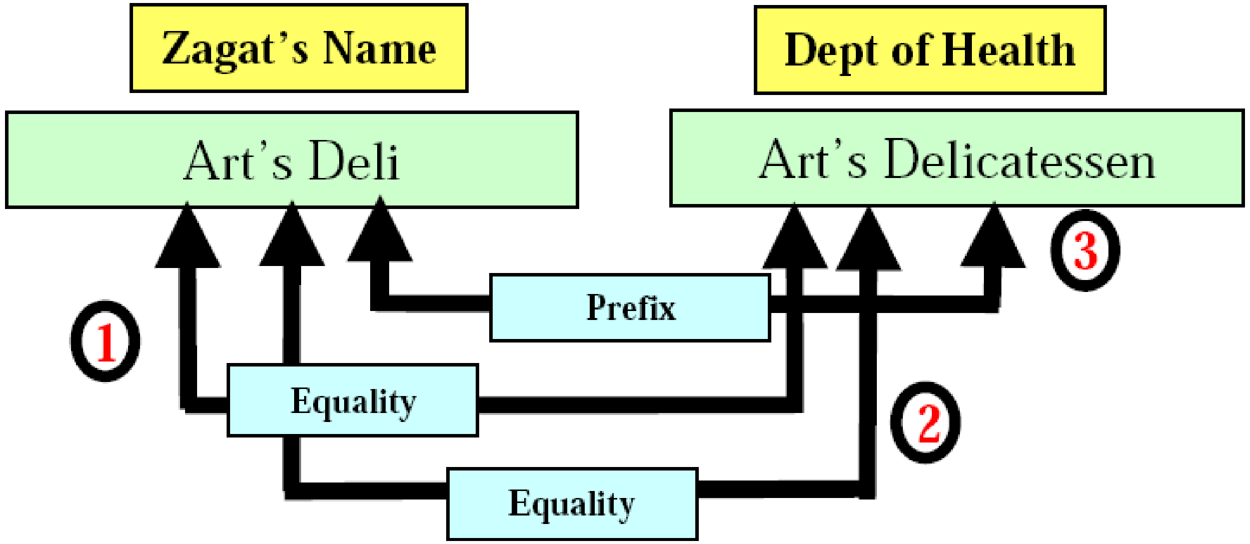
4   5   9

number of intersections over total number of values

Similarity methods for sets

- Using transformations [TKM02]:

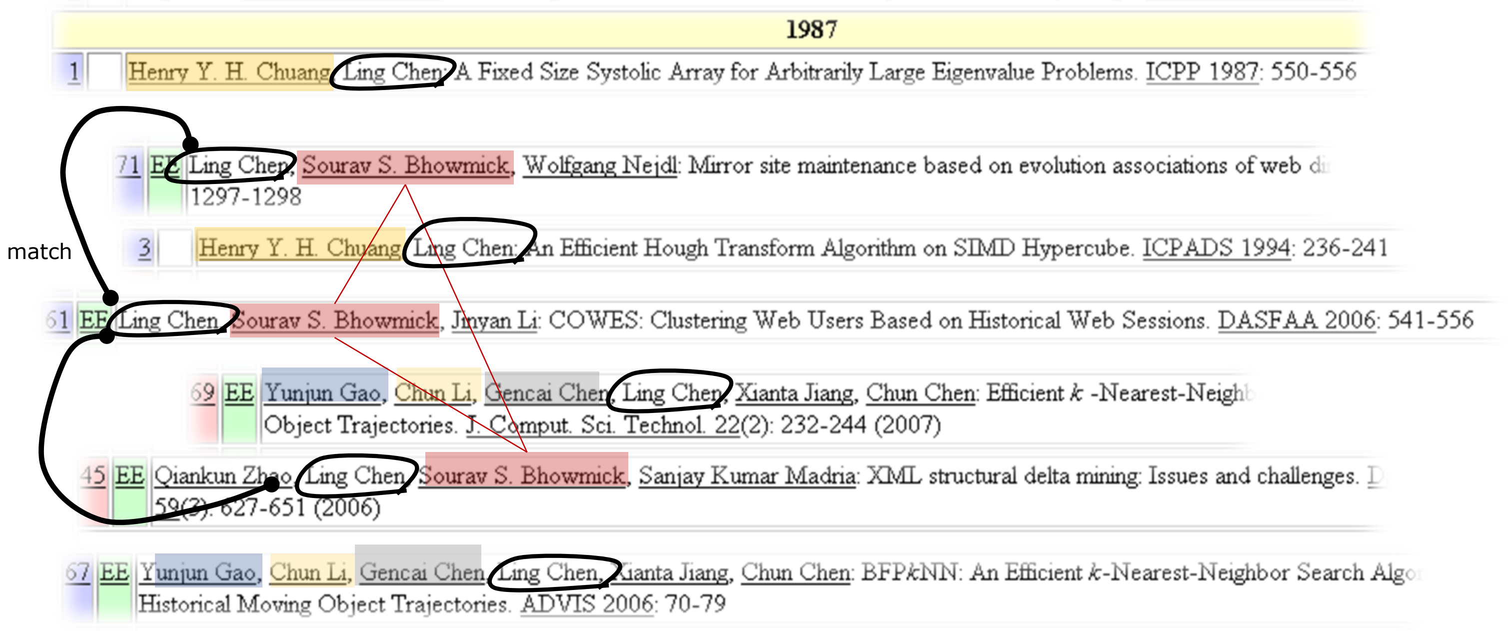
1. Analyze data to generate transformations to make the values comparable
   - Unary transform: just one simple Abbreviations
     - Equality, Stemming, Soundex, Abbreviation (e.g., 3rd or third)
   - N-ary transformations:
     - Initial, Prefix, Suffix, Substring Acronym, Abbreviation
2. Calculate transformation weights
3. Apply on candidate mappings



cost of transformation e.g. inserting/deleting character =1

Similarity methods for sets

- Group Linkage [OKLS07] (Survey [EIV07]):
  - Considers groups of relational records
    - not individual relational records
  - Groups match when:
    1. High similarity between data of individual records "Ling Chen"
    2. Large fraction of matching records, i.e., no. 1



1987

1 Henry Y. H. Chuang, Ling Chen: A Fixed Size Systolic Array for Arbitrarily Large Eigenvalue Problems. ICPP 1987: 550-556

71 EE Ling Chen, Sourav S. Bhowmick, Wolfgang Nejdl: Mirror site maintenance based on evolution associations of web d
1297-1298

3 Henry Y. H. Chuang, Ling Chen: An Efficient Hough Transform Algorithm on SIMD Hypercube. ICPADS 1994: 236-241

61 EE Ling Chen, Sourav S. Bhowmick, Jinyan Li: COWES: Clustering Web Users Based on Historical Web Sessions. DASFAA 2006: 541-556

69 EE Yunjun Gao, Chun Li, Gencai Chen, Ling Chen, Xianta Jiang, Chun Chen: Efficient k -Nearest-Neigh
Object Trajectories. J. Comput. Sci. Technol. 22(2): 232-244 (2007)

45 EE Qiankun Zhao, Ling Chen, Sourav S. Bhowmick, Sanjay Kumar Madria: XML structural delta mining: Issues and challenges. D
59(3): 627-651 (2006)

67 EE Yunjun Gao, Chun Li, Gencai Chen, Ling Chen, Xianta Jiang, Chun Chen: BFPkNN: An Efficient k-Nearest-Neighbor Search Algo
Historical Moving Object Trajectories. ADVIS 2006: 70-79

Outline

1. Motivation: Entity Resolution
2. Atomic similarity methods
3. Similarity methods for sets
4. **Facilitating inner-relationships**

Facilitating inner-relationships

- General idea
  - Heterogeneous data
    - Lack of schema information
    - Variations in entity descriptions
    - Incomplete or missing values
  - Improve effectiveness by considering data semantics
  - Example → Reference Reconciliation

Facilitating inner-relationships

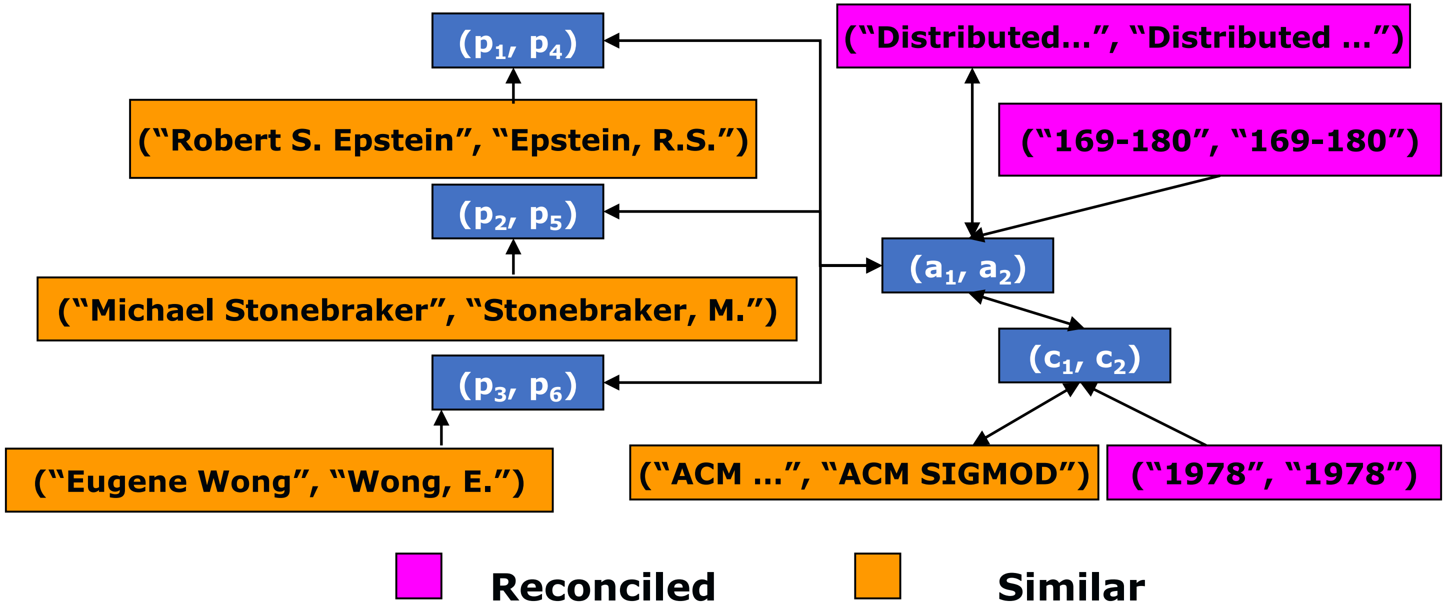
a1/a2 representing the articles

- Reference Reconciliation [DHM05]

1. Build a dependency graph

list of authors changed to abbr. and switched first and last name

a1: (“Distributed…”, “169-180, c1:(1978, “ACM…”), “Robert S. Epstein”, “Michael Stonebraker”, “Eugene Wong”)
a2: (“Distributed…”, “169-180, c2:(1978, “ACM SIGMOD”), “Epstein, R.S.”, “Stonebraker, M.”, “Wong, E.”)



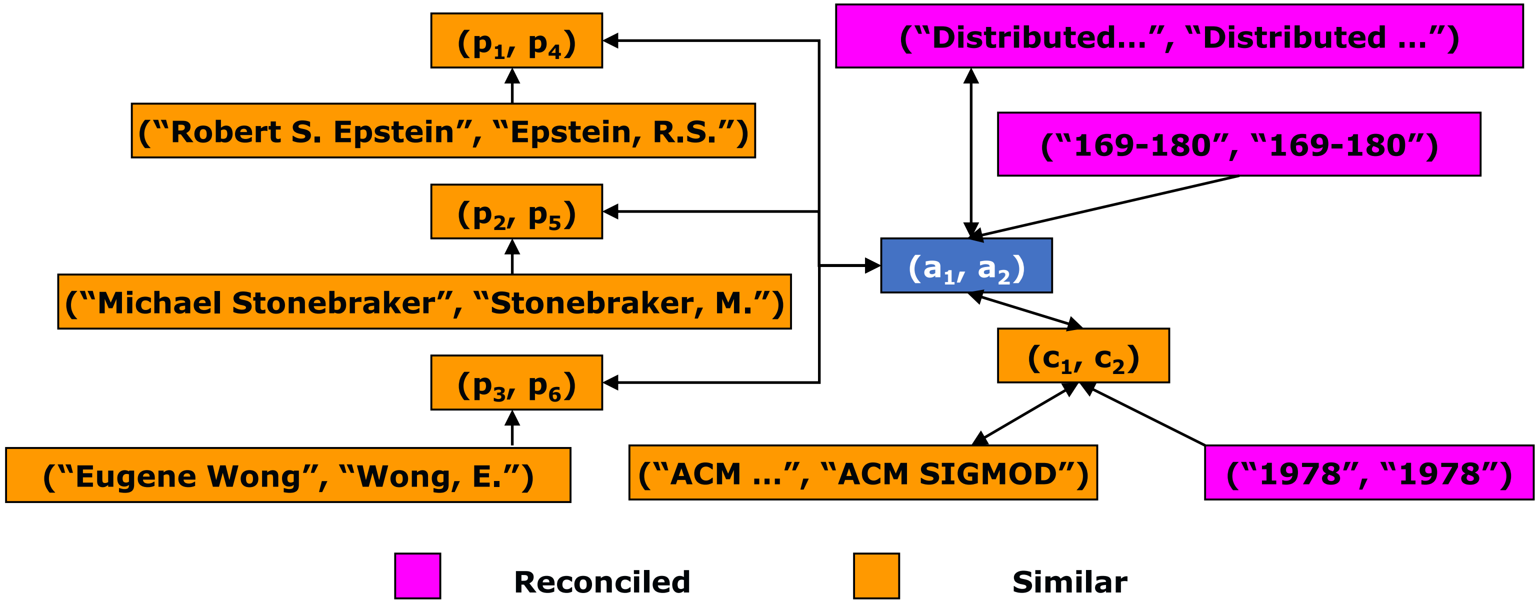
lines between blue elements represent the relations between the attributes

Facilitating inner-relationships

- Reference Reconciliation [DHM05]

1. Build a dependency graph
2. Exploit information and relationships

main idea is to find different information from dif. entities to see what values can match together and find out which records belong to the same attribute



(p_1, p_4)

("Robert S. Epstein", "Epstein, R.S.")

(p_2, p_5)

("Michael Stonebraker", "Stonebraker, M.")

(p_3, p_6)

("Eugene Wong", "Wong, E.")

("Distributed…", "Distributed …")

("169-180", "169-180")

(a_1, a_2)

(c_1, c_2)

("ACM …", "ACM SIGMOD")

("1978", "1978")

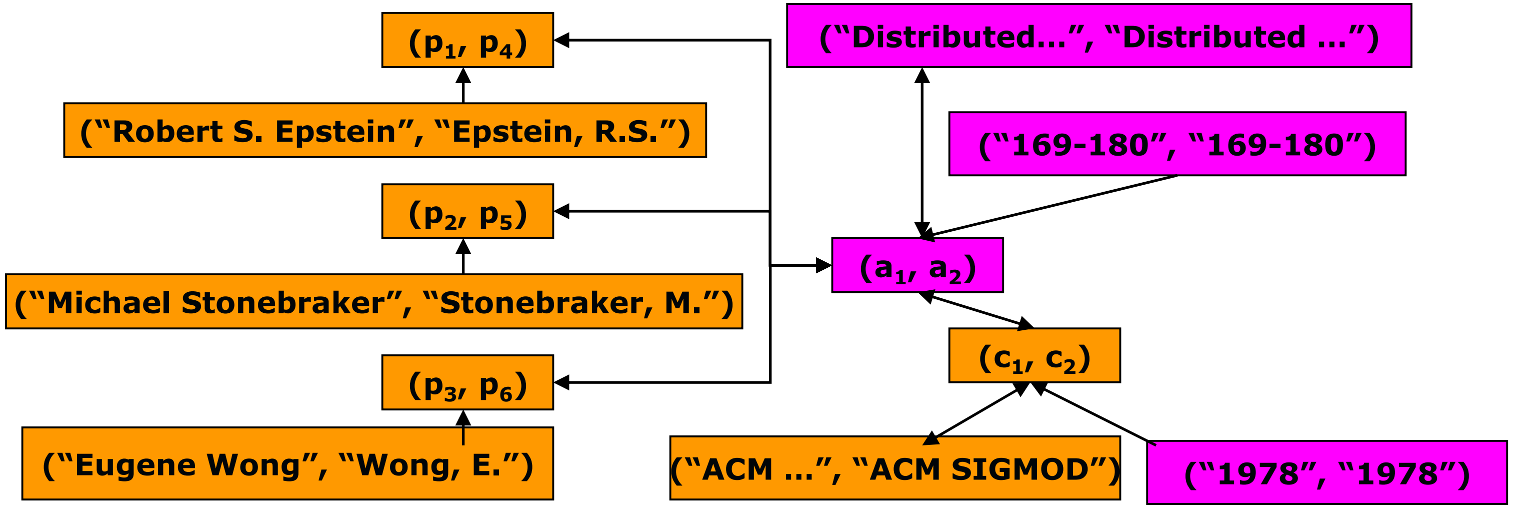
🟪 **Reconciled** 🟧 **Similar**

Utrecht University

Facilitating inner-relationships

- Reference Reconciliation [DHM05]

1. Build a dependency graph
2. Exploit information and relationships

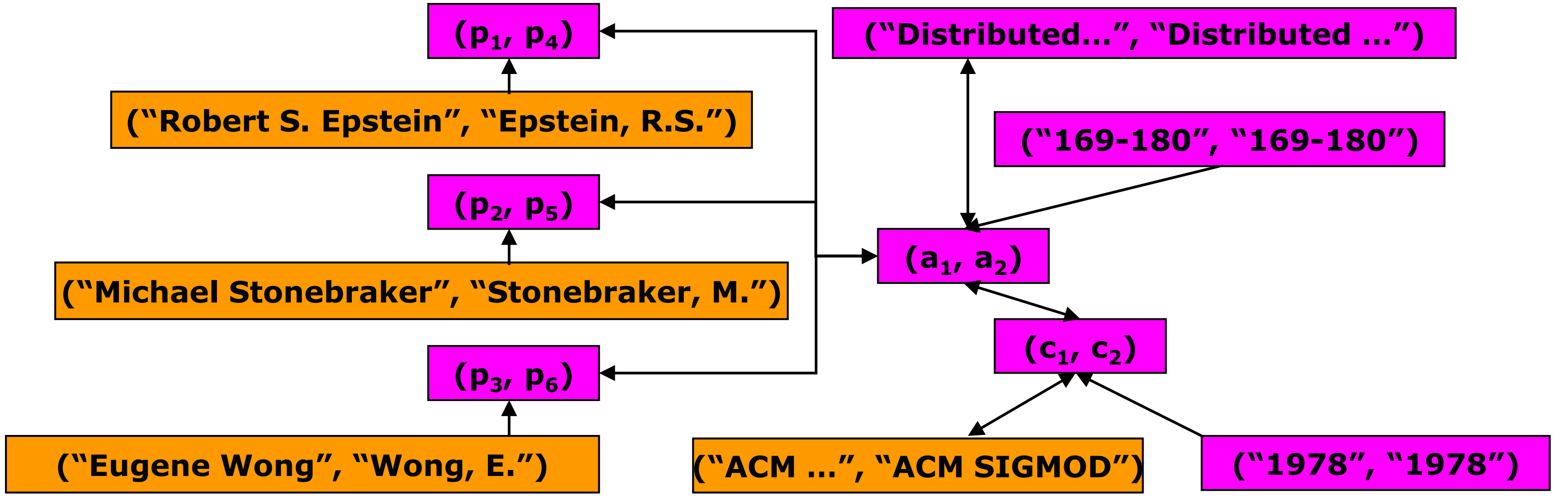


🟪 **Reconciled** 🟧 **Similar**

Facilitating inner-relationships

- Reference Reconciliation [DHM05]

1. Build a dependency graph
2. Exploit information and relationships



(p_1, p_4)

("Robert S. Epstein", "Epstein, R.S.")

(p_2, p_5)

("Michael Stonebraker", "Stonebraker, M.")

(p_3, p_6)

("Eugene Wong", "Wong, E.")

("Distributed…", "Distributed …")

("169-180", "169-180")

(a_1, a_2)

(c_1, c_2)

("ACM …", "ACM SIGMOD")

("1978", "1978")

🟪 **Reconciled** 🟧 **Similar**

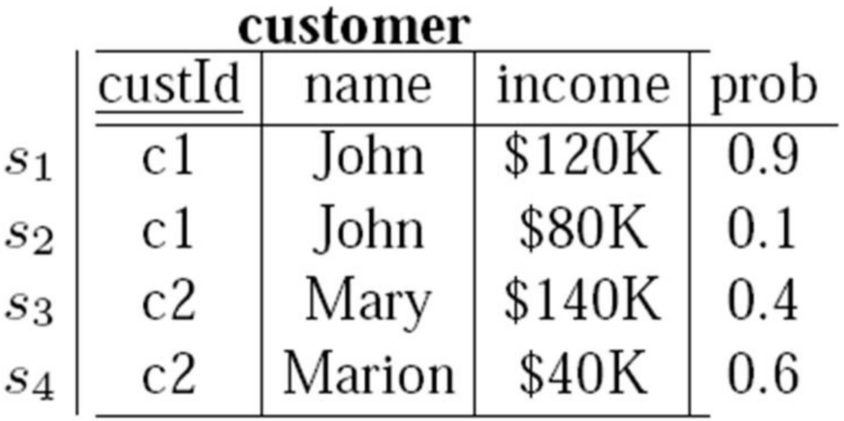
Outline

1. Motivation: Entity Resolution
2. Atomic similarity methods
3. Similarity methods for sets
4. Facilitating inner-relationships
5. **Methods in uncertain data**

Methods in uncertain data

- General idea:
- Keep conflicting relations, e.g., [AFM06], [RDS07], [DS07a], [DHY07]
    - Lack of resolution rules to correctly resolve and merge relations
    - No merging, but maintain results in the database
    - Relations are alternative representations of the same real world object
- Entity representation with probability – indicates…
    - Reliability of the source
    - Output of the matching process
    - Etc.

how strong is the connection between the entities



customer

	custId	name	income	prob
s_1	c1	John	\$120K	0.9
s_2	c1	John	\$80K	0.1
s_3	c2	Mary	\$140K	0.4
s_4	c2	Marion	\$40K	0.6

Methods in uncertain data

we don't want to change the values in the database as long as we don't know for sure that the values are the same

- Entity-Aware querying over prob. linkages (different perspectives)

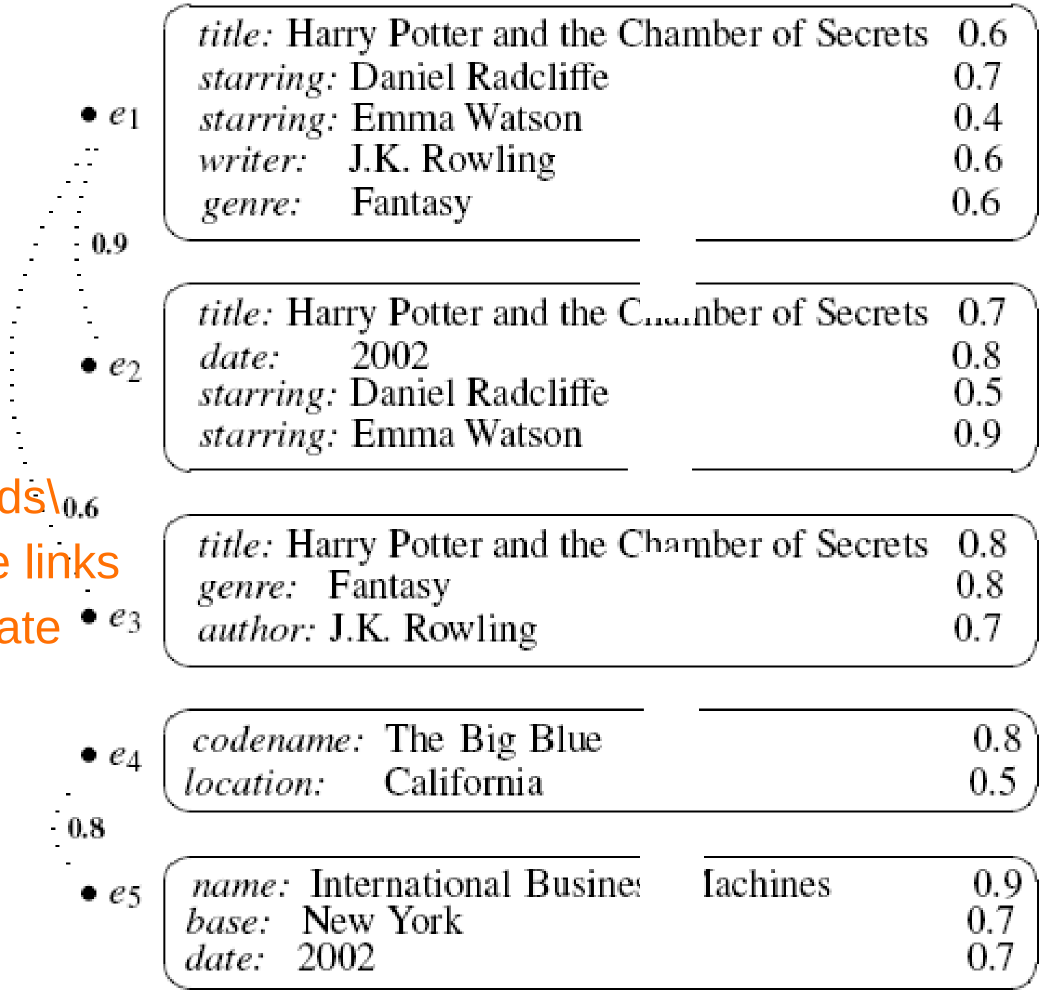
[Ioannou & Velegrakis 2010]

  - Not merging the entities using threshold
  - Keep probabilistic linkages alongside the original data
  - Use them during query processing
  - The idea of possible Worlds

e1 has a high probability to be linked to e2

- Query:
  - “J. K. Rowling” movies in “2002”
- Assume no linkages:
  - zero results
- Possible answer with linkages:
  - merge(e_1, e_2)
  - merge(e_1, e_2, e_3)

we can't find it in indiv. records\ but we can find it through the links to other attributes - e2 has date



title: Harry Potter and the Chamber of Secrets 0.6
starring: Daniel Radcliffe 0.7
starring: Emma Watson 0.4
writer: J.K. Rowling 0.6
genre: Fantasy 0.6

title: Harry Potter and the Chamber of Secrets 0.7
date: 2002 0.8
starring: Daniel Radcliffe 0.5
starring: Emma Watson 0.9

title: Harry Potter and the Chamber of Secrets 0.8
genre: Fantasy 0.8
author: J.K. Rowling 0.7

codename: The Big Blue 0.8
location: California 0.5

name: International Busines lachines 0.9
base: New York 0.7
date: 2002 0.7

Utrecht University

Similarity Methods for Sets

The case of documents

a document has a huge number of values for the same entity

a huge number of features = high dimension

A Common Metaphor

- Many problems can be expressed as finding “similar” sets
  - Find near-neighbors in <u>high-dimensional</u> space
- Examples:
  - Pages with similar words
    - For duplicate detection, classification by topic, plagiarism
  - Customers who purchased similar products (e.g. Movies)
  - Products with similar customer sets (e.g. fans)
  - Images with similar features
    - Users who visited similar websites

Documents as High-Dim Data

- **Converting documents to set**
  - **Simple approaches:**
    - Document = set of words appearing in document
    - Document = set of "important" words

- **Need to account for ordering of words!**
- A different way: **Shingles!**

Define: Shingles

- A *k*-shingle (or *k*-gram) for a document is a sequence of *k* tokens that appears in the doc
  - Tokens can be characters, words or something else, depending on the application
  - Assume tokens = characters for examples

n-grams are ususally words and shingles are characters

- **Example: k=2**; document **D_1** = abcab
  Set of 2-shingles: **$S(D_1)$** = {ab, bc, ca}
  - **Option:** Shingles as a bag (multiset), count ab twice: **$S'(D_1)$ =** {ab, bc, ca, ab}

space is not considered so if a shingle size is 5 and theres abcd then it will take the first letter after the space

Compressing Shingles

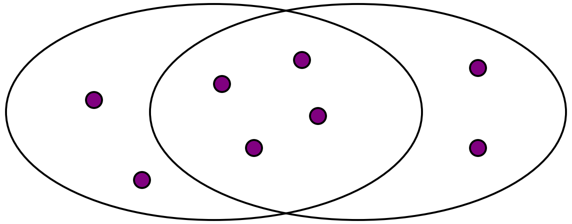
- To **compress long shingles**, we can **hash** them to (say) 4 bytes
- **Represent a document by the set of hash values of its *k*-shingles**
  - **Idea:** Two documents could (rarely) appear to have shingles in common, when in fact only the hash-values were shared
- **Example:** **k=2**; document $\mathbf{D_1}$= abcab
  Set of 2-shingles: $\mathbf{S(D_1)}$ = {ab, bc, ca}
  Hash the shingles: $\mathbf{h(D_1)}$ = {1, 5, 7}

mapped to 1,5,7

Similarity Metric for Shingles

- **Document D_1 is a set of its k-shingles $C_1=S(D_1)$**
- Equivalently, each document is a 0/1 vector in the space of *k*-shingles
  - Each unique shingle is a dimension
  - Vectors are very sparse
- **A natural similarity measure is the Jaccard similarity:**

$$sim(D_1, D_2) = |C_1 \cap C_2| / |C_1 \cup C_2|$$



Working Assumption

- **Documents that have lots of shingles in common have similar text, even if the text appears in different order**

- **Caveat:** You must pick ***k*** large enough, or most documents will have most shingles
  - ***k*** = 5 is OK for short documents
  - ***k*** = 10 is better for long documents

Challenges for ER

- Variety – Semantic
  - Semi-structured data → unprecedented levels of heterogeneity
  - Numerous entity types & vocabularies
  - LOD Cloud*: ~50,000 predicates, ~12,000 vocabularies
- Volume - Performance
  - Millions of entities
  - Billions of name-value pairs describing them
  - LOD Cloud*: >5,5·10^7 entities, ~1,5·10^{11} triples
  - **Too many documents, Too few memory**

Motivation

- **Suppose we need to find near-duplicate documents among $N = 1$ million documents**

- Naïvely, we would have to compute **pairwise Jaccard similarities** for **every pair of docs**
  - $N(N-1)/2 \approx$ **5*10^{11}** comparisons
  - At 10^5 secs/day and 10^6 comparisons/sec, it would take **5 days**

n-1 because we don't compare the value with itself

- For $N = 10$ million, it takes more than a year…

time complexity= how many steps/operations do I need to get my result; how many comparisons one value compared with all the values in a set- n-operations to compare them all with the value

instead of comparing it to all values compare it to candidate values

Find pairs of similar docs

Main idea: Candidates

Instead of keeping a count of each pair, only keep a count of candidate pairs!

-- Pass 1: Take documents and hash them to buckets such that documents that are similar hash to the same bucket

-- Pass 2: Only compare documents that are **candidates** (i.e., they hashed to a same bucket)

Benefits: Instead of $O(N^2)$ comparisons, we need $O(N)$ comparisons to find similar documents

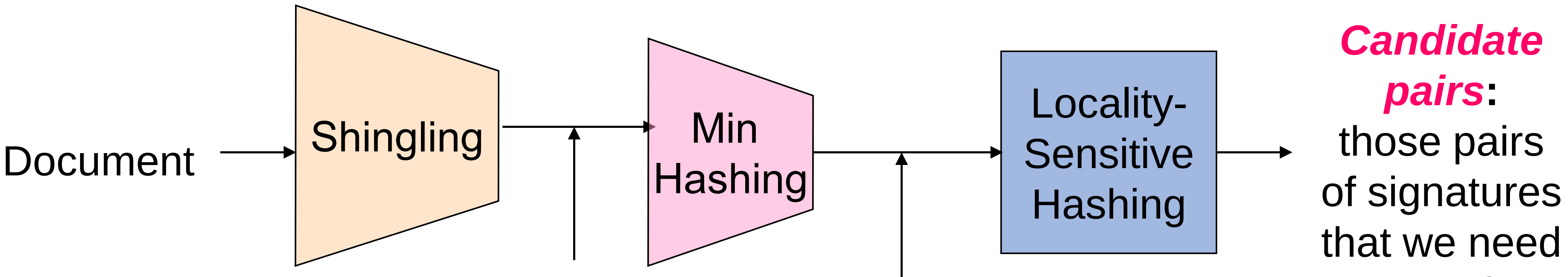
3 Essential Steps for Similar Docs

1. ***Shingling:*** Convert documents to sets

2. ***Min-Hashing:*** Convert large sets to short signatures, while preserving similarity

3. ***Locality-Sensitive Hashing:*** Focus on pairs of signatures likely to be from similar documents
    - **Candidate pairs!**

The Big Picture



Document

Shingling

Min Hashing

Locality-Sensitive Hashing

Candidate pairs: those pairs of signatures that we need to test for similarity

The set of strings of length ***k*** that appear in the document

Signatures: short integer vectors that represent the sets, and reflect their similarity

Distance Measures

- **Goal: Find near-neighbors in high-dim. space**
  - We formally define "near neighbors" as points that are a "small distance" apart
- For each application, we first need to define what "**distance**" means

From Sets to Boolean Matrices

- **Rows** = elements (shingles)
- **Columns** = sets (documents)
  - 1 in row ***e*** and column ***s*** if and only if ***e*** is a member of ***s***
  - Column similarity is the Jaccard similarity of the corresponding sets (rows with value *1)*
  - **Typical matrix is sparse!**
- **Each document is a column:**
  - **Example: sim(C_1 ,C_2) = ?**
    - Size of intersection = 3; size of union = 6, Jaccard similarity (not distance) = 3/6
    - **d(C_1,C_2) = 1 – (Jaccard similarity) = 3/6**

e.g. comparing similarity between columns(documents) 1 and 2?

Documents

Shingles

1	1	1	0
1	1	0	1
0	1	0	1
0	0	0	1
1	0	0	1
1	1	1	0
1	0	1	0

Finding Similar Columns

- **So far:**
  - Documents → Sets of shingles
  - Represent sets as Boolean vectors in a matrix
- **Next goal: Find similar columns while computing small signatures**
  - **Similarity of columns == similarity of signatures**

extracted signature from a column and compared it to another one should give an indication about the similarity between the two columns

Hashing Columns (Signatures)

- **Key idea:** "hash" each column ***C*** to a small ***signature h(C)***, such that:

  (1) ***h(C)*** is small enough that the signature fits in RAM

  (2) ***sim(C_1, C_2)*** is the same as the "similarity" of signatures ***h(C_1)*** and ***h(C_2)***

- **Goal: Find a hash function *h(·)* such that:**
  - If ***sim(C_1,C_2)*** is high, then with high prob. ***h(C_1) = h(C_2)***
  - If ***sim(C_1,C_2)*** is low, then with high prob. ***h(C_1) ≠ h(C_2)***

if similarity between
signatures is large=
similarity between
documents is large

- **Hash docs into buckets. Expect that "most" pairs of near duplicate docs hash into the same bucket!**

documents with similar shingles will be stored in same bucket

- **Clearly, the hash function depends on the similarity metric:**
  - Not all similarity metrics have a suitable hash function
- **There is a suitable hash function for the Jaccard similarity:** It is called **Min-Hashing**

Min-Hashing

- Imagine the rows of the Boolean matrix permuted under **random permutation** π

permutation= it's a combination without repetition

= means that you're shuffling the numbers- new set with random order

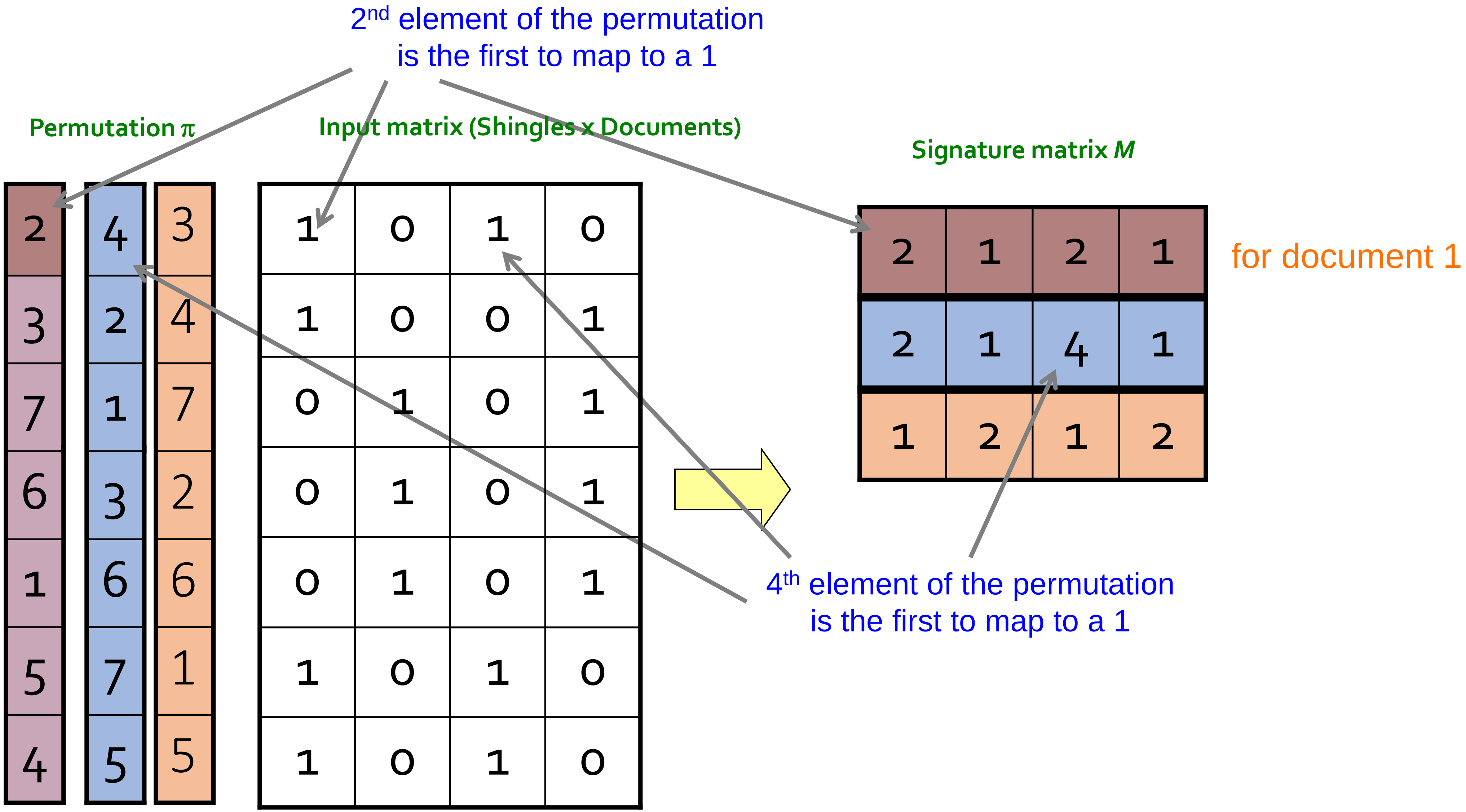
- Define a **"hash" function** $h_\pi(C)$ = the index of the **first** (in the permuted order π) row in which column C has value **1**:

$$h_\pi(C) = \min_\pi \pi(C)$$

- Use several (e.g., 100) independent hash functions (that is, permutations) to create a signature of a column

every time I shuffle the index of the rows and check the index of the first one so signature value is 1

Min-Hashing Example



2nd element of the permutation is the first to map to a 1

Permutation π

2	4	3
3	2	4
7	1	7
6	3	2
1	6	6
5	7	1
4	5	5

Input matrix (Shingles x Documents)

1	0	1	0
1	0	0	1
0	1	0	1
0	1	0	1
0	1	0	1
1	0	1	0
1	0	1	0

Signature matrix *M*

2	1	2	1
2	1	4	1
1	2	1	2

for document 1

4th element of the permutation is the first to map to a 1

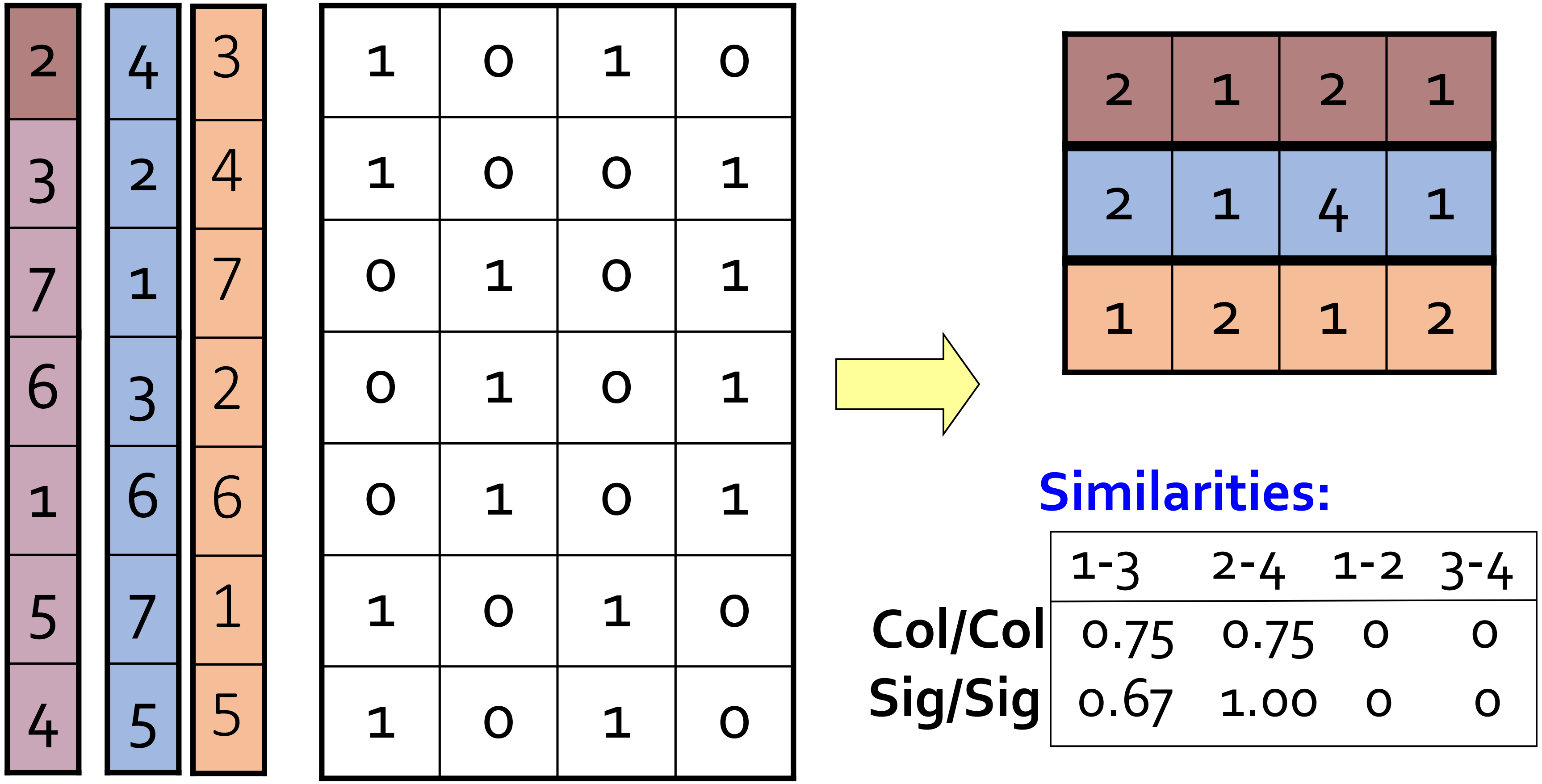
Similarity for Signatures

- We know: **$\Pr[h_\pi(C_1) = h_\pi(C_2)] = sim(C_1, C_2)$**
- Now generalize to multiple hash functions

- **The *similarity of two signatures* is the fraction of the hash functions in which they agree**

- **Note:** Because of the Min-Hash property, the similarity of columns is the same as the expected similarity of their signatures

Min-Hashing Example



Permutation π

2	4	3
3	2	4
7	1	7
6	3	2
1	6	6
5	7	1
4	5	5

Input matrix (Shingles x Documents)

1	0	1	0
1	0	0	1
0	1	0	1
0	1	0	1
0	1	0	1
1	0	1	0
1	0	1	0

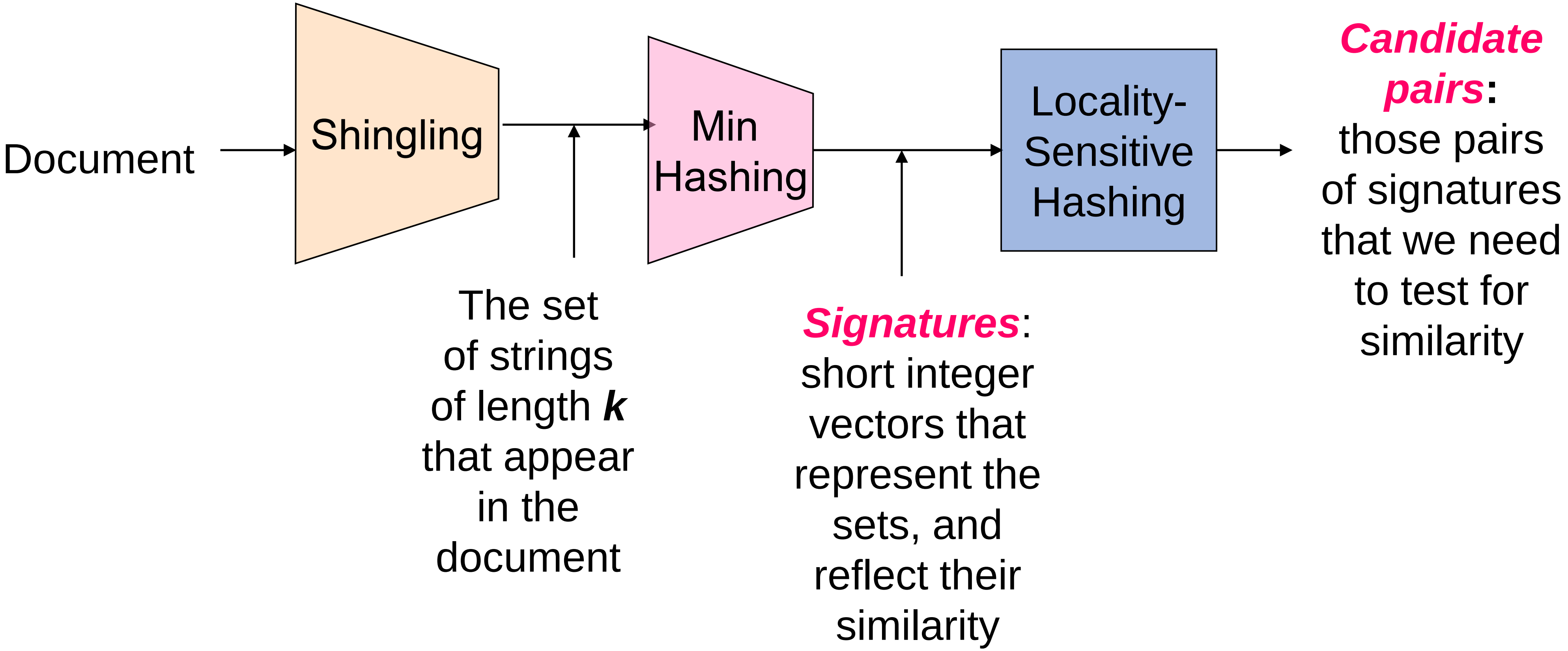
Signature matrix *M*

2	1	2	1
2	1	4	1
1	2	1	2

Similarities:

	1-3	2-4	1-2	3-4
Col/Col	0.75	0.75	0	0
Sig/Sig	0.67	1.00	0	0

Locality-Sensitive Hashing



Document

Shingling

Min Hashing

Locality-Sensitive Hashing

Candidate pairs: those pairs of signatures that we need to test for similarity

The set of strings of length ***k*** that appear in the document

Signatures: short integer vectors that represent the sets, and reflect their similarity

- **Step 3:** ***Locality-Sensitive Hashing:***
  Focus on pairs of signatures likely to be from similar documents

LSH for Min-Hash

- **Big idea: Hash columns of signature matrix *M* several times**

- Arrange that (only) **similar columns** are likely to **hash to the same bucket**, with high probability

- **Candidate pairs are those that hash to the same bucket**

- (Blocking)

Performance

- Merge-purge [HS95],[HS98]:
- Idea: same entities with share information
- Create a key for each relation (e.g., email)
- Sort relations according to key
- Compare only a limited set of relations in each iteration

Standard Blocking

Earliest, simplest form of blocking.

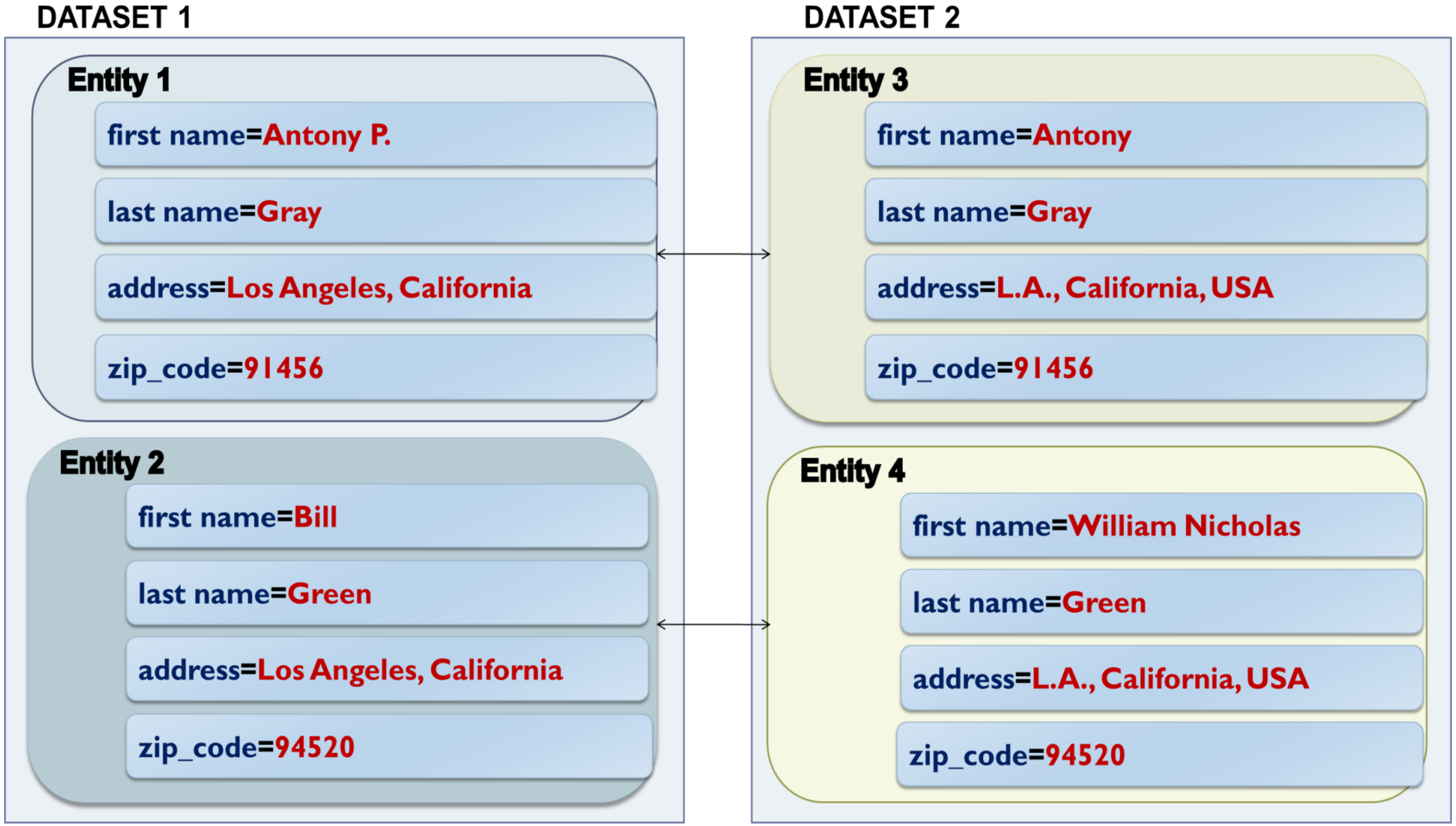
[Fellegi et. al., JASS 1969]

Algorithm:

1. Select the most appropriate attribute name(s) w.r.t. noise and distinctiveness.
2. Transform the corresponding value(s) into a Blocking Key (BK)
3. For each BK, create one block that contains all entities having this BK in their transformation.

Works as a hash function! → Blocks on the **equality** of BKs

Example of Standard Blocking



DATASET 1

Entity 1

first name=Antony P.

last name=Gray

address=Los Angeles, California

zip_code=91456

Entity 2

first name=Bill

last name=Green

address=Los Angeles, California

zip_code=94520

DATASET 2

Entity 3

first name=Antony

last name=Gray

address=L.A., California, USA

zip_code=91456

Entity 4

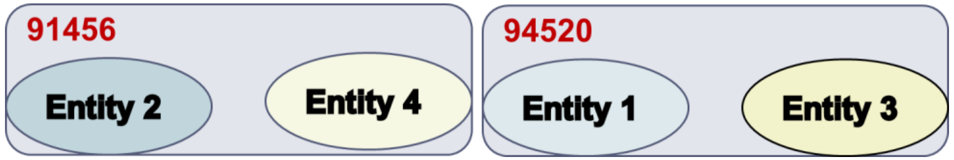
first name=William Nicholas

last name=Green

address=L.A., California, USA

zip_code=94520

Blocks on zip_code:



91456

Entity 2 **Entity 4**

94520

Entity 1 **Entity 3**

(c) Papadakis et al.

Thank you for your attention!

Questions?