

Data Wrangling and Data Analysis

Heterogeneous Data Analysis & String Similarity

Hakim Qahtan

Department of Information and Computing Sciences

Utrecht University



Utrecht University

Reading Material for Today

- Mining of Massive Datasets

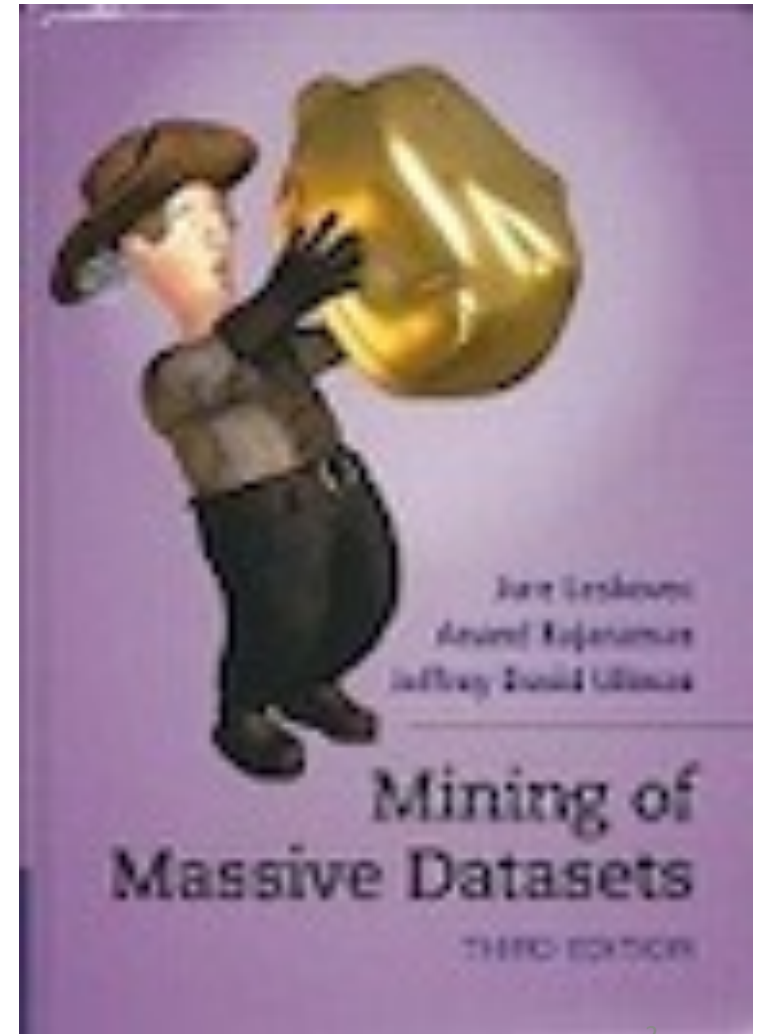
by Jure Leskovec, Anand Rajaraman, Jeff Ullman

<http://www.mmds.org>

Chapter 3.1 – 3.5



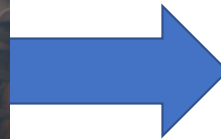
Utrecht University



Entity Linkage



The Wrong Paris



Utrecht University

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



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



London 런던 ਲੰਡਨ ਲंडन லண்டன் லண்டன் லண்டன் லண்டன் லண்டன் லண்டன்
लन्डन லண்டன் இலண்டன் லண்டன் 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 لندن
לונדון לונדון Λονδίνο Лёндан Лондан Лондон Лондон
Лондон Londen 伦敦 ...



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



London 런던 ਲੰਡਨ ਲंडन லண்டன் ロンドン
 लंडन லண்டன் இலண்டன் லண்டன் 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 لوندون
 לונדון לאנדאן Λονδίνο Лѣндан Лондан Лондон Лондон
 Лондон Llundain 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, ...



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



London 런던 ਲੰਡਨ ਲंडन லண்டன் லண்டன் லண்டன் லண்டன் லண்டன்
लंडन லண்டன் இலண்டன் லண்டன் 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 لوندون
לונדון לאנדאן Λονδίνο Лѣндан Лондан Лондон Лондон
Лондон Londen 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 ...

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.

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
identifv new issues and directions for future research and

The **Journal of Web Semantics** is an interdisciplir
various subject areas that contribute to the deve
service Web. These areas include: knowledge te
semantic grid, obviously disciplines like ... [click I](#)

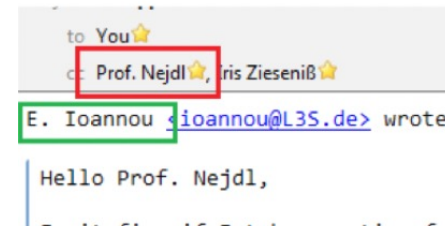


[Enrico Minack](#), [Kaluca](#), [...](#), [...](#), [...](#), [...](#), [...](#)
[Paul-Alexandru Chirita](#), [Wolfgang Nejdl](#): Leveraging personal metadat
system [J. Web Sem.](#) 8(1): 37-54 (2010)



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



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

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

Wolfgang Nejd1
L3S Research Center
Hannover, Germany
nejdl@L3S.de

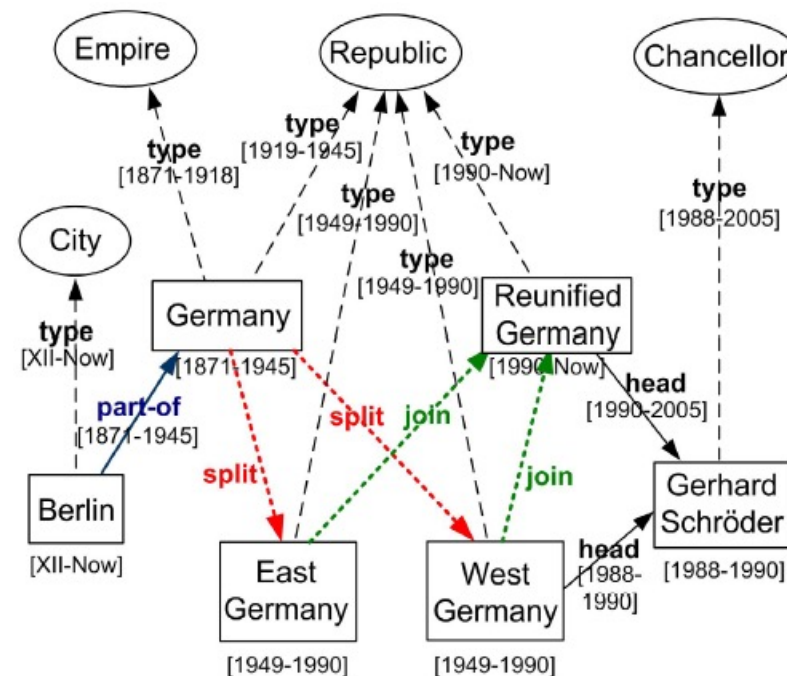
Claudia Niederée
L3S Research Center
Hannover, Germany
niederée@L3S.de

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



Reasons for Different Descriptions

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



Jacqueline Lee Bouvier

[\[Vel09\]](#)



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

[Elm07] :

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



Why it is useful

- 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



Challenges for ER

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

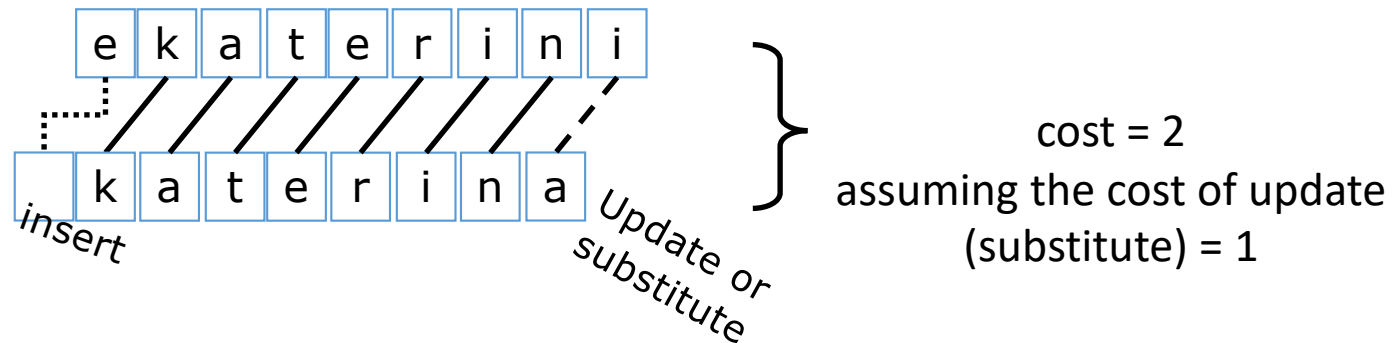


Atomic similarity methods



Atomic String Similarity – Edit Distance

- Number of operations to convert from 1st to 2nd string
- Operations in Levenstein distance [\[Lev66\]](#)
 - delete, insert, and update a character with cost 1



Computing Edit Distance – Another Example

- Example: compute the edit distance between intention and execution

I	N	T	E	*	N	T	I	O	N
*	E	X	E	C	U	T	I	O	N
d	s	s		i	s				

- If each operation has cost of 1
 - Distance between these is 5
- If substitutions cost 2 (Levenshtein)
 - Distance between them is 8



Computing Edit Distance Cont.)

- Dynamic programming: A tabular computation of $D(n, m)$
- Solving problems by combining solutions to subproblems.
- Bottom up
 - We compute $D(i, j)$ for small i, j
 - And compute larger $D(i, j)$ based on previously computed smaller values
 - i.e., compute $D(i, j)$ for all i ($0 < i < n$) and j ($0 < j < m$)



Defining Minimum Edit Distance (Levenshtein)

- Initialization

$$D(i, 0) = i$$

$$D(0, j) = j$$

- Recurrence Relation:

For each $i = 1 \dots M$

For each $j = 1 \dots N$

$$D(i, j) = \min \begin{cases} D(i-1, j) + 1 \\ D(i, j-1) + 1 \\ D(i-1, j-1) + \begin{cases} 2 & \text{if } X(i) \neq Y(j) \\ 0 & \text{if } X(i) = Y(j) \end{cases} \end{cases}$$

- Termination:

$D(N, M)$ is distance



Edit Distance Table – Example

N	9									
O	8									
I	7									
T	6									
N	5									
E	4									
T	3									
N	2									
I	1									
#	0	1	2	3	4	5	6	7	8	9
	#	E	X	E	C	U	T	I	O	N



Edit Distance Table – Example (Cont.)

N	9									
O	8									
I	7									
T	6									
N	5									
E	4									
T	3									
N	2									
I	1									
#	0	1	2	3	4	5	6	7	8	9
	#	E	X	E	C	U	T	I	O	N

$$D(i,j) = \min \begin{cases} D(i-1,j) + 1 \\ D(i,j-1) + 1 \\ D(i-1,j-1) + \begin{cases} 2 & \text{if } w1(i) \neq w2(j) \\ 0 & \text{if } w1(i) = w2(j) \end{cases} \end{cases}$$



Edit Distance Table – Example (Cont.)

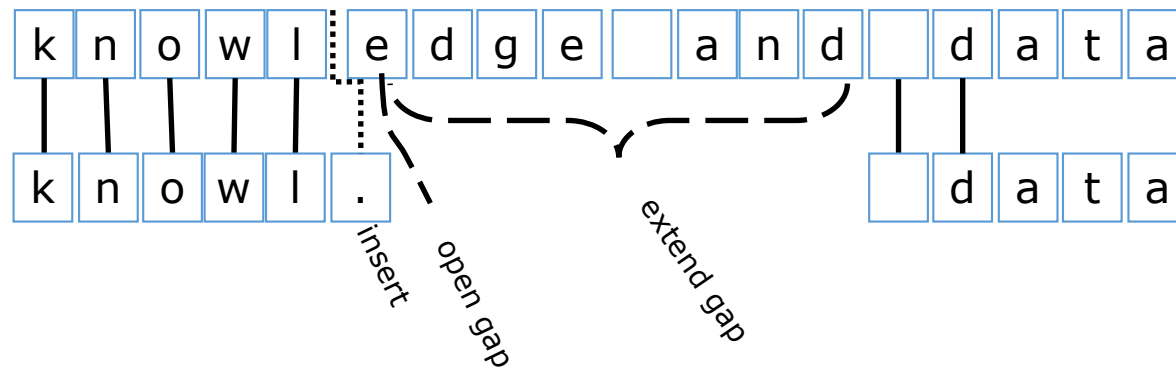
N	9	8	9	10	11	12	11	10	9	8
O	8	7	8	9	10	11	10	9	8	9
I	7	6	7	8	9	10	9	8	9	10
T	6	5	6	7	8	9	8	9	10	11
N	5	4	5	6	7	8	9	10	11	10
E	4	3	4	5	6	7	8	9	10	9
T	3	4	5	6	7	8	7	8	9	8
N	2	3	4	5	6	7	8	7	8	7
I	1	2	3	4	5	6	7	6	7	8
#	0	1	2	3	4	5	6	7	8	9
	#	E	X	E	C	U	T	I	O	N



Atomic String Similarity – Gap Distance

- Overcome limitation of edit distance with shortened strings
- Considers two extra operations

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

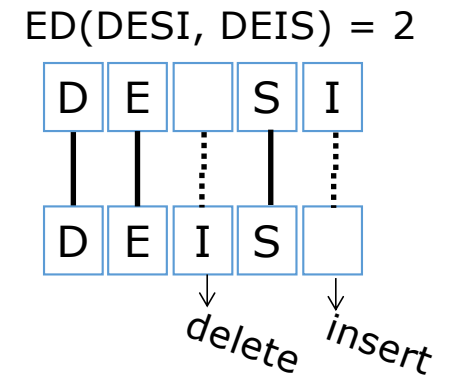


$$\text{cost} = 1 + o + 8e$$



Atomic String Similarity – Jaro Similarity

- Small strings, e.g., first and last names
- C is the set of common characters in S_1 and S_2
 - Two characters from S_1 and S_2 are considered *common* when they are the same and not farther than $\left\lfloor \frac{\max(|S_1|, |S_2|)}{2} \right\rfloor - 1$ characters apart.
- T is the number transpositions/2
 - c_1 and c_2 are a transposition if c_1 and c_2 are **common** but appear in different orders in S_1 and S_2



$$JaroSim(S_1, S_2) = \frac{1}{3} \left(\frac{C}{|S_1|} + \frac{C}{|S_2|} + \frac{C - T}{C} \right) \quad [\text{Jar89}]$$

- Example: “DEIS” vs. “DESI”

$$C=4, T=2/2, \quad JaroSim = \frac{1}{3} \left(\frac{4}{4} + \frac{4}{4} + \frac{4-1}{4} \right) = 0.9167$$



Atomic String Similarity

- 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 $J_w(arnab, aramb)$
 - $JaroSim(arnab, aramb) = \frac{1}{3} \left(\frac{5}{5} + \frac{5}{5} + \frac{4}{5} \right) = 0.933$
 - $J_w(arnab, aramb) = 0.933 + 0.1 * 2 * (1 - 0.933) = 0.9466$



Atomic String Similarity

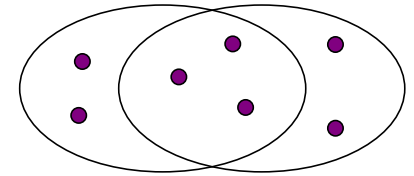
- Soundex: A phonetic algorithm that indexes names by their sounds when pronounced in English.
- Consists of the first letter of the name followed by three numbers. Numbers encode similar sounding consonants.
 - Remove all W, H
 - B, F, P, V encoded as 1, C,G,J,K,Q,S,X,Z as 2
 - D,T as 3, L as 4, M,N as 5, R as 6, Remove vowels
 - Concatenate first letter of string with first 3 numerals
- Ex: great and grate become G6EA3 and G6A3E and then G630
- More recent, metaphone, double metaphone etc.



Similarity methods for sets



Similarity methods for sets



- **Jaccard similarity/distance**

- The **Jaccard similarity** of two **sets** is:

Is J_{dist} a distance measure?

$$J_{sim}(C_1, C_2) = \frac{|C_1 \cap C_2|}{|C_1 \cup C_2|}$$

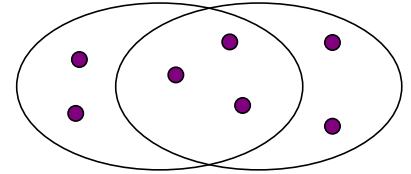
3 in intersection
7 in union
Jaccard similarity = $3/7$
Jaccard distance = $4/7$

- **Jaccard distance:** $J_{dist} = 1 - J_{sim}(C_1, C_2) = 1 - \frac{|C_1 \cap C_2|}{|C_1 \cup C_2|}$

- 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 $\{a, a, a, b\}$ and $\{a, a, b, b, c\} = 3/9 = 1/3$



Similarity methods for sets



- **Sørensen Coefficient** (also called Coefficient of Community CC)

- The **Sørensen similarity** of two **sets** is computed as:

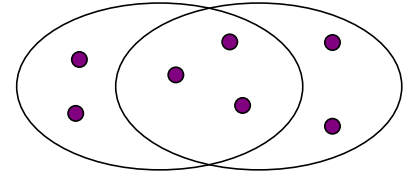
$$CC(C_1, C_2) = \frac{2 * |C_1 \cap C_2|}{|C_1| + |C_2|}$$

3 in intersection
5 in each set
Sørensen similarity= 6/10

- Similarity between {a, b, c, d} and {a, b, e, f} = 4/8 = 1/2
- Gives more weight for the number of common elements



Similarity methods for sets



- **Tversky Index:** a generalized form of Jaccard and Sørensen
 - The **Tversky Index** of two **sets** is computed as:

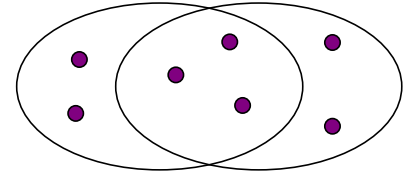
$$S(C_1, C_2) = \frac{|C_1 \cap C_2|}{|C_1 \cap C_2| + \alpha |C_1 - C_2| + \beta |C_2 - C_1|}$$

- $\alpha, \beta \geq 0$
- $\alpha = \beta = 1 \Rightarrow$ Jaccard similarity
- $\alpha = \beta = 0.5 \Rightarrow$ Sørensen similarity

3 in intersection
2 in the difference
 $\alpha = 0.2$ & $\beta = 0.8$
Tversky similarity = $3/5$



Similarity methods for sets



- **Overlap Coefficient:** also called Szymkiewicz–Simpson coefficient
 - It is defined as:

$$OC(C_1, C_2) = \frac{|C_1 \cap C_2|}{\min(|C_1|, |C_2|)}$$

3 in intersection
5 in each set
Overlap coefficient = 3/5



Case of Documents



A Common Metaphor

- Many problems can be expressed as finding “similar” sets
 - Find near-neighbors in high-dimensional 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



Shingles

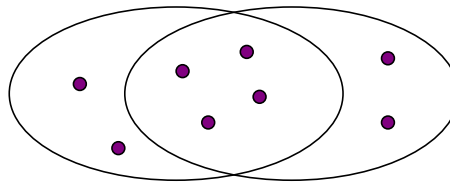
- A **k -shingle** (or **k -gram**) for a document is a sequence of k tokens that appears in the doc
 - Tokens can be **characters**, **words** or something else, depending on the application
 - Assume tokens = characters for examples
- **Example:** $k=2$; document $D_1 = \text{abcab}$
Set of 2-shingles: $S(D_1) = \{\text{ab}, \text{bc}, \text{ca}\}$
 - **Option:** Shingles as a bag (multiset), count ab twice: $S'(D_1) = \{\text{ab}, \text{bc}, \text{ca}, \text{ab}\}$



Similarity Metric for Shingles

- Represent document D_1 as 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**:

$$J_{sim}(C_1, C_2) = \frac{|C_1 \cap C_2|}{|C_1 \cup C_2|}$$



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*: $>5,5 \cdot 10^7$ entities, $\sim 1,5 \cdot 10^{11}$ 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**
 - $N(N - 1)/2 \approx 5 \cdot 10^{11}$ comparisons
 - At 10^5 secs/day and 10^6 comparisons/sec, it would take **5 days**
- For $N = 10$ million, it takes more than a year...



Find Pairs of Similar Documents

- **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



How could we use hashing to convert a document to a Sparse Boolean Vector (where each index represents a different word)?

Hash Tables: Basic Idea

- Use a key (arbitrary string or number) to index directly into an array – $O(1)$ time to access records
 - $A["brand"] = "Ford"$
 - Need a *hash function* to convert the key to an integer

	Key	Data
0	brand	ford
1	color	orange
2	kiwi	Australian fruit

Characteristics of a Good Hashing Function

- Returns an integer between 0 and the table size
- Efficiently computable
- Does not waste extra space
- At least one key is hashed to every integer between 0 and the table size
- Minimizes the collisions: the different keys that hash to the same number



Examples of Hashing Functions

- For **integer keys**: x is the key and m is the table size

- $h_1(x) = x \% m$ (% is the modulus function)

- $h_2(x) = x(x + 3) \% m$

- Multiplication hashing function

- Select $0 < c < 1$ and compute $w = xc$
- Take $u = \text{fraction part of } w$
- $h_3(x) = \lfloor um \rfloor$

$$m = 15, \quad c = 0.3$$

x	$h_1(x)$	$h_2(x)$	$h_3(x)$
36	6	9	12
51	6	9	4
8	8	13	6
18	3	3	6
9	9	3	10
47	2	10	1



Examples of Hashing Functions

- For **string keys**: x is the key and m is the table size
 - $h_1(x) = \text{sum}(\text{ascii}(x[i])) \% m, \quad 0 \leq i < \text{length}(x)$
 - **Problem**: string with the same set of characters hash to the same number ('abc', 'bca', 'acb', ...)
 - **Solution**: consider the string to be integer with base 128
 - $h_2(x) = \text{sum}(\text{ascii}(x[i]) * 128^i) \% m, \quad 0 \leq i < \text{length}(x)$
- **Example**: use h_1, h_2 to hash the strings ``abc'', ``acb'' (table size $m = 15$)
 - $h_1(abc) = 97 + 98 + 99 = 294 \% 15 = 9$
 - $h_1(acb) = 294 \% 15 = 9$
 - $h_2(abc) = ((97 * 128^2) + (98 * 128) + (99 * 1)) \% 15 = 11$
 - $h_2(acb) = ((97 * 128^2) + (99 * 128) + (98 * 1)) \% 15 = 3$



Finding similar documents requires more than simple hashing functions

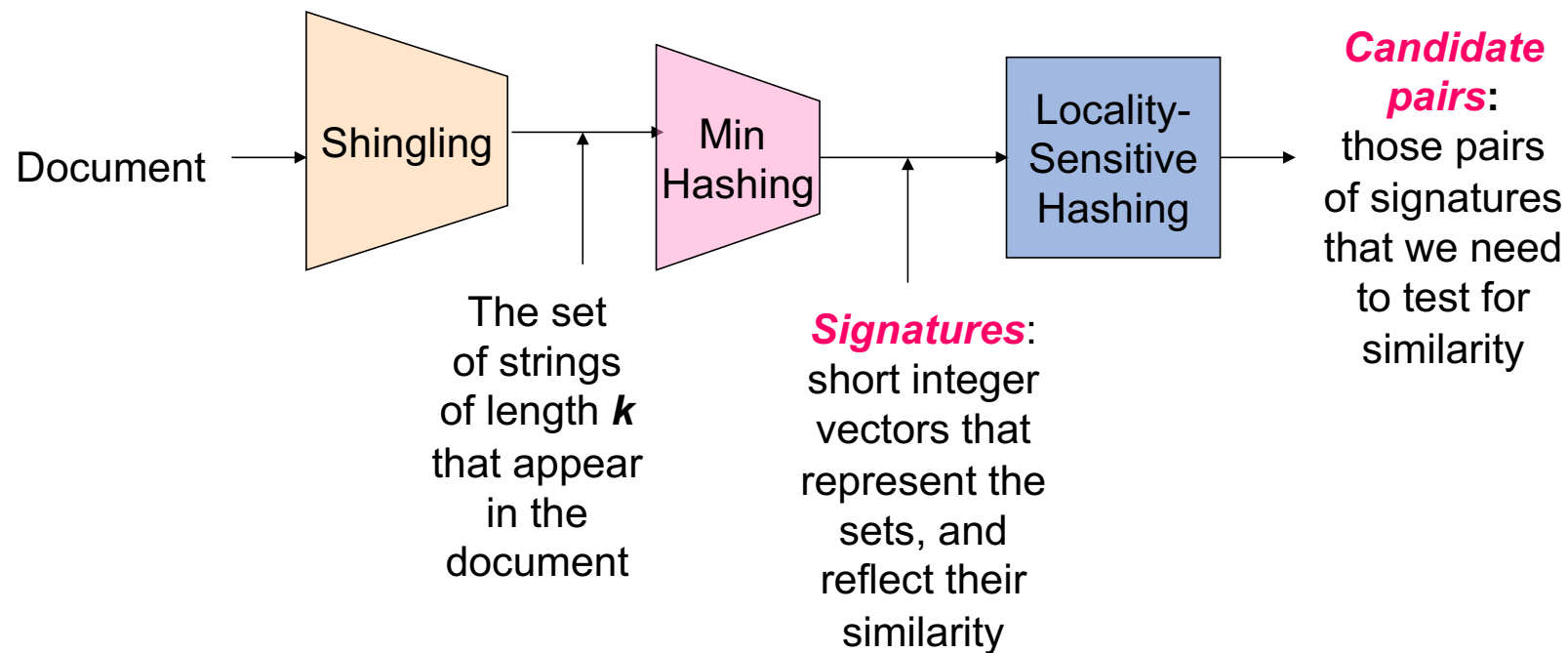


3 Essential Steps for Similar Docs

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!**



3 Essential Steps for Similar Docs



3 Essential Steps for Similar Docs

- **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:** $J_{sim}(C_1, C_2) = ?$
 - Size of intersection = 3; size of union = 6, Jaccard similarity (not distance) = $3/6$
 - $d(C_1, C_2) = 1 - (\text{Jaccard similarity}) = 3/6$

		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 $\rightarrow$ Sets of shingles
 - Represent sets as Boolean vectors in a matrix
- **Next goal: Find similar columns while computing small signatures**
 - Similarity of columns $\approx$ similarity of signatures



Hashing

- **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) $\text{sim}(C_1, C_2)$ is the same as the “similarity” of signatures $h(C_1)$ and $h(C_2)$
- **Goal:** Find a hash function $h(\cdot)$ such that:
 - If $\text{sim}(C_1, C_2)$ is high, then with high prob. $h(C_1) = h(C_2)$
 - If $\text{sim}(C_1, C_2)$ is low, then with high prob. $h(C_1) \neq h(C_2)$
- **Hash docs into buckets. Expect that “most” pairs of near duplicate docs hash into the same bucket!**



Min-Hashing

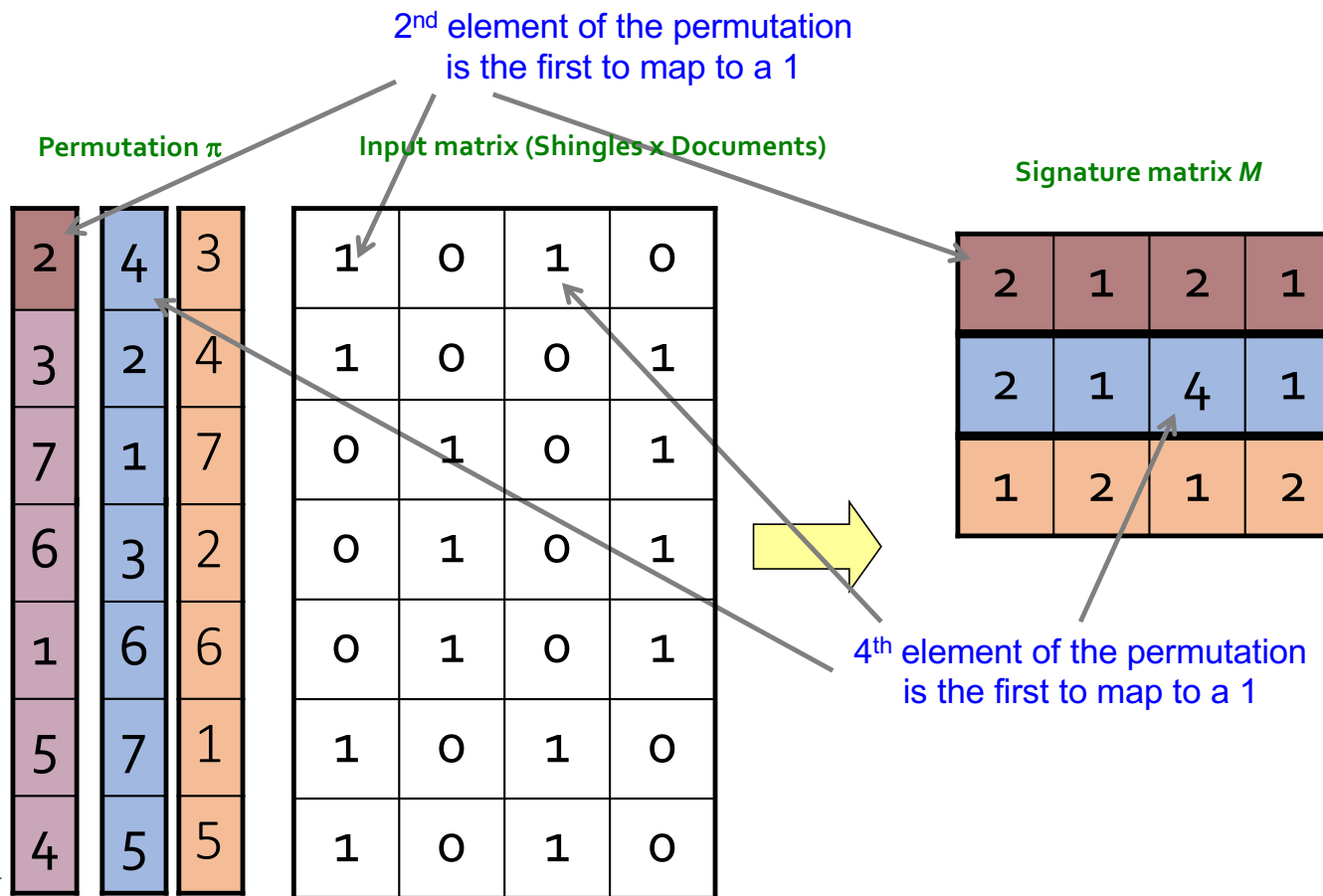
- Imagine the rows of the Boolean matrix permuted under **random permutation** π
- Define a **“hash” function** $h_{\pi}(C)$ = the index of the **first** (in the permuted order π) row in which column C has value **1**:

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

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



Min-Hashing Example



Similarity of Signatures

- Clearly: $\Pr[h_{\pi}(C_1) = h_{\pi}(C_2)] = \text{sim}(C_1, C_2)$
- 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



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)



Standard Blocking

Algorithm:

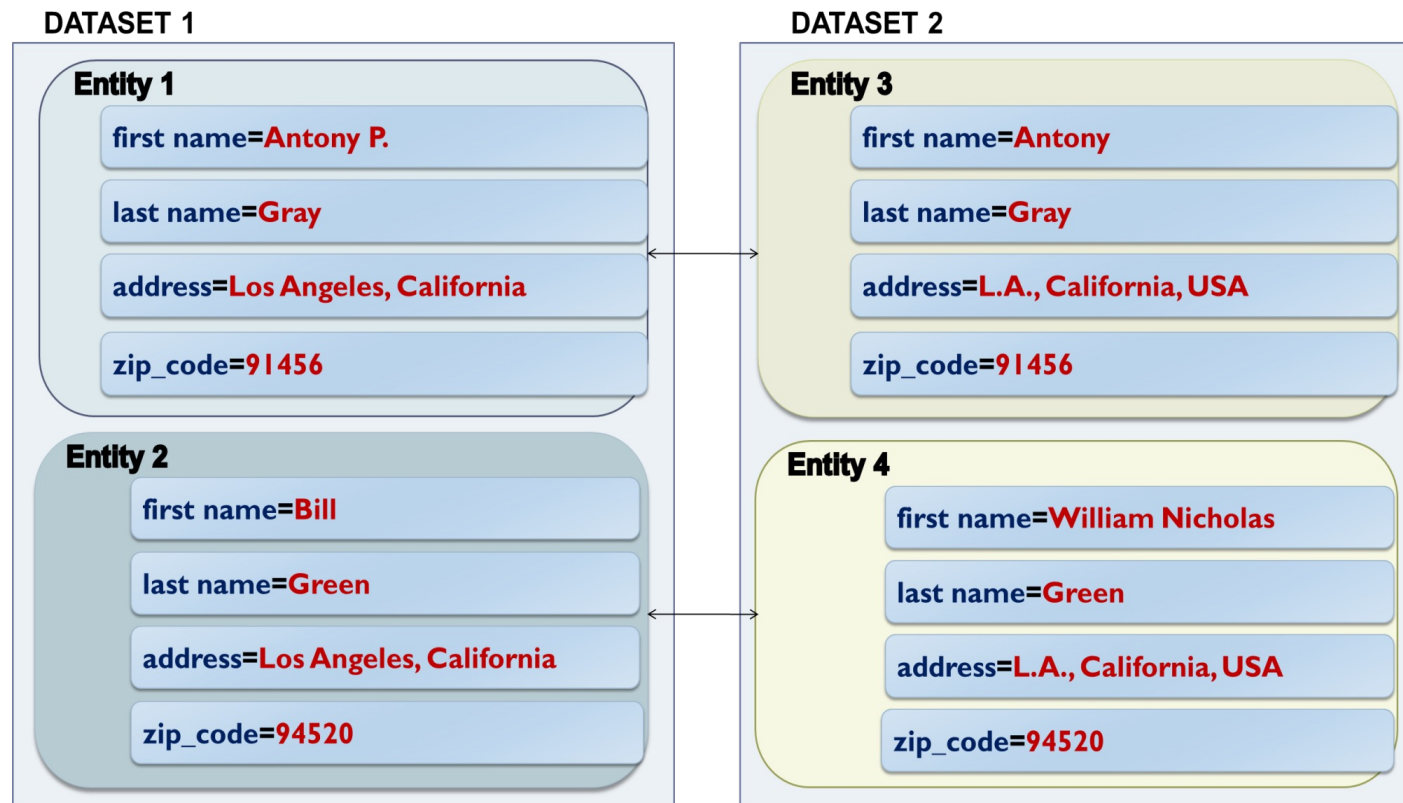
1. Select the most appropriate attribute name(s) w.r.t. noise and distinctiveness.
2. Transform the corresponding value(s) into a Blocking Key (BK)
3. For each BK, create one block that contains all entities having this BK in their transformation.

[Fellegi et. al., JASS 1969]

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



Standard Blocking – Example



Blocks on zip_code:



Thank you for your attention!

Questions?



Disclaimer: Much of the material presented originates from a number of different presentations and courses of the following people: Yannis Velegrakis (Utrecht University), Jeff Ullman (Stanford University), Bill Howe (U of Washington), Martin Fouler (Thought Works), Ekaterini Ioannou (Tilburg University), Themis Palpanas (U of Paris-Descartes). Copyright stays with the authors. No distribution is allowed without prior permission by the authors.



Additional References

- [Jar89] M. A. Jaro: Advances in record linkage methodology as applied to matching the 1985 census of Tampa, Florida. Journal of the American Statistical Association 84: 414-420.
- [Win99] William E. Winkler: The state of record linkage and current research problems. IRS publication R99/04 (<http://www.census.gov/srd/www/byname.html>)
- Fellegi, I. P. and Sunter, A. B. (1969). A theory for record linkage. Journal of the American Statistical Association, 64(328):1183–1210.
- [Lev66] Levenshtein, Vladimir I. (February 1966). "Binary codes capable of correcting deletions, insertions, and reversals". Soviet Physics Doklady. 10 (8): pp. 707–710.
- [Vel09] Rizzolo, F., Velegrakis, Y., Mylopoulos, J., Bykau, S. (2009). Modeling Concept Evolution: A Historical Perspective. In: Laender, A.H.F., Castano, S., Dayal, U., Casati, F., de Oliveira, J.P.M. (eds) Conceptual Modeling - ER 2009. ER 2009. Lecture Notes in Computer Science, vol 5829. Springer, Berlin, Heidelberg.
- [Elm07] A. K. Elmagarmid, P. G. Ipeirotis and V. S. Verykios, "Duplicate Record Detection: A Survey," in IEEE Transactions on Knowledge and Data Engineering, vol. 19, no. 1, pp. 1-16, Jan. 2007.
- [Don15]





- Summarize what you learned today in 2-minutes



The information in this presentation has been compiled with the utmost care,
but no rights can be derived from its contents.