Record Linkage

Everything Data

CompSci 290.01 Spring 2014
Announcements (Wed. Jan. 28)

• **Homework #3** will be posted by tomorrow morning
  – Due midnight Sunday
Recap: Querying Relational Databases in SQL

**SELECT** columns or expressions

(or for each group of them if query has grouping/aggregation)

**FROM** tables

1. Generate all combinations of rows, one from each table; each combination forms a “wide row”

**WHERE** conditions

2. Filter—keep only “wide rows” satisfying conditions

**GROUP BY** columns

3. Group—“wide rows” with matching values for columns go into the same group

**ORDER BY** output columns;

4. Sort the output rows

5. Compute one output row for each “wide row”
Problem

• Forbes magazine article: “Wall Street’s favorite senators”
Problem

• Forbes magazine article: “Wall Street’s favorite senators”

• What are their ages?
Solution

• Join with the persons table (from govtrack)

• But there is no key to join on ...
Record Linkage

• Problem of finding duplicate entities across different sources (or even within a single dataset).

Record linkage

From Wikipedia, the free encyclopedia
(Redirected from Entity resolution)

Record linkage (RL) refers to the task of finding records in a data set that refer to the same entity across different data sources (e.g., data files, books, websites, databases). Record linkage is necessary when joining data sets based on entities that may or may not share a common identifier (e.g., database key, URI, National identification number), as may
Ironically, Record Linkage has many names

- Coreference resolution
- Entity Resolution
- Duplicate detection
- Reference reconciliation
- Object consolidation
- Object identification
- Entity clustering
- Approximate match
- Hardening soft databases
- Merge/purge
- Reference matching
- Household matching
- Householding
- Doubles
- Fuzzy match
- Deduplication
- Identity uncertainty
Motivating Example 1: Web
Motivating Example 1: Web
Motivating Example 1: Web

Auto Pro to Call
- ★★★★★ (6 Reviews)
- (919) 967-2271
- 1809 Fordham Blvd, Chapel Hill, NC 27514
- Directions | Send to Phone
- autoprotocall.com

These guys are crooks. They wanted $100 just to put the meter on my check engine light a task that takes 2 minutes. $100 just to diagnose it not to do any repairs. Places like Advance Auto... more

Swedish Imports
- (919) 493-4545
- 5404 Durham Chapel Hill Blvd, Durham, NC 27707
- Directions | Send to Phone
- swedishimports.net

N-Tune Automotive
- ✅ Merchant verified
- (919) 401-2612
- 411 Erwin Rd, Durham, NC 27707
- Directions | Send to Phone
- www.naturegenuine.com

Raleigh Auto Repair
- A & J Automotive since 1996
- Dependable Service, Honest Answers
- www.ajautorepair.com

10% Off Any Auto Repair
- Plus Oil Change Combo Coupons for $21.95 or Less on Any Make or Model
- www.LocalBizNow.com

Auto Mechanic School
- Become a mechanic with the Auto Repair Technician program.
- www.penfoster.edu
Motivating Example 2: Network Science

• Measuring the topology of the internet ... using traceroute
IP Aliasing Problem [Willinger et al. 2009]

Figure 2. The IP alias resolution problem. Paraphrasing Fig. 4 of [50], traceroute does not list routers (boxes) along paths but IP addresses of input interfaces (circles), and alias resolution refers to the correct mapping of interfaces to routers to reveal the actual topology. In the case where interfaces 1 and 2 are aliases, (b) depicts the actual topology while (a) yields an “inflated” topology with more routers and links than the real one.
Figure 3. The IP alias resolution problem in practice. This is reproduced from [48] and shows a comparison between the Abilene/Internet2 topology inferred by Rocketfuel (left) and the actual topology (top right). Rectangles represent routers with interior ovals denoting interfaces. The histograms of the corresponding node degrees are shown in the bottom right plot. © 2008 ACM.
And many many more examples

- Linking Census Records
- Public Health
- Medical records
- Web search – query disambiguation
- Comparison shopping
- Maintaining customer databases
- Law enforcement and Counter-terrorism
- Scientific data
- Genealogical data
- Bibliographic data
Opportunity

http://lod-cloud.net/
Back to our example

• Join with the persons table (from govtrack)

• But there is no key to join on ...

• What about (firstname, lastname)?
Attempt 1:

```
SELECT w.*, date_part('year', current_date) - date_part('year', p.birthday) AS age
FROM wallst w, persons p
WHERE w.first_name = p.first_name
    and w.last_name = p.last_name;
```
Problems

• Join condition is too specific
  – Nicknames used instead of real first names
Attempt 2:

• Join on Last name + Age < 100 (senator must be alive)

```
SELECT w.*, date_part('year', current_date) - date_part('year', p.birthday) AS age
FROM wallst w, persons p
WHERE w.lastname = p.last_name and date_part('year', current_date) - date_part('year', p.birthday) < 100;
```
Problem:

• Join condition is too inclusive
  – Many individuals share the same last name.

<table>
<thead>
<tr>
<th>Surname</th>
<th>Approx #</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smith</td>
<td>2.4 M</td>
<td>1</td>
</tr>
<tr>
<td>Johnson</td>
<td>1.8 M</td>
<td>2</td>
</tr>
<tr>
<td>Williams</td>
<td>1.5 M</td>
<td>3</td>
</tr>
<tr>
<td>Brown</td>
<td>1.4 M</td>
<td>4</td>
</tr>
<tr>
<td>Jones</td>
<td>1.4 M</td>
<td>5</td>
</tr>
</tbody>
</table>

http://www.census.gov/genealogy/www/data/2000surnames/
“Where is Joe Liebermen?”

• Spelling mistake
  – Liebermen vs Lieberman

• Need an approximate matching condition!
Levenshtein (or edit) distance

• The minimum number of character edit operations needed to turn one string into the other.

  LIEBERMAN
  LIEBERMEN

  – Substitute A to E. Edit distance = 1
Levenshtein (or edit) distance

• Distance between two string s and t is the shortest sequence of edit commands that transform s to t.

• Commands:
  – Copy character from s to t  (cost = 0)
  – Delete a character from s     (cost = 1)
  – Insert a character into t     (cost = 1)
  – Substitute one character for another (cost = 1)
Levenshtein (or edit) distance

Ashwin Machanavajjhala
Aswhin Maachanavajjhala
Levenshtein (or edit) distance

String s: Ashwin Mahchanavajihala

String t: Ashwhin Mahachanavajghala

Total cost: 4
Computing the edit distance

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>S</th>
<th>W</th>
<th>A</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Cost of changing “G” → “A”

Cost of changing “ASWH” → “AS”
Computing the edit distance

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>S</th>
<th>W</th>
<th>A</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>W</td>
<td>3</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Cost of changing “ASW” → “AS”:

Minimum of:

- Cost of “AS” → “AS” + 1 (delete W)
- Cost of “ASW” → “A” + 1 (insert S)
- Cost of “AS” → “A” + 1 (substitute W with S)
### Computing the edit distance

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>S</th>
<th>W</th>
<th>A</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>A</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>S</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>W</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>H</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>I</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>N</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>
Computing the edit distance

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>S</th>
<th>W</th>
<th>A</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>A</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>S</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>W</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>H</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>I</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>N</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

Remember the minimum in each step and retrace your path.
Edit Distance Variants

• Needleman-Munch
  – Different costs for each operation

• Affine Gap distance
  – John Reed vs John Francis “Jack” Reed
  – Consecutive inserts cost less than the first insert.
Back to our example … Attempt 3

SELECT w.firstname, w.lastname, w.state, w.party,
p.first_name, p.last_name, date_part('year',
current_date) - date_part('year', p.birthday) AS age
FROM wallst w, persons p
WHERE levenshtein(w.lastname, p.last_name) <= 1
and date_part('year', current_date) -
date_part('year', p.birthday) < 100;
Jaccard Distance

• Useful similarity function for sets
  – *(and for… long strings).*

• Let A and B be two sets
  – Words in two documents
  – Friends lists of two individuals

\[
\text{Jaccard}(A, B) = \frac{|A \cap B|}{|A \cup B|}
\]
Jaccard similarity for names

• Use character trigrams

\[ \text{LIEBERMAN} = \{GGL, GLI, LIE, IEB, EBE, BER, ERM, RMA, MAN, ANG, NGG}\]
\[ \text{LIEBERMEN} = \{GGL, GLI, LIE, IEB, EBE, BER, ERM, RMA, MEN, ENG, NGG}\]

\[\text{Jaccard}(s,t) = \frac{9}{13} = 0.69\]
Attempt 4:

SELECT w.firstname, w.lastname, w.state, w.party, p.first_name, p.last_name, date_part('year', current_date) - date_part('year', p.birthday) AS age
FROM wallst w, persons p
WHERE similarity(w.lastname, p.last_name) >= 0.5
  and date_part('year', current_date) - date_part('year', p.birthday) < 100;
Translation / Substitution Tables

• Strings that are usually used interchangeably
  – New York vs Big Apple
  – Thomas vs Tom
  – Robert vs Bob
Attempt 5

select w.firstname, w.lastname, w.state, p.first_name, p.last_name, date_part('year', current_date) - date_part('year', p.birthday) AS age
from wallst w, persons p
where levenshtein(w.lastname, p.last_name) <= 1
and date_part('year', current_date) - date_part('year', p.birthday) < 100
and (w.firstname = p.first_name or w.firstname IN (select n.nickname from nicknames n where n.firstname = p.first_name));
Almost there …

• Tim matches both Timothy and Tim
  – Can fix it by matching on STATE
  – *Homework exercise 😊*
Summary of Similarity Methods

Easiest and most efficient

- Equality on a boolean predicate
- Edit distance
  - Levenstein, Affine
- Set similarity
  - Jaccard
- Vector Based
  - Cosine similarity, TFIDF
- Translation-based
- Numeric distance between values
- Phonetic Similarity
  - Soundex, Metaphone
- Other
  - Jaro-Winkler, Soft-TFIDF, Monge-Elkan
Summary of Similarity Methods

- Equality on a boolean predicate
- Edit distance
  - Levenstein, Affine
- Set similarity
  - Jaccard
- Vector Based
  - Cosine similarity, TFIDF

Used for Text (reviews/tweets), sets, class membership, ...

- Translation-based
- Numeric distance between values
- Phonetic Similarity
  - Soundex, Metaphone
- Other
  - Jaro-Winkler, Soft-TFIDF, Monge-Elkan

Good for Names

Handle Typographical errors

Useful for abbreviations, alternate names.
Evaluating Record Linkage

• Hard to get all the matches to be exactly correct in real world problems
  – As we saw in real examples

• Need to quantify how good the matching is.
Property Testing

• Consider a universe U of objects
  – Documents (in web search)
  – Pairs of records (in record linkage)

• Suppose you want to identify a subset M in U that satisfies a specific property
  – Relevance to a query (in web search)
  – Do the records match (in record linkage)
Property Testing

• Consider a universe $U$ of objects
• Suppose you want to identify a subset $M$ in $U$ that satisfies a specific property

Let $A$ be an (imperfect) algorithm that guesses whether or not an element in $U$ satisfies the property

– Let $M_A$ be the subset of objects that $A$ identifies as satisfying the property.
## Property Testing

### Real World

<table>
<thead>
<tr>
<th>Algorithm Guess</th>
<th>Satisfies P</th>
<th>Doesn’t Satisfy P</th>
</tr>
</thead>
<tbody>
<tr>
<td>Satisfies P</td>
<td><em>True positives (TP)</em></td>
<td><em>False positives (FP)</em></td>
</tr>
<tr>
<td>Doesn’t satisfy P</td>
<td><em>False negatives (FN)</em></td>
<td><em>True negatives (TN)</em></td>
</tr>
</tbody>
</table>

[Crying Wolf!]

\[ \text{M}_A \quad \text{U} - \text{M}_A \]
Venn diagram view

- True positives (TP)
- True negatives (TN)
- False negatives (FN)
- False positives (FP)
Error: Precision / Recall

\[
\text{Precision} = \frac{TP}{(TP + FP)} = \frac{|M \cap M_A|}{|M_A|}
\]

fraction of answers returned by A that are correct

\[
\text{Recall} = \frac{TP}{(TP + FN)} = \frac{|M \cap M_A|}{|M|}
\]

fraction of correct answers that are returned by A
Error: F-measure

\[
\text{Precision} = \frac{|M \cap M_A|}{|M_A|}
\]

\[
\text{Recall} = \frac{|M \cap M_A|}{|M|}
\]

\[
\text{F1 score} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}
\]
Example

- M:

<table>
<thead>
<tr>
<th>firstname</th>
<th>lastname</th>
<th>state</th>
<th>first_name</th>
<th>last_name</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bob</td>
<td>Bennett</td>
<td>UT</td>
<td>Robert</td>
<td>Bennett</td>
<td>81</td>
</tr>
<tr>
<td>Tom</td>
<td>Carper</td>
<td>DE</td>
<td>Thomas</td>
<td>Carper</td>
<td>67</td>
</tr>
<tr>
<td>Mike</td>
<td>Crapo</td>
<td>ID</td>
<td>Michael</td>
<td>Crapo</td>
<td>63</td>
</tr>
<tr>
<td>Chris</td>
<td>Dodd</td>
<td>CT</td>
<td>Christopher</td>
<td>Dodd</td>
<td>70</td>
</tr>
<tr>
<td>Mike</td>
<td>Enzi</td>
<td>WY</td>
<td>Michael</td>
<td>Enzi</td>
<td>70</td>
</tr>
<tr>
<td>Tim</td>
<td>Johnson</td>
<td>SD</td>
<td>Tim</td>
<td>Johnson</td>
<td>68</td>
</tr>
<tr>
<td>Joe</td>
<td>Lieberman</td>
<td>CT</td>
<td>Joseph</td>
<td>Lieberman</td>
<td>72</td>
</tr>
<tr>
<td>Jack</td>
<td>Reed</td>
<td>RI</td>
<td>John</td>
<td>Reed</td>
<td>65</td>
</tr>
<tr>
<td>Charles</td>
<td>Schumer</td>
<td>NY</td>
<td>Charles</td>
<td>Schumer</td>
<td>64</td>
</tr>
<tr>
<td>Richard</td>
<td>Shelby</td>
<td>AL</td>
<td>Richard</td>
<td>Shelby</td>
<td>80</td>
</tr>
</tbody>
</table>

(10 rows)
Example:

Algorithm A:

```sql
select * from wallst w, persons p
where w.lastname = p.last_name and
date_part('year', current_date) - date_part('year', p.birthday) < 100
and (w.firstname = p.first_name or w.firstname IN
(select n.nickname from nicknames n where n.firstname = p.first_name));
```

- Exact match on last name
- Age < 100
- First name is same or a nickname
Example

- $M_A$:

<table>
<thead>
<tr>
<th>firstname</th>
<th>lastname</th>
<th>state</th>
<th>first_name</th>
<th>last_name</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bob</td>
<td>Bennett</td>
<td>UT</td>
<td>Robert</td>
<td>Bennett</td>
<td>81</td>
</tr>
<tr>
<td>Charles</td>
<td>Schumer</td>
<td>NY</td>
<td>Charles</td>
<td>Schumer</td>
<td>64</td>
</tr>
<tr>
<td>Chris</td>
<td>Dodd</td>
<td>CT</td>
<td>Christopher</td>
<td>Dodd</td>
<td>70</td>
</tr>
<tr>
<td>Jack</td>
<td>Reed</td>
<td>RI</td>
<td>John</td>
<td>Reed</td>
<td>65</td>
</tr>
<tr>
<td>Mike</td>
<td>Crapo</td>
<td>ID</td>
<td>Michael</td>
<td>Crapo</td>
<td>63</td>
</tr>
<tr>
<td>Mike</td>
<td>Enzi</td>
<td>WY</td>
<td>Michael</td>
<td>Enzi</td>
<td>70</td>
</tr>
<tr>
<td>Richard</td>
<td>Shelby</td>
<td>AL</td>
<td>Richard</td>
<td>Shelby</td>
<td>80</td>
</tr>
<tr>
<td>Tim</td>
<td>Johnson</td>
<td>SD</td>
<td>Timothy</td>
<td>Johnson</td>
<td>68</td>
</tr>
<tr>
<td>Tim</td>
<td>Johnson</td>
<td>SD</td>
<td>Tim</td>
<td>Johnson</td>
<td>68</td>
</tr>
<tr>
<td>Tom</td>
<td>Carper</td>
<td>DE</td>
<td>Thomas</td>
<td>Carper</td>
<td>67</td>
</tr>
</tbody>
</table>

(10 rows)
Example

Precision = \frac{|M \cap M_A|}{|M_A|} = \frac{9}{10} = 0.9

Recall = \frac{|M \cap M_A|}{|M|} = \frac{9}{10} = 0.9

F1 score = 2 \frac{0.9 \times 0.9}{0.9 + 0.9} = 0.9
Summary

• Many interesting data analyses require reasoning across different datasets

• May not have access to keys that uniquely identify individual rows in both datasets
Summary

• Use combinations of attributes that are approximate keys (or quasi-identifiers)

• Use similarity measures for fuzzy or approximate matching
  – Levenshtein or Edit distance
  – Jaccard Similarity

• Use translation tables
Summary

• Record Linkage is rarely perfect
  – Missing attributes
  – Messy data errors
  – …

• **Precision/Recall** is used to measure the quality of linkage.
The Ugly side of Record Linkage
[Sweeney IJUFKS 2002]

- Name
- SSN
- Visit Date
- Diagnosis
- Procedure
- Medication
- Total Charge
- Zip
- Birth date
- Sex

Medical Data
The Ugly side of Record Linkage
[Sweeney IJUFKS 2002]

Medical Data

- Name
- SSN
- Visit Date
- Diagnosis
- Procedure
- Medication
- Total Charge

Voter List

- Name
- Address
- Date Registered
- Party affiliation
- Date last voted

Governor of MA uniquely identified using ZipCode, Birth Date, and Sex.

Name linked to Diagnosis
The Ugly side of Record Linkage
[Sweeney IJUFKS 2002]

- Name
- SSN
- Visit Date
- Diagnosis
- Procedure
- Medication
- Total Charge

- Name
- Address
- Date Registered
- Party affiliation
- Date last voted

- Zip
- Birth date
- Sex

87% of US population uniquely identified using ZipCode, Birth Date, and Sex.

Quasi Identifier

Medical Data       Voter List