Collective Entity Resolution in Relational Data (contd)

CompSci 590.03
Instructor: Ashwin Machanavajjhala

Slides adapted from [Singla et al ICDM06], [Rastogi et al VLDB '11]

Duke UNIVERSITY

This class

- Collective Entity Resolution using Markov Logic Networks
- Scaling Collective Entity Resolution

Markov Logic

[Richardson & Domingos, 06]

- A logical KB is a set of **hard constraints** on the set of possible worlds
- Let us make them **soft constraints**
- When a world violates a formula, it becomes less probable but not impossible
- Give each formula a **weight**
  - Higher weight ⇒ Stronger constraint

$$P(world) \propto \exp\left(\sum weights \;\; of \;\; formulas \;\; it \;\; satisfies\right)$$

Markov Logic

- A **Markov Logic Network (MLN)** is a set of pairs **(F, w)** where
  - **F** is a formula in first-order logic
  - **w** is a real number

true groundings of *ith* clause

$$P(X) = \frac{1}{Z} \exp\left(\sum_{i \in F} w_i n_i(x) \right)$$

Normalization Constant

Iterate over all first-order MLN formulas

Inference

- Given weights, computing the probability of a world can be computed using the following techniques

- MCMC

- Gibbs Sampling

- WalkSAT
  - Find an assignment of truth values to variables that maximizes the total weight of the satisfied formulae (or clauses)

Duke UNIVERSITY

Problem Formulation

- **Given**
  - A database of records representing entities in the real world e.g. citations
  - A set of fields e.g. author, title, venue
  - Each record represented as a set of typed predicates e.g. *HasAuthor(citation,author), HasVenue(citation,venue)*
- **Goal**
  - To determine which of the records/fields refer to the same underlying entity

Duke UNIVERSITY

Example: Bibliography Database

Citation	Title	Author	Venue
C1	Entity Resolution	J. Cox	ICDM 06
C2	Entity Resolution and Logic	Cox J.	Sixth ICDM
C3	Learning Boolean Formulas	Jacob C.	ICDM 06
C4	Learning of Boolean Formulas	Jacob Coxe	Sixth ICDM

Problem Formulation

- Entities in the real world represented by one or more strings appearing in the DB e.g. ***"J. Cox", "Cox J."***
- String constant for each record e.g. ***"C1", "C2"***
- Goal: for each pair of string constants $<x_1, x_2>$ of the same type, is $x_1 = x_2$?

Handling Equality

- Introduce ***Equals(x,y)*** for $x = y$
- Introduce the axioms of equality
  - Reflexivity: $x = x$
  - Symmetry: $x = y \Rightarrow y = x$
  - Transitivity: $x = y \wedge y = z \Rightarrow z = x$

Predicate Equivalence

$$R(x_1,y_1) \wedge x_1 = x_2 \wedge y_1 = y_2 \Rightarrow R(x_2,y_2)$$

- If (x1,x2) and (y1,y2) are the same, then if x1,y1 are related, then x2,y2 are also related.
  - Hard constraints like the equality axioms.
  - Infinite weight

Reverse Predicate Equivalence

- Same relation with the same entity gives evidence about two entities being same

  $R(x_1,y_1) \wedge R(x_2,y_2) \wedge x_1 = x_2 \Rightarrow y_2 = y_2$

- Not true logically, but gives useful information
  - Soft constraint with weights
  - Weight determines strength of the constraint
- Example

  HasAuthor(C1, J. Cox) ∧ HasAuthor(C2, Cox J.) ∧ C1 = C2 ⇒ (J. Cox = Cox J.)

Model for Entity Resolution

- Model is in the form of an MLN
  - Each formula has a weight (which can be specified by humans or learnt from training data)
- Evidence predicates are relations which hold according to the DB
- Goal: Query predicate is ***Equality***
  - Compute likelihood of the equality predicated being true
  - Equality predicates are related to evidence via predicate and reverse predicate equivalence.

Duke UNIVERSITY

Enriching the model

- Predicate and reverse predicate equivalence only fire when
  - Either, x1, x2 are constants and are identical
  - Or, Equality(x1, x2) is satisfied.

  - Need to be able to encode similarity functions

- Can add other constraints.

Encoding Similarity Functions

- Each field is a string composed of tokens
- Introduce ***HasWord(field, word)***
- Use reverse predicate equivalence

$$HasWord(f_1,w_1) \wedge HasWord(f_2,w_2) \wedge w_1 = w_2 \Rightarrow f_1 = f_2$$

- Example

HasWord(J. Cox, Cox) ∧ HasWord(Cox J., Cox) ∧ (Cox = Cox) ⇒ (J. Cox = Cox J.)

Duke UNIVERSITY

Encoding Similarity

$$HasWord(f_1,w_1) \wedge HasWord(f_2,w_2) \wedge w_1 = w_2 \Rightarrow f_1 = f_2$$

- If these rules have the same weight for all rules,

  $$\Pr[f1 = f2 \mid n \text{ words in common}] = e^{wn} / (e^{wn} + 1)$$

- Different weight for each word
  - Similar to a learnable similarity measure of [Bilenko & Mooney 2003]

Two-level Similarity

- Individual words as units: Can't deal with spelling mistakes
- Break each word into ngrams: Introduce ***HasEngram(word, ngram)***
- Use reverse predicate equivalence for word comparisons
- Gives a two level similarity measure.

Fellegi-Sunter Model

- Uses Naïve Bayes for match decisions with field comparisons used as predictors
- Simplest Version: Field similarities measured by presence/ absence of words in common

$$HasWord(f_1, w_1) \wedge HasWord(f_2, w_2) \wedge HasField(r_1, f_1) \wedge HasField(r_2, f_2) \wedge w_1 = w_2 \Rightarrow r_1 = r_2$$

- Example

$$HasWord(J.\ Cox, Cox) \wedge HasWord(Cox\ J., Cox) \wedge HasAuthor(C1, J.\ Cox) \wedge HasAuthor(C2, Cox\ J.) \wedge (Cox = Cox) \Rightarrow (C1 = C2)$$

Duke UNIVERSITY

Relational Models

- Fellegi-Sunter + transitivity [McCallum & Wellner 2005]

  $(f_1 = f_2) \wedge (f_2 = f_3) \Rightarrow (f_3 = f_1)$

- Fellegi-Sunter + reverse predicate equivalence for records/fields [Singla & Domingos 2005]

  $HasField(r_1, f_1) \wedge HasField(r_2, f_2) \wedge f_1 = f_2 \Rightarrow r_1 = r_2$

  $HasAuthor(C1, J.\ Cox) \wedge HasAuthor(C2, Cox\ J.) \wedge (J.\ Cox = Cox\ J.) \Rightarrow C1 = C2$

Relational Models

- Co-authorship relation for entity resolution [Bhattacharya & Getoor, TKDD'07]

$$HasAuthor(c,a_1) \wedge HasAuthor(c,a_2) \Rightarrow Coauthor(a_1,a_2)$$

$$Coauthor(a_1, a_2) \wedge Coauthor(a_3, a_4) \wedge a_1 = a_3 \Rightarrow a_2 = a_4$$

This class

- Collective Entity Resolution using Markov Logic Networks
- Scaling Collective Entity Resolution

Scalability

ER with Markov Logic Networks:

(+) High ***accuracy***

(-) Often scale only to a few 1000 entities

How can we scale

Collective Entity Matching

to millions of entities?

Duke UNIVERSITY

Approach 1

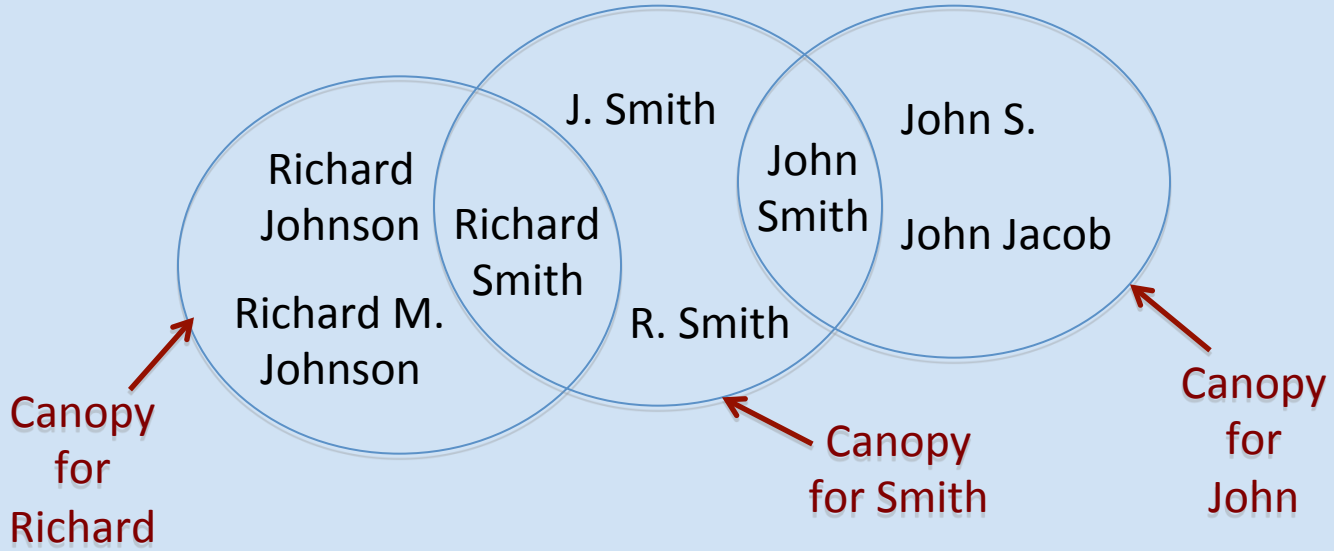
- Generates overlapping canopies (e.g., Canopy clustering)
- Run collective matcher on each canopy

Id	Author-1	Author-2	Paper
A_1	John Smith	Richard Johnson	Indices and Views
A_2	J Smith	R Johnson	SQL Queries
A_3	Dr. Smyth	R Johnson	Indices and Views

Efficiency: Use Canopies [McCallum et. al.]

Reduces # of candidate pairs from:

$O(|Records|^2)$ to $|Candidate\ pairs|$



Blocking is not sufficient for scaling

Inference on MLNs : $\Omega(|\text{Candidate pairs}|^2)$

Example (using canopy clustering):

- |Records| = 1000, |Candidate pairs| = 15,000,
  - Time ~ 5 minutes
- |Records| = 50,000, |Candidate pairs| = 10 million
  - Time required = 2,500 hours ~ 3 months

Duke UNIVERSITY

Distribute

Run collective entity-matching over canopies separately

Example for Collective methods[SD06]

- |References| = 1000, |Candidates| = 15,000,
  - Time = 5 minutes
- One canopy: |References| = 100, |Candidates| ~ 1000,
  - Time ~ 10 Seconds
- |References| = 50,000, # of canopies ~ 13k
  - Time ~ 20 hours << 3 months!

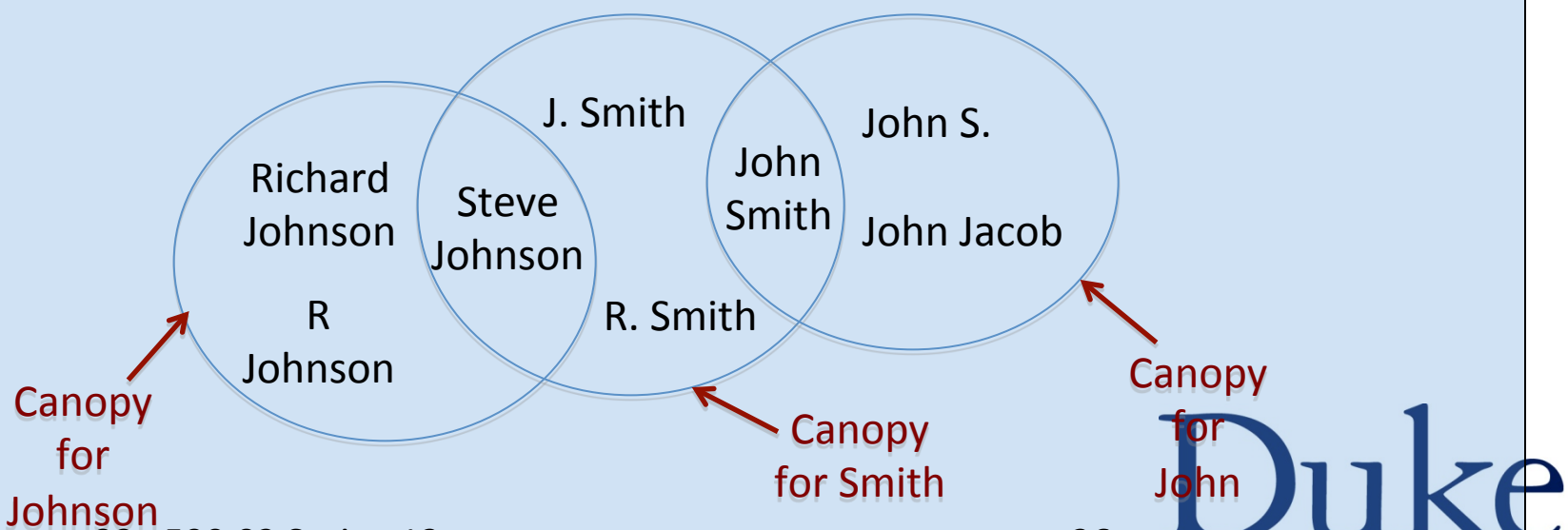
Duke UNIVERSITY

Problem: Correlations across canopies will be lost

$$\text{CoAuthor}(A_1, B_1) \wedge \text{CoAuthor}(A_2, B_2) \wedge \text{match}(B_1, B_2) \rightarrow \text{match}(A_1, A_2)$$

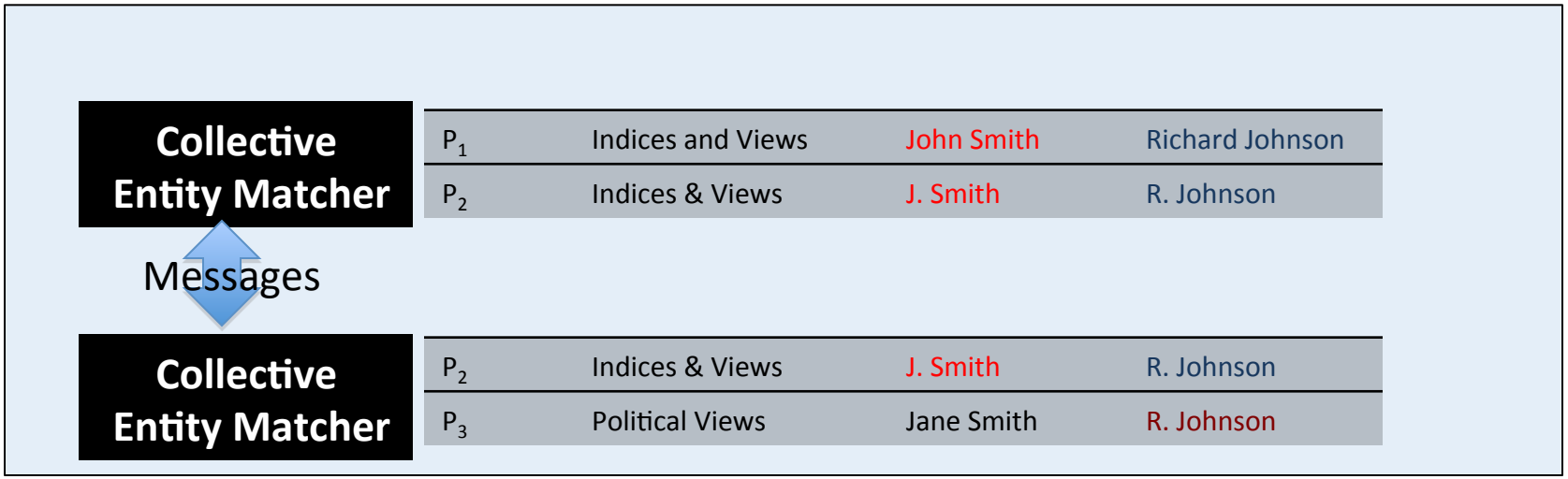
Example: CoAuthor rule grounds to the correlation

match(Richard Johnson, R Johnson) => match(J. Smith, John Smith)



Approach 1

- Generates overlapping canopies (e.g., Canopy clustering)
- Run collective matcher on each canopy
- Pass messages between canopies and iterate.



Message Passing

Simple Message Passing (SMP)

1. Run entity matcher M locally in each canopy
2. If M finds a match(r_1,r_2) in some canopy, pass it as evidence to all canopies
3. Rerun M within each canopy using new evidence
4. Repeat until no new matches found in each canopy

Runtime (per iteration): $O(k^2 f(k) c)$

- k : maximum size of a canopy
- f(k): Time taken by ER on canopy of size k
- c : number of canopies

Formal Properties

for a well behaved ER method …

Convergence: No. of steps ≤ no. of matches

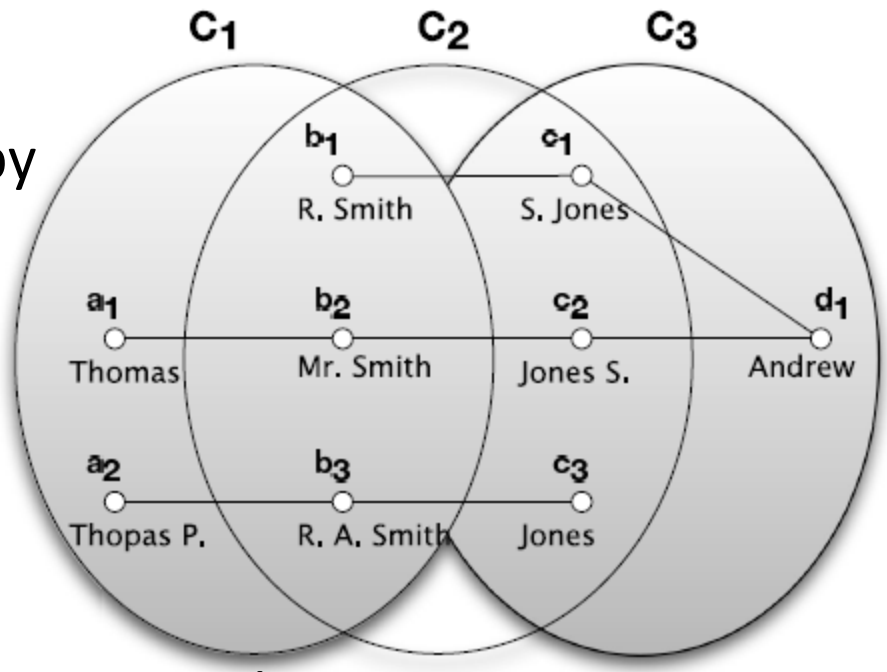
Consistency: Output independent of the canopy order

Soundness: Each output match is actually a true match

~~***Completeness***. Each true match is also a output match~~

Completeness

Papers 2 and 3 match only if a canopy knows that
- match(a1,a2)
- match(b2,b3)
- match(c2,c3)



Simple message passing will not find any matches
- thus, no messages are passed, no progress

Solution: Maximal message passing
- Send a message if there is a potential for match

Summary

- Markov Logic Networks provide a general abstraction for formulating entity resolution tasks
  - Pro: Can encode any type of constraints into the problem
  - Pro: High accuracy can be achieved
  - Con: Not scalable beyond problems with a few thousand records
- Collective ER can be scaled up using message passing

Duke UNIVERSITY