DSE 203 (Fall 2020)

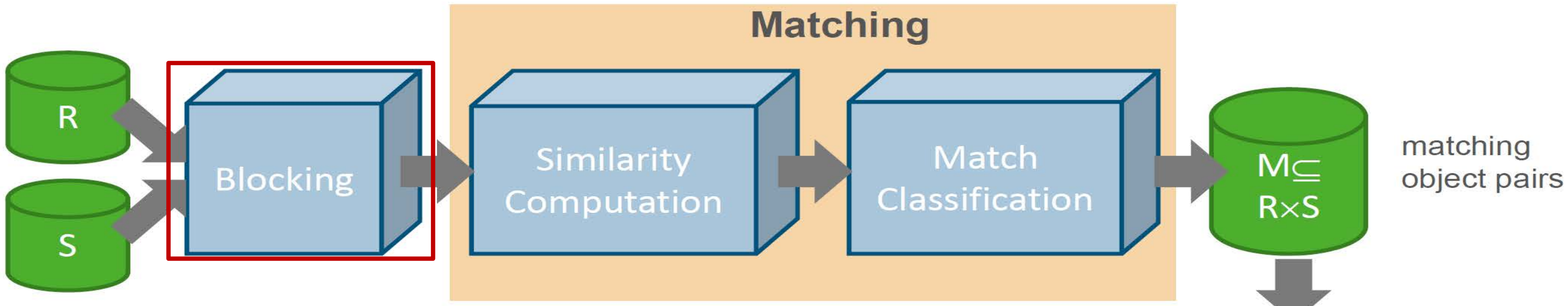
The Entity Resolution Problem (AI + DB)

From a Company that sells Entity Resolution Products

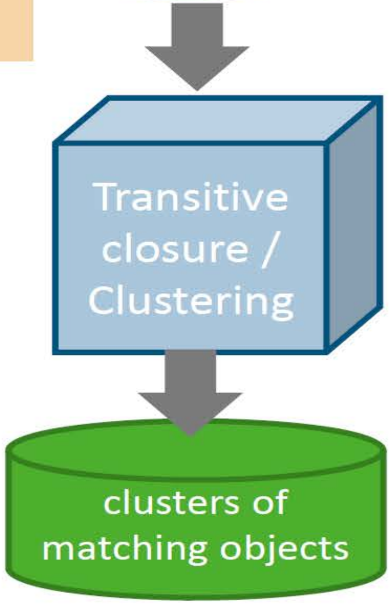
Entity Resolution – The Problem

- Many databases contain uncertain and imprecise references to real-world entities
- The absence of proper identifiers for the underlying entities often results in one or more databases which contain multiple references to as well as multiple properties of the same real-world entity
- Entity resolution involves discovering the underlying entities and mapping each database reference to these entities
  - Using all available information about the entities
- Many variants
  - Deduplication, record linking, entity matching, record matching, …

The General Architecture for Entity Matching



- **Special cases:**
  - only one input data source
  - 2 clean (duplicate-free) sources -> 1:1 mapping
  - 1 clean, 1 unclean source -> 1:n mapping
  - 2 unclean sources -> n:m mappings



Blocking

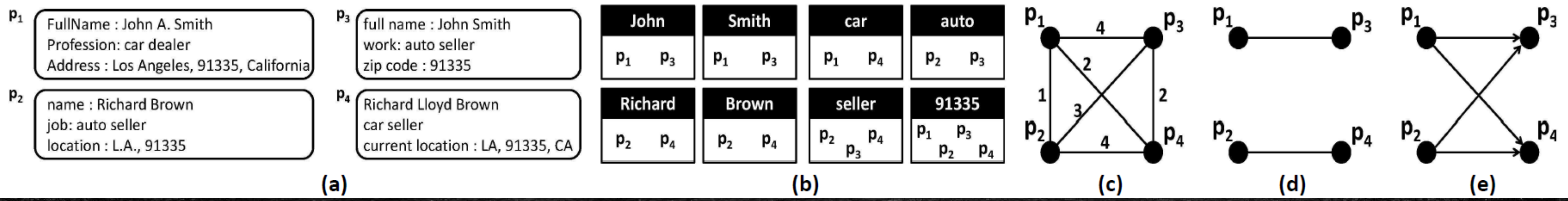
- An indexing technique
  - Splits the databases into non-overlapping blocks, such that only records within each block are compared with each other.
  - A blocking criterion, commonly called a blocking key is either based on a single record field (attribute), or the concatenation of values from several fields.

Record fields					Blocking keys and BKVs		
Identifiers	Givennames	Surnames	Postcodes	Suburb names	Sndx(GiN)+PC	Fi2D(PC)+DMe(SurN)	Sndx(SubN)+La2D(PC)
R1	Peter	Christen	2010	North Sydney	P360-2010	**20-KRST**	N632-10
R2	Pedro	Kristen	2000	Sydeny	P360-2000	**20-KRST**	S530-00
R3	Paul	Smith	2600	Canberra	P400-2600	26-SM0	**C516-00**
R4	Pablo	Smyth	2700	Canberra Sth	P140-2700	27-SM0	**C516-00**

Soundex (Sndx) encoded givenname (GiN)
First two digits (Fi2D) of postcode
Double-Metaphone (DMe) encoded surname (SurN) values

Blocking Methods

- The "standard" (token-ased) method
  - a parameter-free technique
  - every distinct token t_i in the input creates a separate block b_i that contains all entities having t_i in their attribute values as long as t_i is shared by at least 2 entities



- Attribute Clustering
  - partitions attribute names into a set K of non-overlapping clusters according to the similarity of their values
  - applies the standard method independently inside each cluster

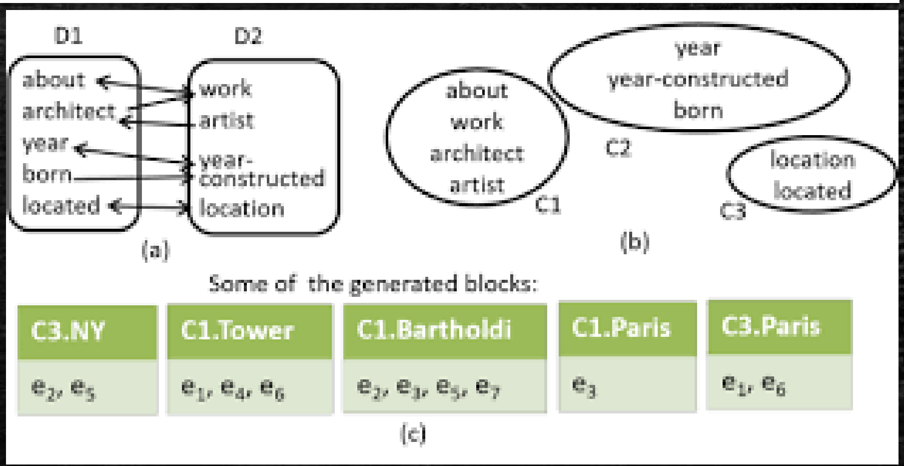
Attribute Clustering for Blocking

- Each attribute name from N1 is associated with the most similar attribute name of N2 (Lines 2-5) and vice versa (Line 6)
- The link between two attribute names is stored in a data structure (Line 5)
  - Based on the condition that the similarity of their values exceeds zero (Line 4)
- The transitive closure of the stored links is then computed (Line 7) to form the basis for partitioning attribute names into clusters
- Each connected component of the transitive closure corresponds to an attribute cluster (Line 8).
- All singleton clusters are merged into a new one, called the Glue Cluster and symbolized as k_{glue} (Line 10).

Algorithm 1: Attribute Clustering Blocking.

Input: Attribute name sets: N_1, N_2, Attribute values: V_1, V_2
Output: Set of attribute names clusters: K

1 $links \leftarrow \{\}$; $k_{glue} \leftarrow \{\}$;
2 **foreach** $n_{i,1} \in N_1$ **do**
3 $n_{j,2} \leftarrow getMostSimilarAttribute(n_{i,1}, N_2, V_2)$;
4 **if** $0 < sim(n_{i,1}.getValues(), n_{j,2}.getValues())$ **then**
5 $links.add(newLink(n_{i,1}, n_{j,2}))$;
6 **foreach** $n_{i,2} \in N_2$ **do** ... ; // same as with N_1
7 $links' \leftarrow computeTransitiveClosure(links)$;
8 $K \leftarrow getConnectedComponents(links')$;
9 **foreach** $k_i \in K$ **do**
10 **if** $|k_i| = 1$ **then** $K.remove(k_i)$; $k_{glue}.add(k_i)$;
11 $K.add(k_{glue})$;
12 **return** K;



Meta-Blocking

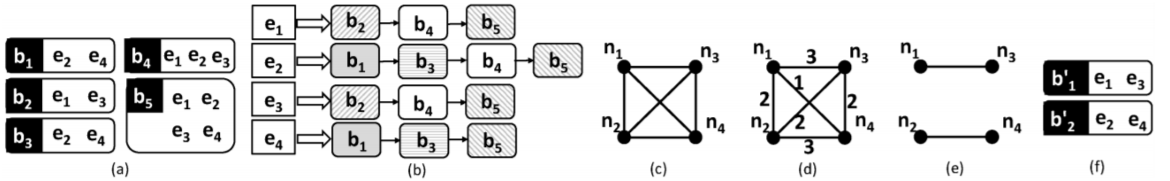
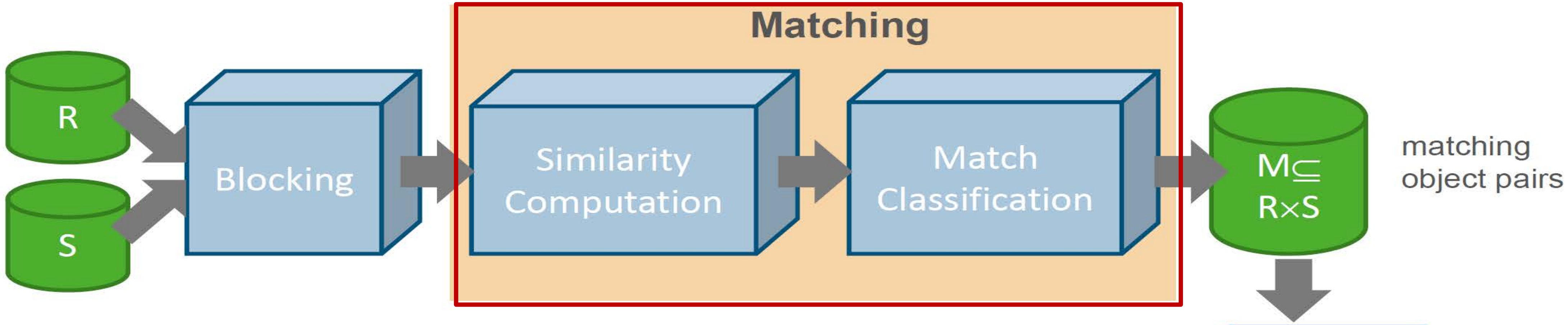


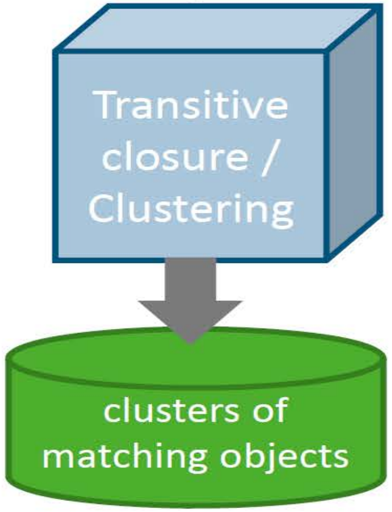
Fig. 5. (a) A block collection B with $e_1 \equiv e_3$ and $e_2 \equiv e_4$, (b) the corresponding Entity Index, (c) the corresponding blocking graph G_B, (d) the weighted G_B, (e) the pruned G_B, and (f) the new block collection B'.

- Construct the blocking graph
  - Edges are weighted proportionately to the likelihood that the adjacent entities are matching
  - Edges with low weights are pruned, because they correspond to superfluous comparisons
    - **Weighted Edge Pruning (WEP)** removes all edges that do not exceed a specific threshold
    - **Cardinality Edge Pruning (CEP)** retains the globally K top weighted edges
    - **Weighted Node Pruning (WNP)** retains in each node neighborhood the entities that exceed a local threshold, which may be the average edge weight of each neighborhood
    - **Cardinality Node Pruning (CNP)** retains the top-k weighted edges in each node neighborhood
  - The resulting pruned blocking graph GB′ is transformed into a restructured block collection B′ by forming one block for every retained edge

The General Architecture for Entity Matching



- **Special cases:**
  - only one input data source
  - 2 clean (duplicate-free) sources -> 1:1 mapping
  - 1 clean, 1 unclean source -> 1:n mapping
  - 2 unclean sources -> n:m mappings



Matching Strategy

- **Blocking** to reduce search space
  - Group similar objects within blocks based on blocking key
  - Restrict object matching to objects from the same block
  - Alternative approach: Sorted Neighborhood
- **Combined use** of several matchers
  - Attribute-level matching
    - Based on generic or domain-specific similarity functions, e.g., string similarity (edit distance, n-gram, TF/IDF, etc.)
  - Context-based matchers
  - Learning-based or manual specification of matcher combination
  - Optional: **Transitive closure** of matches to identify indirect matches

Rule-based Matching

- The developer writes rules that specify when two tuples match
  - typically after examining many matching and non-matching tuple pairs, using a development set of tuple pairs
  - rules are then tested and refined, using the same development set or a test set
- Many types of rules exist
  - linearly weighted combination of individual similarity scores
  - logistic regression combination
  - more complex rules

Linearly Weighted Combination Rules

- Tuple Matching
  - Choose a set of attributes to be used for matching
  - Compute the similarity score between tuples *x* and *y* as a linearly weighted combination of individual similarity scores

    sim(x,y) = $\sum_{i=1}^{n} \alpha_i * sim_i(x,y)$

    n is number of attributes in each table

    $\mathbf{s}_i(\mathbf{x},\mathbf{y})$ is a sim score between the i-th attributes of x and y

    $\alpha_i \in [0,1]$ is a pre-specified weight that indicates the important of the i-th attribute to sim(x,y), such that $\sum_{i=1}^{n} \alpha_i = 1$
  - *sim(x,y)* is a match if *sim(x,y)* ≥ β
  - Learn the weights with a suitable version (e.g., with regularization) of linear regression

Example

Table X

	Name	Phone	City	State
x_1	Dave Smith	(608) 395 9462	Madison	WI
x_2	Joe Wilson	(408) 123 4265	San Jose	CA
x_3	Dan Smith	(608) 256 1212	Middleton	WI

(a)

Table Y

	Name	Phone	City	State
y_1	David D. Smith	395 9426	Madison	WI
y_2	Daniel W. Smith	256 1212	Madison	WI

(b)

Matches
(x_1, y_1)
(x_3, y_2)

(c)

- *sim(x,y)* =

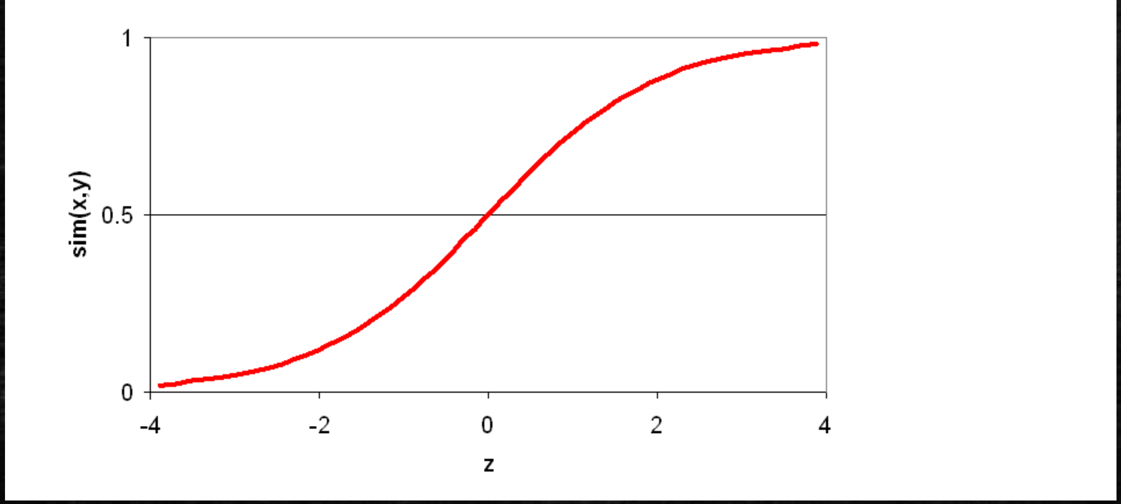
  $0.3s_{name}(x,y) + 0.3s_{phone}(x,y) + 0.1s_{city}(x,y) + 0.3s_{state}(x,y)$
  - $s_{name}(x,y)$: based on Jaro-Winkler
  - $s_{phone}(x,y)$: based on edit distance between *x*'s phone (after removing area code) and *y*'s phone
  - $s_{city}(x,y)$: based on edit distance
  - $s_{state}(x,y)$: based on exact match; yes → 1, no → 0

Pros and Cons

- Pros
  - conceptually simple, easy to implement
- Cons
  - an increase δ in the value of any $\mathbf{s}_i$ will cause a linear increase $\alpha_i * \delta$ in the value of s
  - in certain scenarios this is not desirable,
  - after a certain threshold, an increase in $\mathbf{s}_i$ should count less (i.e., "diminishing returns" should kick in)
  - e.g., if $\mathbf{s}_{name}(\mathbf{x},\mathbf{y})$ is already 0.95 then the two names already very closely match
    - so any increase in $\mathbf{s}_{name}(\mathbf{x},\mathbf{y})$ should contribute only minimally

Logistic Regression Rules

- Addressing the diminishing returns problem
- $sim(x, y) = 1/(1 + e^{-z})$, where $z = \sum_{i=1}^{n} \alpha_i * sim_i(x, y)$
- Notice that
  - α_i are not constrained to be between 0 and 1 and are not required to sum to 1
  - Even if z goes to infinity, $sim(x, y)$ increases only gradually



Logistic Regression Rules

- Are also very useful in situations where
  - there are many "signals" (e.g., 10-20) that can contribute to whether two tuples match
  - we don't need all of these signals to "fire" in order to conclude that the tuples match
  - as long as a reasonable number of them fire, we have sufficient confidence
- Logistic regression is a natural fit for such cases
- Hence is quite popular as a first matching method to try

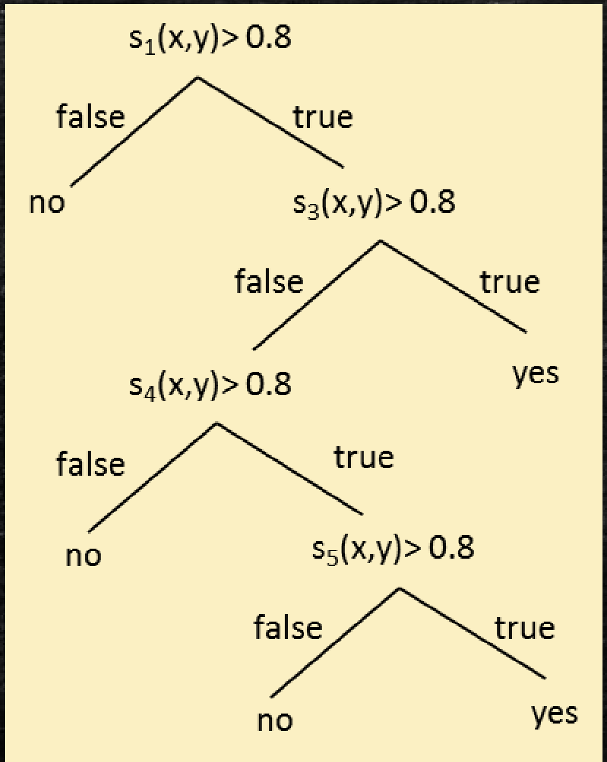
Matching with a Decision Tree

match names match phones match cities match states check area code against city

$v_1 = \langle [s_1(a_1,b_1), s_2(a_1,b_1), s_3(a_1,b_1), s_4(a_1,b_1), s_5(a_1,b_1), s_6(a_1,b_1)], \text{yes} \rangle$

$v_2 = \langle [s_1(a_2,b_2), s_2(a_2,b_2), s_3(a_2,b_2), s_4(a_2,b_2), s_5(a_2,b_2), s_6(a_2,b_2)], \text{yes} \rangle$

$v_3 = \langle [s_1(a_3,b_3), s_2(a_3,b_3), s_3(a_3,b_3), s_4(a_3,b_3), s_5(a_3,b_3), s_6(a_3,b_3)], \text{no} \rangle$



Now the labels are yes/no, not 1/0

More Complex Rules

- Appropriate when we want to encode more complex matching knowledge
  - e.g., two persons match if names match approximately and either phones match exactly or addresses match exactly
    1. If $\mathbf{s}_{name}(\mathbf{x},\mathbf{y}) < 0.8$ then return “not matched”
    2. Otherwise if $\mathbf{e}_{phone}(\mathbf{x},\mathbf{y})$ = true then return “matched”
    3. Otherwise if $\mathbf{e}_{city}(\mathbf{x},\mathbf{y})$ = true and $\mathbf{e}_{state}(\mathbf{x},\mathbf{y})$ = true then return “matched”
    4. Otherwise return “not matched”

Constraints

- Important forms of constraints:
  - Transitivity – If M1 and M2 match, M2 and M3 match, then M1 and M3 match
  - Exclusivity: If M1 matches with M2, then M3 cannot match with M2
  - Functional Dependency: If M1 and M2 match, then M3 and M4 must match

- Transitivity is key to deduplication
- Exclusivity is key to record linkage
- Functional dependencies for data cleaning

http://users.umiacs.umd.edu/~getoor/Tutorials/ER_VLDB2012.pdf

Constraint Types

Some constraints might need Joins

	Hard Constraint	Soft Constraint
Positive Evidence	If M1, M2 match then M3, M4 must match *If two papers match, their venues match*	If M1, M2 match then M3, M4 more likely to match *If two venues match, then their papers are more likely to match*
Negative Evidence	Mention M1 and M2 must refer to distinct entities **(Uniqueness)** *Coauthors are distinct* If M1, M2 don't match then M3, M4 cannot match *If two venues don't match, then their papers don't match*	If M1, M2 don't match then M3, M4 less likely to match *If institutions don't match, then authors less likely to match*

Constraints may be directional or bidirectional

Constraints can be recursive, e.g., if two authors have matching co-authors, then they match

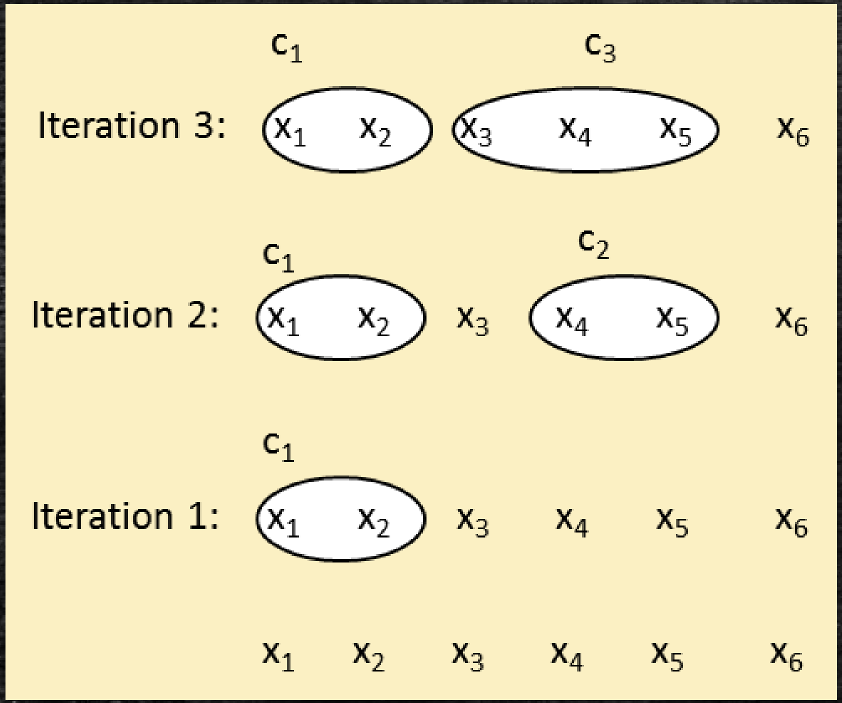
Additional Constraints

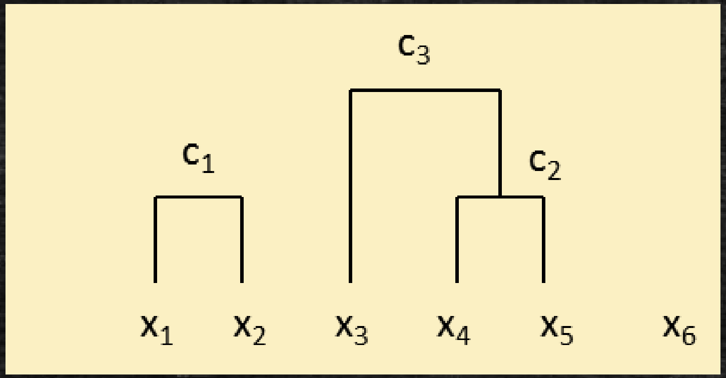
- Aggregate Constraints
  - count constraints
    - Entity *A* can link to at most *N* *B*s – Authors have at most 5 papers at any conference
  - Other aggregates like sum, average more complex
    - Examples?

Matching by Clustering

- Many common clustering techniques have been used
  - agglomerative hierarchical clustering (AHC), k-means, graph-theoretic, …
  - here we focus on AHC, a simple yet very commonly used one
- AHC
  - partitions a given set of tuples D into a set of clusters
    - all tuples in a cluster refer to the same real-world entity, tuples in different clusters refer to different entities
  - begins by putting each tuple in D into a single cluster
  - iteratively merges the two most similar clusters
  - stops when a desired number of clusters has been reached, or until the similarity between two closest clusters falls below a pre-specified threshold

Example





- sim(x,y) =
  $0.3s_{name}(x,y) + 0.3s_{phone}(x,y) + 0.1s_{city}(x,y) + 0.3s_{state}(x,y)$

Computing a Similarity Score between Two Clusters

- Let c and d be two clusters
- Single link: $s(c,d) = \mathbf{min}_{xi \in c,\, yj \in d}\ \mathbf{sim(x_i, y_j)}$
- Complete link: $s(c,d) = \mathbf{max}_{xi \in c,\, yj \in d}\ \mathbf{sim(x_i, y_j)}$
- Average link: $s(c,d) = [\sum_{xi \in c,\, yj \in d} \mathbf{sim(x_i, y_j)}]$ / [# of $(\mathbf{x_i},\mathbf{y_j})$ pairs]
- Canonical tuple construction/canonicalization
  - create a canonical tuple that represents each cluster
  - sim between c and d is the sim between their canonical tuples
  - canonical tuple is created from attribute values of the tuples
    - e.g., "Mike Williams" and "M. J. Williams" ➔ "Mike J. Williams"
    - (425) 247 4893 and 247 4893 ➔ (425) 247 4893

Key Ideas underlying the Clustering Approach

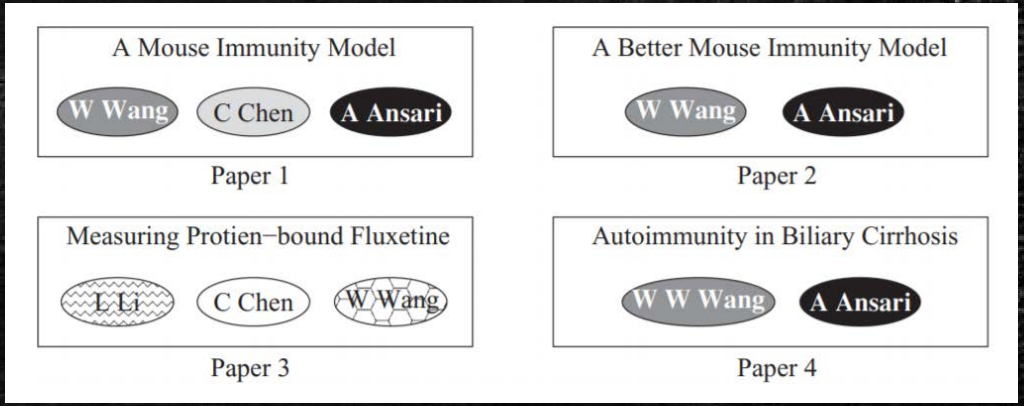
- View matching tuples as the problem of constructing entities (i.e., clusters)
- The process is iterative
  - leverage what we have known so far to build “better” entities
- In each iteration merge all matching tuples within a cluster to build an “entity profile”, then use it to match other tuples → merging then exploiting the merged information to help matching
- These same ideas appear in subsequent approaches that we will cover

Relational Clustering

- A bibliographic database
  - W. Wang, C. Chen, A. Ansari, A mouse immunity model
  - W. Wang, A. Ansari, A better mouse immunity model
  - L. Li, C. Chen, W. Wang, Measuring protein-bound fluxetine
  - W. W. Wang, A. Ansari, Autoimmunity in biliary cirrhosis
- Goal
  - Which of these author names refer to the same author entities?

6 underlying author entities – Wang1 and Wang 2, Chen1 and Chen2, Ansari and Li
But there are 10 references

The Target Result



Match Dependencies and Extent

- Match Dependency
  - When matching decisions depend on other matching decisions (in other words, matching decisions are not made independently), we refer to the approach as collective
- Match Extent
  - Global
    - If two papers match, then their venues match
      - This constraint can be applied to all instances of venue mentions – All occurrences of 'SIGMOD' can be matched to 'International Conference on Management of Data'
  - Local
    - If two papers match, then their authors match
      - This constraint can only be applied locally – Don't want to match all occurrences of 'J. Smith' with 'Jeff Smith' who is mentioned in this paper

Semantic Integrity Constraints

Type	Example
Aggregate	C1 = No researcher has published more than five AAAI papers in a year
Subsumption	C2 = If a citation X from DBLP matches a citation Y in a homepage, then each author mentioned in Y matches some author mentioned in X
Neighborhood	C3 = If authors X and Y share similar names and some co-authors, they are likely to match
Incompatible	C4 = No researcher exists who has published in both HCI and numerical analysis
Layout	C5 = If two mentions in the same document share similar names, they are likely to match
Key/Uniqueness	C6 = Mentions in the PC listing of a conference is to different researchers
Ordering	C7 = If two citations match, then their authors will be matched in order
Individual	C8 = The researcher with the name “Mayssam Saria” has fewer than five mentions in DBLP (new graduate student)

Handling Constraints

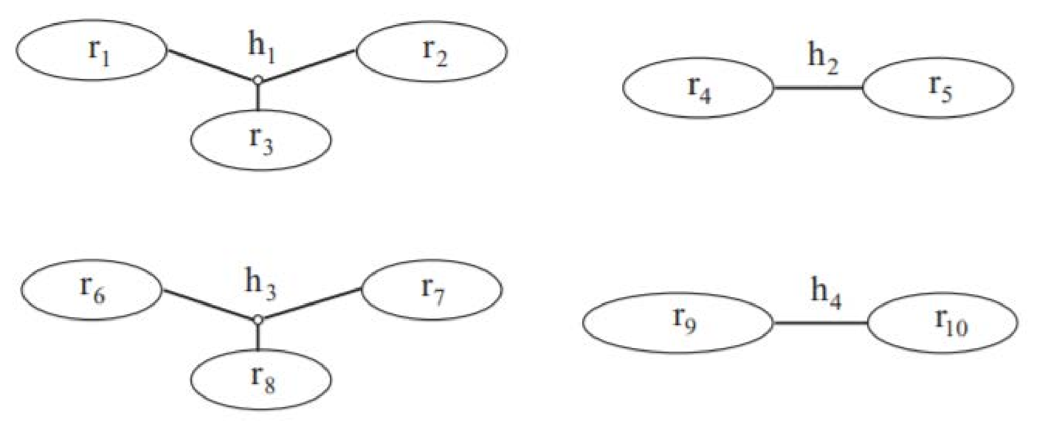
- Record linkage – propagation through exclusivity
  - Weighted k-partite matching
- Deduplication – propagation through transitivity
  - Correlation clustering
- Collective – propagation through general constraints
  - Similarity propagation
    - Dependency graphs, Collective Relational Clustering
  - Probabilistic approaches
    - LDA, CRFs, Markov Logic Networks, Probabilistic Relational Models

Two sub-problems

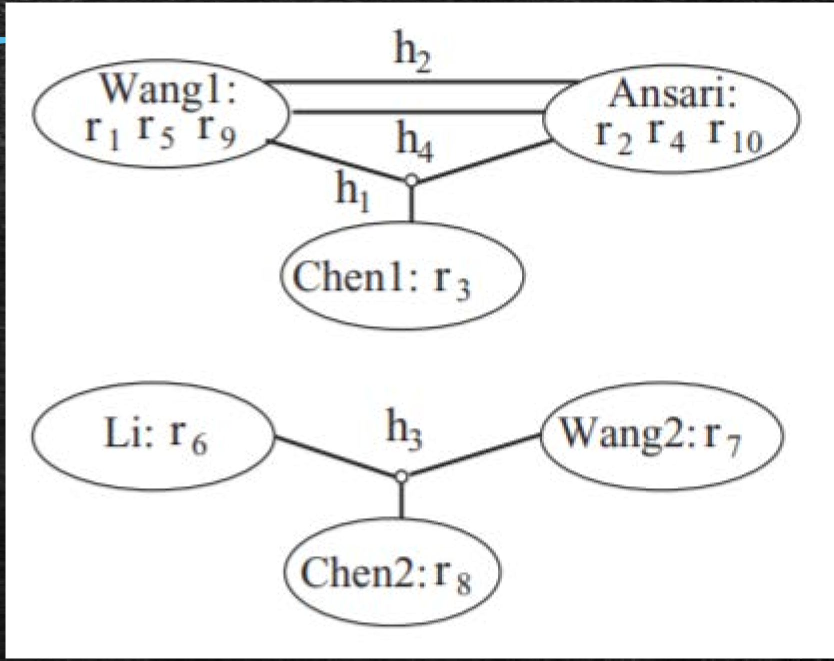
- Identification problem
  - A set of different name references the same entity
  - Wei Wei Wang can be represented as Wei Wang, W. Wang, W. W. Wang, …
- Disambiguation problem
  - Same name refers to different entities
- Relational Clustering
  - Don't look only at values, look at the relationships between the entities
  - Simultaneously solve both the identification and the disambiguation problems
- The relationship that we would like to exploit is that these authors co-occur (co-authorship) in various combinations

Re-representing Data

The r's are references and the h's are hyperedges



- Each hyperedge *h* may have attributes as well which we denote $h.A_1$, $h.A_2$, … , $h.A_k$
- $h.R$ denotes the set of references that it connects
- A reference *r* can belong to zero or more hyperedges
- $r.H$ denotes the set of hyperedges in which *r* participates



An abstract representation for the entity graph for the author resolution example; the nodes are the entities, the set of references they correspond to are listed, and the h's are hyperedges.

Naïve Relational Entity Resolution

- The similarity function has two components
  - Attribute based similarity $sim_A(r_i, r_j)$ – same as what we have seen before
    - How well do the names match? How well do other attributes match?
  - Relationship-based similarity
    - The idea: To determine if two author references in two different papers are coreferent, we can compare the names of their coauthors
      - The naive relational decision about the references W. Wang and W. W. Wang, would consider that both have coauthors with the name A. Ansari
  - Take a linear combination of the two scores
    - $$\boxed{sim_{NR}(r_i, r_j) = (1 - \alpha) \times sim_A(r_i, r_j) + \alpha \times sim_H(r_i, r_j), \quad 0 \leq \alpha \leq 1}$$
      - If two references belong to just one hyperedge, then we compare the similarity between the set of hyperedges that each reference belongs to
      - Otherwise, we make pairwise computation of the similarity of these hyperedges

Collective Relational Entity Resolution

- Problem with the naïve Relational Entity Resolution Technique
  - It can mislead in domains where most names are frequent and hyperedges are dense
  - the naive relational approach is it does not reason about the identities of the related references
    - For the two Wang references the two C. Chen coauthors match regardless of whether they refer to Chen1 or Chen2.
    - The correct evidence to use here is that the Chens are not coreferent.
    - Therefore, **to resolve the W. Wang references, it is necessary to resolve the C Chen references as well and not just consider their name similarity**.

The Basic Approach

- Cluster the references so that only those that correspond to the same entity are assigned to the same cluster
- Use a **greedy agglomerative clustering** algorithm where
  - At any stage, the current set $C = \{ci\}$ of entity clusters reflects the current belief about the mapping of the references to entities
- Here, instead of similarity measures for references, we need to define *similarities between clusters of references*

$$\boxed{sim(c_i, c_j) = (1 - \alpha) \times sim_A(c_i, c_j) + \alpha \times sim_R(c_i, c_j), \qquad 0 \le \alpha \le 1}$$

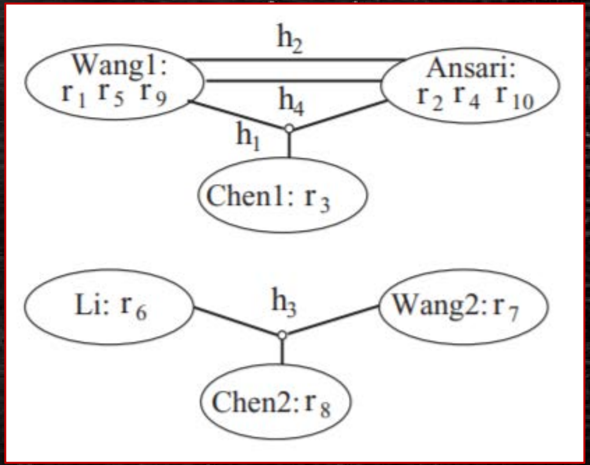
  - *$sim_R()$* is the relational similarity between the references in the two entity clusters
- The similarity of two clusters depends on the current cluster labels of their neighbors and therefore changes as their labels are updated
  - The similarity between W. Wang and W. W. Wang increases once the Ansari references are given the same cluster label

Computing Cluster Similarity – 1

- Each reference *r* is associated with one or more hyperedges
- The set of hyperedges *c.H* for a cluster *c* is given by $$c.H = \bigcup_{r \in \mathcal{R} \wedge r.C = c} \{h \mid h \in \mathcal{H} \wedge r \in h.R\}.$$
  - The union of all hyperedges associated with each reference in the cluster
  - These hyperedges connect c to other clusters
- The relational similarity for two clusters needs to compare their connectivity patterns to other clusters
- For any cluster *c*, the set of other clusters to which c is connected via its hyperedge set *c.H* form the neighborhood *Nbr(c)* of cluster *c*:

$$Nbr(c) = \bigcup_{h \in c.H, r \in h.R} \{c_j \mid c_j = r.C\}.$$

  - The neighborhood of the cluster for Wang1 consists of the clusters for Ansari and Chen1; alternatively it is the bag of clusters {Ansari, Ansari, Ansari, Chen1}

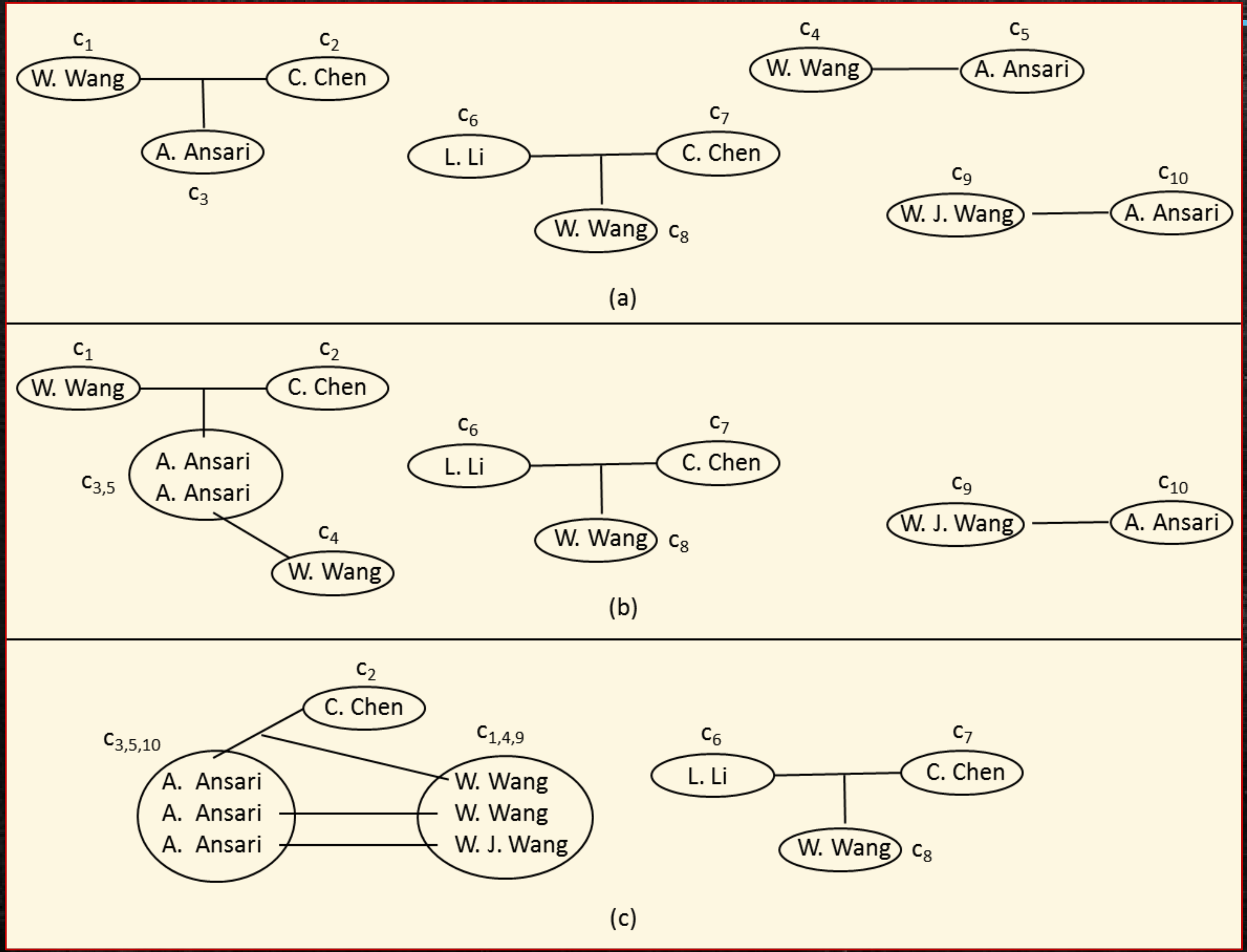


Computing Cluster Similarity – 2

- Jaccard coefficient for two clusters
  - The bag semantics variant
    - JaccardCoeff + $Fr(ci, c_j)$, by using $Nbr_B(ci)$ and $Nbr_B(c_j)$ in the formula

$$JaccardCoeff(c_i, c_j) = \frac{|Nbr(c_i) \bigcap Nbr(c_j)|}{|Nbr(c_i) \bigcup Nbr(c_j)|}.$$

Agglomerative Hierarchical Relational Clustering




```
Find similar references using blocking
Initialize clusters using bootstrapping

For clusters c_i, c_j such that similar(c_i, c_j)
    Insert ⟨sim(c_i, c_j), c_j, c_j⟩ into priority queue

While priority queue not empty
    Extract ⟨sim(c_i, c_j), c_i, c_j⟩ from queue
    If sim(c_i, c_j) less than threshold, then stop
    Merge c_i and c_j to new cluster c_ij
    Remove entries for c_i and c_j from queue
    For each cluster c_k such that similar(c_ij, c_k)
        Insert ⟨sim(c_ij, c_k), c_ij, c_k⟩ into queue
    For each cluster c_n neighbor of c_ij
        For c_k such that similar(c_k, c_n)
            Update sim(c_k, c_n) in queue
```

Scaling up Rule-based Matching

- Sorting
  - use a key to sort tuples, then scan the sorted list and match each tuple with only the previous (w-1) tuples, where w is a pre-specified window size
  - key should be strongly "discriminative": brings together tuples that are likely to match, and pushes apart tuples that are not
    - example keys: soc sec, student ID, last name, soundex value of last name
  - employs a stronger heuristic than hashing: also requires that tuples likely to match be within a window of size w
    - but is often faster than hashing because it would match fewer pairs

Scaling up Rule-based Matching

- Indexing
  - index tuples such that given any tuple a, can use the index to quickly locate a relatively small set of tuples that are likely to match a
    - e.g., inverted index on names
- Canopies
  - use a computationally cheap sim measure to quickly group tuples into overlapping clusters called canopines (or umbrella sets)
  - use a different (far more expensive) sim measure to match tuples within each canopy
  - e.g., use TF/IDF to create canopies

Scaling up Rule-based Matching

- Using representatives
  - applied during the matching process
  - assigns tuples that have been matched into groups such that those within a group match and those across groups do not
  - create a representative for each group by selecting a tuple in the group or by merging tuples in the group
  - when considering a new tuple, only match it with the representatives
- Combining the techniques
  - e.g., hash houses into buckets using zip codes, then sort houses within each bucket using street names, then match them using a sliding window

Scaling up Rule-based Matching

- For the second goal of minimizing time it takes to match each pair
  - no well-established technique as yet
  - tailor depending on the application and the matching approach
  - e.g., if using a simple rule-based approach that matches individual attributes then combines their scores using weights
    - can use short circuiting: stop the computation of the sim score if it is already so high that the tuple pair will match even if the remaining attributes do not match