Entity Linkage

Shobeir Fakhraei
University of Southern California

Slides based on material provided by Lise Getoor, Ashwin Machanavajjhala, Craig Knoblock, Jose-Luis Ambite, and others.

Where are we?

- Information Extraction
  - Semi-structured: Web, HTML
  - Unstructured: Text, Adds, Tweets, …
- Entity Linkage, Data Cleaning, Normalization
- Logical Data Integration
  - Mediators, Query Rewriting
  - Warehouse, Logical Data Exchange
- Automatic Source Modeling/Learning Schema Mappings
- Semantic Web
  - RDF, SPARQL, OWL, Linked Data
- Advanced Topics
  - Geospatial Data Integration, Knowledge Graphs

Information Extraction

As a family of techniques:

Information Extraction = segmentation + classification + association + clustering

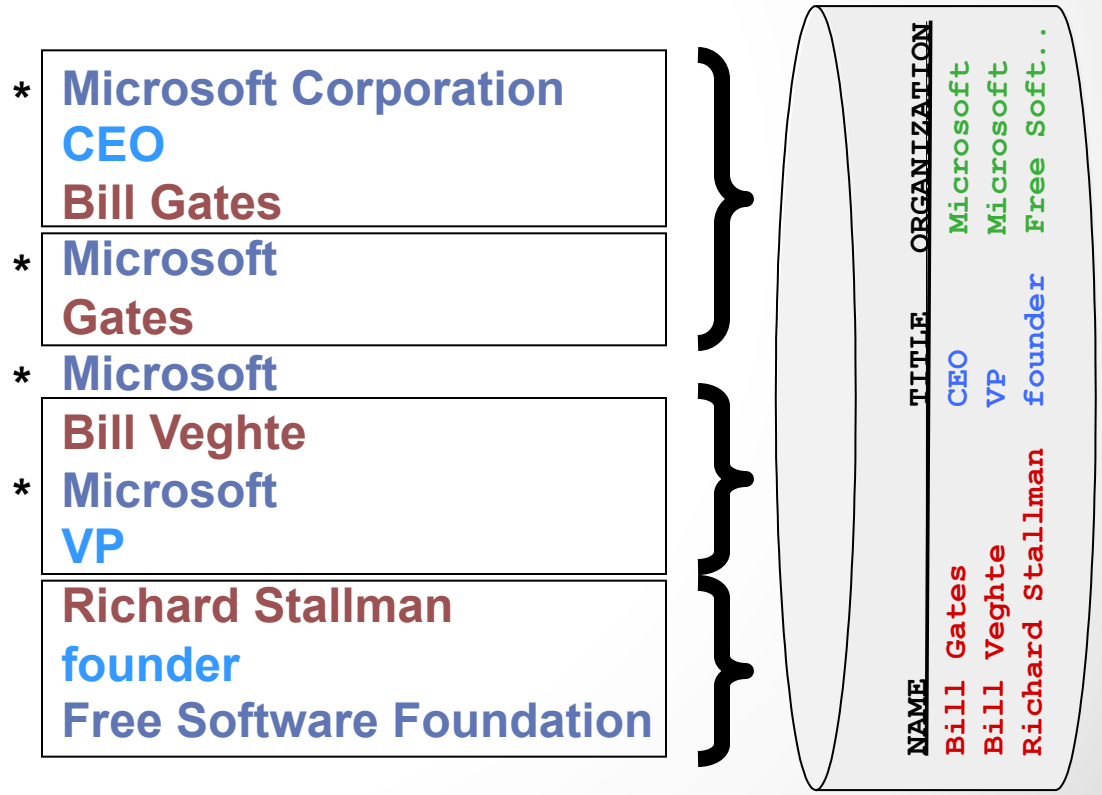
October 14, 2002, 4:00 a.m. PT

For years, Microsoft Corporation CEO Bill Gates railed against the economic philosophy of open-source software with Orwellian fervor, denouncing its communal licensing as a "cancer" that stifled technological innovation.

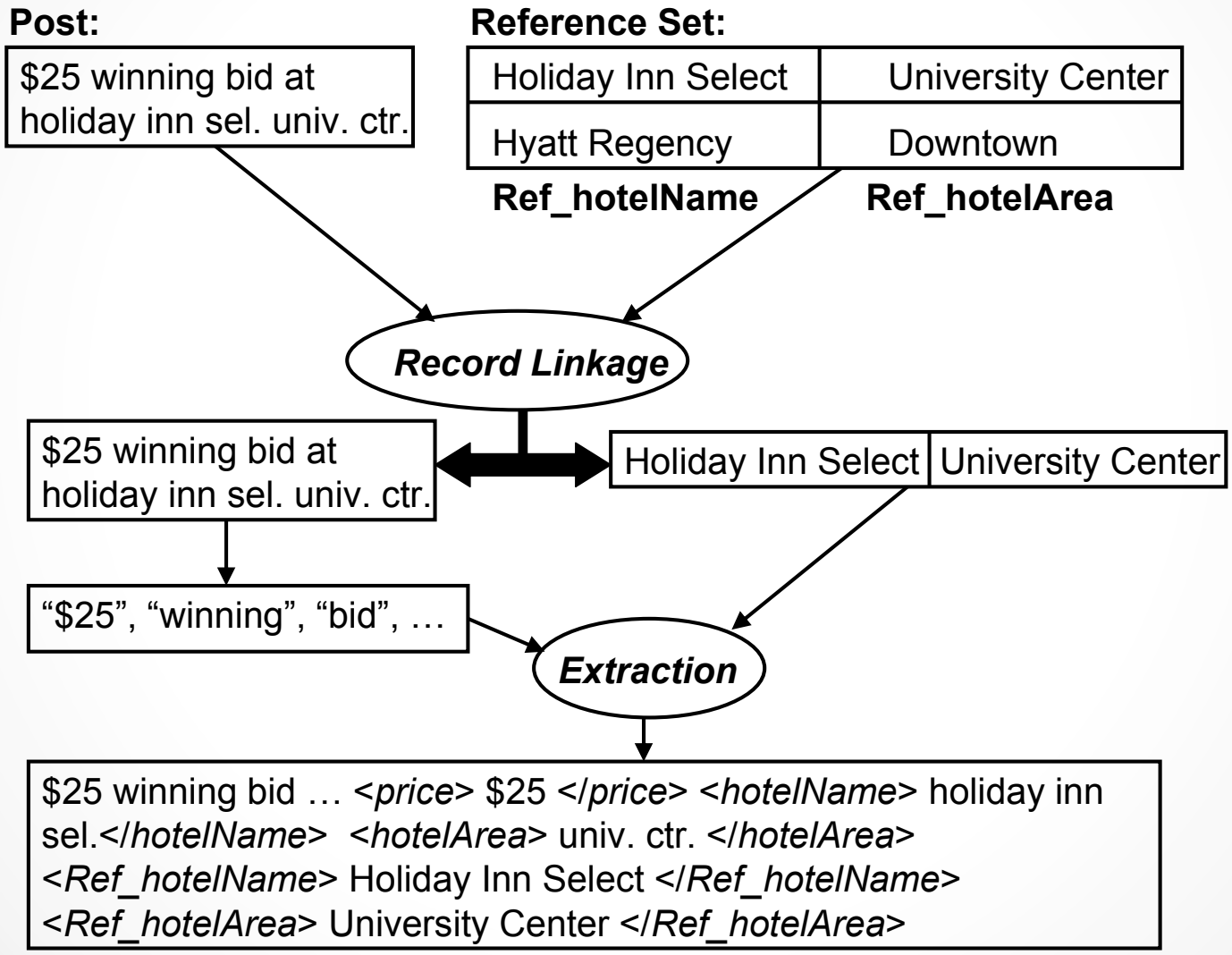
Today, Microsoft claims to "love" the open-source concept, by which software code is made public to encourage improvement and development by outside programmers. Gates himself says Microsoft will gladly disclose its crown jewels--the coveted code behind the Windows operating system--to select customers.

"We can be open source. We love the concept of shared source," said Bill Veghte, a Microsoft VP. "That's a super-important shift for us in terms of code access."

Richard Stallman, founder of the Free Software Foundation, countered saying…



IE form Ungrammatical Text



What is Entity Linkage?

Problem of identifying and linking/grouping different manifestations of the same real world object.

Examples of manifestations and objects:

- Different ways of addressing (names, email addresses, FaceBook accounts) the same person in text.
- Web pages with differing descriptions of the same business.
- Different photos of the same object.
- …

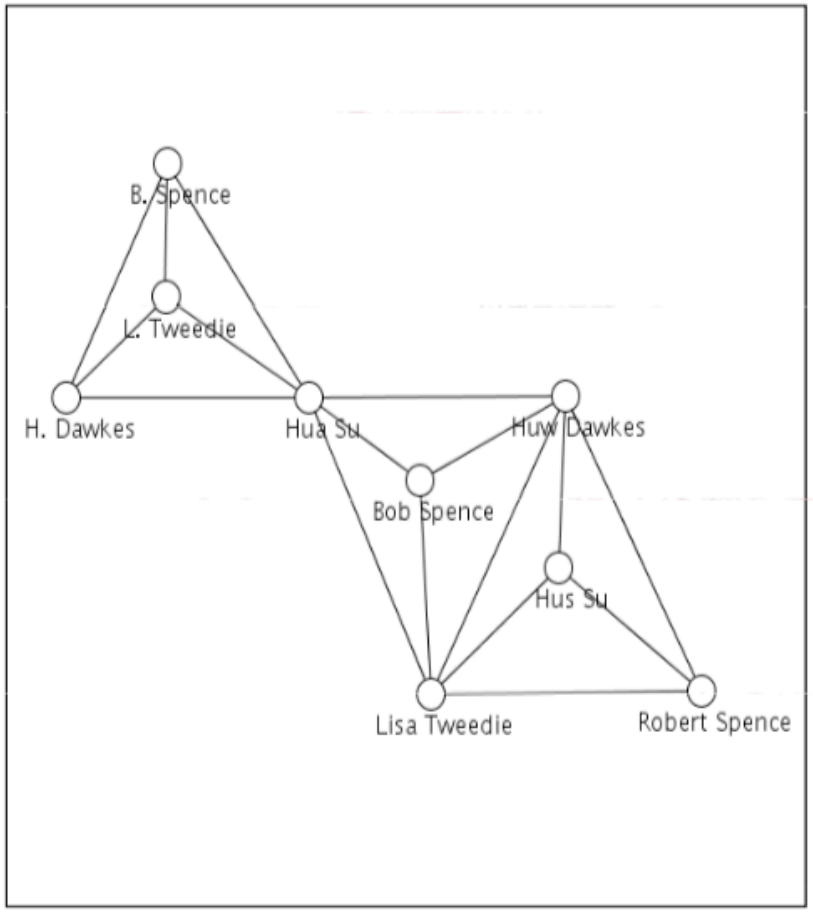
Ironically, Entity Linkage has many duplicate names

- Record linkage
- Duplicate detection
- Coreference resolution
- Reference reconciliation
- Fuzzy match
- Object consolidation
- Object identification
- Deduplication
- Entity clustering
- Approximate match
- Identity uncertainty
- Merge/purge
- Household matching
- Hardening soft databases
- Householding
- Reference matching
- Doubles

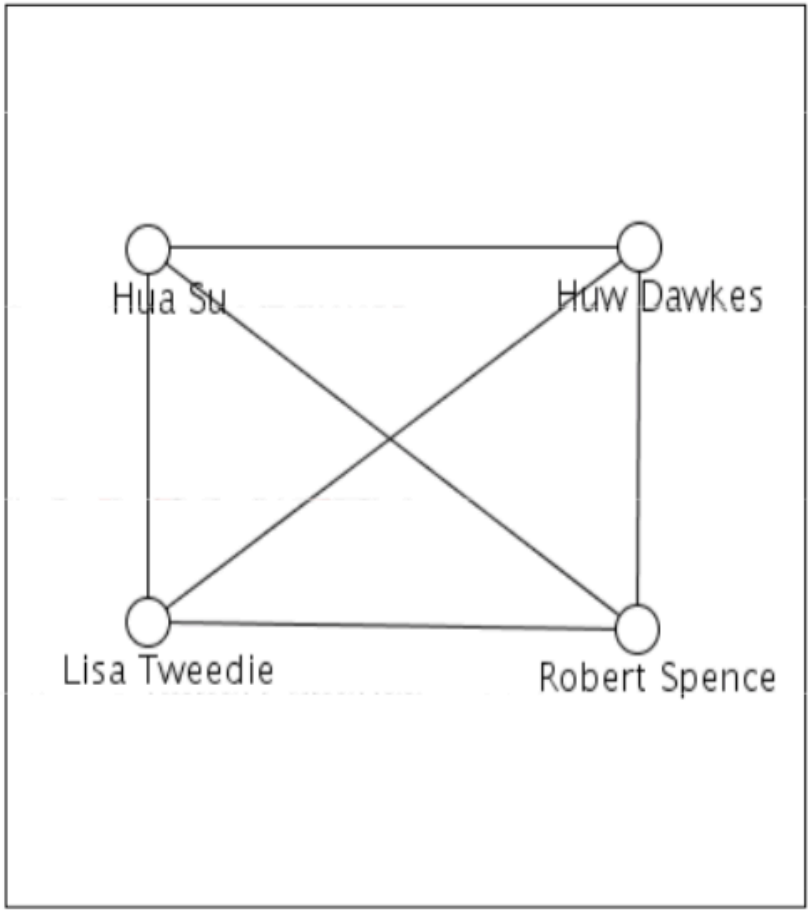
EL Motivating Examples

- *Linking Census Records*
- *Public Health*
- *Web search*
- *Comparison shopping*
- *Counter-terrorism*
- *Spam detection*
- *Machine Reading*
- …

EL and Network Analysis



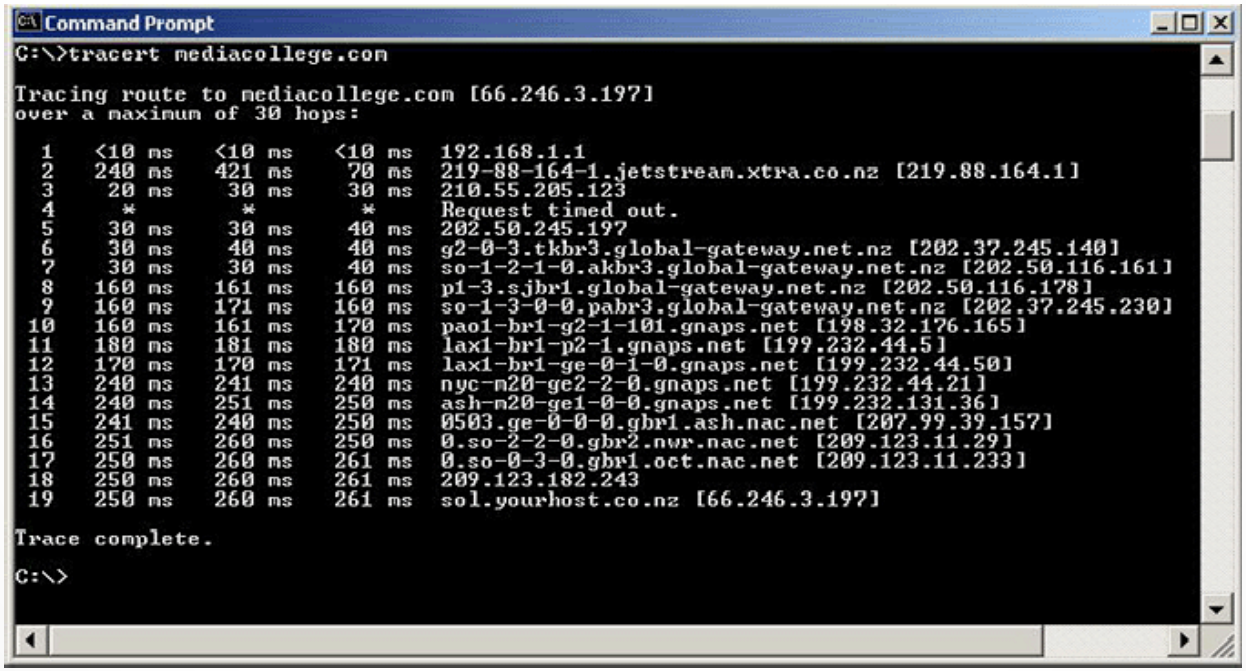
before



after

Motivation: Network Science

- Measuring the topology of the internet … using **traceroute**



IP Aliasing Problem

[Willinger et al. 2009]

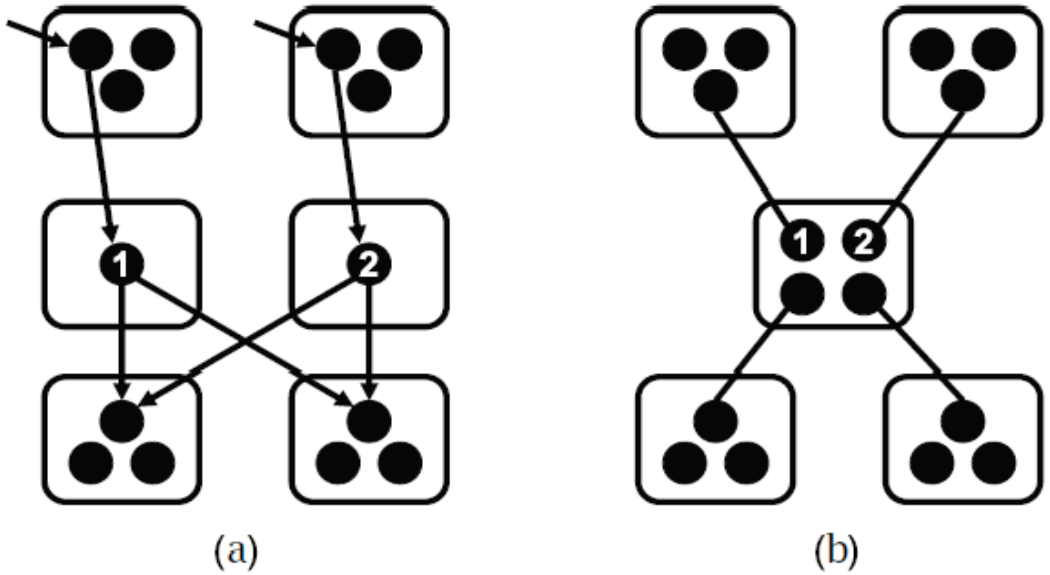


Figure 2. The IP alias resolution problem. Paraphrasing Fig. 4 of [50], `traceroute` does not list routers (boxes) along paths but IP addresses of input interfaces (circles), and alias resolution refers to the correct mapping of interfaces to routers to reveal the actual topology. In the case where interfaces 1 and 2 are aliases, (b) depicts the actual topology while (a) yields an "inflated" topology with more routers and links than the real one.

IP Aliasing Problem

[Willinger et al. 2009]

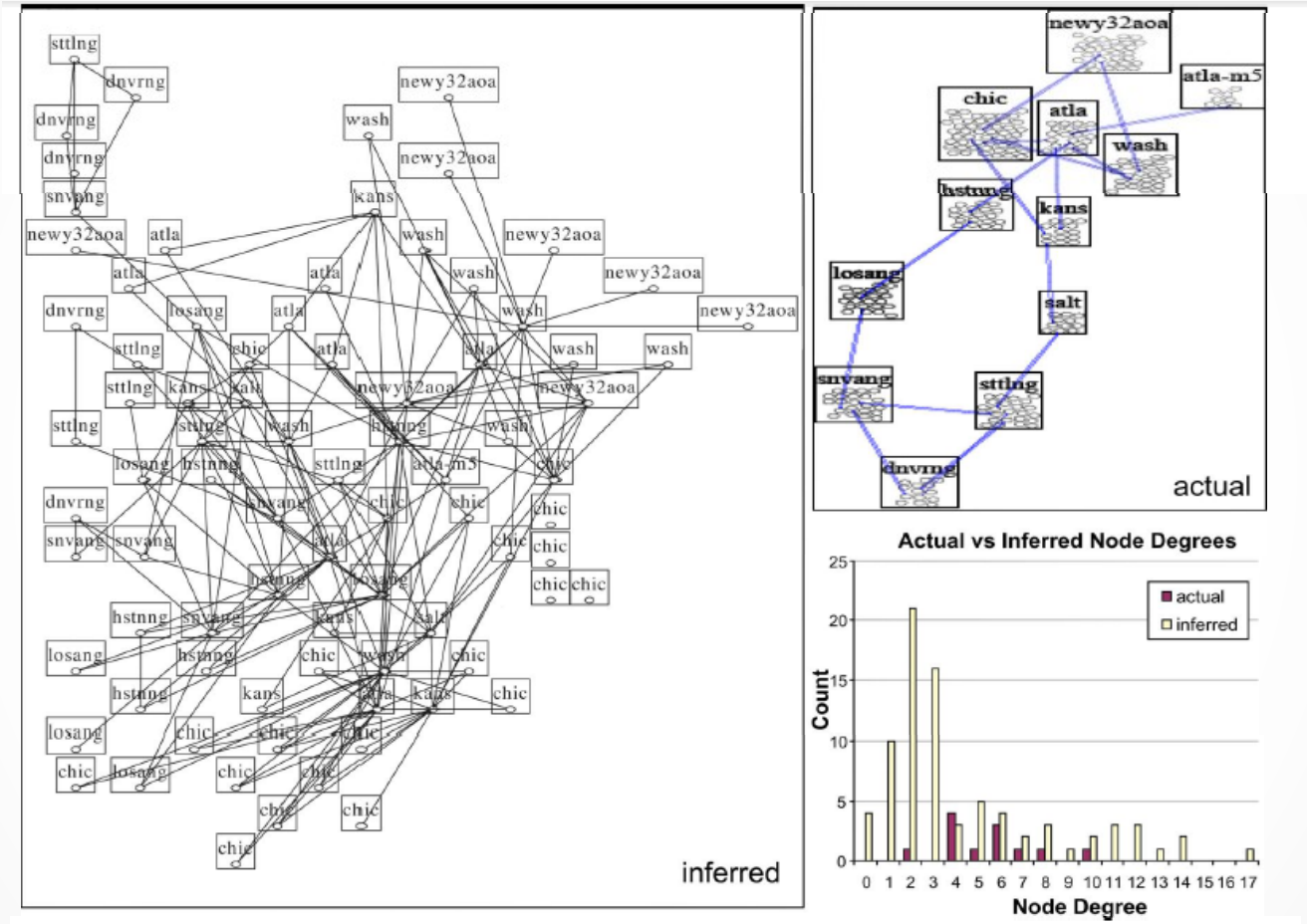
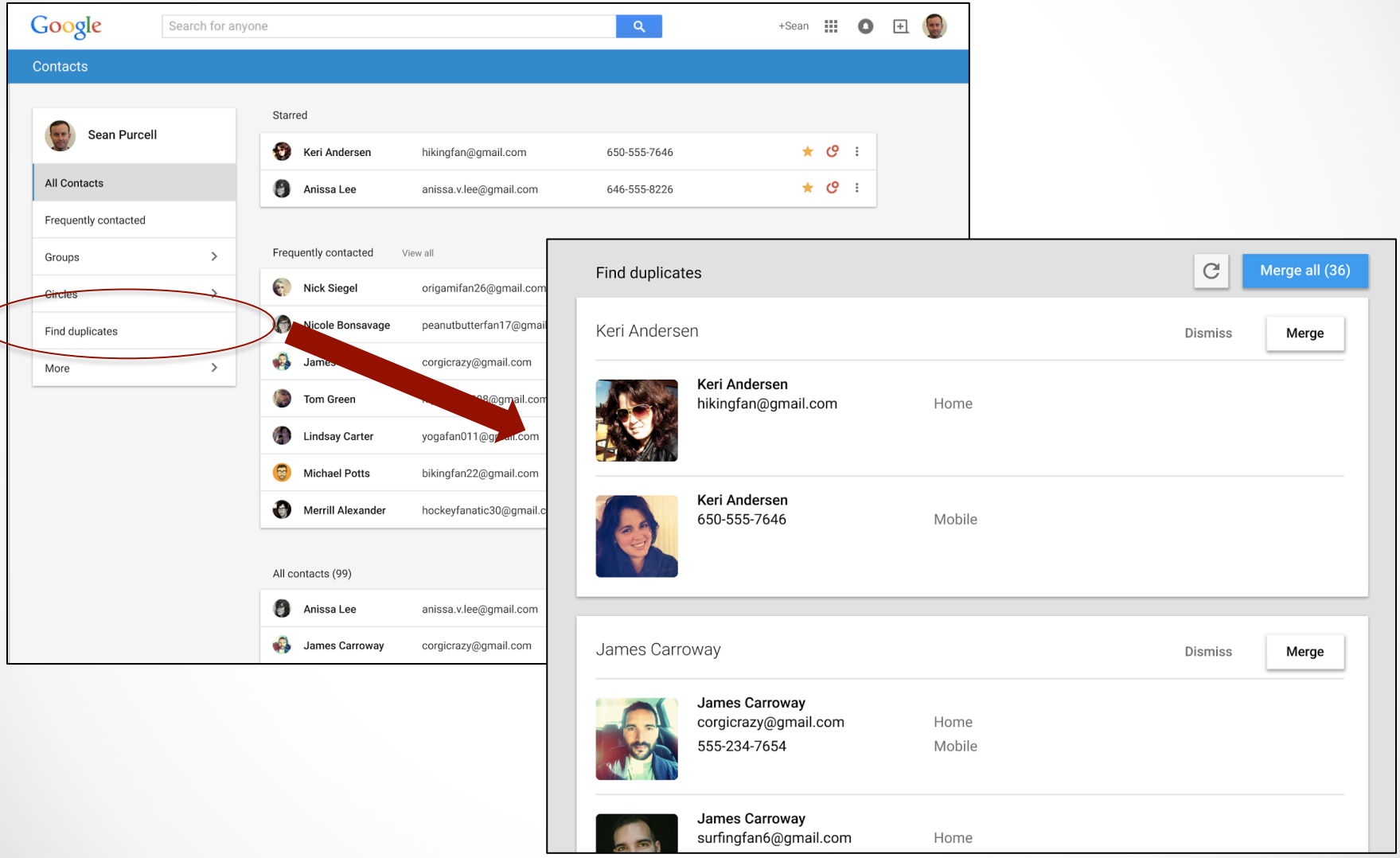


Figure 3. The IP alias resolution problem in practice. This is re-produced from [48] and shows a comparison between the Abilene/Internet2 topology inferred by Rocketfuel (left) and the actual topology (top right). Rectangles represent routers with interior ovals denoting interfaces. The histograms of the corresponding node degrees are shown in the bottom right plot. © 2008 ACM,

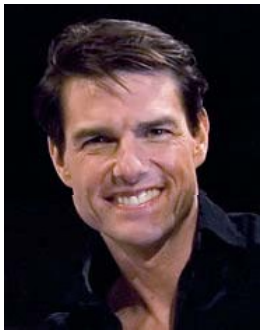
Example: Google Contacts



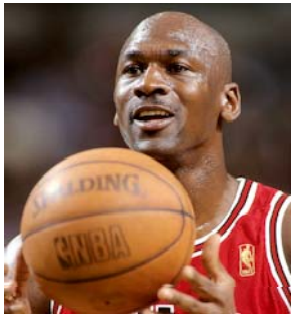
Traditional Challenges in ER

- Name/Attribute ambiguity

Thomas Cruise



Michael Jordan



Traditional Challenges in ER

- Name/Attribute ambiguity
- Errors due to data entry

↓	C1	C2
	Total Cholesterol_1	Total Cholesterol_2
682	214.4	214.4
683	184.4	184.4
684	183.5	183.5
685	240.7	240.7
686	215.1	215.1
687	198.6	198.6
688	2800.0	280.0
689	210.8	210.8
690	182.5	182.5
691	192.6	192.6

Traditional Challenges in ER

- Name/Attribute ambiguity
- Errors due to data entry
- Missing Values

Exhibit 2: Examples of variables that are set to unknown values

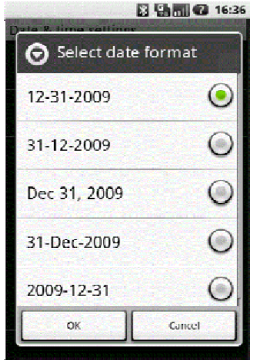
Administrative dates:	set to 0101YY, 010199, 999999
Date of Birth	0101YY, 1506YY, 3006YY, 0107YY, 1507YY, 0101YEAR
Names:	set to spaces, NK, UNKNOWN, or ZZZZ BABY, MALE, FEMALE, TWIN, TRIPLET, INFANT
Other variables:	set to 9, 99, 9999, -1 NK (Not Known) NA (Not applicable) NC (Not coded) U (Unknown)

[Gill et al; Univ of Oxford2003]

Traditional Challenges in ER

- Name/Attribute ambiguity
- Errors due to data entry
- Missing Values
- Changing Attributes

- Data formatting



- Abbreviations / Data Truncation

ABSTRACT PROBLEM STATEMENT

Abstract Problem Statement

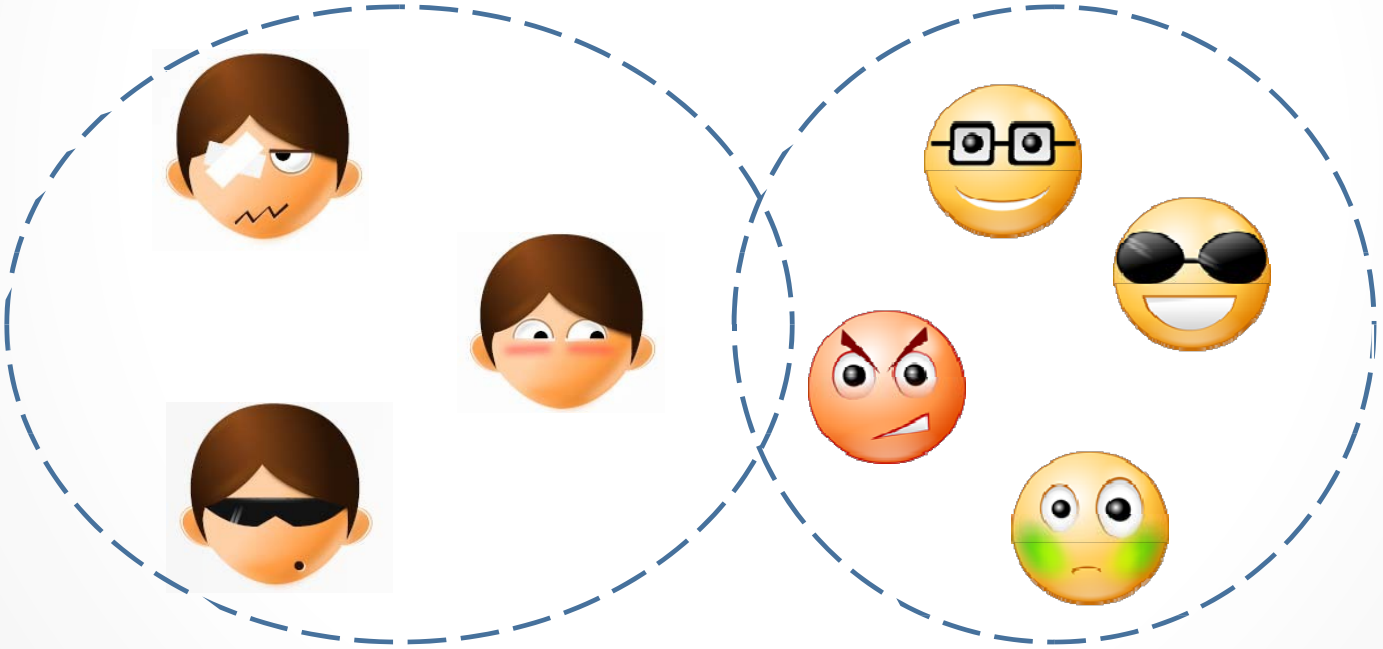
Real World

Digital World

Records / Mentions

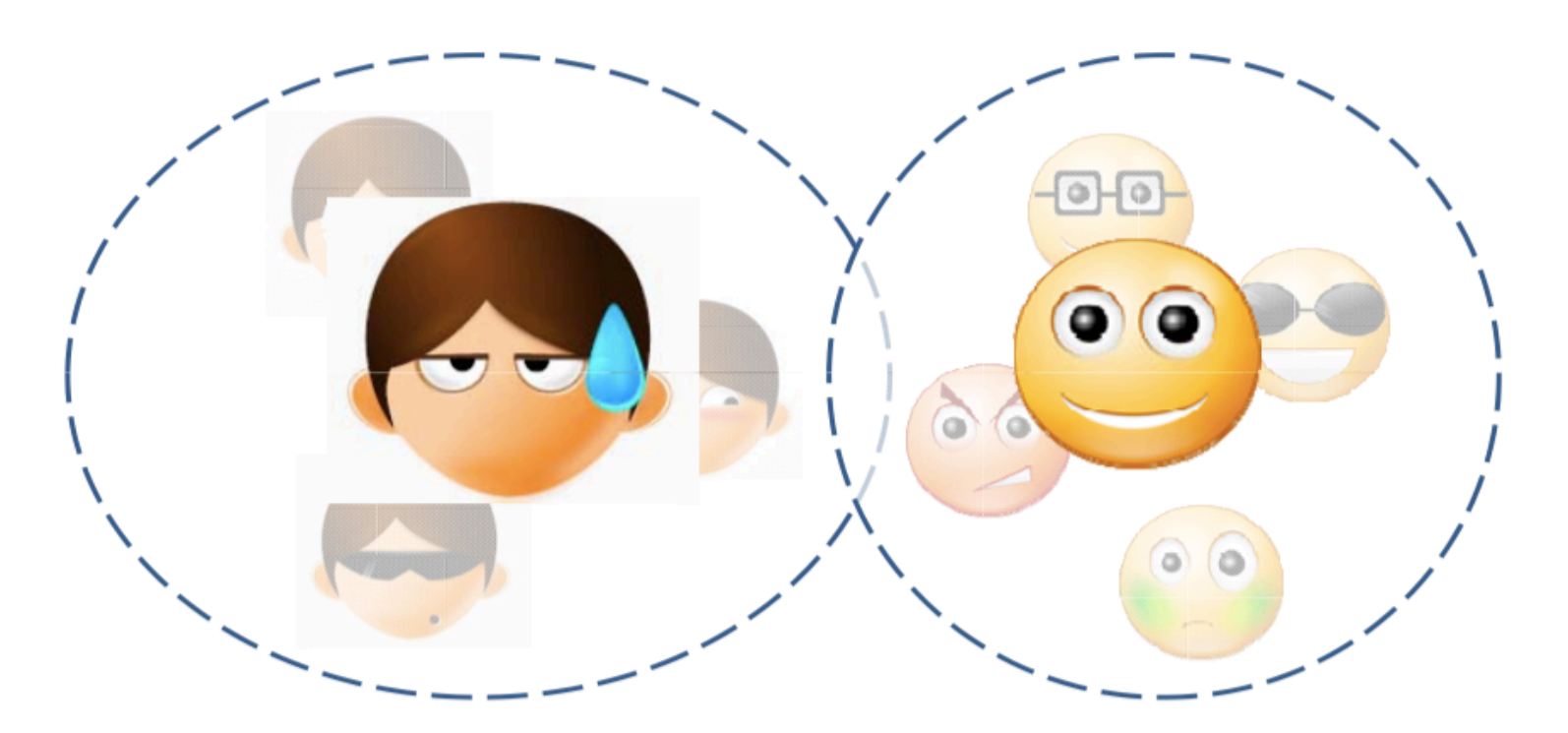
Deduplication Problem Statement

- Cluster the records/mentions that correspond tosame entity



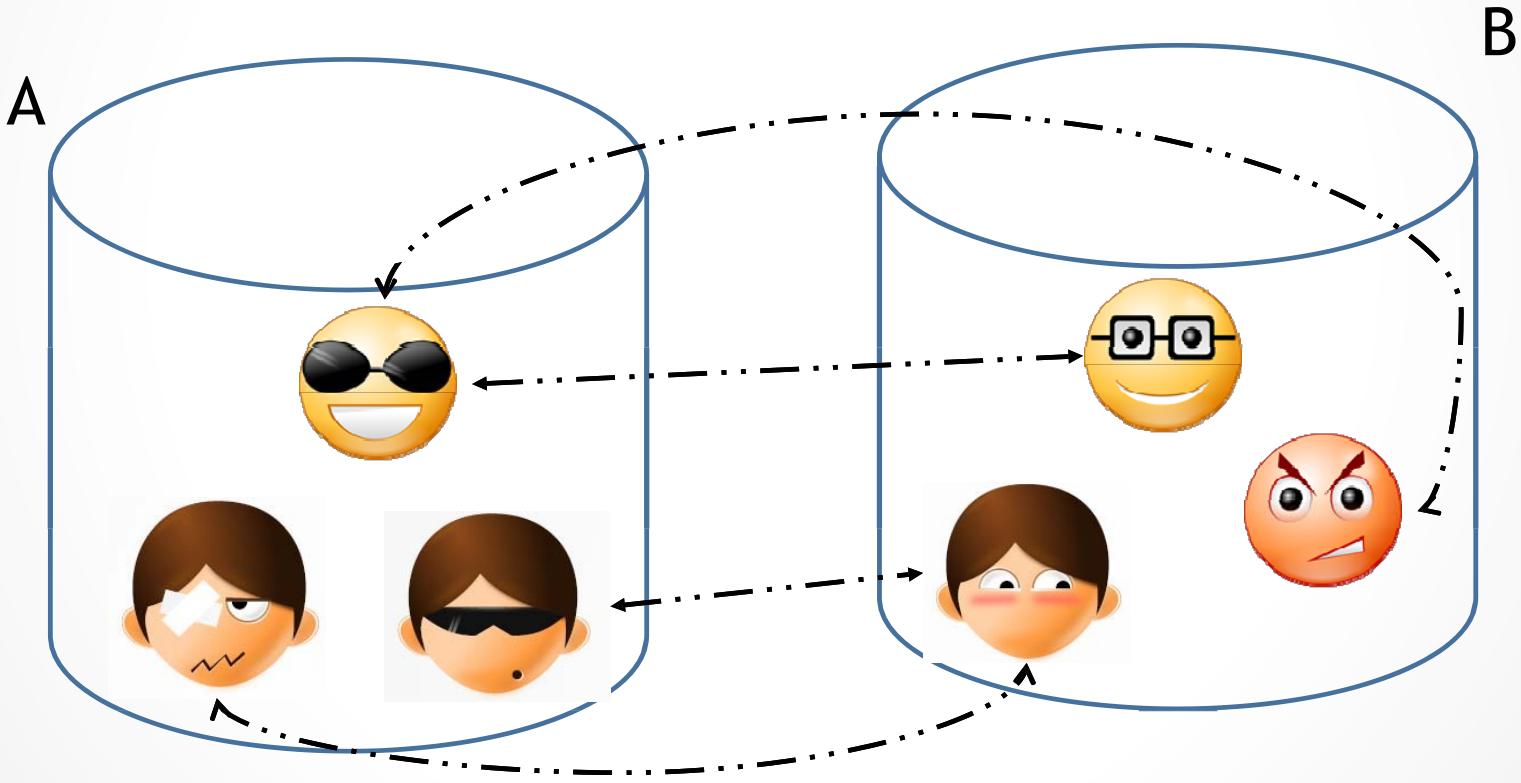
Deduplication Problem Statement

- Cluster the records/mentions that correspond tosame entity
  - Compute cluster representative



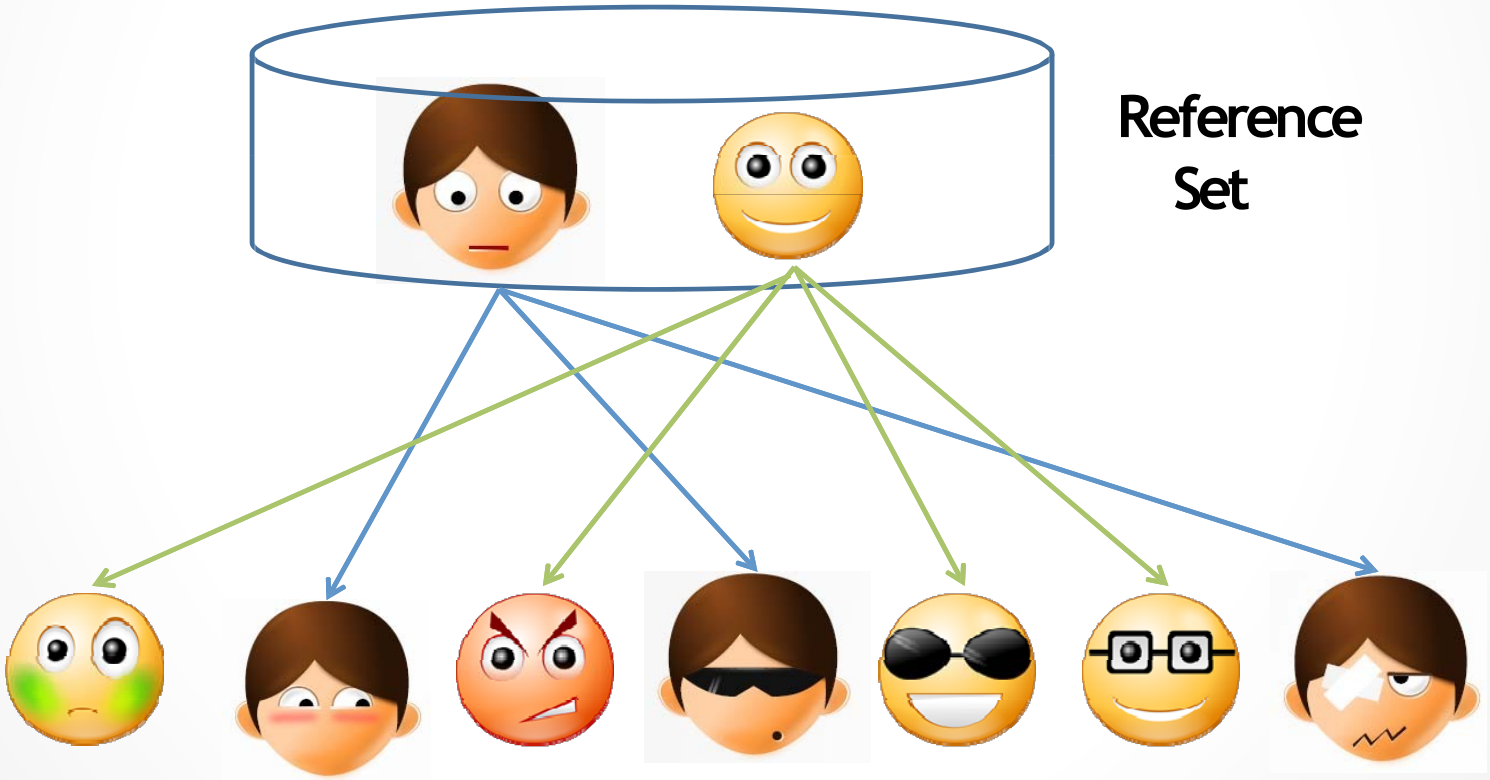
Record Linkage Problem Statement

- Link records that match across databases

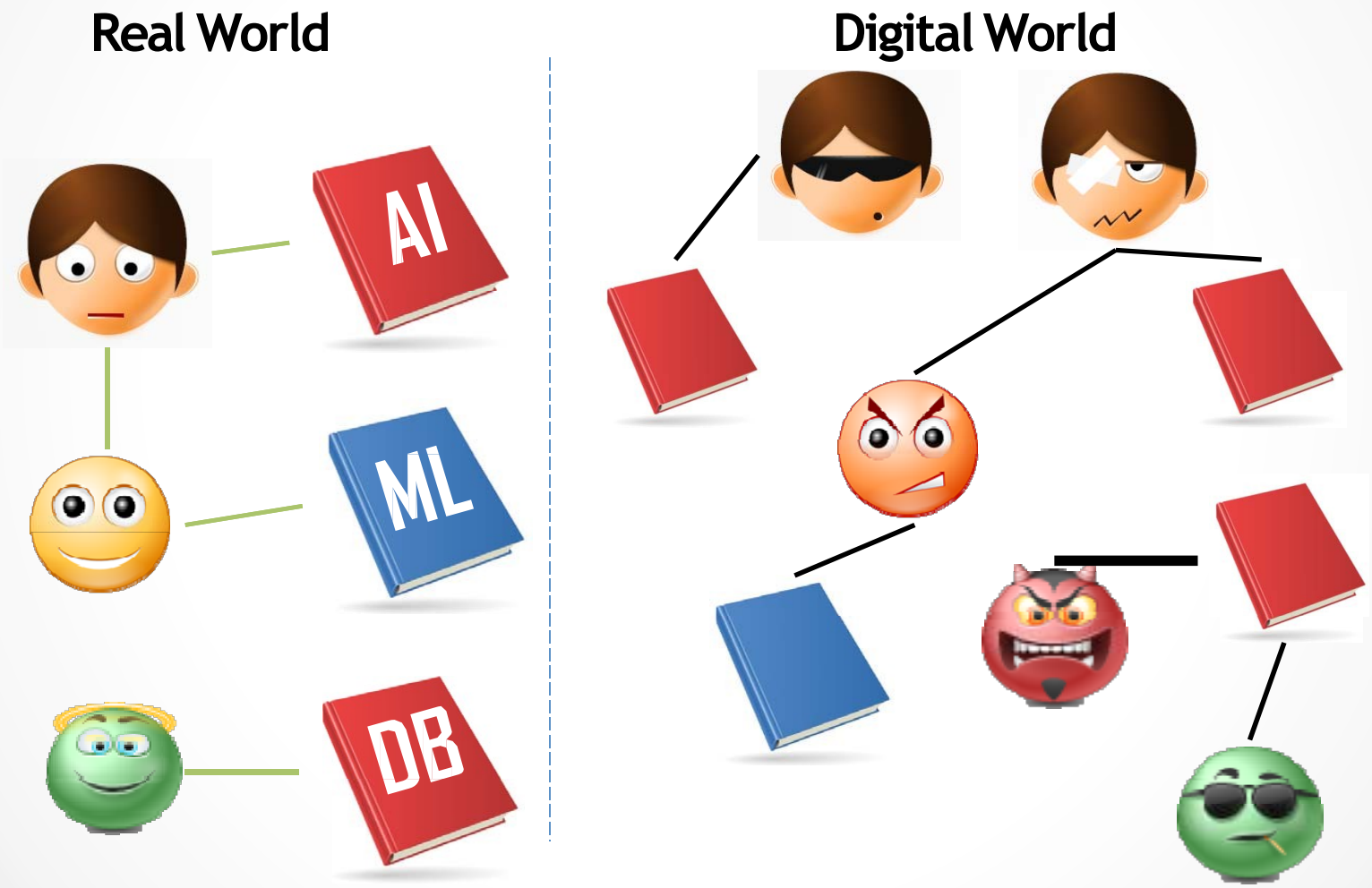


Reference Matching Problem

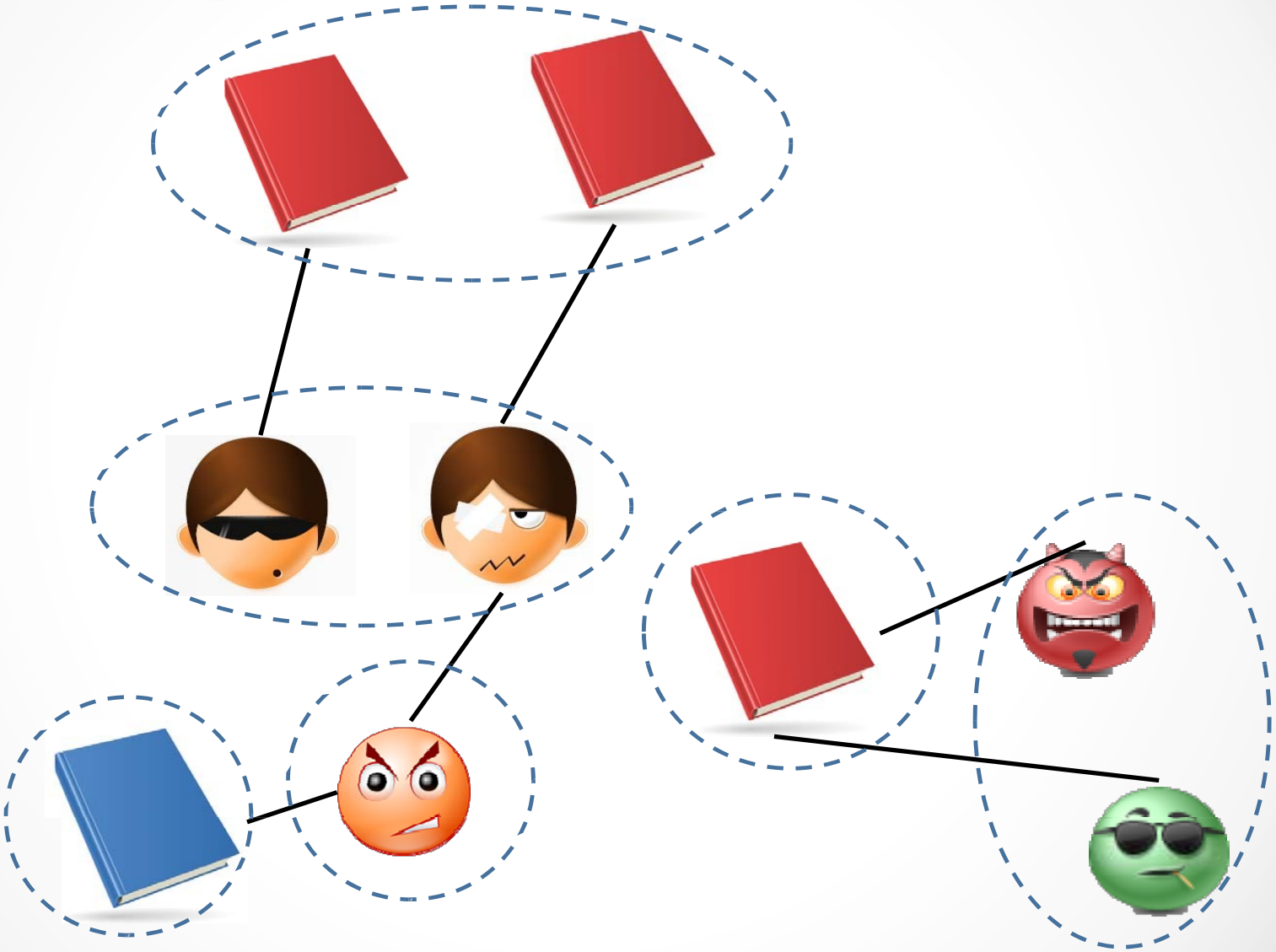
- Match noisy records to clean records in a reference table



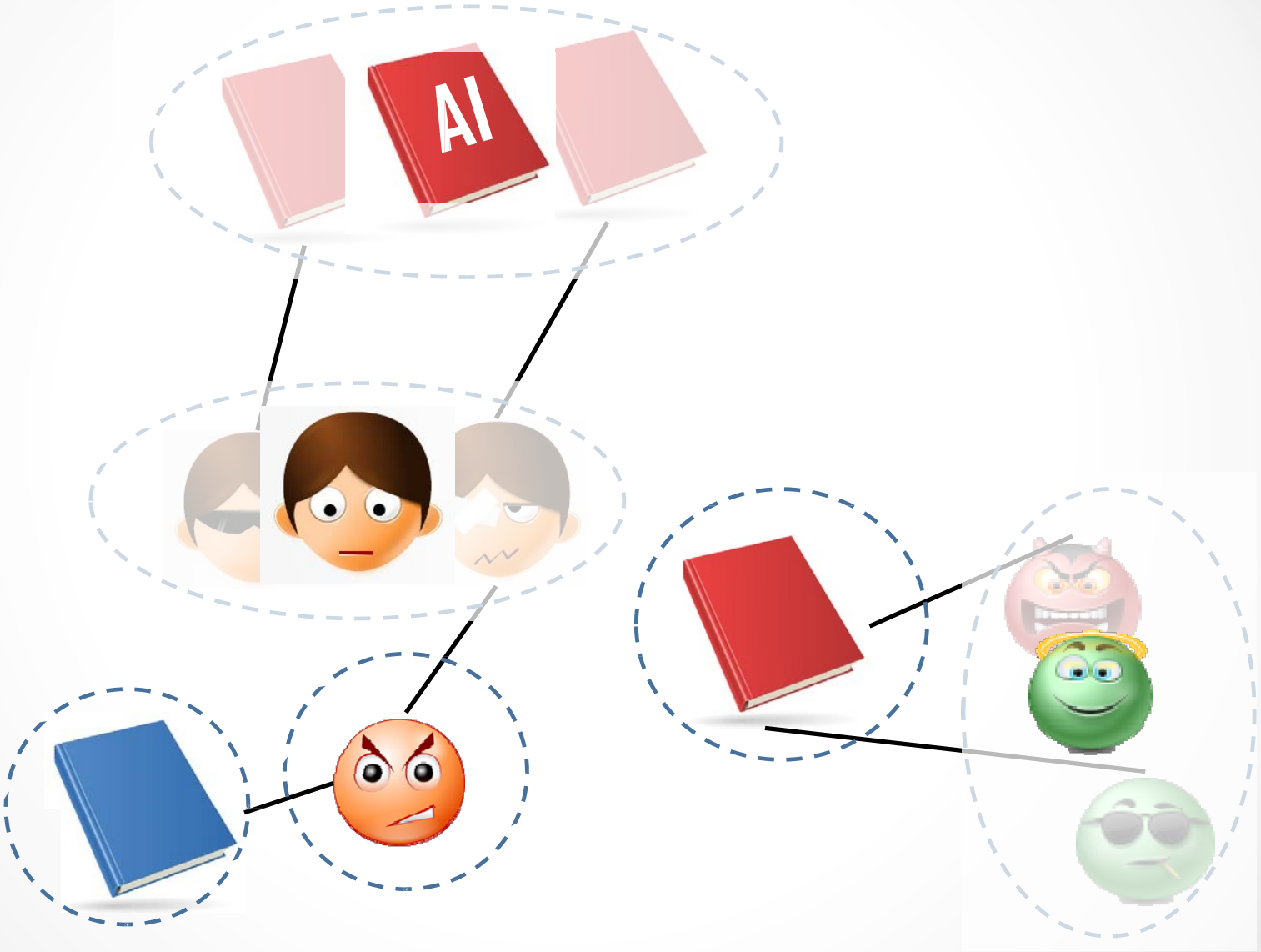
Abstract Problem Statement



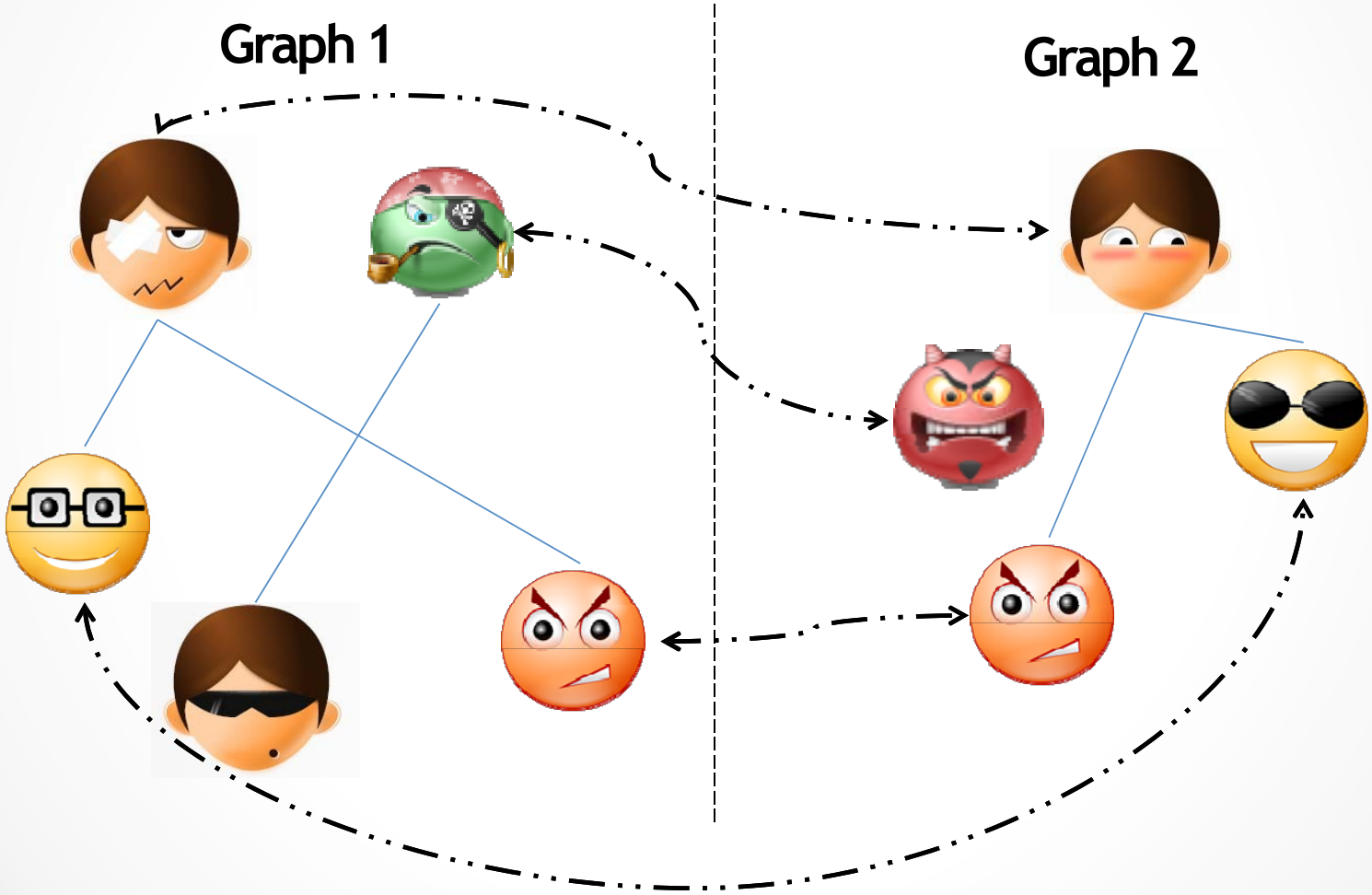
Deduplication Problem Statement



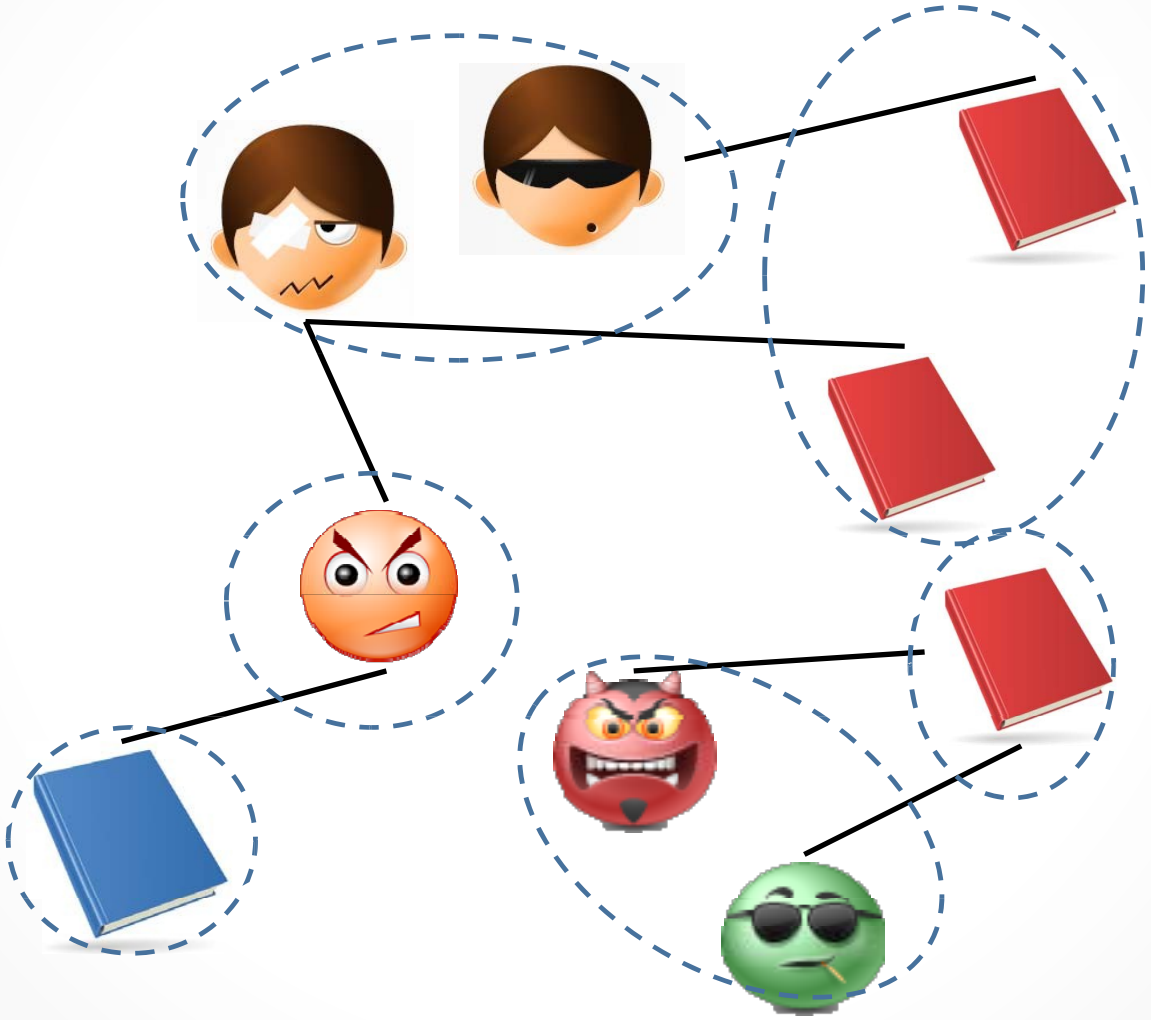
Deduplication with Canonicalization



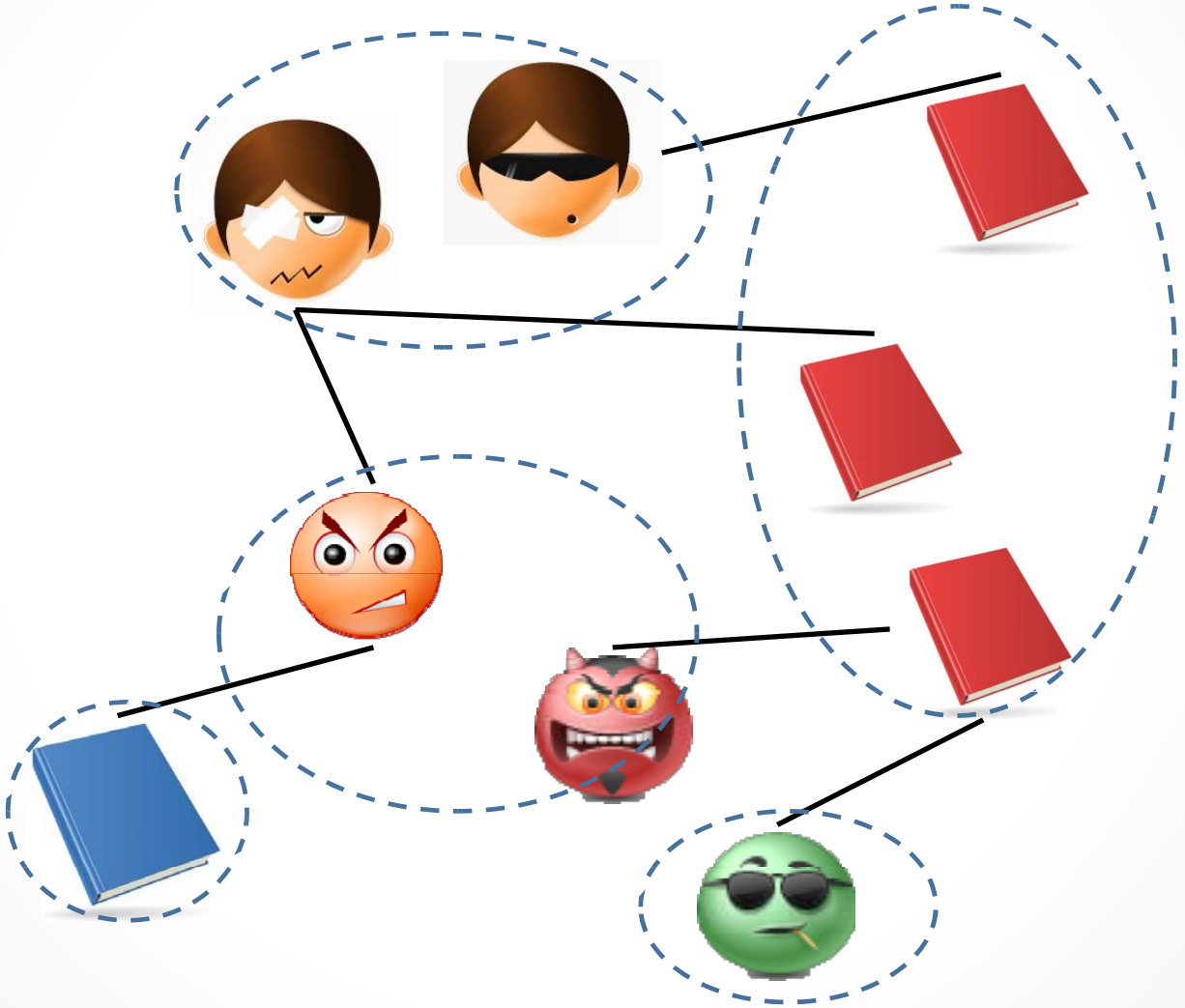
Graph Alignment



Relationships are crucial



Relationships are crucial



Problem Definition (textbook)

- Given two relational tables X and Y with identical schemas
  - assume each tuple in X and Y describes an entity (e.g., person)
- We say tuple x from X matches tuple y from Y if they refer to the same real-world entity
  - (x,y) is called a match
- Goal: find all matches between X and Y

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)

- Other variations
  - Tables X and Y have different schemas
  - Match tuples within a single table X
  - The data is not relational, e.g., XML or RDF

Why is This Different than String Matching?

- In theory, can treat each tuple as a string by concatenating the fields, then apply string matching techniques
- But doing so makes it hard to apply sophisticated techniques and domain-specific knowledge
- E.g., consider matching tuples that describe persons
  - suppose we know that in this domain two tuples match if the names and phone match exactly
  - this knowledge is hard to encode if we use string matching
  - it is better to keep the fields apart

Challenges

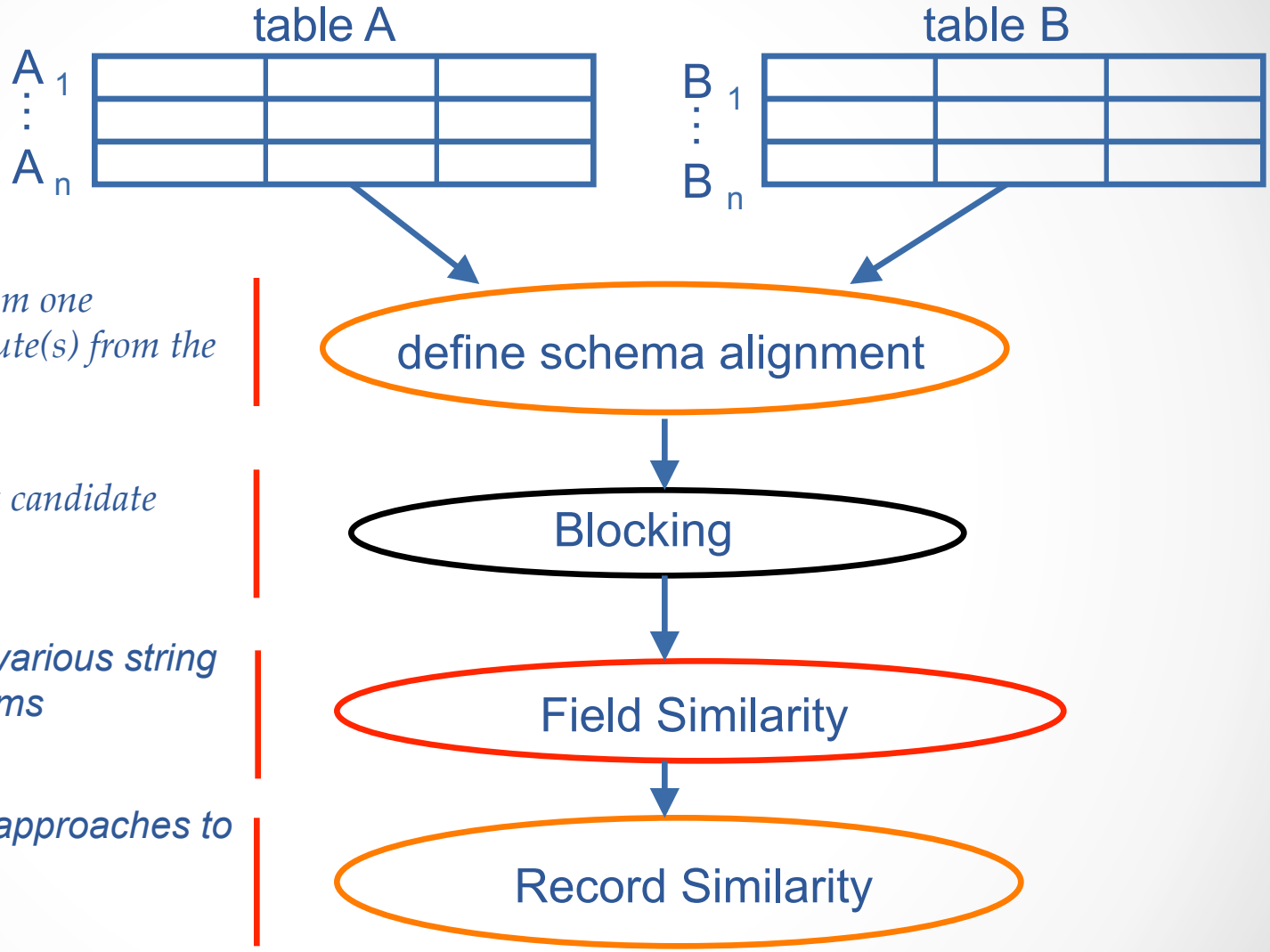
- Same as in string matching
- How to match accurately?
  - difficult due to variations in formatting conventions, use of abbreviations, shortening, different naming conventions, omissions, nicknames, and errors in data
  - several common approaches: rule-based, learning-based, clustering, probabilistic, collective
- How to scale up to large data sets
  - again many approaches have been developed, as we will discuss

General Approach to Record Linkage

1. Identification of candidate pairs (blocking)
   - Comparing all possible record pairs would be computationally wasteful
2. Compute Field Similarity
   - String similarity between individual fields is computed
3. Compute Record Similarity
   - Field similarities are combined into a total record similarity estimate

slide by Craig Knoblock

Overview



Map attribute(s) from one datasource to attribute(s) from the other datasource.

Find the most likely candidate pairs

Using one of the various string matching algorithms

Using one of the approaches to data matching

slide by Craig Knoblock

CC BY NC SA

Rule-based Matching

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, we will consider
  - linearly weighted combination of individual similarity scores
  - logistic regression combination
  - more complex rules

Linearly Weighted Combination Rules

- Compute the sim score between tuples x and y as a linearly weighted combination of individual sim scores
  - sim(x,y) = $\sum_{i=1}^{n} \alpha_i * sim_i(x, y)$
  - n is number of attributes in each table
  - $\mathbf{s}_i$**(x,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 $\sum_{i=1}^{n} \alpha_i = 1$
- We declare x and y matched if sim(x,y) $\geq \beta$ for a pre-specified β, and not matched otherwise
  - in another variation: declare x and y matched if sim(x,y) $\geq \beta$, not matched if sim(x,y) $\leq \gamma$, and subject to human review otherwise; we do not consider this in the chapter

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

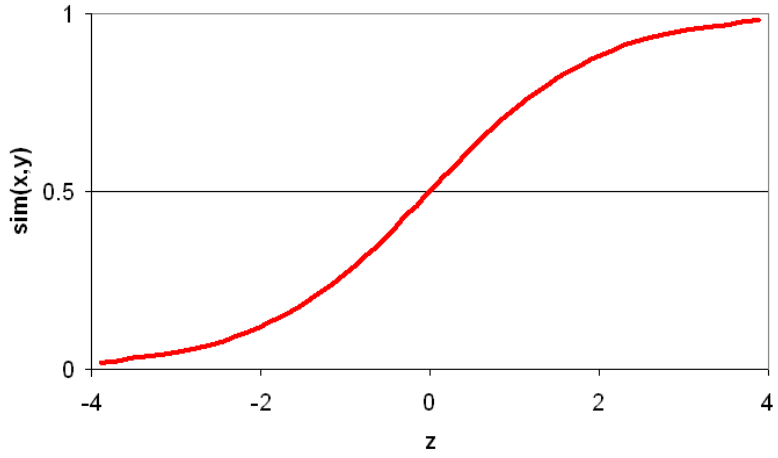
Pros and Cons

- Pros
  - conceptually simple, easy to implement
  - can learn weights α_i from training data

- Cons
  - an increase ± in the value of any $\mathbf{s}_i$ will cause a linear increase α_i * ± in the value of s
  - in certain scenarios this is not desirable, there 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

- address the diminishing-returns problem
- sim(x,y) = 1 / (1 + e^{-z}), where z = $\sum_{i=1}^{n} \alpha_i * sim_i(x, y)$
  - here \alpha_i are not constrained to be in [0,1] and sum to 1
  - so z goes from -∞ to +∞, in which case sim(x,y) gradually increases, but minimally so after z has exceeded a certain value ➔ ensuring diminishing returns



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

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"

Pros and Cons of Rule-Based Approaches

- Pros
  - easy to start, conceptually relatively easy to understand, implement, debug
  - typically run fast
  - can encode complex matching knowledge
- Cons
  - can be labor intensive, it takes a lot of time to write good rules
  - can be difficult to set appropriate weights
  - in certain cases it is not even clear how to write rules
  - learning-based approaches address these issues

Learning-based Matching

Learning-based Matching

- Here we consider supervised learning
  - learn a matching model M from training data, then apply M to match new tuple pairs
  - will consider unsupervised learning later

Learning-based Matching

- The training:
  1. start with training data: $T = \{(x_1,y_1,l_1), \ldots (x_n,y_n,l_n)\}$, where each (x_i,y_i) is a tuple pair and l_i is a label: "yes" if x_i matches y_i and "no" otherwise

Learning-based Matching

- The training:
  1. start with training data: $T = \{(x_1,y_1,l_1), \ldots (x_n,y_n,l_n)\}$, where each (x_i,y_i) is a tuple pair and l_i is a label: "yes" if x_i matches y_i and "no" otherwise
  2. define a set of features $f_1, \ldots, f_m$, each quantifying one aspect of the domain judged possibly relevant to matching the tuples

Learning-based Matching

- The training:
  1. start with training data: T = {(x_1,y_1,l_1), … (x_n,y_n,l_n)}, where each (x_i,y_i) is a tuple pair and l_i is a label: "yes" if x_i matches y_i and "no" otherwise
  2. define a set of features f_1, …, f_m, each quantifying one aspect of the domain judged possibly relevant to matching the tuples
  3. convert each training example (x_i,y_i,l_i) in T to a pair (<f_1(x_i,y_i), …, f_m(x_i,y_i)>, c_i)
     1. v_i = <f_1(x_i,y_i), …, f_m(x_i,y_i)> is a feature vector that encodes (x_i,y_i) in terms of the features
     2. c_i is an appropriately transformed version of label l_i (e.g., yes/no or 1/0, depending on what matching model we want to learn)

Learning-based Matching

- The training:
  1. start with training data: T = {(x_1,y_1,l_1), … (x_n,y_n,l_n)}, where each (x_i,y_i) is a tuple pair and l_i is a label: "yes" if x_i matches y_i and "no" otherwise
  2. define a set of features f_1, …, f_m, each quantifying one aspect of the domain judged possibly relevant to matching the tuples
  3. convert each training example (x_i,y_i,l_i) in T to a pair (<f_1(x_i,y_i), …, f_m(x_i,y_i)>, c_i)
     1. v_i = <f_1(x_i,y_i), …, f_m(x_i,y_i)> is a feature vector that encodes (x_i,y_i) in terms of the features
     2. c_i is an appropriately transformed version of label l_i (e.g., yes/no or 1/0, depending on what matching model we want to learn)
  4. thus T is transformed into T' = {(v_1,c_1), …, (v_n,c_n)}

Learning-based Matching

- The training:
  1. start with training data: T = {(x_1,y_1,l_1), … (x_n,y_n,l_n)}, where each (x_i,y_i) is a tuple pair and l_i is a label: "yes" if x_i matches y_i and "no" otherwise
  2. define a set of features f_1, …, f_m, each quantifying one aspect of the domain judged possibly relevant to matching the tuples
  3. convert each training example (x_i,y_i,l_i) in T to a pair (<f_1(x_i,y_i), …, f_m(x_i,y_i)>, c_i)
     1. v_i = <f_1(x_i,y_i), …, f_m(x_i,y_i)> is a feature vector that encodes (x_i,y_i) in terms of the features
     2. c_i is an appropriately transformed version of label l_i (e.g., yes/no or 1/0, depending on what matching model we want to learn)
  4. thus T is transformed into T' = {(v_1,c_1), …, (v_n,c_n)}
  5. apply a learning algorithm (e.g. decision trees, SVMs) to T' to learn a matching model M

Learning-based Matching

- Applying model M to match new tuple pairs
  - given pair (x,y), transform it into a feature vector

    $v = \langle f_1(x,y), \ldots, f_m(x,y) \rangle$
  - apply M to v to predict whether x matches y

Example: Learning a Linearly Weighted Rule

<a_1 = (Mike Williams, (425) 247 4893, Seattle, WA), b_1 = (M. Williams, 247 4893, Redmond, WA), yes>
<a_2 = (Richard Pike, (414) 256 1257, Milwaukee, WI), b_2 = (R. Pike, 256 1237, Milwaukee, WI), yes>
<a_3 = (Jane McCain, (206) 111 4215, Renton, WA), b_3 = (J. M. McCain, 112 5200, Renton, WA), no>

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

$v_1 = \langle [s_1(a_1,b_1), s_2(a_1,b_1), s_3(a_1,b_1), s_4(a_1,b_1), s_5(a_1,b_1), s_6(a_1,b_1)], 1 \rangle$

$v_2 = \langle [s_1(a_2,b_2), s_2(a_2,b_2), s_3(a_2,b_2), s_4(a_2,b_2), s_5(a_2,b_2), s_6(a_2,b_2)], 1 \rangle$

$v_3 = \langle [s_1(a_3,b_3), s_2(a_3,b_3), s_3(a_3,b_3), s_4(a_3,b_3), s_5(a_3,b_3), s_6(a_3,b_3)], 0 \rangle$

- s_1 and s_2 use Jaro-Winkler and edit distance
- s_3 uses edit distance (ignoring area code of a)
- s_4 and s_5 return 1 if exact match, 0 otherwise
- s_6 encodes a heuristic constraint

Example: Learning a Linearly Weighted Rule

- Goal: learn rule s(a,b) = $\sum_{i=1}^{6} \alpha_i s_i(a,b)$
- Perform a least-squares linear regression on training data

$v_1 = <[s_1(a_1,b_1), s_2(a_1,b_1), s_3(a_1,b_1), s_4(a_1,b_1), s_5(a_1,b_1), s_6(a_1,b_1)], 1>$

$v_2 = <[s_1(a_2,b_2), s_2(a_2,b_2), s_3(a_2,b_2), s_4(a_2,b_2), s_5(a_2,b_2), s_6(a_2,b_2)], 1>$

$v_3 = <[s_1(a_3,b_3), s_2(a_3,b_3), s_3(a_3,b_3), s_4(a_3,b_3), s_5(a_3,b_3), s_6(a_3,b_3)], 0>$

to find weights α_i that minimizes the squared error

$$\sum_{i=1}^{3} (c_i - \sum_{j=1}^{6} \alpha_j s_j(v_i))^2$$

- $\mathbf{c}_i$ is the label associated with $\mathbf{v}_i$
- $\mathbf{s}_j(\mathbf{v}_i)$ is the value of the j-th element of $\mathbf{v}_i$

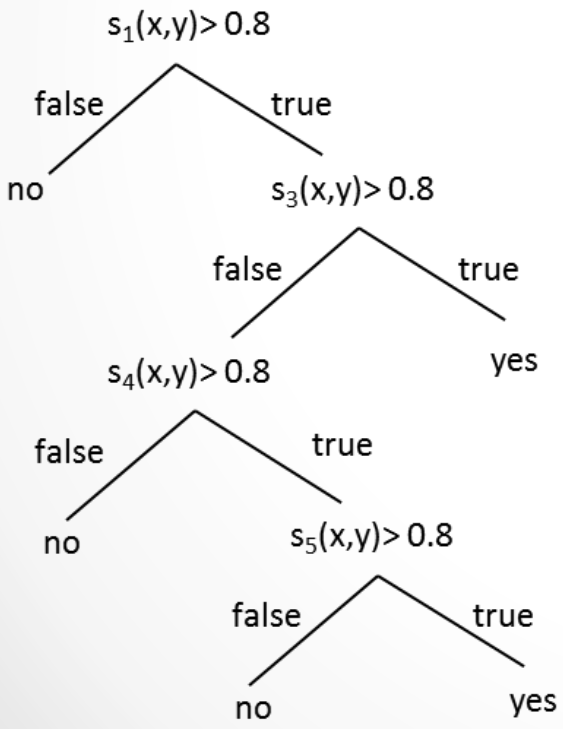
Example: Learning a Decision Tree

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

$v_1 = <[s_1(a_1,b_1), s_2(a_1,b_1), s_3(a_1,b_1), s_4(a_1,b_1), s_5(a_1,b_1), s_6(a_1,b_1)]$, yes>

$v_2 = <[s_1(a_2,b_2), s_2(a_2,b_2), s_3(a_2,b_2), s_4(a_2,b_2), s_5(a_2,b_2), s_6(a_2,b_2)]$, yes>

$v_3 = <[s_1(a_3,b_3), s_2(a_3,b_3), s_3(a_3,b_3), s_4(a_3,b_3), s_5(a_3,b_3), s_6(a_3,b_3)]$, no>



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

The Pros and Cons of Learning-based Approach

- Pros compared to rule-based approaches
  - in rule-based approaches must manually decide if a particular feature is useful ➔ labor intensive and limit the number of features we can consider
  - learning-based ones can automatically examine a large number of features
  - learning-based approaches can construct very complex rules
- Cons
  - still require training examples, in many cases a large number of them, which can be hard to obtain
  - clustering can address this problem

Matching by Clustering …

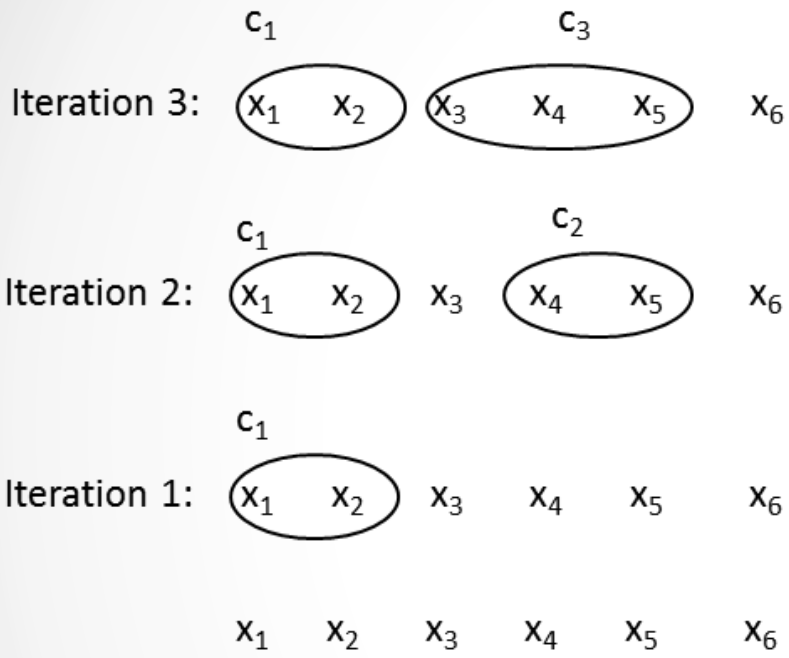
Matching by Clustering

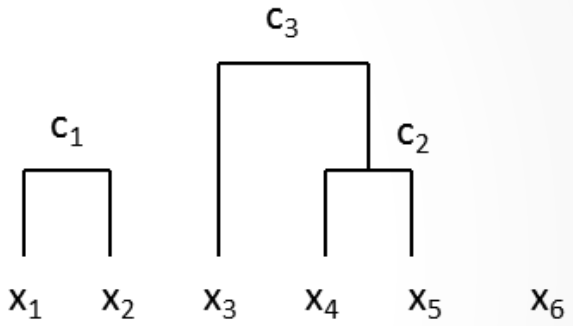
- Many common clustering techniques have been used
  - agglomerative hierarchical clustering (AHC), k-means, graph-theoretic, …
  - here we focus on AHC, a simple yet very commonly used one

Matching by Clustering

- 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:

$$s(c,d) = \mathbf{min}_{xi2c,\ yj2d}\ \mathbf{sim}(x_i, y_j)$$

- Complete link:

$$s(c,d) = \mathbf{max}_{xi2c,\ yj2d}\ \mathbf{sim}(x_i, y_j)$$

- Average link:

$$s(c,d) = [\sum_{xi2c,\ yj2d}\ \mathbf{sim}(x_i, y_j)] /\ [\#\text{ of } (x_i, y_j) \text{ pairs}]$$

Computing a Similarity Score between Two Clusters

- Canonical tuple:
  - 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" & "M. J. Williams" ➔ "Mike J. Williams"
    (425) 247 4893 & 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

Scaling Up Data Matching …

Scaling up Data Matching

- Two goals:
  - minimize # of tuple pairs to be matched (Blocking)
  - minimize time it takes to match each pair

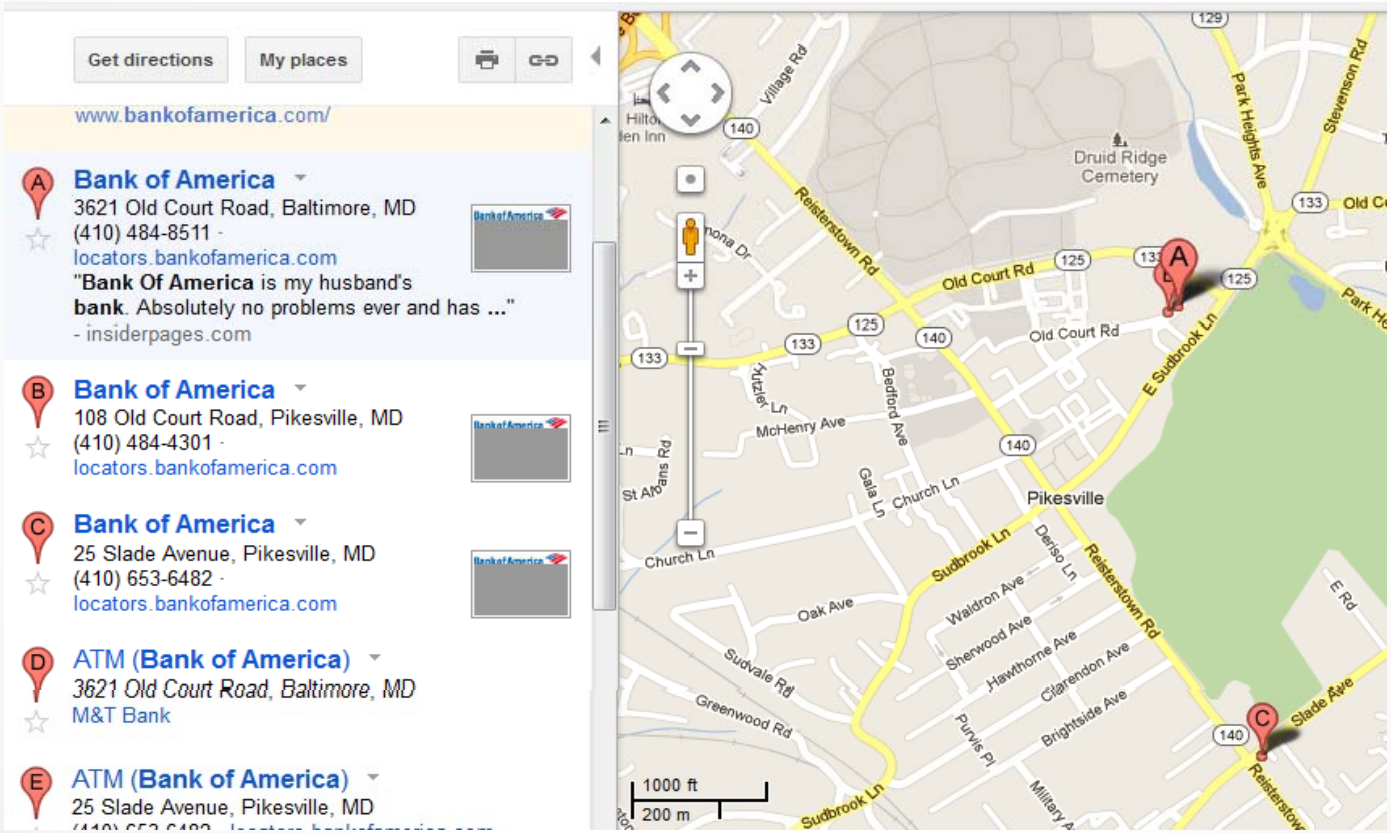
Blocking: Motivation

- Naïve pairwise: $|R|^2$ pairwise comparisons
  - 1000 business listings each from 1,000 different cities across the world
  - 1 trillion comparisons
  - 11.6 days (if each comparison is 1 μs)

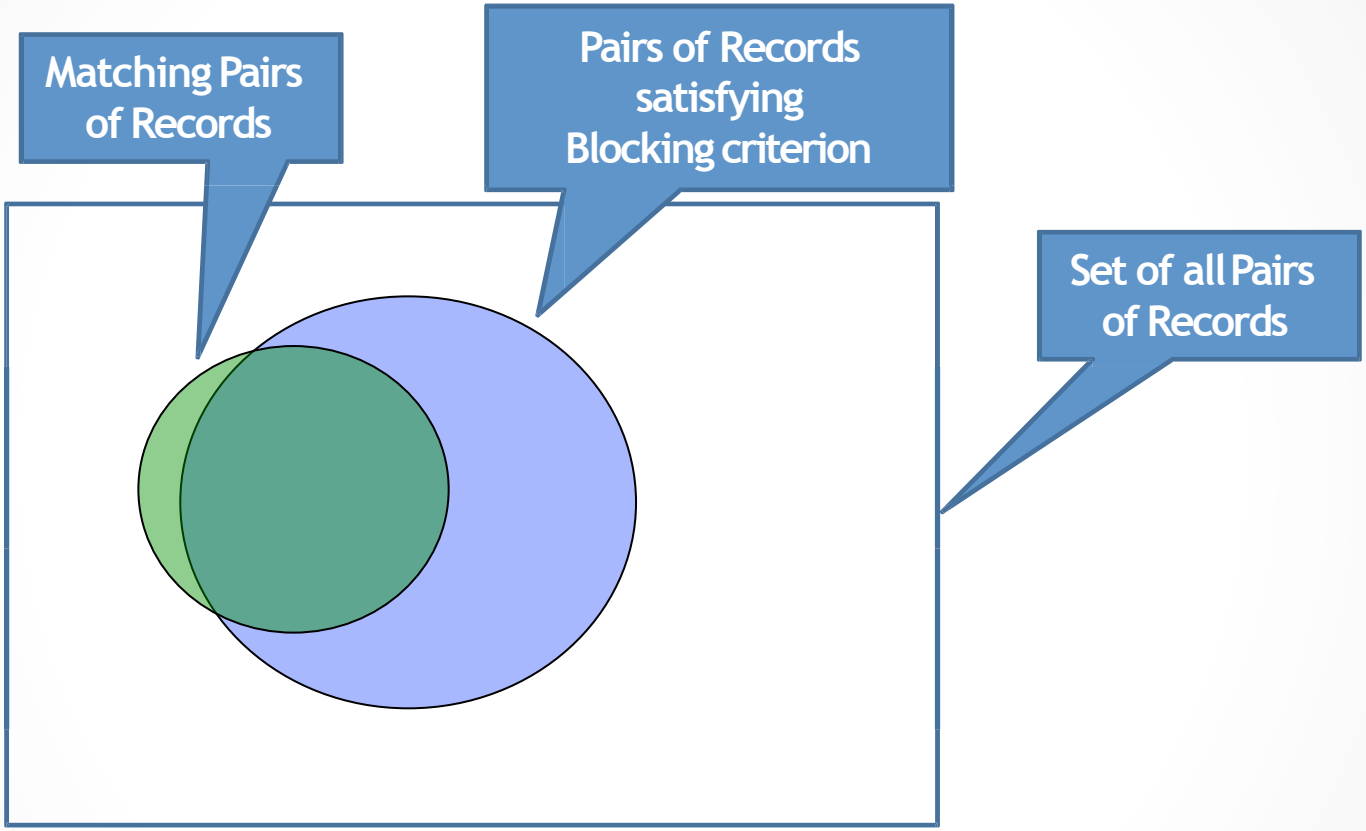
- Mentions from different cities are unlikely to be matches
  - **Blocking Criterion: City**
  - 1 billion comparisons
  - 16 minutes (if each comparison is 1 μs)

Blocking: Motivation

- Mentions from different cities are unlikely to be matches
  - May miss potential matches



Blocking: Motivation



Scaling up Data Matching

- Blocking:
  - hashing
  - sorting
  - indexing
  - canopies
  - using representatives
  - combining the techniques

Scaling up Data Matching

- Simple Hashing
  - hash tuples into buckets, match only tuples within each bucket
  - e.g., hash house listings by zipcode, then match within each zip

Scaling up Data 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

Scaling up Data 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

Scaling up Data Matching

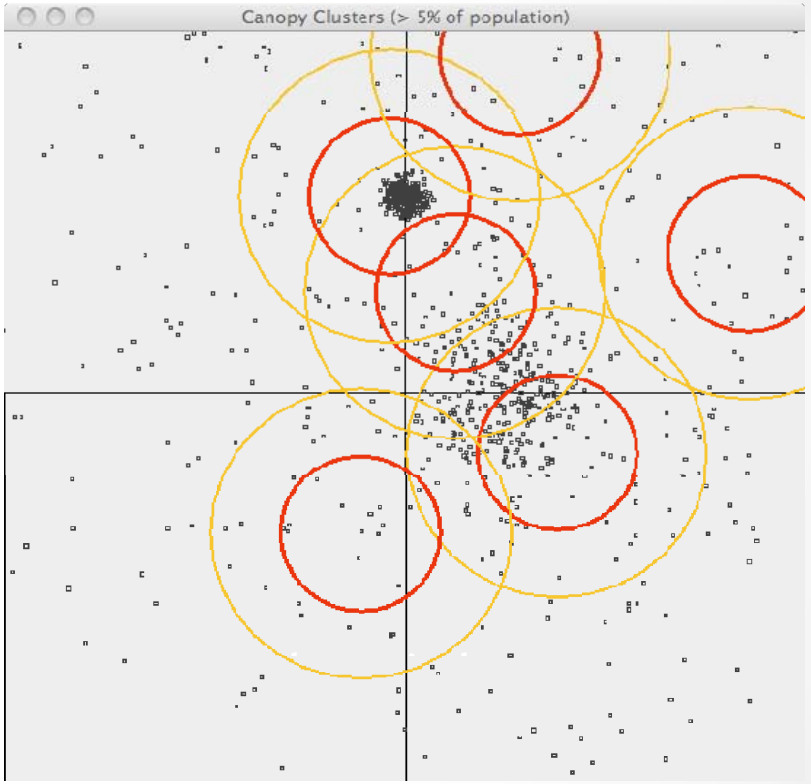
- Canopies
  - use a computationally cheap sim measure to quickly group tuples into overlapping clusters called canopines (or umbrella sets)
  - use a different (far more expensive) sim measure to match tuples within each canopy
  - e.g., Strings that differ by at least 3 in length can not have an edit distance smaller than 3.

Canopy Clustering [McCallum et al KDD'00]

Input: Mentions M,
d(x,y), a distance metric,
thresholds T1 > T2

Algorithm:

1. Pick a random element x from M
2. Create new canopy Cx using mentions y s.t. d(x,y) < T1
3. Delete all mentions y from M s.t. d(x,y) < T2
4. Return to Step 1 if M is not empty.



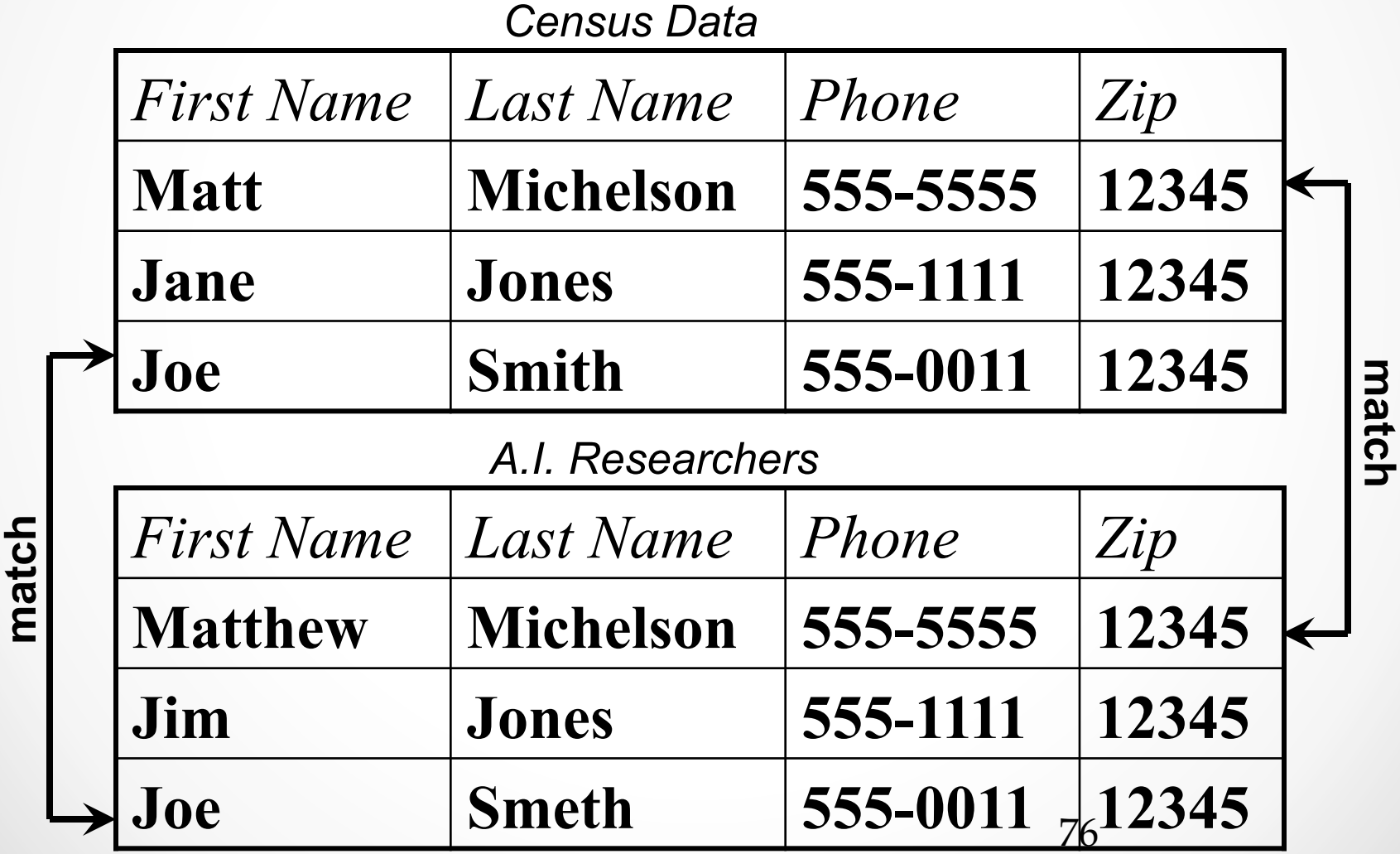
Scaling up Data Matching

- 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

Scaling up Data Matching

- 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

Multi-Pass Blocking: Example



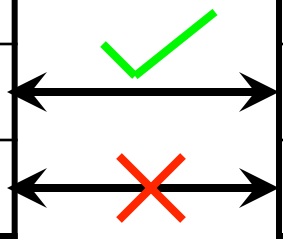
Slide by Matthew Michelson

Rule-based Blocking: Generating Candidates

(method, attribute)

(token, last name) AND (1st letter, first name) = block-key

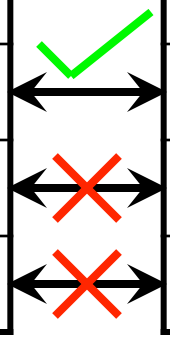
First Name	*Last Name*
Matt	**Michelson**
Jane	**Jones**



First Name	*Last Name*
Matthew	**Michelson**
Jim	**Jones**

(token, zip)

First Name	Last Name	Zip
Matt	Michelson	12345
Matt	Michelson	12345
Matt	Michelson	12345



First Name	Last Name	Zip
Matthew	Michelson	12345
Jim	Jones	12345
Joe	Smeth	12345

Blocking – Multi-pass
(Hernandez & Stolfo, 1998)

- Sort neighborhoods on block keys
- Multiple independent runs using keys
  - runs capture different match candidates
- Example:
    - 1st → (token, last name)
    - 2nd → (token, first name) & (token, phone)

Slide by Matthew Michelson

Blocking – Multi-pass

- Terminology:
  - Each pass is a “conjunction”
    - (token, first) AND (token, phone)
  - Combine passes to form “disjunction”
    - [(token, last)] OR
      [(token, first) AND (token, phone)]
  - Disjunctive Normal Form rules
    - form “Blocking Schemes”

Slide by Matthew Michelson

Blocking Effectiveness

- Determined by rules
- Determined by choices for attributes and methods
  - (token, zip) captures all matches, but all pairs too
  - (token, first) AND (token, phone) gets half the matches, and only 1 candidate generated
  - Which is better? Why?
- How to quantify??

Slide by Matthew Michelson

Blocking Effectiveness

Reduction Ratio (RR) = 1 – ||C|| / (||S|| *|| T||)

S,T are data sets;
C is the set of candidates

Pairs Completeness (PC) [Recall] = S_m / N_m

S_m = # true matches in candidates,
N_m = # true matches between S and T

Examples:

(token, last name) AND (1st letter, first name)

RR = 1 – 2/9 ≈ 0.78

PC = 1 / 2 = 0.50

(token, zip)

RR = 1 – 9/9 = 0.0

PC = 2 / 2 = 1.0

Census Data

First Name	*Last Name*	*Phone*	*Zip*
Matt	**Michelson**	**555-5555**	**12345**
Jane	**Jones**	**555-1111**	**12345**
Joe	**Smith**	**555-0011**	**12345**

A.I. Researchers

First Name	*Last Name*	*Phone*	*Zip*
Matthew	**Michelson**	**555-5555**	**12345**
Jim	**Jones**	**555-1111**	**12345**
Joe	**Smeth**	**555-0011**	**12345**

match

match

Slide by Matthew Michelson

How to choose methods and attributes?

- Blocking Goals:
  - Small number of candidates (High RR)
  - Don't leave any true matches behind! (High PC)
- Previous approaches:
  - Ad-hoc by researchers or domain experts
- New Approach:
  - Blocking Scheme Learner (BSL) – modified Sequential Covering Algorithm

Michelson & Knoblock, Learning Blocking Schemes for Record Linkage, AAAI 2006

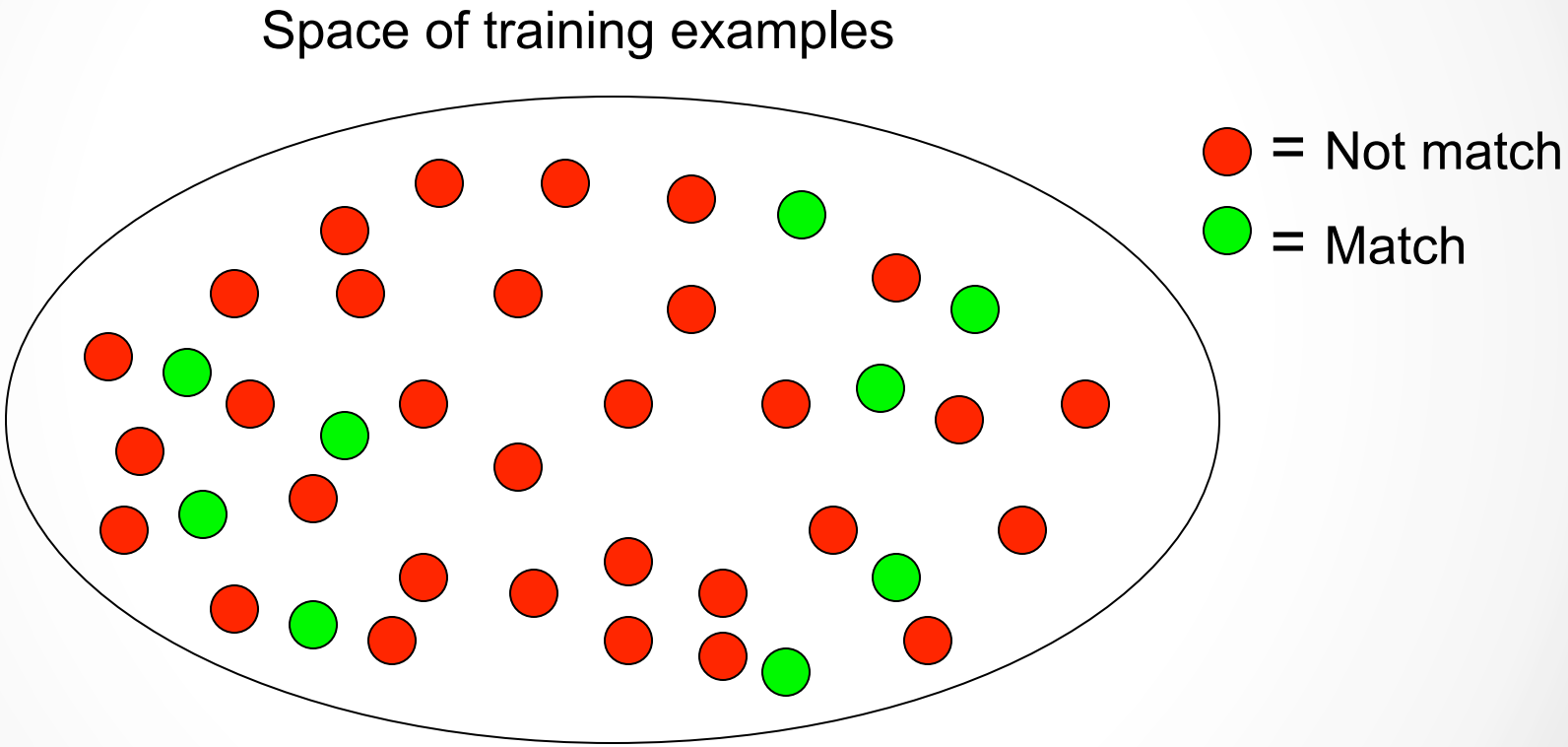
Slide by Matthew Michelson

Learning Schemes – Intuition

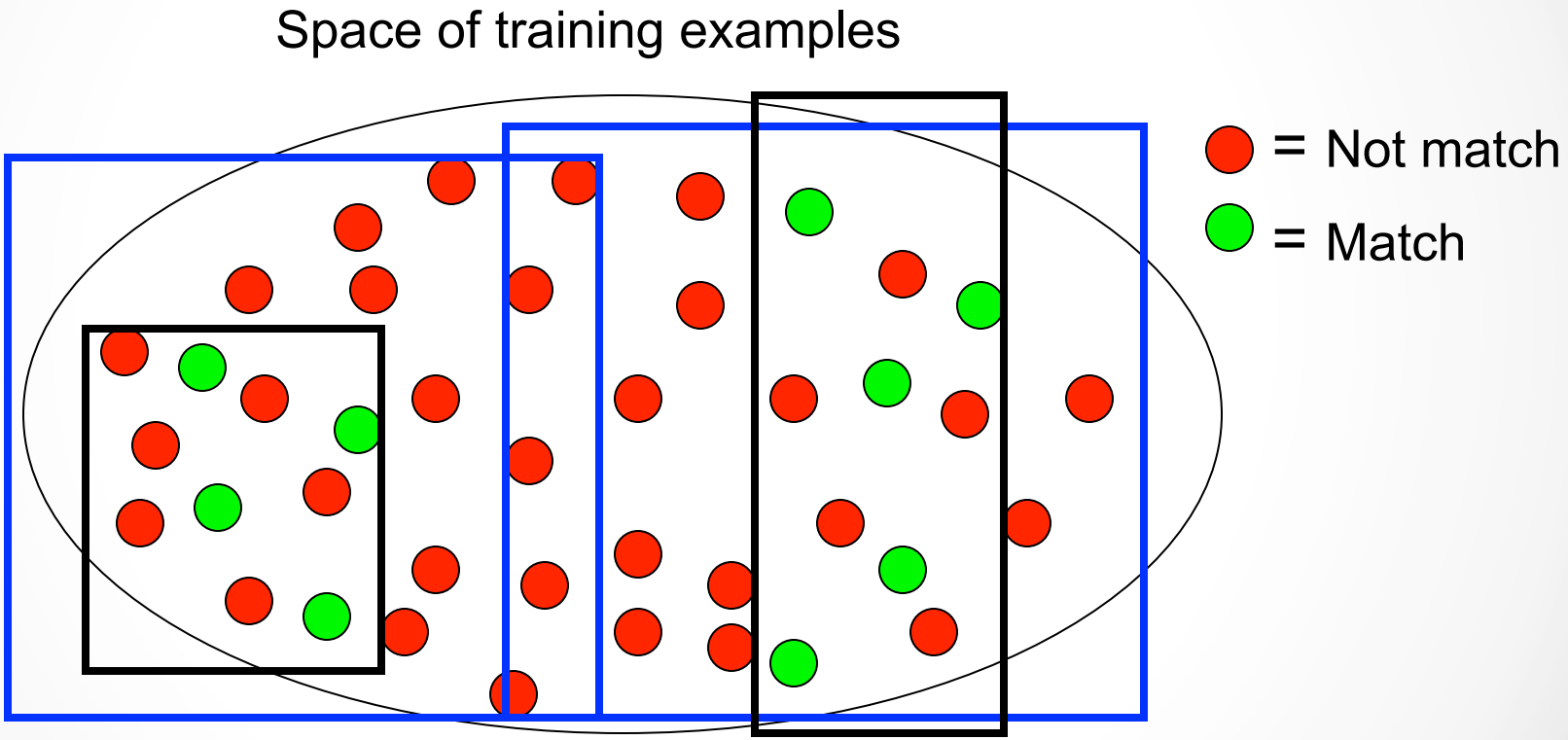
- Learn restrictive conjunctions
  - partition the space → minimize False Positives
- Union restrictive conjunctions
  - Cover all training matches
  - Since minimized FPs, conjunctions should not contribute many FPs to the disjunction

Slide by Matthew Michelson

Example to clear things up!



Example to clear things up!



Rule 1 :- (zip|token) & (first|token)

Final Rule :- [(zip|token) & (first|token)] UNION [(last|1st Letter) & (first|1st letter)]

Rule 2 :- (last|1st Letter) & (first|1st Letter)

Slide by Matthew Michelson

SCA: propositional rules

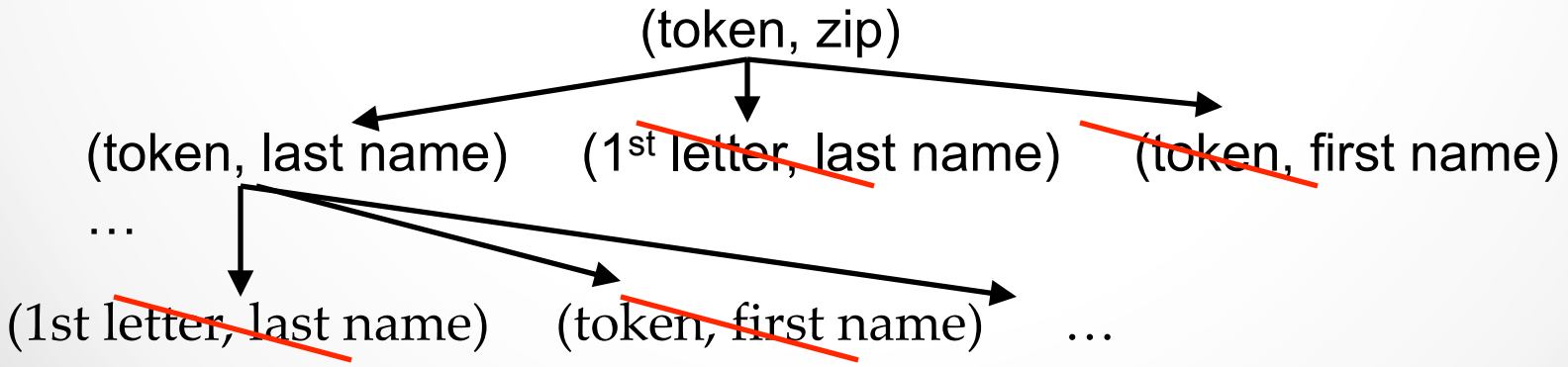
- Multi-pass blocking = disjunction of conjunctions
- Learn conjunctions and union them together!
- Cover all training matches to maximize PC

```
SEQUENTIAL-COVERING( class, attributes, examples, threshold)
LearnedRules ← {}
Rule ← LEARN-ONE-RULE(class, attributes, examples)
While examples left to cover, do
    LearnedRules ← LearnedRules U Rule
    Examples ← Examples – {Examples covered by Rule}
    Rule ← LEARN-ONE-RULE(class, attributes, examples)
    If Rule contains any previously learned rules, remove them
Return LearnedRules
```


Slide by Matthew Michelson

Learn-One-Rule

- Learn conjunction that maximizes RR
- General-to-specific beam search
  - Keep adding/intersecting (attribute, method) pairs
    - Until can't improve RR
    - Must satisfy minimum PC



Slide by Matthew Michelson

Scaling up Data Matching

- Two goals:
  - minimize # of tuple pairs to be matched (Blocking)
  - minimize time it takes to match each pair

Scaling up Data 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 (i.e., matches individual attributes then combines their scores using weights)
    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

Scaling up Using Parallel Processing

- Commonly done in practice
- Examples: hash tuples into buckets, then match each bucket in parallel