Record Linkage

Craig Knoblock
University of Southern California

These slides are based in part on slides from
Matt Michelson, Sheila Tejada, Misha Bilenko,
Jose Luis Ambite, Claude Nanjo, and Steve Minton
### Record Linkage Problem

- **Task:**
  - **identify syntactically different records that refer to the same entity**
- **Common sources of variation:** database merges, typographic errors, abbreviations, extraction errors, OCR scanning errors, etc.

<table>
<thead>
<tr>
<th>Restaurant Name</th>
<th>Address</th>
<th>City</th>
<th>Phone</th>
<th>Cuisine</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fenix</td>
<td>8358 Sunset Blvd.</td>
<td>Hollywood</td>
<td>213/848-6677</td>
<td>American</td>
</tr>
<tr>
<td>Fenix at the Argyle</td>
<td>8358 Sunset Blvd.</td>
<td>W. Hollywood</td>
<td>213-848-6677</td>
<td>French (new)</td>
</tr>
</tbody>
</table>
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
Overview

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

Tokenize, then label tokens

Eliminate highly unlikely candidate record pairs.

Use learned distance metric to score field

Pass feature vector to SVM classifier to get overall score for candidate pair.

---

define schema alignment

Parsing

Blocking

Field Similarity

Record Similarity
Outline

- Blocking
- Field Matching
- Record Matching
- Entity Matching
- Conclusion
Outline

- Blocking
- Field Matching
- Record Matching
- Entity Matching
- Conclusion
# Record Linkage – Finding Matches

## Census Data

<table>
<thead>
<tr>
<th>First Name</th>
<th>Last Name</th>
<th>Phone</th>
<th>Zip</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matt</td>
<td>Michelson</td>
<td>555-5555</td>
<td>12345</td>
</tr>
<tr>
<td>Jane</td>
<td>Jones</td>
<td>555-1111</td>
<td>12345</td>
</tr>
<tr>
<td>Joe</td>
<td>Smith</td>
<td>555-0011</td>
<td>12345</td>
</tr>
</tbody>
</table>

## A.I. Researchers

<table>
<thead>
<tr>
<th>First Name</th>
<th>Last Name</th>
<th>Phone</th>
<th>Zip</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matthew</td>
<td>Michelson</td>
<td>555-5555</td>
<td>12345</td>
</tr>
<tr>
<td>Jim</td>
<td>Jones</td>
<td>555-1111</td>
<td>12345</td>
</tr>
<tr>
<td>Joe</td>
<td>Smeth</td>
<td>555-0011</td>
<td>12345</td>
</tr>
</tbody>
</table>
Record Linkage – Finding Matches

• Can’t compare all records!
  • Just 5,000 to 5,000 → 25,000,000 comparisons!
  • At 0.01s/comparison → 250,000 s → ~3 days!

• Need to use a subset of comparisons
  • “Candidate matches”
  • Want to cover true matches
  • Want to throw away non-matches
## Blocking – Generating Candidates

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

<table>
<thead>
<tr>
<th>First Name</th>
<th>Last Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matt</td>
<td>Michelson</td>
</tr>
<tr>
<td>Jane</td>
<td>Jones</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>First Name</th>
<th>Last Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matthew</td>
<td>Michelson</td>
</tr>
<tr>
<td>Jim</td>
<td>Jones</td>
</tr>
</tbody>
</table>

### (token, zip)

<table>
<thead>
<tr>
<th>First Name</th>
<th>Last Name</th>
<th>Zip</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matt</td>
<td>Michelson</td>
<td>12345</td>
</tr>
<tr>
<td>Matt</td>
<td>Michelson</td>
<td>12345</td>
</tr>
<tr>
<td>Matt</td>
<td>Michelson</td>
<td>12345</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>First Name</th>
<th>Last Name</th>
<th>Zip</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matthew</td>
<td>Michelson</td>
<td>12345</td>
</tr>
<tr>
<td>Jim</td>
<td>Jones</td>
<td>12345</td>
</tr>
<tr>
<td>Joe</td>
<td>Smith</td>
<td>12345</td>
</tr>
</tbody>
</table>
### Census Data

<table>
<thead>
<tr>
<th>First Name</th>
<th>Last Name</th>
<th>Zip</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matt</td>
<td>Michelson</td>
<td>12345</td>
</tr>
<tr>
<td>Jane</td>
<td>Jones</td>
<td>12345</td>
</tr>
<tr>
<td>Joe</td>
<td>Smith</td>
<td>12345</td>
</tr>
</tbody>
</table>

- **Zip** = ‘12345’
- 1 Block of **12345** Zips
- Compare to the “block-key”
- Group & Check to reduce Checks
• Can we use a better blocking key than tokens?
  • What about “fuzzy” tokens?
    • Matt → Matthew, William → Bill? (similarity)
    • Michael → Mychael (spelling)
  • Bi-Gram Indexing
Bi-Gram indexing

- **Step 1**: Take token and break it into bigrams
  - Token: **matt**
  - (‘ma,’ ‘at,’ ‘tt,’)

- **Step 2**: Generate all sub-lists
  - (# bigrams) \( \times \) (threshold) = sub-list length
  - \(3 \times .7 = 2\)

- **Step 3**: Sort sub-lists and put them into inverted index
  - (‘at’ ‘ma’) (‘at’ ‘tt’) (‘ma’ ‘tt’) → record w/ matt

---

Block key
BiGram Indexing: Properties

• **Threshold properties**
  - lower = shorter sub-lists → more lists
  - higher = longer sub-lists → less lists, less matches

• Now we can find spelling mistakes, close matches, etc…
Blocking – Multi-pass

- Sort neighborhoods on block keys
- Multiple independent runs using keys
  - runs capture different match candidates
- Attributed to (Hernandez & Stolfo, 1998)
- E.g.) 1\textsuperscript{st} → (token, last name)
  
  2\textsuperscript{nd} → (token, first name) &
  
  (token, phone)
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”
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??
Blocking Effectiveness

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

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

Pairs Completeness (PC) [Recall] = Sm / Nm

Sm = # true matches in candidates,
Nm = # 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
Old Techniques: Ad-hoc rules
New Techniques: Learn rules!

*Learned rules justified by quantitative effectiveness*

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
Learning Schemes – Intuition

- **Learn restrictive conjunctions**
  - partition the space $\rightarrow$ minimize False Positives
- **Union restrictive conjunctions**
  - Cover all training matches
  - Since minimized FPs, conjunctions should not contribute many FPs to the disjunction
Space of training examples

Rule 1: \((\text{zip|token}) \& (\text{first|token})\)

Final Rule: \([(\text{zip|token}) \& (\text{first|token})] \cup [(\text{last|1st Letter}) \& (\text{first|1st Letter})]\)

Rule 2: \((\text{last|1st Letter}) \& (\text{first|1st Letter})\)
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
```
• LEARN-ONE-RULE is greedy
  • rule containment as you go, instead of comparison afterward
  • Ex) rule: (token|zip) & (token|first)
    (token|zip) CONTAINS (token|zip) & (token|first)
  • Guarantee later rule is less restrictive – If not how are there examples left to cover?
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

(token, zip)

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

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

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

(token, last name)
### Experiments

**HFM** = \( \{\text{token, make}\} \cap \{\text{token, year}\} \cap \{\text{token, trim}\} \)
\( \cup \)
\( \{\text{1st letter, make}\} \cap \{\text{1st letter, year}\} \cap \{\text{1st letter, trim}\} \)
\( \cup \) \( \{\text{synonym, trim}\} \)

**BSL** = \( \{\text{token, model}\} \cap \{\text{token, year}\} \cap \{\text{token, trim}\} \)
\( \cup \) \( \{\text{token, model}\} \cap \{\text{token, year}\} \cap \{\text{synonym, trim}\} \)

<table>
<thead>
<tr>
<th>Cars</th>
<th>RR</th>
<th>PC</th>
</tr>
</thead>
<tbody>
<tr>
<td>HFM</td>
<td>47.92</td>
<td>99.97</td>
</tr>
<tr>
<td>BSL</td>
<td>99.86</td>
<td>99.92</td>
</tr>
<tr>
<td>BSL (10%)</td>
<td>99.87</td>
<td>99.88</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Census</th>
<th>RR</th>
<th>PC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best 5 Winkler</td>
<td>99.52</td>
<td>99.16</td>
</tr>
<tr>
<td>Adaptive Filtering</td>
<td>99.9</td>
<td>92.7</td>
</tr>
<tr>
<td>BSL</td>
<td>98.12</td>
<td>99.85</td>
</tr>
<tr>
<td>BSL (10%)</td>
<td>99.50</td>
<td>99.13</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Restaurants</th>
<th>RR</th>
<th>PC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Marlin</td>
<td>55.35</td>
<td>100.00</td>
</tr>
<tr>
<td>BSL</td>
<td>99.26</td>
<td>98.16</td>
</tr>
<tr>
<td>BSL (10%)</td>
<td>99.57</td>
<td>93.48</td>
</tr>
</tbody>
</table>
Conclusions

- Automatic Blocking Schemes using Machine Learning
  - Not created by hand
    - cheaper
    - easily justified
  - Better than non-experts ad-hoc and comparable to domain expert’s rules
    - Nice reductions – scalable record linkage
    - High coverage – don’t hinder record linkage
Outline

- Blocking
- Field Matching
- Record Matching
- Entity Matching
- Conclusion
Field Matching Approaches

- Expert-system rules
  - Manually written
- Token similarity
  - Used in Whirl
- String similarity
  - Used in Marlin
- Learned transformation weights
  - Used in HFM
Token-based Metrics

- Any string can be treated as a *bag of tokens*.
  - “8358 Sunset Blvd” → \{8358, Sunset, Blvd\}
    ‘nset’, ‘set ‘,

- Each token corresponds to a dimension in Euclidean space; string similarity is the normalized dot product (cosine) in the vector space.

- Weighting tokens by Inverse Document Frequency (IDF) is a form of *unsupervised* string metric learning.
Token Similarity
[Cohen, 1998]

- Follows the same approach used by classical IR algorithms (including web search engines).
- First, “stemming” is applied to each entry.
  - E.g. “Joe’s Diner” -> “Joe [‘s] Diner”
- Then, entries are compared by counting the number of words in common.
- Note: Infrequent words weighted more heavily by TF/IDF metric = Term Frequency / Inverse Document Frequency
Sequence-based String Metrics: String Edit Distance [Levenshtein, 1966]

• Minimum number of character deletions, insertions, or substitutions needed to make two strings equivalent.
  • “misspell” to “mispell” is distance 1 (‘delete s’)
  • “misspell” to “mistell” is distance 2 (‘delete s’, ‘substitute p with t’ OR ‘substitute s with t’, ‘delete p’)
  • “misspell” to “misspelling” is distance 3 (‘insert i’, ‘insert n’, ‘insert g’)

• Can be computed efficiently using dynamic programming in $O(mn)$ time where $m$ and $n$ are the lengths of the two strings being compared.

• Unit cost is typically assigned to individual edit operations, but individual costs can be used.
String Edit Distance with Affine Gaps
[Gotoh, 1982]

• Cost of gaps formed by contiguous deletions/insertions should be lower than the cost of multiple non-contiguous operators.
  • Distance from “misspell” to “misspelling” is <3.

• Affine model for gap cost: $\text{cost}(\text{gap})=s+e|\text{gap}|$, $e<s$

• Edit distance with affine gaps is more flexible since it is less susceptible to sequences of insertions/deletions that are frequent in natural language text (e.g. ‘Street’ vs. ‘Str’).
Learnable Edit Distance with Affine Gaps (Bilenko & Moody)

- **Motivation:**
  - Significance of edit operations depends on a particular domain
  - *Substitute* ‘/’ with ‘-’ insignificant for phone numbers.
  - *Delete* ‘Q’ significant for names.
  - Gap start/extension costs vary: sequence deletion is common for addresses (‘Street’ ➔ ‘Str’), uncommon for zip codes.

- Using individual weights for edit operations, as well as learning gap operation costs allows adapting to a particular field domain.

- [Ristad & Yianilos, ‘97] proposed a one-state generative model for regular edit distance.
Learnable Edit Distance with Affine Gaps – the Generative Model

• Matching/substituted pairs of characters are generated in state $M$.
• Deleted/inserted characters that form gaps are generated in states $D$ and $I$.
• Special termination state “#” ends the alignment of two strings.
• Similar to pairwise alignment HMMs used in bioinformatics [Durbin et al. ’98].
Learnable Edit Distance with Affine Gaps: Training

- Given a corpus of *matched* string pairs, the model is trained using Expectation-Maximization.
- The model parameters take on values that result in high probability of producing duplicate strings.
  - Frequent edit operations and typos have *high* probability.
  - Rare edit operations have *low* probability.
  - Gap parameters take on values that are optimal for duplicate strings in the training corpus.
- Once trained, distance between any two strings is estimated as *the posterior probability of generating the most likely alignment between the strings as a sequence of edit operations.*
- Distance computation is performed in a simple dynamic programming algorithm.
Learning Transformation Weights for Field Matching

- **Synonym**: Robert → {Bob, Robbie, Rob} ← Rob
- **Acronym**: International Business Machines ← I.B.M.
- **Misspelling**: Smyth ← Smith
- **Concatenation**: Mc Donalds ← McDonalds
- **Prefix/Abbreviation**: Inc ← Incorporated
- **Suffix**: Reformat ← format
- **Substring**: Garaparandaseu ← Paranda
- **Stemming**: George’s Golfing Range ← George’s Golfer Range
- **Levenstein**: the ← teh
Training the Field Learner

Transformations =

\{ Equal, Synonym, Misspelling, Abbreviation, Prefix, Acronym, Concatenation, Suffix, Soundex, Missing… \}

Transformation Graph

Training the Field Learner

Another Transformation Graph

“Apartment 16 B, 3101 Eades St” $\leftrightarrow$ “3101 Eads Street NW Apt 16B”
Generic Preference Ordering

Equal > Synonym > Misspelling > Missing …

Training Algorithm:

I. For each training record pair
   i. For each aligned field pair (a, b)
      i. build transformation graph T(a, b)
         ➢ “complete / consistent”
         ➢ Greedy approach: preference ordering over transformations
Training the Field Learner
Step 2: Calculating the probabilities

• For each transformation type \( v_i \) (e.g. Synonym), calculate the following two probabilities:

\[
p(v_i|M) = \frac{\text{freq. of } v_i \text{ in } M}{\text{size } M} \\
p(v_i|\neg M) = \frac{\text{freq. of } v_i \text{ in } \neg M}{\text{size } \neg M}
\]

• Note: Here we make the Naïve Bayes assumption
Scoring unseen instances

An Example

\[ a = "\text{Giovani Italian Cucina Int’l}" \]
\[ b = "\text{Giovani Italian Kitchen International}" \]
\[ T(a, b) = \{ \text{Equal(Giovani, Giovani), Equal(Italian, Italian), Synonym(Cucina, Kitchen), Abbreviation(Int’l, International)} \} \]

Training:
\[ p(M) = 0.31 \quad p(\neg M) = 0.69 \]
\[ p(\text{Equal} | M) = 0.17 \quad p(\text{Equal} | \neg M) = 0.027 \]
\[ p(\text{Synonym} | M) = 0.29 \quad p(\text{Synonym} | \neg M) = 0.14 \]
\[ p(\text{Abbreviation} | M) = 0.11 \quad p(\text{Abbreviation} | \neg M) = 0.03 \]

\[ p(M) \prod p(v_i | M) = 2.86 \times 10^{-4} \]
\[ p(\neg M) \prod p(v_i | \neg M) = 2.11 \times 10^{-6} \]

Score\text{HFM} = 0.993 \rightarrow \text{Good Match!}
Outline

- Blocking
- Field Matching
- Record Matching
- Entity Matching
- Conclusion
Combining String Similarity Across Fields

- Some fields are more indicative of record similarity than others:
  - For addresses, *street address* similarity is more important than *city* similarity.
  - For bibliographic citations, *author* or *title* similarity are more important than *venue* (i.e. conference or journal name) similarity.

- Field similarities should be weighted when combined to determine record similarity.

- Weights can be learned using a learning algorithm [Cohen & Richman ‘02], [Sarawagi & Bhamidipaty ‘02], [Tejada et. al. ‘02].
Record Matching Approaches

- Learning Decision Trees
  - Used in Active Atlas (Tejada et al.)
- Support Vector Machines (SVM)
  - Used in Marlin (Bilenko & Moody)
- Unsupervised Learning
  - Used in matching census records (Winkler 1998)
**Learning Mapping Rules with Decision Trees**

**Zagat’s Restaurants**

<table>
<thead>
<tr>
<th>Name</th>
<th>Street</th>
<th>Phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>Art’s Deli</td>
<td>12224 Ventura Boulevard</td>
<td>818-756-4124</td>
</tr>
<tr>
<td>Teresa’s</td>
<td>80 Montague St.</td>
<td>718-520-2910</td>
</tr>
<tr>
<td>Steakhouse The</td>
<td>128 Fremont St.</td>
<td>702-382-1600</td>
</tr>
<tr>
<td>Les Celebrites</td>
<td>155 W. 58th St.</td>
<td>212-484-5113</td>
</tr>
</tbody>
</table>

**Dept. of Health**

<table>
<thead>
<tr>
<th>Name</th>
<th>Street</th>
<th>Phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>Art’s Delicatessen</td>
<td>12224 Ventura Blvd.</td>
<td>818/755-4100</td>
</tr>
<tr>
<td>Teresa’s</td>
<td>103 1st Ave. between 6th and 7th Sts.</td>
<td>212/228-0604</td>
</tr>
<tr>
<td>Binion’s Coffee Shop</td>
<td>128 Fremont St.</td>
<td>702/382-1600</td>
</tr>
<tr>
<td>Les Celebrites</td>
<td>160 Central Park S</td>
<td>212/484-5113</td>
</tr>
</tbody>
</table>

**Mapping rules:**

Name > .9 & Street > .87 => mapped

Name > .95 & Phone > .96 => mapped
Learning Mapping Rules

Set of Similarity Scores

<table>
<thead>
<tr>
<th>Name</th>
<th>Street</th>
<th>Phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>.967</td>
<td>.973</td>
<td>.3</td>
</tr>
<tr>
<td>.17</td>
<td>.3</td>
<td>.74</td>
</tr>
<tr>
<td>.8</td>
<td>.542</td>
<td>.49</td>
</tr>
<tr>
<td>.95</td>
<td>.97</td>
<td>.67</td>
</tr>
</tbody>
</table>

Mapping Rules

- Name > .8 & Street > .79 => mapped
- Name > .89 => mapped
- Street < .57 => not mapped
Mapping Rule Learner with Active Learning

Choose initial examples

Generate committee of learners

Learn Rules

Classify Examples

Votes

Choose Example

Learn Rules

Classify Examples

Votes

Learn Rules

Classify Examples

Votes

Set of Mapped Objects

Label

USER

Label
Choose an example based on the disagreement of the query committee:

<table>
<thead>
<tr>
<th>Examples</th>
<th>M1</th>
<th>M2</th>
<th>M3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Art’s Deli, Art’s Delicatessen</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>CPK, California Pizza Kitchen</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Ca’Brea, La Brea Bakery</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

In this case CPK, California Pizza Kitchen is the most informative example based on disagreement.
SVM Learned Record Similarity

- String similarities for each field are used as components of a feature vector for a pair of records.

- SVM is trained on labeled feature vectors to discriminate duplicate from non-duplicate pairs.

- Record similarity is based on the distance of the feature vector from the separating hyperplane.
Learning Record Similarity (cont.)

<table>
<thead>
<tr>
<th>Name</th>
<th>Address</th>
<th>City</th>
<th>Cuisine</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fenix</td>
<td>8358 Sunset Blvd. West</td>
<td>Hollywood</td>
<td>American</td>
</tr>
<tr>
<td>Fenix at the Argyle</td>
<td>8358 Sunset Blvd.</td>
<td>W. Hollywood</td>
<td>French (new)</td>
</tr>
</tbody>
</table>

Learned distance measure

Feature vector: $d_{1n}, d_{2n}, d_{1a}, d_{2a}, d_{1c}, d_{2c}, d_{1cu}, d_{2cu}$

SVM

Duplicate records → Non-duplicate records
Learnable Vector-space Similarity

Each string is converted to vector-space representation.

The pair vector is classified as “similar” or “dissimilar”.

The pair vector is created.

Similarity between strings is obtained from the SVM output.

\[ \text{Sim}(x, y) \propto f(p^{(x,y)}) \]
Unsupervised Record Linkage

• Idea: Analyze data and automatically cluster pairs into three groups:
  • Let $R = \frac{P(\text{obs} | \text{Same})}{P(\text{obs} | \text{Different})}$
  • Matched if $R > \text{threshold } T_U$
  • Unmatched if $R < \text{threshold } T_L$
  • Ambiguous if $T_L < R < T_U$

• This model for computing decision rules was introduced by Felligi & Sunter in 1969

• Particularly useful for statistically linking large sets of data, e.g., by US Census Bureau
• Winkler (1998) used EM algorithm to estimate $P(\text{obs} \mid \text{Same})$ and $P(\text{obs} \mid \text{Different})$

• EM computes the *maximum likelihood estimate*. The algorithm iteratively determines the parameters most likely to generate the observed data.

• Additional mathematical techniques must be used to adjust for “relative frequencies”, i.e. last name of “Smith” is much more frequent than “Knoblock”.

Unsupervised Record Linkage (cont.)
Outline

• Blocking
• Field Matching
• Record Matching
• Entity Matching
• Conclusion
EntityBases: Compiling, Organizing and Querying Massive Entity Repositories

• Today:
  • Lots of data & documents available
  • NLP technology for extracting simple entities & facts

• Opportunity: Collect and query billions of facts about millions of entities (e.g., people, companies, locations, …)
DUBAI TRADER AND IRANIAN COMPANY OFFICER INDICTED FOR SCHEME TO ILLEGALLY SHIP U.S. GOODS TO IRAN

Washington, D.C. - United States Attorney Kenneth L. Wainstein, Department of Homeland Security Assistant Secretary for Immigration and Customs Enforcement (ICE) Michael J. Garcia, U.S. Department of Commerce Undersecretary for Industry and Security Kenneth I. Juster, and jointly announced that a federal grand jury in the District of Columbia has returned an indictment charging Khalid Mahmood, also known as Khalid Mahmood Chaudhary, 52, of Dubai, United Arab Emirates, and his age unknown, of Iran, with violations of the International Economic Powers Act, the Iranian Transactions Regulations, and the Export Administration Regulations.

The indictment alleges that Mahmood was doing business as Sharp Line Trading with offices in the United Arab Emirates, and Sharbaf was a principal officer of a forklift manufacturing firm located in Iran. According to the indictment, in early June 2004, an employee of Sepahan Lifter Company contacted a United States company by email and requested a price quotation for particular radiators for heavy-duty 5-ton capacity forklift trucks manufactured in Iran by

LEXINGTON, KENTUCKY MAN SENTENCED TO 39 MONTHS IN PRISON FOR VIOLATING TRADE EMBARGO AGAINST IRAN

Washington, D.C. - United States Attorney Kenneth L. Wainstein, Darryl W. Jackson, United States Department of Commerce Assistant Secretary for Export Enforcement, and Mark Gendar, Acting Special Agent-in-Charge for U.S. Immigration and Customs Enforcement (ICE), Department of Homeland Security, announced yesterday that the Honorable John D. Bates sentenced Robert E. Quinn, 54, of Lexington, Kentucky, to 39 months of incarceration and a fine of $6,000. Quinn was found guilty by a federal jury on December 7, 2005, of one count of conspiring to violate the U.S. trade embargo against Iran and five counts of illegal exports to Iran.

In October 2005, a federal grand jury in the District of Columbia returned a six-count superseding indictment against Quinn, 54, and Michael H. Holland, also of Lexington, Kentucky, and of Iran, charging them with violating the United States embargo on trade with Iran. Quinn and Holland were sales executives employed by Clark Material Handling Corporation ("CMHC"), a Kentucky-based forklift truck manufacturer. Sharbaf is President and Managing Director of a forklift truck manufacturer in Esfahan, Iran.
The Idea

- **EntityBases**: Large-scale, organized entity knowledgebases
  - composed of billions of facts/millions of entities
- Each Entitybase:
  - **Aggregates** information about a single entity type
    - e.g. PeopleBase, CompanyBase, AssetBase, GeoBase, …
    - Simple representation, broad coverage.
  - **Integrates** data collected from numerous, heterogeneous sources
  - **Consolidates** data, resolving multiple references to the same entity
- Requires scalable algorithms and tools to populate, organize and query massive EntityBases
Why this is a Hard Problem: Limitations of Previous Work

• EntityBase is not a straightforward extension of past work on data integration and record linkage

• New challenges include:
  • Real-world entities have attributes with multiple values
    • Ex: name: maiden name, transliterations, aliases, …
    • Previous work only dealt with records with single values for attributes (e.g., a single name, phone number, address, etc.)
  • Need to link arbitrary number of sources, with different schemas
    • Most previous work on record linkage focused on merging two tables with similar schemas
  • In addition, real-time response must be considered

• We had to extend previous work on both data integration and record linkage to support massive-scale entity bases
  • Without compromising efficiency!
EntityBase Architecture

Candidate evaluation

Matching

Blocking

Inverted index

<table>
<thead>
<tr>
<th>token</th>
<th>entities</th>
</tr>
</thead>
<tbody>
<tr>
<td>steve</td>
<td>1, 3, 9, 298, 4929, ...</td>
</tr>
<tr>
<td>minton</td>
<td>6, 54, 238, 787, ...</td>
</tr>
</tbody>
</table>

Mediator

Entity model

Local Entity Repository

Numerous heterogeneous data sources

Database

Legacy program

Web site

Legacy program

Database

Database

Legacy program

Web site

Candidate identification

Query processing

Information Gathering

EntityBase System

Queries
Data Gathering

- Sample Linkable Company Sources
  - Kompass
  - Ameinfo
- Used Fetch Agent Platform
EntityBase Integration Architecture

- **Local Entity Repository (LER):**
  - stores entity identifying attributes
  - record linkage reasoning on these attributes
- **Materialized Sources**
  - Entity-identifying attributes fed into core entity base
  - Additional attributes materialized, but not copied into LER for performance
- **Remote Sources**
  - Cannot be materialized due to organizational constraints, security, or rapid data change
- **Mediator**
  - Assigns common semantics to data from LER and sources
  - Integrates data from LER and sources in response to user queries
Data Representation Approach

- EntityBase uses a **Mediated Schema** to integrate data from different sources
  - Assigns common semantics
  - Handle multiple values
  - Normalizes/Parses values
  - Object-relational representation
  - General, but still efficient for record linkage processing
Mediated Schema

- **Entity**: main object type
  - Ex: Company
- Each entity has several *multi-valued* attributes (units):
  - Ex: name, address, phone, keyperson, code (ticker, D&B DUNS, ISIN,...), email, product/service, ...
- **Unit**: structured (sub)object with *single-valued* attributes
  - Ex: address( FullAddress, StreetAddress, City, Region, PostalCode, Country, GeoExtent)
- Some units extended with geospatial extent
  - Ex: address, phone
### Sources

<table>
<thead>
<tr>
<th>SRID</th>
<th>Name</th>
<th>Office Street</th>
<th>Office City</th>
<th>Office State</th>
<th>Office Zip</th>
<th>Factory Street</th>
<th>Factory City</th>
<th>Factory State</th>
<th>Factory Zip</th>
<th>Ceo</th>
<th>Cto</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Fetch Technologies</td>
<td>PO Box 1000</td>
<td>El Segundo</td>
<td>CA</td>
<td>90245</td>
<td>2041 Rosecrans Ave</td>
<td>El Segundo</td>
<td>CA</td>
<td>90245</td>
<td>Robert Landes</td>
<td>Steve Minton</td>
</tr>
</tbody>
</table>

### Address

<table>
<thead>
<tr>
<th>RID</th>
<th>Src</th>
<th>SRID</th>
<th>FullAddress</th>
<th>StreetAddress</th>
<th>City</th>
<th>Region</th>
<th>Postal Code</th>
<th>Country</th>
<th>Geo Extent</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>s1</td>
<td>3</td>
<td>PO Box 1000 El Segundo CA 90245</td>
<td>PO Box 1000</td>
<td>El Segundo</td>
<td>CA</td>
<td>90245</td>
<td>USA</td>
<td>SDO_GEOMETRY(…)</td>
</tr>
<tr>
<td>22</td>
<td>s1</td>
<td>3</td>
<td>2041 Rosecrans Ave El Segundo CA 90245</td>
<td>2041 Rosecrans Ave</td>
<td>El Segundo</td>
<td>CA</td>
<td>90245</td>
<td>USA</td>
<td>SDO_GEOMETRY(…)</td>
</tr>
<tr>
<td>23</td>
<td>s2</td>
<td>10</td>
<td>CA 90245</td>
<td>null</td>
<td>null</td>
<td>CA</td>
<td>90245</td>
<td>USA</td>
<td>SDO_GEOMETRY(…)</td>
</tr>
</tbody>
</table>

### Entity

<table>
<thead>
<tr>
<th>EID</th>
<th>unit</th>
<th>RID</th>
<th>name</th>
<th>Source</th>
<th>SRID</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>name</td>
<td>14</td>
<td>7</td>
<td>Name</td>
<td>10</td>
</tr>
<tr>
<td>7</td>
<td>name</td>
<td>56</td>
<td>7</td>
<td>Name</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>addres s</td>
<td>21</td>
<td>7</td>
<td>Address</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>addres s</td>
<td>22</td>
<td>7</td>
<td>Address</td>
<td>10</td>
</tr>
<tr>
<td>7</td>
<td>addres s</td>
<td>23</td>
<td>7</td>
<td>Address</td>
<td>10</td>
</tr>
</tbody>
</table>

Importing Data from DB/Fetch Agents

- Import Rule for address:

  \texttt{address( RID, Source, SRID, FullAddress, StreetAddress, City, PostalCode, Country, GeoExtent) :-}

  \texttt{IranYellowPages( Name, ManagingDirector, CommercialManager,}
  \texttt{ CentralOffice, OfficeTelephone, OfficeFax, OfficePOBox,}
  \texttt{ Factory, FactoryTelephone, FactoryFax, FactoryPOBox, Email,}
  \texttt{ WebAddress, ProductsServices, SRID) ^}

  \texttt{ParseAddress( Centraloffice, StreetAddress, City) ^}

  \texttt{Concat( StreetAddress, City, Region, OfficePOBox, Country, FullAddress) ^}

  \texttt{ComputeGeoextent( FullAddress, GeoExtent) ^}

  \texttt{GenRID( SRID, Source, "1", RID) ^ (Source = "IranYellowPages ") ^}

  \texttt{(OfficePOBox = PostalCode) ^ (Country = "Iran")}
Entity linkage

..., said A. Baroul of Tehran-based Adaban, ....

Find the company that most closely matches the case where company name ~ “Adaban”, address ~ “Tehran”, key person ~ “A. Baroul”

EntityBase

Quickly identify promising candidates from large EB

Closely judge candidates

...and the best match is:

Craig Knoblock
Efficient blocking

Want to quickly identify promising candidates

But...

- We need to use fast comparison methods
  - e.g., string or word ID comparisons
  - edit distance computations are likely too expensive
- We are working with many potential entities
  - Do not want to return too large of a block size (will impact RL perf)

Core issue

- Computing set intersections / unions efficiently
  → Novel Union/Count Algorithm
Intelligent Field Matching with Transformations

- Transformations relate two values and provide a more precision definition of how well they match.
- Using fine-grained transformations in the matching phase increases accuracy.
Matching

Candidate set (from blocking)

1. (E-5640) Abadan Intl Transportation
   (E-109) Abadan Petrochemical Co.
   (E-71) Kavian Industrial
   (E-89276) Adaban Partners

2. Compute attribute value metrics

3. Classification, based on attribute value evaluation

4. (E-5640) Abadan Intl Transportation  85%
   (E-109) Abadan Petrochemical Co.  99.5%

- Classifier judges the importance of the combination of attribute value metrics
  - e.g., complete mismatch on address may be offset by strong matches on phone and name

Steve Minton                  Fletch Software                   El Segundo, CA
Steve N Minton          Fetch Technologies               El Segundo

Craig Knoblock
University of Southern California
Outline

- Blocking
- Field Matching
- Record Matching
- Entity Matching
- Conclusion
Related Work

- Record linkage [Newcombe et al. ’59; Fellegi & Sunter ’69; Winkler ’94, ’99, ‘02]
- Database hardening [Cohen et al. ’00]
- Merge/purge [Hernandez & Stolfo ’95]
- Field matching [Monge & Elkan ’96]
- Data cleansing [Lee et al. ’99]
- Name matching [Cohen & Richman ’01, Cohen et al. ’03]
- De-duplication [Sarawagi & Bhamidipaty ’02]
- Object identification [Tejada et al. ’01, ’02]
- Fuzzy duplicate elimination [Ananthakrishna et al. ’02]
- Identity uncertainty [Pasula et. al. ’02, McCallum & Wellner ‘03]
- Object consolidation [Michalowski et al. ’03]
Conclusions

- Technical choices in record linkage:
  - Approach to blocking
  - Approach to field matching
  - Approach to record matching
  - Is the matching done pairwise or based on entities

- Learning approaches have the advantage of being able to
  - Adapt to specific application domains
  - Learn which fields are important
  - Learn the most appropriate transformations

- Optimal classifier choice is sensitive to the domain and the amount of available training data.