

PART 2

CLASSICAL SINGLE ENTITY ER

Outline

1. Introduction & Motivation
2. Classical Single Entity ER
 - a) Problem Statement
 - b) Data Preparation
 - c) Matching Features
 - d) Pairwise Approaches
 - e) Clustering based Approaches
 - f) Canonicalization
3. Efficiency: Blocking/Canopies
4. Relational & MultiEntity ER
5. Demo
6. Challenges & Future Directions

Key References: [Christen, 2012], [Koudas et al, SIGMOD06]

PART 2-a

ER PROBLEM STATEMENT

Abstract Problem Statement

Real World



Digital World

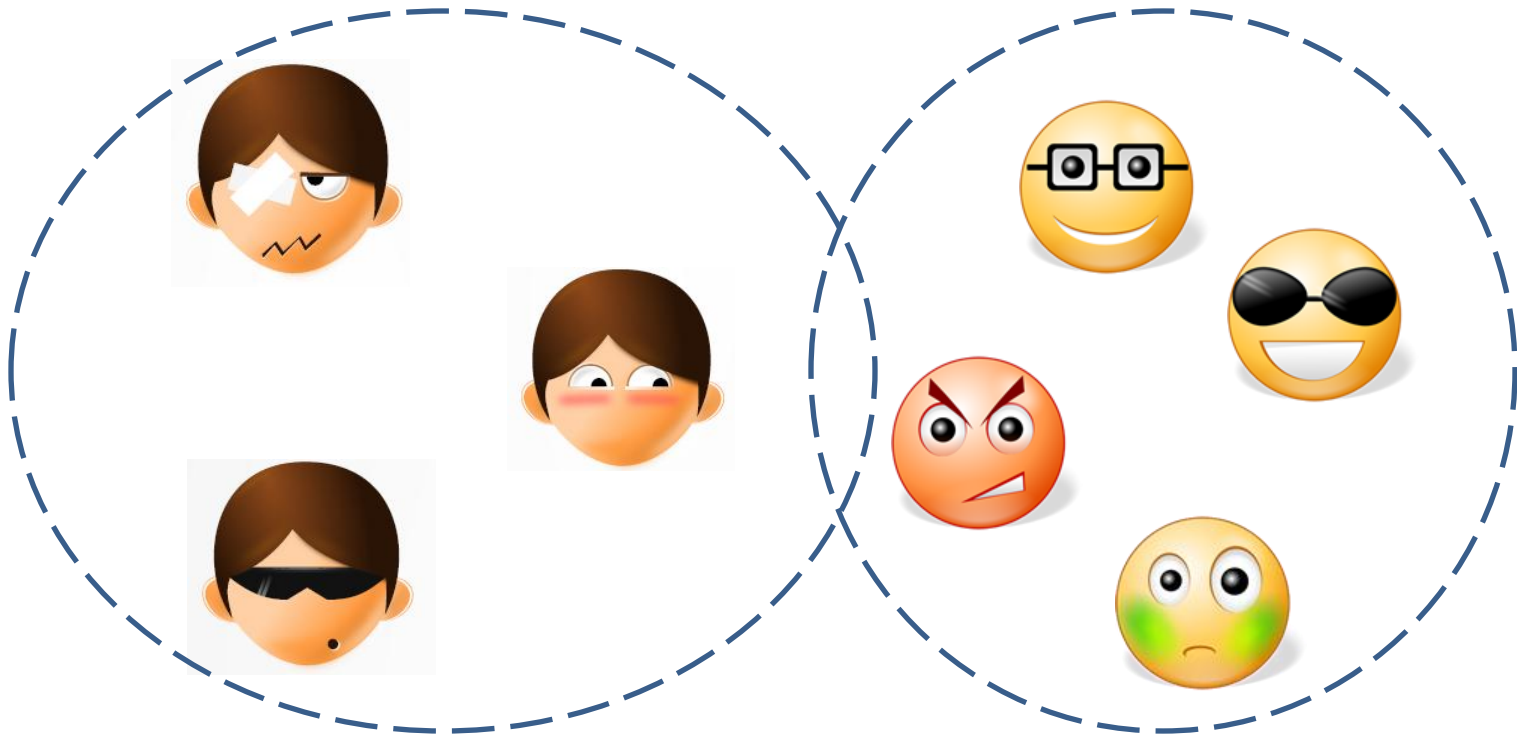


Records /
Mentions



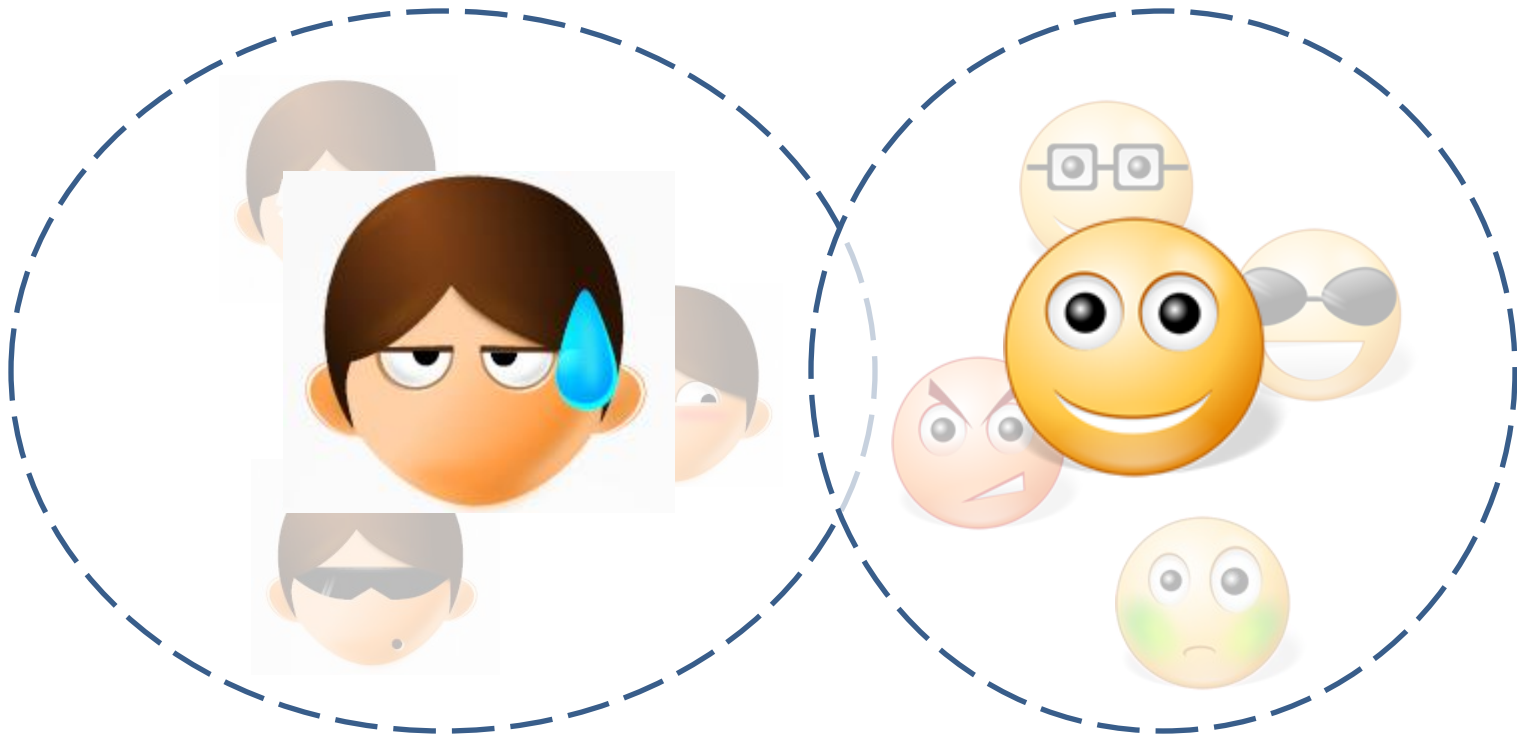
Deduplication Problem Statement

- Cluster the records/mentions that correspond to same entity



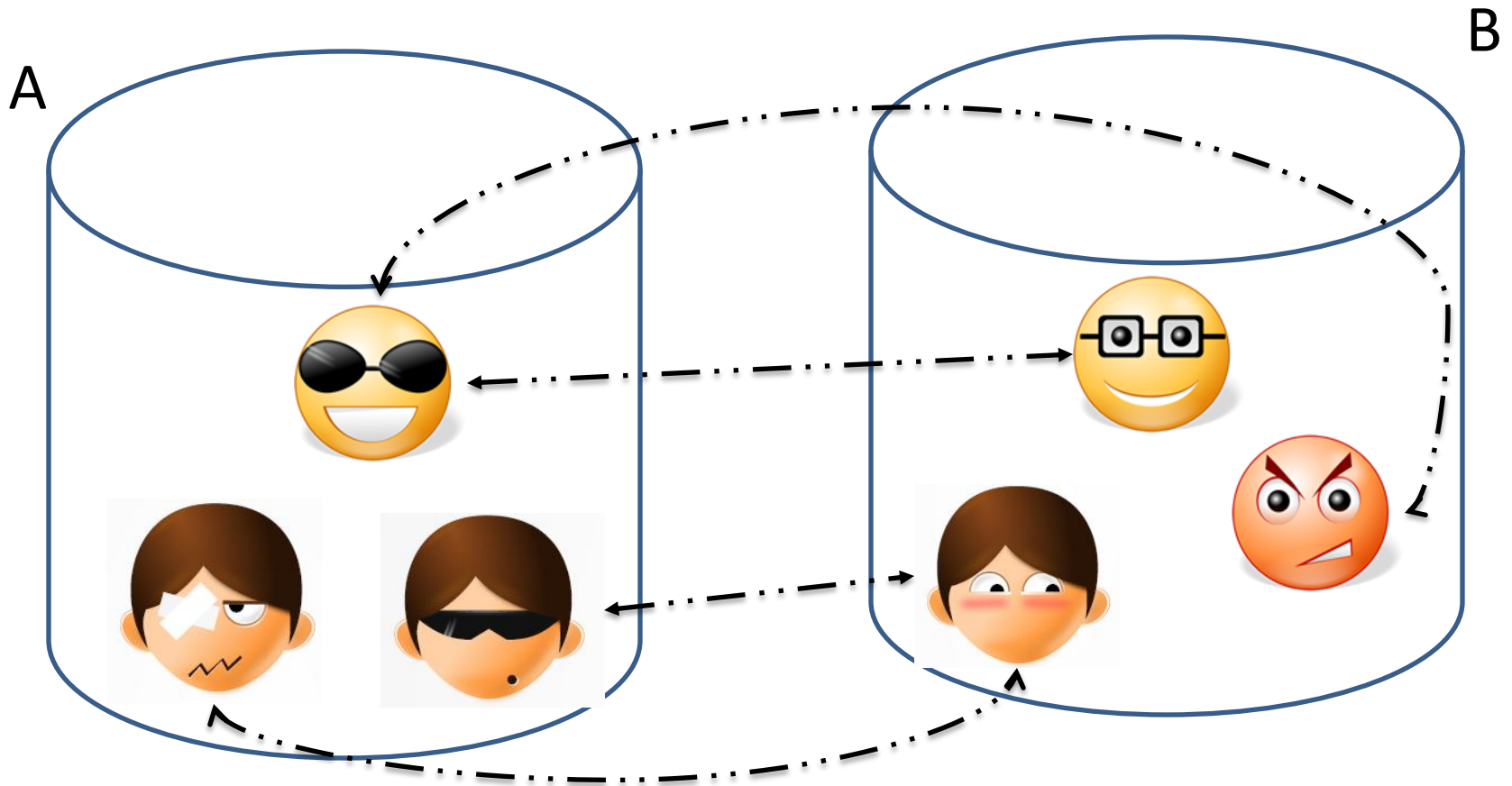
Deduplication Problem Statement

- Cluster the records/mentions that correspond to same entity
 - **Intensional Variant:** Compute cluster representative



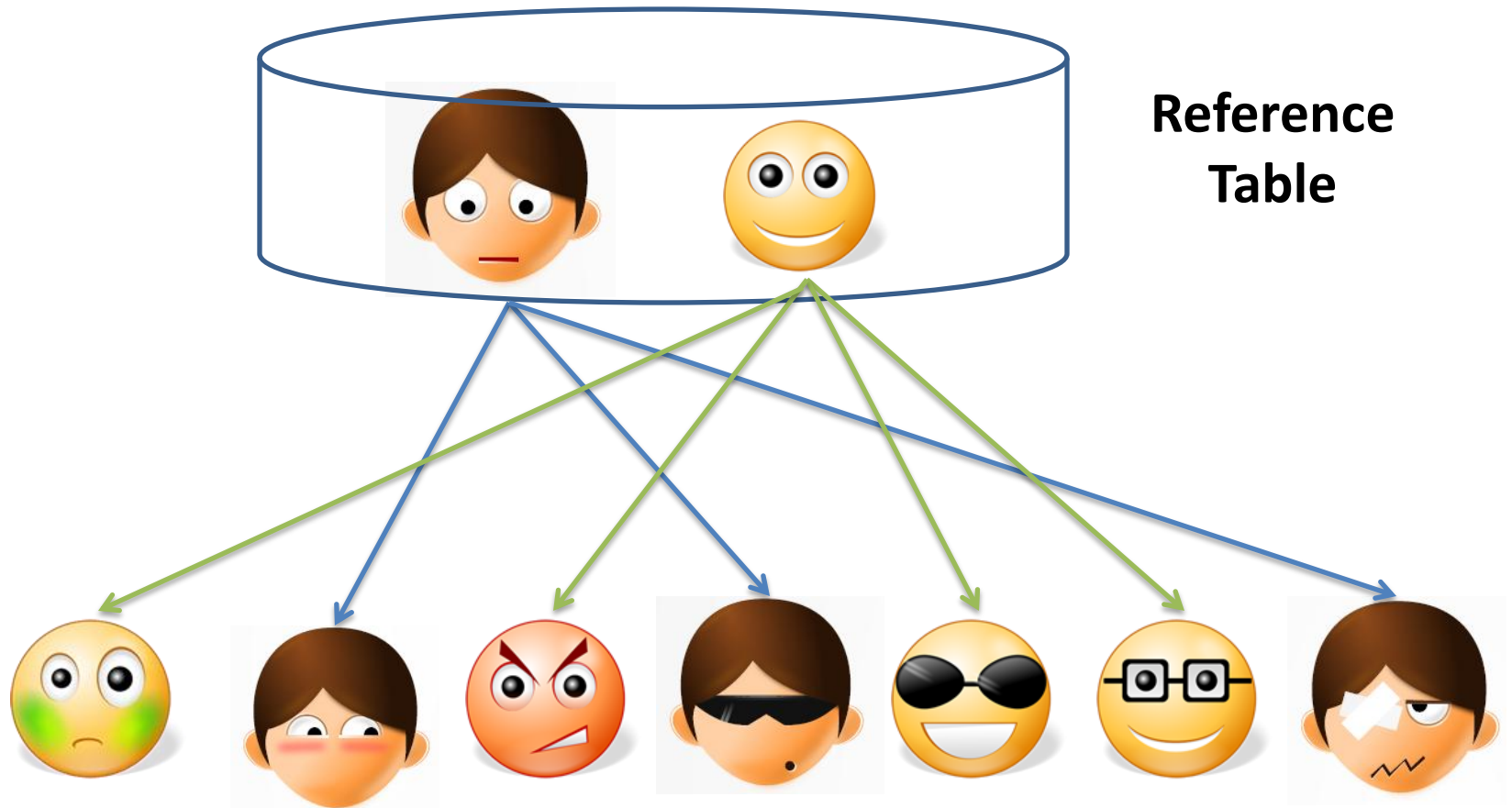
Record Linkage Problem Statement

- Link records that match across databases



Reference Matching Problem

- Match noisy records to clean records in a reference table



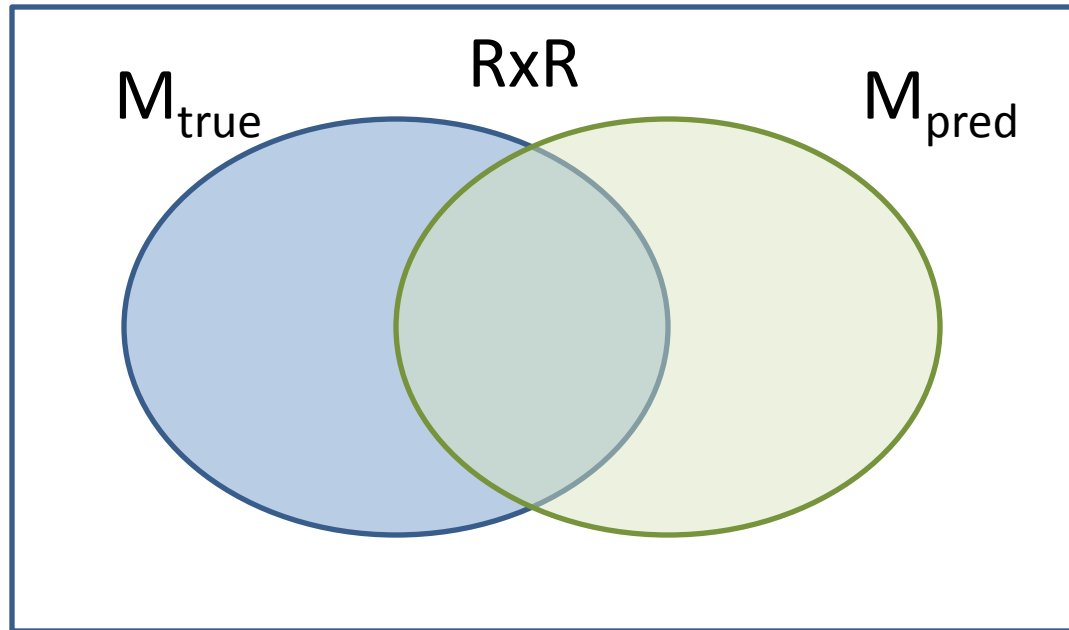
Notation & Assumptions

- R : set of records / mentions
- M : set of *matches* (record pairs that correspond to same entity)
- N : set of *non-matches* (record pairs corresponding to different entities)
- E : set of entities

- True ($M_{true}, N_{true}, E_{true}$): according to real world
vs Predicted ($M_{pred}, N_{pred}, E_{pred}$): by algorithm

Relationship between M_{true} and M_{pred}

- M_{true} (SameAs , Equivalence)
- M_{pred} (Similar representations and similar attributes)

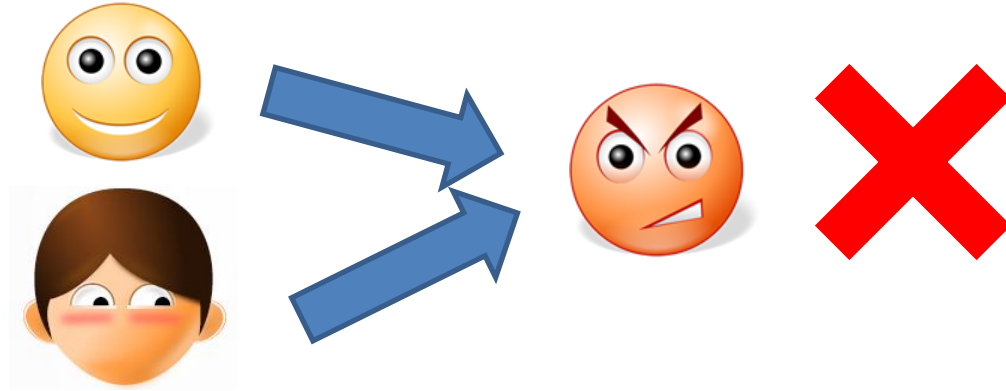


Metrics

- Pairwise metrics
 - Precision/Recall, F1
 - # of predicted matching pairs
- Cluster level metrics
 - purity, completeness, complexity
 - Precision/Recall/F1: Cluster-level, closest cluster, MUC, B^3 , Rand Index
 - Generalized merge distance [Menestrina et al, PVLDB10]

Typical Assumptions Made

- *Each record/mention is associated with a single real world entity.*



- *In record linkage, no duplicates in the same source*
- *If two records/mentions are identical, then they are true matches*

$$(\text{🕵️}, \text{🕵️}) \in M_{\text{true}}$$

ER versus Classification

Finding matches vs non-matches is a classification problem



- Imbalanced: typically $O(R)$ matches, $O(R^2)$ non-matches
- Instances are pairs of records. Pairs are not IID

  $\in M_{\text{true}}$

AND



  $\in M_{\text{true}}$

  $\in M_{\text{true}}$

ER vs Clustering

Computing entities from records is a clustering problem

- In typical clustering algorithms (k-means, LDA, etc.) *number of clusters is a constant or sub linear in R .*
- In ER: *number of clusters is linear in R , and average cluster size is a constant. Significant fraction of clusters are singletons.*

PART 2-b

DATA PREPARATION

Data Hygiene

- As with any data analysis process, the initial data preparation and preprocess is half the work; often times smart normalization can do a lot of the work toward solving ER problems!
- *Real data is dirty*
- Common Issues:
 - Typographical mistakes
 - Variations in single attributes
 - missing attribute values
 - values that are swapped between attributes
 - attributes that were incorrectly matched across different database schemas
- Important issue: assessing apriori data source quality

Example: Names

- Why are names important?
 - Most business, government, news, web sources are about people
 - Even scientific documents reference authors, affiliations, etc.
 - identifying data collected about individuals generally include names and addresses, dates of birth, social security or drivers license numbers, telephone numbers, and email addresses.
- Why are names hard?
 - Many variations – easy to misspell ‘Lise’ or ‘Machanavajjhala’
 - Names are different from regular text
 - Nicknames
 - Changing names (marriage/divorce)
 - Language/ cultural differences
 - *No unique correct name*

Schema Normalization

- Schema Matching – contact number and phone number
- Compound attributes – full address vs str,city,state,zip
- Nested attributes
 - List of features in one dataset (air conditioning, parking)
 - Each feature is a boolean attribute
- Set valued attributes
 - Set of phones vs primary/secondary phone
- Record segmentation from text

- Systems: e.g., Clio [Fagin et al, CM09], Google Refine

Data Normalization

- Common Preprocessing:
 - Convert to all lower/all upper
 - Remove whitespace
 - Rule-based/Table lookup mapping of characters and tokens
- Common Normalizations
 - detecting and correcting values that contain known typographical errors or variations,
 - expanding abbreviations and replacing them with standard forms
 - replacing nicknames with their proper name forms
 - Usually done based on dictionaries (e.g., commercial dictionaries, names, postal addresses, etc.)
- E.g., for ontology matching, intelligent use of camel case is important
 - BooksOrMagazines poor string matching with MagazinesOrBooks.
 - However, if you break it up according to camel case rules you get:
 - "Books or Magazines" and "Magazines or Books" and applying a bag of words matching to that yields a 100% match.

PART 2-c

MATCHING FUNCTIONS

Matching Functions

- For two references x and y , compute a “comparison” vector, which most commonly is the similarity of each component attribute
- Typically normalized, $0 \leq \text{sim}(x, y) \leq 1$
- Attributes most often strings, also numeric
- Distance metric: idempotent, non-negative, symmetric, triangle inequality
- Not all commonly used ER distance functions are metrics, e.g., non-linear elastic matching (NEM)
- From distance, can convert to similarity:
 - $S = 1 / d$, or if d is normalized, $s = 1 - d$

Taxonomy of Common Similarity functions

- Equality on a boolean predicate
- Edit distance
- Set similarity
- Alignment-based or Two-tiered
- Phonetic Similarity
- Translation-based
- Numeric distance between values
- Domain-specific

Edit Distance

- Levenstein

- $\text{dist}_{\text{Levenstein}}(s_1, s_2)$: min # single character insertions, deletions and substitutions required to convert one string into another
 - Dynamic programming algorithm $O(s_1 \times s_2)$ using $O(\min(s_1, s_2))$ space; additional optimization for long strings, etc.
 - Can add table to have different costs for different character pairs
- $\text{sim}_{\text{Levenstein}}(s_1, s_2) = 1 - \text{dist}_{\text{Levenstein}}(s_1, s_2) / \max(|s_1|, |s_2|)$

- Other variations

- Damerau-Levenshtein: adds transposition
- different costs for insertions/deletions vs. transpositions (useful for fixed length strings)
- Affine gaps: adds open gap, extend gap
- Smith-Waterman: gaps + mismatches at beginning/end lower cost; originally developed to find optimal alignment in biological sequences.
 - Also employs dynamic programming algorithm
 - Well-suited for names, with abbreviations, etc.; more computationally expensive than Levenstein

Set Similarity

$$sim_{overlap}(s_1, s_2) = \frac{|s_1 \cap s_2|}{\min(|s_1|, |s_2|)}$$

$$sim_{jaccard}(s_1, s_2) = \frac{|s_1 \cap s_2|}{|s_1 \cup s_2|}$$

$$sim_{dice}(s_1, s_2) = \frac{2 \cdot |s_1 \cap s_2|}{|s_1| + |s_2|}$$

- Computational complexity $|s_1| + |s_2|$ --- much better than edit distance

Vector Similarity

- Cosine similarity

$$sim_{\cos}(x, y) = \frac{\sum_i x_i y_i}{\sqrt{\sum_i x_i^2} \sqrt{\sum_i y_i^2}}$$

- Weighted: TF*IDF

- Example, qgrams:

- Compute all length q substrings in each string
- Compare using set/vector similarity

- Variations:

- Pad beginning and end with special characters, to weight matching at begin and end more heavily
- Positional q-grams: include position, and only count as the same if within a certain window

Combination & Alignment-based

- Approaches that combine edit distance and q-grams
 - Jaro: counts the number of agreeing characters c (*characters that are the same*) that are in common within half the length of the longer string, and the number of transpositions t in the sets of common strings
 - Jaro-Winkler: adds in; higher weights for matching at beginning of string, adjust for length, adds weight to common substitutions
- Alignment-based:
 - Monge-Elkan: Token-based, finds the best matching tokens, and then uses secondary string similarity to calculate the match of the tokens
 - SoftTFIDF: like Monge-Elkan, but uses TF-IDF weighting of the tokens

Phonetic Similarity Measures

- Soundex
 - The first letter of the name, followed by three digits which encode the remaining consonants, with similar consonants mapped to the same digit.
 - [from wikipedia] both "Robert" and "Rupert" return the same string "R163" while "Rubin" yields "R150"
- New York State Identification and Intelligence System (NYSIIS)
- Metaphone and Double Metaphone

Similarity functions on strings

String 1	String 2	Jaro	Winkler	Bigram	Trigram	PBigram	SkGram	LE-Dist	DLE-Dist	BagDist	Editex	ComZip	LCS2	LCS3	SW-Dist	SyllADist
shackleford	shackelford	0.970	0.982	0.750	0.692	0.750	0.821	0.818	0.909	1.000	0.806	0.684	0.818	0.818	0.691	0.645
dunningham	cunningham	0.896	0.896	0.667	0.522	0.667	0.722	0.800	0.800	0.800	0.769	0.722	0.842	0.842	0.716	0.678
nichleson	nichulson	0.926	0.956	0.700	0.636	0.700	0.696	0.778	0.778	0.889	0.769	0.647	0.778	0.778	0.622	0.698
jones	johnson	0.790	0.832	0.429	0.250	0.286	0.355	0.429	0.429	0.571	0.500	0.533	0.333	0.333	0.333	0.500
massey	massie	0.889	0.933	0.571	0.500	0.571	0.645	0.667	0.667	0.833	0.714	0.714	0.667	0.667	0.733	0.774
abroms	abrams	0.889	0.922	0.714	0.625	0.714	0.677	0.833	0.833	0.833	0.889	0.571	0.833	0.500	0.900	0.905
hardin	martinez	0.722	0.722	0.250	0.000	0.250	0.306	0.500	0.500	0.500	0.542	0.500	0.571	0.286	0.629	0.078
itman	smith	0.467	0.467	0.167	0.000	0.167	0.115	0.000	0.000	0.600	0.133	0.615	0.400	0.400	0.400	0.000
geraldine	geraldine	0.926	0.926	0.800	0.727	0.800	0.848	0.889	0.889	0.889	0.926	0.882	0.889	0.889	0.933	0.714
marhta	martha	0.944	0.961	0.571	0.500	0.571	0.710	0.667	0.833	1.000	0.765	0.571	0.500	0.500	0.500	0.615
michelle	michael	0.869	0.921	0.588	0.421	0.588	0.597	0.625	0.625	0.750	0.700	0.625	0.800	0.533	0.640	0.930
tanya	tonya	0.867	0.880	0.667	0.571	0.667	0.654	0.800	0.800	0.800	0.857	0.769	0.600	0.600	0.880	0.867
dwayne	duane	0.822	0.840	0.462	0.400	0.462	0.456	0.667	0.667	0.667	0.688	0.571	0.364	0.364	0.364	0.759
sean	susan	0.783	0.805	0.545	0.462	0.545	0.468	0.600	0.600	0.600	0.667	0.615	0.444	0.444	0.489	0.529
jon	john	0.917	0.933	0.667	0.545	0.667	0.595	0.750	0.750	0.750	0.818	0.667	0.571	0.571	0.571	0.889
brookhaven	brookhaven	0.933	0.947	0.909	0.750	0.909	0.843	0.900	0.900	0.900	1.000	0.778	0.900	0.700	0.800	0.769
higbee	higbee	0.889	0.922	0.714	0.625	0.714	0.677	0.833	0.833	0.833	0.800	0.571	0.833	0.500	0.667	0.312
cunningham	cunningham	0.967	0.980	0.857	0.783	0.857	0.866	0.900	0.900	0.900	0.885	0.722	0.947	0.947	0.821	0.949
campell	campbell	0.958	0.975	0.824	0.737	0.824	0.779	0.875	0.875	0.875	0.900	0.688	0.933	0.933	0.773	0.578
galloway	calloway	0.917	0.917	0.778	0.700	0.778	0.829	0.875	0.875	0.875	0.895	0.875	0.875	0.875	0.875	0.652
michele	michelle	0.958	0.975	0.941	0.842	0.941	0.857	0.875	0.875	0.875	1.000	0.750	0.800	0.800	0.800	1.000
jonathon	jonathan	0.917	0.950	0.778	0.700	0.778	0.780	0.875	0.875	0.875	0.913	0.750	0.750	0.750	0.925	0.929
dickson	dixon	0.790	0.832	0.571	0.500	0.571	0.387	0.571	0.571	0.571	0.684	0.533	0.667	0.333	0.333	0.837
gail	gayle	0.783	0.827	0.364	0.308	0.364	0.426	0.600	0.600	0.600	0.643	0.615	0.444	0.444	0.444	0.857
sydney	sydeny	0.944	0.961	0.571	0.500	0.571	0.710	0.667	0.833	1.000	0.667	0.571	0.500	0.500	0.500	0.486
tsetung	zedong	0.643	0.643	0.267	0.235	0.267	0.239	0.429	0.429	0.429	0.571	0.533	0.308	0.308	0.585	0.143

Fig. 5.4 Example string pairs and the similarities calculated on them. 'PBigram' stands for positional bigrams, 'SkGram' for skip-grams, 'LE-Dist' for the Levenshtein edit distance, 'DLE-Dist' for the Damerau-Levenshtein edit distance, 'ComZip' for compression based similarity using the ZLib compressor, 'LCS2' and 'LCS3' for the longest common sub-string comparison with the minimum length set to 2 and 3, respectively, 'SW-Dist' denotes the Smith-Waterman edit distance and 'SyllADist' the syllable alignment distance. In each row, the largest similarity value is shown in boldface and the lowest in italics.

Translation

- Can use translation tables to support
 - Abbreviations
 - Nicknames
 - Other synonyms
- There are techniques for using and learning translation tables
 - Most likely transformation
 - Computing all transformations
 - Learning from web
 - See [Arasu et al., SIGMOD08, ICDE08]

Numeric Similarity

- Often: financial data, salary, taxes, transactions, etc.
- Two approaches:
 - Maximum absolute difference
 - Maximum relative difference

Dates

- Date:
 - Calculate difference in days, and use maximum absolute difference (e.g., within 30 days of each other)
 - If month and day are swapped, use some default similarity value
 - If only difference is month, but day and year match, then use some default similarity value
- Age: if given by date, calculate # of days since birth
- Time can be handled similar to dates

Other Common Types

- Geographic location
- Complex data: xml, image, audio, video
- Records: sum individual components, or weight them

Summary of Matching Function

- Equality on a boolean predicate
- Edit distance
- Set similarity
- Alignment-based or Two-tiered
- Phonetic Similarity
- Translation-based
- Numeric distance between values
- Domain-specific
- Useful packages:
 - SecondString, <http://secondstring.sourceforge.net/>
 - Simmetrics: <http://sourceforge.net/projects/simmetrics/>
 - LingPipe, <http://alias-i.com/lingpipe/index.html>

Matching Algorithms

- Pairwise Approaches
 - Match decisions made independently
- Cluster-based Approaches
 - Match decisions depend other decisions

PART 2-d

PAIRWISE ER

Simple Threshold Method

- Most basic (and surprisingly commonly used by novices):
- Compute similarity and choose threshold
 - Predict:
 - Everything above threshold match
 - Everything below threshold non-match
 - Variation:
 - Choose two thresholds, t_u , t_l
 - Above t_u , match
 - Below t_l , non-match
 - Between: potential match, region saved for clerical review
- Issues: don't take into account importance of different attributes, or importance of different attribute values

Rule-based

RecID	GivenName	Surname	StrNum	StrName	Suburb	BDay	BMonth	BYear	SimSum
-------	-----------	---------	--------	---------	--------	------	--------	-------	--------

$$(\mathcal{s}(\text{GivenName})[r_i, r_j] \geq 0.9) \wedge (\mathcal{s}(\text{Surname})[r_i, r_j] = 1.0) \\ \wedge (\mathcal{s}(\text{BMonth})[r_i, r_j] = 1.0) \wedge (\mathcal{s}(\text{BYear})[r_i, r_j] = 1.0) \Rightarrow [r_i, r_j] \rightarrow \text{Match}$$

$$(\mathcal{s}(\text{GivenName})[r_i, r_j] \geq 0.7) \wedge (\mathcal{s}(\text{Surname})[r_i, r_j] \geq 0.8) \\ \wedge (\mathcal{s}(\text{BDay})[r_i, r_j] = 1.0) \wedge \mathcal{s}(\text{BMonth})[r_i, r_j] = 1.0) \\ \wedge (\mathcal{s}(\text{BYear})[r_i, r_j] = 1.0) \Rightarrow [r_i, r_j] \rightarrow \text{Match}$$

$$(\mathcal{s}(\text{GivenName})[r_i, r_j] \geq 0.7) \wedge (\mathcal{s}(\text{Surname})[r_i, r_j] \geq 0.8) \\ \wedge (\mathcal{s}(\text{StrName})[r_i, r_j] \geq 0.8) \wedge (\mathcal{s}(\text{Suburb})[r_i, r_j] \geq 0.8) \Rightarrow [r_i, r_j] \rightarrow \text{Match}$$

$$(\mathcal{s}(\text{GivenName})[r_i, r_j] \geq 0.7) \wedge (\mathcal{s}(\text{Surname})[r_i, r_j] \geq 0.8) \\ \wedge (\mathcal{s}(\text{BDay})[r_i, r_j] \leq 0.5) \wedge (\mathcal{s}(\text{BMonth})[r_i, r_j] \leq 0.5) \\ \wedge (\mathcal{s}(\text{BYear})[r_i, r_j] \leq 0.5) \Rightarrow [r_i, r_j] \rightarrow \text{Non-Match}$$

$$(\mathcal{s}(\text{GivenName})[r_i, r_j] \geq 0.7) \wedge (\mathcal{s}(\text{Surname})[r_i, r_j] \geq 0.8) \\ \wedge (\mathcal{s}(\text{StrName})[r_i, r_j] \leq 0.6) \wedge (\mathcal{s}(\text{Suburb})[r_i, r_j] \leq 0.6) \Rightarrow [r_i, r_j] \rightarrow \text{Non-Match}$$

- Rule set may be
 - Manually constructed
 - Learned based on training data often using sequential covering algorithm
- Rule set evaluated based on accuracy, coverage; generally prefer a smaller rule set

Probabilistic Record Linkage

- Ideas introduced in [Newcombe, Science59], formalized in [Fellegi& Sunter, JASS69]
- William Winkler, Census Bureau popularized and extended [Winkler99, Herzog et al, 07]
- Matching records between two files A and B
- Intuition: two records that match on last name “Machanavajjhala” are more likely to match than two records that match on “Smith”

F&S Model

- r is record pair, γ is comparison vector, M matches, U non-matches
- In the original work, γ is binary, 0/1, match/not match

- Decision rule
$$R = \frac{P(\gamma \mid r \in M)}{P(\gamma \mid r \in U)}$$

$$R \geq t_u \Rightarrow r \rightarrow \text{Match}$$

$$t_l < R < t_u \Rightarrow r \rightarrow \text{Potential Match}$$

$$R \leq t_u \Rightarrow r \rightarrow \text{Non - Match}$$

- Thresholds t_u and t_l determined by apriori bounds on false matches and false non-matches

Computing Probabilities

- Typically make an independence assumption
- Agreement weight w_i is calculated for each attribute i based on m and u probabilities:
 - $m_i = P(a_i=b_i, a \in A, b \in B \mid r \in M)$
 - $u_i = P(a_i=b_i, a \in A, b \in B \mid r \in N)$
- Can also estimate probabilities using EM [Winkler99,Hertzog et al., 07]

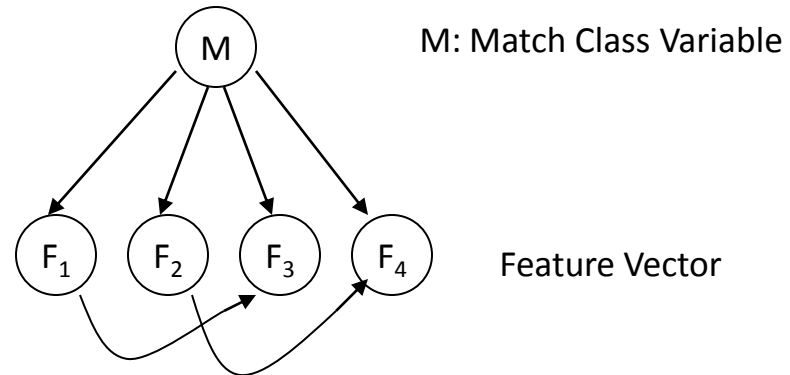
ML Pairwise Approaches

- Supervised machine learning algorithms
 - Decision trees
 - [Cochinwala et al, IS01]
 - Support vector machines
 - [Bilenko & Mooney, KDD03]; [Christen, KDD08]
 - Ensembles of classifiers
 - [Chen et al., SIGMOD09]
 - Conditional Random Fields (CRF)
 - [Wellner & McCallum, NIPS04]
- Issues:
 - Training set generation
 - Imbalanced classes – many more negatives than positives (even after blocking)
 - Misclassification cost

Generative Models for Linkage [Ravikumar & Cohen, UAI04]

Main ideas:

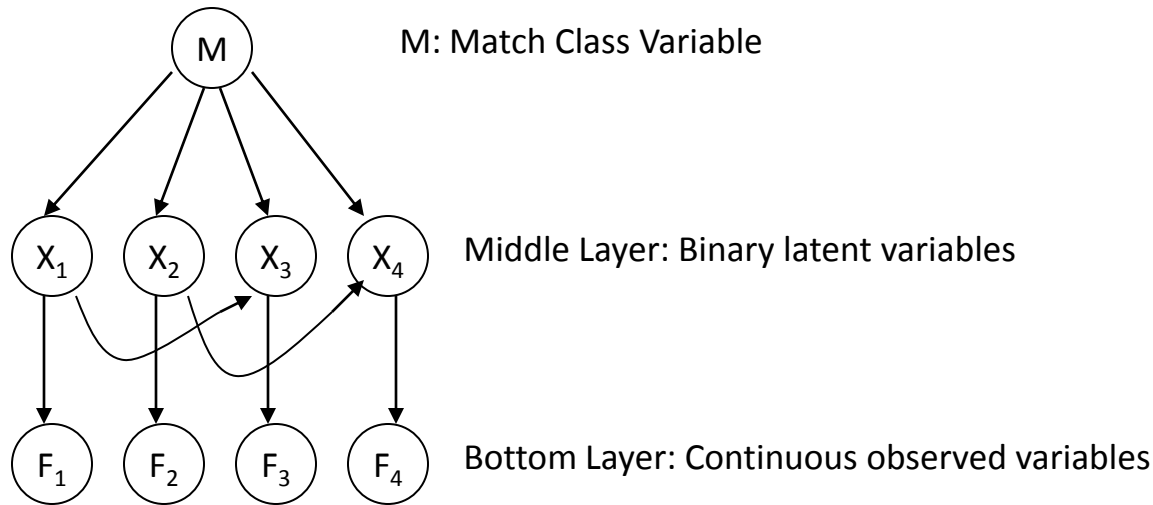
- Construct a *generative model* of the record-pair feature vector.
- Learn *parameters* and adjust *structure* of model using data with *unknown* match status.
- Avoid overfitting with “semantic” constraints, monotonicity constraints, and “bootstrapping”



Result:

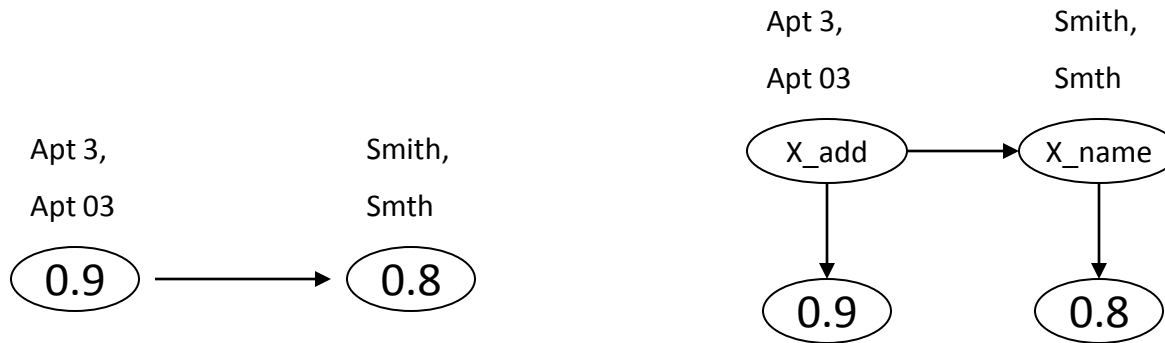
- Competitive with *supervised* learning methods.

A Hierarchical Model



- Latent match-class for each field given its features
- Global match-class depends solely on these latent match-class variables
- Dependencies only in the middle layer of binary latent variables.
- Learn parameters and dependencies with structural EM

Intuition

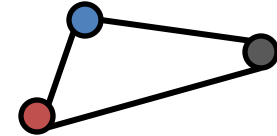


- Last-name pair depends on Address pair
 - But *error* in last-name pair does not depend on *error* in address pair
 - Intuition of *dependency*: Match Status of last-name pair depends on match-status of address pair.
 - So, instead of modelling the dependency between continuous values, we need model only the dependency between the binary latent match-class variables.

Pairwise + Transitive Closure

- Often pairwise ER algorithm output “inconsistent” results

– $(x, y) \in M_{\text{pred}}, (y, z) \in M_{\text{pred}}, \text{ but } (x, z) \notin M_{\text{pred}}$

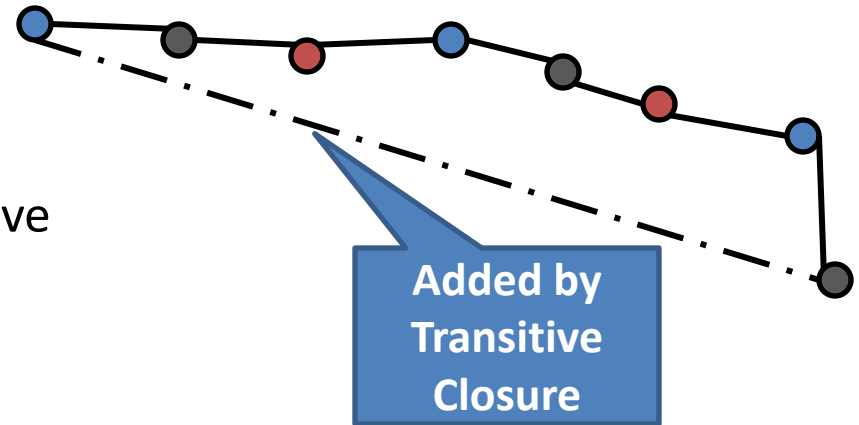


- Idea: Correct this by adding additional matches using transitive closure

- In certain cases, this is a bad idea.

– Graphs resulting from pairwise ER have diameter > 20

[Das Sarma et al 2012 CIKM]



- Clustering solutions deal with this problem directly by reasoning about record jointly.

Summary of Pairwise ER

- Simple thresholds
- Rule-based Methods
- Supervised Machine Learning
- Graphical Models
- Pairwise methods may not be sufficient for deduplication
 - Transitive constraints not satisfied

PART 2-e

CLUSTERING-BASED ER

Clustering-based ER

- Resolution decisions are not made independently for each pair of records
- Based on variety of clustering algorithms, but
 - Number of clusters unknown apriori
 - Many, many small (possibly singleton) clusters
- Often take a pair-wise similarity graph as input
- May require the construction of a *cluster representative* or *canonical entity*

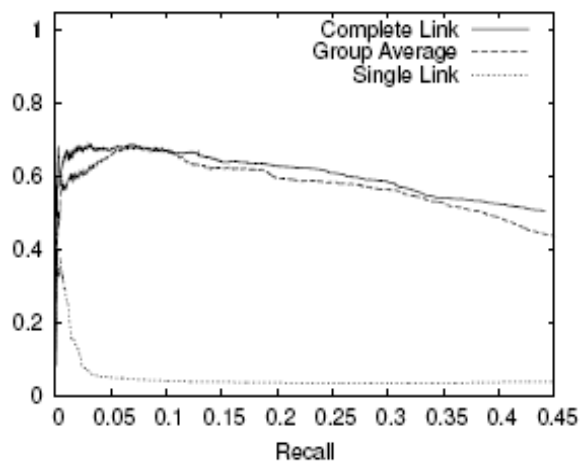
Clustering Methods for ER

- **Hierarchical Clustering**
- Nearest Neighbor based methods
- Correlation Clustering

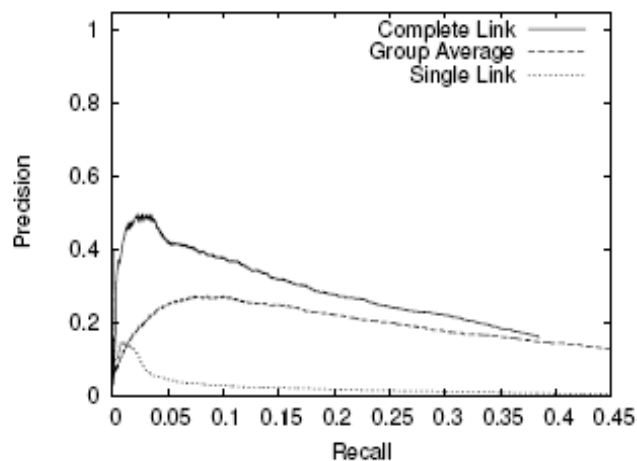
Hierarchical Clustering

- Bottom-Up Greedy Agglomerative Clustering
 - Starts with each reference in a separate cluster
 - then repeatedly joins the closest pair of clusters
- Cluster distances
 - Single-Link
 - Nearest Neighbor: their closest members.
 - Complete-Link
 - Furthest Neighbor: their furthest members.
 - Centroid:
 - Distance between centroids
 - Group Average:
 - average of all cross-cluster pairs.

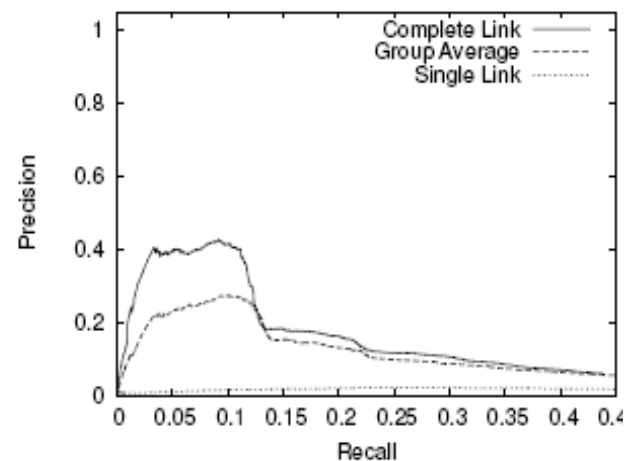
Empirical results on data partitioning



Digital cameras



Camcorder



Luggage

- Setup: Online comparison shopping,
 - Fields: name, model, description, price
 - Learner: Online perceptron learner
- Complete-link clustering \gg single-link clustering(transitive closure)

Clustering Methods for ER

- Hierarchical Clustering
- **Nearest Neighbor based methods**
- Correlation Clustering

Shortcoming of Hierarchical Clustering

- Hierarchical clustering must choose a threshold to determine clusters.

- 1 and 2 are duplicates
7 and 8 are not
But, $\text{Sim}(1, 2) \approx \text{Sim}(7, 8)$

- How to choose?

ID	ArtistName	TrackName
1*	The Doors	LA Woman
2*	Doors	LA Woman
3*	The Beatles	A Little Help from My Friends
4*	Beatles, The	With A Little Help From My Friend
5*	Shania Twain	Im Holdin on to Love
6*	Twian, Shania	I'm Holding On To Love
7	4 th Elemynt	Ears/Eyes
8	4 th Elemynt	Ears/Eyes - Part II
9	4th Elemynt	Ears/Eyes - Part III
10	4 th Elemynt	Ears/Eyes - Part IV
11	Aaliyah	Are You Ready
12	AC DC	Are You Ready
13	Bob Dylan	Are You Ready
14	Creed	Are You Ready

Table 1: Examples from a media database.

[Chaudhuri et al ICDE 05]

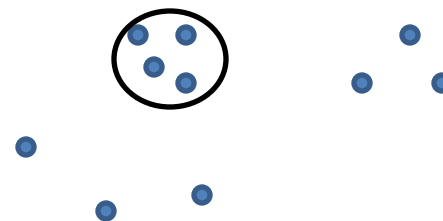
Leveraging Local Structure

Compact Set Property:

duplicate tuples are closer to each other than to other tuples

For all v, v' in S , and v'' not in S ,

$$d(v, v') < d(v, v'')$$



Sparse Neighborhood Property:

the “local neighborhood” of duplicate tuples is sparse

Let $nn(v)$ be the nearest neighbor to v

Let $ng(v)$ be the number of nodes within distance $p \times nn(v)$

A set S has sparse neighborhood (with param c) if

$$\max (\{ ng(v) : v \text{ in } S \}) < c$$

Dense/Sparse Partitioning Problem

$DE(k)$: Identify the smallest number of compact SN sets S_1, S_2, \dots, S_m , such that $|S_i| \leq k$.

Algorithm: [Chaudhuri et al ICDE 05]

Step 1: Determine the best k nearest neighbors of every record.

Step 2: Partition the records into compact SN sets

- A set of p records are a compact SN set if their p -nearest neighbors are the same.

Clustering Methods for ER

- Hierarchical Clustering
- Nearest Neighbor based methods
- **Correlation Clustering**

Integer Linear Programming view of ER

- $r_{xy} \in \{0,1\}$, $r_{xy} = 1$ if records x and y are in the same cluster.
- $w_{xy}^+ \in [0,1]$, cost of clustering x and y together
- $w_{xy}^- \in [0,1]$, cost of placing x and y in different clusters

$$\text{minimize } \sum r_{xy} w_{xy}^+ + (1 - r_{xy}) w_{xy}^-$$

$$\text{s.t. } \forall x, y, z \in R,$$

$$r_{xy} + r_{xz} + r_{yz} \neq 2$$



Transitive
closure

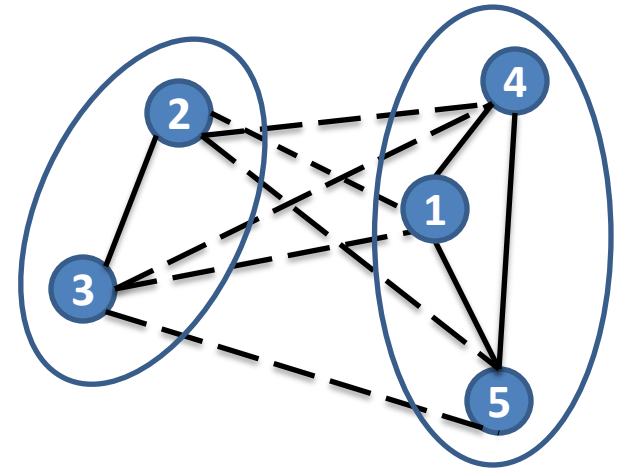
Correlation Clustering

$$\begin{aligned} & \text{minimize } \sum r_{xy} w_{xy}^+ + (1 - r_{xy}) w_{xy}^- \\ & \text{s.t. } \forall x, y, z \in R, \\ & \quad r_{xy} + r_{xz} + r_{yz} \neq 2 \end{aligned}$$

- Cluster mentions such that total cost is minimized

Solid edges contribute w_{xy}^+ to the objective

Dashed edges contribute w_{xy}^- to the objective



- Cost based on pairwise similarities

$$\{p_{xy} \mid \forall (x, y) \in R \times R\}$$

- Additive: $w_{xy}^+ = p_{xy}$ and $w_{xy}^- = (1 - p_{xy})$
- Logarithmic: $w_{xy}^+ = \log(p_{xy})$ and $w_{xy}^- = \log(1 - p_{xy})$

Correlation Clustering

- Solving the ILP is NP-hard [Ailon et al 2008 JACM]
- A number of heuristics [Elsner et al 2009 ILP-NLP]
 - Greedy BEST/FIRST/VOTE algorithms
 - Greedy PIVOT algorithm (5-approximation)
 - Local Search

Greedy Algorithms

Step 1: Permute the nodes according a random π

Step 2: Assign record x to the cluster that maximizes *Quality*

Start a new cluster if *Quality* < 0

Quality:

- BEST: Cluster containing the closest match $\max_{y \in C} w_{xy}^+$
 - [Ng et al 2002 ACL]
- FIRST: Cluster contains the most recent vertex y with $w_{xy}^+ > 0$
 - [Soon et al 2001 CL]
- VOTE: Assign to cluster that minimizes objective function.
 - [Elsner et al 08 ACL]

Practical Note:

- Run the algorithm for many random permutations , and pick the clustering with best objective value (better than average run)

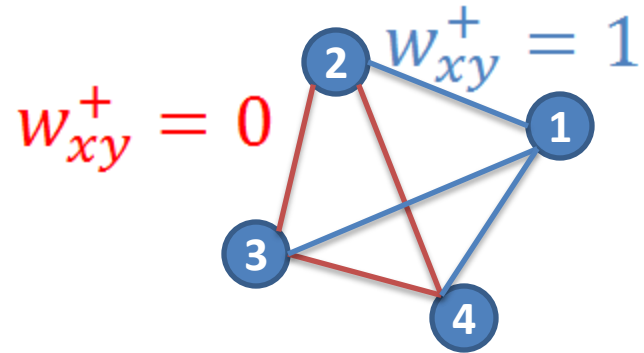
Greedy with approximation guarantees

PIVOT Algorithm

[Ailon et al 2008 JACM]

- Pick a random (*pivot*) record p .
- New cluster = $\{x \mid w_{px}^+ > 0\}$

- $\pi = \{1,2,3,4\}$ $C = \{\{1,2,3,4\}\}$
- $\pi = \{2,4,1,3\}$ $C = \{\{1,2\}, \{4\}, \{3\}\}$
- $\pi = \{3,2,4,1\}$ $C = \{\{1,3\}, \{2\}, \{4\}\}$



When weights are 0/1, $E(\text{cost}(\text{greedy})) < \mathbf{3} \text{ OPT}$

For $w_{xy}^+ + w_{xy}^- = 1$, $E(\text{cost}(\text{greedy})) < \mathbf{5} \text{ OPT}$

Local Search

BOEM Algorithm [Gionis et al 2007 TKDD]

- Start with an initial clustering (e.g. output of greedy)
- Remove one random element from a cluster
- Make the Best One Element Move (BOEM)
 - Move it to another cluster or Create a new cluster.

Empirical Comparison [Elsner et al 2009 ILP-NLP]

Logarithmic Weights			
VOTE/BOEM	93.80	33	41
SA	93.56	31	36
PIVOT/BOEM	93.63	32	39
BEST/BOEM	93.57	31	38
FIRST/BOEM	93.65	30	36
VOTE	93.41	29	35
BOEM	93.51	30	35
PIVOT	90.85	17	27
BEST	87.11	20	29
FIRST	40.97	11	8

1-1 score: Compute a maximum matching between proposed clustering and ground truth. Report overlap between the two.

Pairwise F measure

Fraction of pairs of points for which clustering agrees with ground truth.

PART 2-f

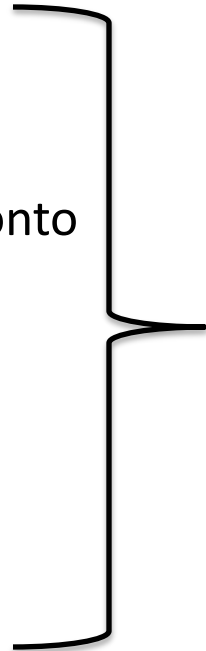
CANONICALIZATION

Canonicalization

- Merge information from duplicate mentions to construct a cluster representative with *maximal* information

- Starbucks,
123 Queen St, Toronto
Ph: *null*

- Starbucks,
null, Toronto
Ph: 333-4444



**Starbucks,
123 Queen St, Toronto
Ph: 333-4444**

Canonicalization

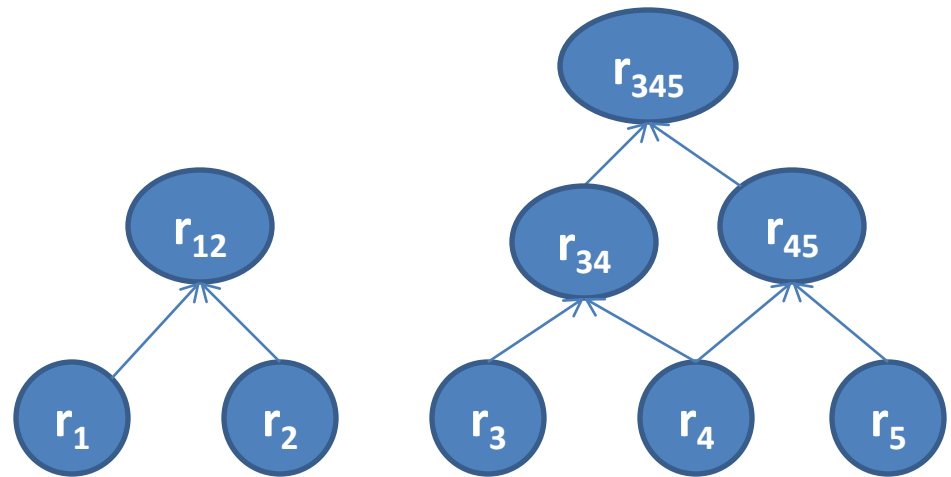
- Critically important in Web portals where users must be shown a consolidated view of the duplicate cluster.
 - Each mention only contains a subset of the attributes.
 - Mentions contain variations (of names, addresses).
 - Some of the mentions can have wrong values.

Canonicalization

- For names: typically longest names are used.
- For set values attributes: UNION is used.
- For strings, [Culotta et al KDD07] learn an edit distance for finding the most representative “centroid”.
- Can use “majority rule” to fix errors
(if 4 out of 5 duplicates say a business is closed, then business is closed).
 - This may not always work due to copying [Dong et al VLDB09], or when underlying data changes [Pal et al WWW11]

Canonicalization for Efficiency

- Stanford Entity Resolution Framework [Benjelloun VLDBJ09]
 - Consider a blackbox match and merge function
 - Match is a pairwise boolean operator
 - Merge: construct canonical version of a matching pair
- Problem: Minimize number of invocations to Match and Merge



G-Swoosh

- Pick a random record x .
- Find all matches to x (y_1, y_2, \dots).
- Compute merged records $\langle x, y_1 \rangle, \langle x, y_2 \rangle, \dots$
- Add merged records to the set of records.
- Remove dominated records.
 - A record x is dominated by a merge $\langle y, z \rangle$ if $\langle y, z \rangle$ contains more information.

ICAR properties

- Idempotence: $\langle x, x \rangle = x$
- Commutativity: $x \approx y$ iff $y \approx x$, and $\langle x, y \rangle = \langle y, x \rangle$
- Associativity: $\langle x, \langle y, z \rangle \rangle = \langle \langle x, y \rangle, z \rangle$
- Representativity: if $z = \langle x, y \rangle$. If $w \approx x$, then $w \approx z$.
- Merge functions like UNION satisfy ICAR properties.
- If ICAR properties are satisfied and $x \approx y$, then $\langle x, y \rangle$ always dominates x and y .

R-swoosh

- Use ICAR properties.
- Derivation:
 - Merge: add $\langle x, y \rangle$ to record set if $x \approx y$
 - Purge: remove all dominated records (x, y are dominated by $\langle x, y \rangle$)
- Any maximal derivation sequence computes the correct ER answer.
 - Number of match merge steps is $<$ number of records.

Summary of Clustering & Canonicalization

- Hierarchical techniques are directly relate ER to clustering
- ER differs from clustering since local structures may be different.
- Correlation clustering is a popular (and rigorous) method for ER.
- Canonicalization can also help speed up ER in certain cases.