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]
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 \( \Rightarrow \) Stronger constraint

\[
P(\text{world}) \propto \exp\left( \sum \text{weights of formulas it satisfies} \right)
\]
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

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

- \(Z\) is the normalization constant
- \(n_i(x)\) is the number of true groundings of the \(i\)th clause
- Iterate over all first-order MLN formulas

Lecture 22 : 590.02 Spring 13
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)
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. 
    \(\text{HasAuthor(citation,author)}, \text{HasVenue(citation,venue)}\)

• **Goal**
  
  – To determine which of the records/fields refer to the same underlying entity
Example: Bibliography Database

<table>
<thead>
<tr>
<th>Citation</th>
<th>Title</th>
<th>Author</th>
<th>Venue</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>Entity Resolution</td>
<td>J. Cox</td>
<td>ICDM 06</td>
</tr>
<tr>
<td>C2</td>
<td>Entity Resolution and Logic</td>
<td>Cox J.</td>
<td>Sixth ICDM</td>
</tr>
<tr>
<td>C3</td>
<td>Learning Boolean Formulas</td>
<td>Jacob C.</td>
<td>ICDM 06</td>
</tr>
<tr>
<td>C4</td>
<td>Learning of Boolean Formulas</td>
<td>Jacob Coxe</td>
<td>Sixth ICDM</td>
</tr>
</tbody>
</table>
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: $x = y \land y = z \Rightarrow z = x$
Predicate Equivalence

\[ R(x_1, y_1) \land x_1 = x_2 \land y_1 = y_2 \implies R(x_2, y_2) \]

- If \((x_1, x_2)\) and \((y_1, y_2)\) are the same, then if \(x_1, y_1\) are related, then \(x_2, y_2\) 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

\[ R(x_1, y_1) \land R(x_2, y_2) \land x_1 = x_2 \Rightarrow y_2 = y_2 \]

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

- Example

\[ \text{HasAuthor}(C_1, J. Cox) \land \text{HasAuthor}(C_2, Cox J.) \land C_1 = C_2 \Rightarrow (J. Cox = Cox J.) \]
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.
Enriching the model

• Predicate and reverse predicate equivalence only fire when
  – Either, \( x_1, x_2 \) are constants and are identical
  – Or, \( \text{Equality}(x_1, x_2) \) 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 \textit{HasWord(field, word)}
- Use reverse predicate equivalence

\[
\text{HasWord}(f_1, w_1) \land \text{HasWord}(f_2, w_2) \land w_1 = w_2 \implies f_1 = f_2
\]

- Example

\[
\text{HasWord}(J.\ Cox, Cox) \land \text{HasWord} (Cox J., Cox) \land (Cox = Cox) \implies (J.\ Cox = Cox J.)
\]
Encoding Similarity

\[ \text{HasWord}(f_1, w_1) \land \text{HasWord}(f_2, w_2) \land w_1 = w_2 \Rightarrow f_1 = f_2 \]

- If these rules have the same weight for all rules,
  \[ \Pr[f_1 = f_2 \mid n \text{ words in common}] = \frac{e^{wn}}{e^{wn} + 1} \]

- 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 \textit{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

\[
\text{HasWord}(f_1, w_1) \land \text{HasWord}(f_2, w_2) \land \text{HasField}(r_1, f_1) \land \text{HasField}(r_2, f_2) \land \\
w_1 = w_2 \iff r_1 = r_2
\]

- Example

\[
\text{HasWord}(J. \ Cox, Cox) \land \text{HasWord}(Cox J., Cox) \land \text{HasAuthor}(C1, J. \ Cox) \land \\
\text{HasAuthor}(C2, Cox J.) \land \ (Cox = Cox) \ \Rightarrow \ (C1 = C2)
\]
Relational Models

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

\[(f_1 = f_2) \land (f_2 = f_3) \Rightarrow (f_3 = f_1)\]

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

\[\text{HasField}(r_1, f_1) \land \text{HasField}(r_2, f_2) \land f_1 = f_2 \Rightarrow r_1 = r_2\]

\[\text{HasAuthor}(C1, J. Cox) \land \text{HasAuthor}(C2, Cox J.) \land (J. Cox = Cox J.) \Rightarrow C1 = C2\]
Relational Models

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

\[ \text{HasAuthor}(c, a_1) \land \text{HasAuthor}(c, a_2) \implies \text{Coauthor}(a_1, a_2) \]

\[ \text{Coauthor}(a_1, a_2) \land \text{Coauthor}(a_3, a_4) \land a_1 = a_3 \implies a_2 = a_4 \]
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?
Approach 1

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

<table>
<thead>
<tr>
<th>Id</th>
<th>Author-1</th>
<th>Author-2</th>
<th>Paper</th>
</tr>
</thead>
<tbody>
<tr>
<td>A₁</td>
<td>John Smith</td>
<td>Richard Johnson</td>
<td>Indices and Views</td>
</tr>
<tr>
<td>A₂</td>
<td>J Smith</td>
<td>R Johnson</td>
<td>SQL Queries</td>
</tr>
<tr>
<td>A₃</td>
<td>Dr. Smyth</td>
<td>R Johnson</td>
<td>Indices and Views</td>
</tr>
</tbody>
</table>
Efficiency: Use Canopies

Reduces # of candidate pairs from:

\[ O(|\text{Records}|^2) \text{ to } |\text{Candidate pairs}| \]
Blocking is not sufficient for scaling

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

Example (using canopy clustering):

- \(|\text{Records}| = 1000, |\text{Candidate pairs}| = 15,000, \)
  - Time ~ 5 minutes
- \(|\text{Records}| = 50,000, |\text{Candidate pairs}| = 10 \text{ million} \)
  - Time required = 2,500 hours ~ 3 months
Distribute

Run collective entity-matching over canopies separately

Example for Collective methods\([SD06]\)

- \(|\text{References}| = 1000, |\text{Candidates}| = 15,000,\)  
  - Time = 5 minutes
- One canopy: \(|\text{References}| = 100, |\text{Candidates}| \sim 1000,\)  
  - Time \sim 10 Seconds
- \(|\text{References}| = 50,000, \# \text{ of canopies} \sim 13k\)  
  - Time \sim 20 hours \ll 3 months!
Problem: Correlations across canopies will be lost

\[ \text{CoAuthor}(A_1, B_1) \land \text{CoAuthor}(A_2, B_2) \land \text{match}(B_1, B_2) \Rightarrow \text{match}(A_1, A_2) \]

Example: CoAuthor rule grounds to the correlation

\[ \text{match}(\text{Richard Johnson, R Johnson}) \Rightarrow \text{match}(\text{J. Smith, John Smith}) \]
Approach 1

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

<table>
<thead>
<tr>
<th>P_1</th>
<th>Indices and Views</th>
<th>John Smith</th>
<th>Richard Johnson</th>
</tr>
</thead>
<tbody>
<tr>
<td>P_2</td>
<td>Indices &amp; Views</td>
<td>J. Smith</td>
<td>R. Johnson</td>
</tr>
<tr>
<td>P_3</td>
<td>Political Views</td>
<td>Jane Smith</td>
<td>R. Johnson</td>
</tr>
</tbody>
</table>

Lecture 22 : 590.02 Spring 13
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
- \text{match}(a_1,a_2)
- \text{match}(b_2,b_3)
- \text{match}(c_2,c_3)

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