Record Linkage

Craig Knoblock
University of Southern California

These slides are based in part on slides from
Sheila Tejada and Misha Bilenko
Record Linkage Problem

<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. West</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>

- 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.


Outline

• Introduction
• Candidate Generation
• Field Matching
• Record Matching
• Discussion
Integrating Restaurant Sources

Zagat’s Restaurant Guide Source

Department of Health Restaurant Rating Source

ARIADE Information Mediator

Question: What is the Review and Rating for the Restaurant “Art’s Deli”?
Ariadne Information Mediator

Extract web objects in the form of database records

<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>

<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>5432 Sunset Blvd</td>
<td>212/484-5113</td>
</tr>
</tbody>
</table>
Application Dependent Mapping

Observations:

• Mapping objects can be application dependent
• Example:
  
  Mapped?

Steakhouse The  128 Fremont Street  702-382-1600

Binion's Coffee Shop  128 Fremont St.  702/382-1600

• The mapping is in the application, not the data
• User input is needed to increase accuracy of the mapping
General Approach to Record Linkage

1. Identification of candidate pairs
   • 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

4. Linkage/Merging
   • Records with similarity higher than a threshold are labeled as matches
   • Equivalence classes are found by transitive closure
Outline

• Introduction
• Candidate Generation
• Field Matching
• Record Matching
• Discussion
Candidate Generation

- Comparing all possible matches across two datasets would require $n^2$ comparisons
- On large datasets this is impractical and wasteful
- Instead, we compare only those that could possibly be matched
- Also referred to as blocking
Approach to Candidate Generation

- Construct an inverted index of all tokens in a document
  - Links the token to the documents in which it appears
  - Place each token in a hash table
- Apply transformations on the tokens to find closely related tokens
  - Transformations include equal, stemming, soundex, and other unary transformations
- Use a stop list to avoid common tokens
  - Tokens such as “the”, “s”, etc. would be on the stop list
### Example: Partial Inverted Index for LA Department of Health

Document (Object) 5 Restaurant name: “Art’s Delicatessen”

Tokens: “Art”, “s”, “Delicatessen”


<table>
<thead>
<tr>
<th>Transformed Tokens</th>
<th>Transformations</th>
<th>Original Tokens</th>
<th>Object Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Art”</td>
<td>Equal</td>
<td>“Art”</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>Stemming</td>
<td>“Arte”</td>
<td>57</td>
</tr>
<tr>
<td>“A630”</td>
<td>Soundex</td>
<td>“Art”</td>
<td>5</td>
</tr>
<tr>
<td>“s”</td>
<td>Equality</td>
<td>“s”</td>
<td>5,6,9,71,79,97,111</td>
</tr>
<tr>
<td>“S000”</td>
<td>Soundex</td>
<td>“s”</td>
<td>5,6,9,71,79,97,111</td>
</tr>
<tr>
<td>“Del”</td>
<td>Stemming</td>
<td>“Dell”</td>
<td>57</td>
</tr>
<tr>
<td></td>
<td>Stemming</td>
<td>“Deli”</td>
<td>7,93</td>
</tr>
<tr>
<td></td>
<td>Equality</td>
<td>“Del”</td>
<td>60</td>
</tr>
<tr>
<td>“Deli”</td>
<td>Equality</td>
<td>”Deli”</td>
<td>7,93</td>
</tr>
<tr>
<td>“D400”</td>
<td>Soundex</td>
<td>”Deli”</td>
<td>7,93</td>
</tr>
<tr>
<td></td>
<td>Soundex</td>
<td>“Dell”</td>
<td>57</td>
</tr>
<tr>
<td>“Delicatessen”</td>
<td>Equality</td>
<td>“Delicatessen”</td>
<td>5</td>
</tr>
<tr>
<td>“D423”</td>
<td>Soundex</td>
<td>“Delicatessen”</td>
<td>5</td>
</tr>
<tr>
<td>“Dell”</td>
<td>Equality</td>
<td>“Dell”</td>
<td>57</td>
</tr>
</tbody>
</table>
Outline

• Introduction
• Candidate Generation
• Field Matching
• Record Matching
• Discussion
Field Matching Approaches

- Expert-system rules
  - Manually written
- Information retrieval
- General string similarity
  - Used in Marlin
- Learned weights for domain-specific transformations
  - Used in Active Atlas
Information Retrieval Approach
[Cohen, 1998]

• Idea: Evaluate the similarity of records via textual similarity. Used in Whirl (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 TFIDF metric = Term Frequency Inverse Document Frequency
Token-based Metrics

- Any string can be treated as a *bag of tokens*.
  - “8358 Sunset Blvd” ▶ \{8358, Sunset, Blvd\}

- 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.
String Similarity Measures

- Metrics based on *sequence comparison*:
  - String edit distance
  - Variants: Length of longest common subsequence, Smith-Waterman distance, etc.
  - [Gusfield ‘97]

- Metrics based on *vector-space similarity*:
  - Rely on representing strings as sets of tokens
  - Variants include word tokenization, n-grams, etc.
  - [Baeza-Yates & Ribeiro-Neto ‘98]
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

• 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.
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

- Learn general transformations to recognize related objects

<table>
<thead>
<tr>
<th>Zagat’s</th>
<th>Transformations</th>
<th>Dept of Health</th>
</tr>
</thead>
<tbody>
<tr>
<td>Art’s Deli</td>
<td>Prefix</td>
<td>Art’s Delicatessen</td>
</tr>
<tr>
<td>California Pizza Kitchen</td>
<td>Acronym</td>
<td>CPK</td>
</tr>
<tr>
<td>Philippe The Original</td>
<td>Stemming</td>
<td>Philippe’s The Original</td>
</tr>
</tbody>
</table>
Transformation Weights

• Transformations can be more appropriate for a specific application domain
  - Restaurants, Companies or Airports

• Or for different attributes within an application domain
  - Acronym more appropriate for the attribute Restaurant Name than for the Phone attribute

• Learn likelihood that if transformation is applied then the objects are mapped

  Transformation Weight = P(mapped | transformation)
Types of Transformations

Unary Transformations

• Equality (Exact match)
• Stemming
• Soundex (e.g. “Celebrities” => “C453”)
• Abbreviation (e.g. “3rd” => “third”)

Binary Transformations

• Initial
• Prefix (e.g. “Deli” & “Delicatessen”)
• Suffix
• Substring
• Acronym (e.g. “California Pizza Kitchen” & “CPK”)
• Drop Word
Applying Unary Transformations

Employs Information Retrieval Techniques

- One set of attribute values broken into words or tokens
  - “Art” “s” “Delicatessen”
- Apply Type I transformations to tokens
  - “Art” “A630” “s” “S000” “Delicatessen” “D423”
- Enter tokens into inverted index
- Tokens from second set used to query the index
  - Transformed query set: “Art” “A630” “s” “S000” “Deli” “Del” “D400”
Applying Binary Transformations

- Binary transformations improve measurement of similarity
Calculate Transformation Weights

\[ P(\text{mapped} \mid \text{transformation}) = \]

\[ \frac{P(\text{transformation} \mid \text{mapped}) \cdot P(\text{mapped})}{P(\text{transformation})} \]

<table>
<thead>
<tr>
<th>Examples</th>
<th>Classification</th>
<th>Labeled by</th>
</tr>
</thead>
<tbody>
<tr>
<td>Art’s Deli, Art’s Delicatessen</td>
<td>Mapped</td>
<td>Learner</td>
</tr>
<tr>
<td>CPK, California Pizza Kitchen</td>
<td>Mapped</td>
<td>User</td>
</tr>
<tr>
<td>Ca’Brea, La Brea Bakery</td>
<td>Not Mapped</td>
<td>Learner</td>
</tr>
</tbody>
</table>
Computing Textual Similarity

Zagat’s Restaurant Objects

Name  Street  Phone
Z1,  Z2,  Z3

Department of Health Objects

Name  Street  Phone
D1,  D2,  D3

• Candidate Generator returns sets of similarity scores

Name        Street        Phone
.9          .79           .4
.17          .3           .74
...
Outline

- Introduction
- Candidate Generation
- Field Matching
- Record Matching
- Discussion
Record Matching Approaches

- Learning Decision Trees
- Support Vector Machines (SVM)
- Unsupervised Learning
Learning Mapping Rules with Decision Trees

- Learning important attributes for determining a mapping

<table>
<thead>
<tr>
<th>Name</th>
<th>Street</th>
<th>Phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zagat’s</td>
<td>Art’s Deli 12224 Ventura Blvd</td>
<td>818-756-4124</td>
</tr>
<tr>
<td>Dept of Health</td>
<td>Art’s Delicatessen 12224</td>
<td>818/755-4100</td>
</tr>
<tr>
<td></td>
<td>Ventura Blvd.</td>
<td></td>
</tr>
</tbody>
</table>
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>
<tr>
<td>...</td>
<td></td>
<td></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

Set of Mapped Objects

Label

USER

Learn Rules

Classify Examples

Votes
Committee Disagreement

- Chooses an example based on the disagreement of the query committee

<table>
<thead>
<tr>
<th>Committee</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
Learnable Vector-space Similarity

Each string is converted to vector-space representation.

The pair vector is created.

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

Similarity between strings is obtained from the SVM output.

\[ \text{Sim}(x, y) \propto f(p^{(x,y)}) \]
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].
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>

![Diagram showing a learned distance measure and feature vector leading to an SVM, with duplicate records on one end and non-duplicate records on the other.](diagram.png)
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
Unsupervised Record Linkage (cont.)

- Winkler (1998) used EM algorithm to estimate $P(\text{obs | Same})$ and $P(\text{obs | Different})$
- 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”.

Craig Knoblock  University of Southern California
Outline

• Introduction
• Candidate Generation
• Field Matching
• Record Matching
• Discussion
Enforcing One-to-One Relationship

- Viewed as weighted bipartite matching problem

Given weights $W$, matching method determines mostly likely Matching Assignment
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 candidate generation
  - Approach to field matching
  - Approach to record matching
- 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.