Data matching
(or Duplicate Detection or Record Linkage)

Helena Galhardas
DEI IST

References

- Slides “Data Quality and Data Cleansing” course, Felix Naumann, Winter 2014/15
- “Data Matching”, Peter Christen, Springer.
Problem definition

- Given two relational tables \( R \) and \( S \) with identical schema, we say tuple \( r \) in \( R \) matches a tuple \( s \) in \( S \) if they refer to the same real-world entity
  - Those kind of pairs are called matches
    - We want to find all such matches

Duplicate Detection

\[ R \times S \]

**Similarity measure Algorithm**

- \( \text{sim} > \theta \rightarrow \text{Duplicate} \)
- \( \text{sim} < \theta \rightarrow \text{Non-duplicate} \)
Example

<table>
<thead>
<tr>
<th>Name</th>
<th>SSN</th>
<th>Addr</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jack Lemmon</td>
<td>430-871-8294</td>
<td>Maple St</td>
</tr>
<tr>
<td>Harrison Ford</td>
<td>292-918-2913</td>
<td>Culver Blvd</td>
</tr>
<tr>
<td>Tom Hanks</td>
<td>234-762-1234</td>
<td>Main St</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>SSN</th>
<th>Addr</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tom Hanks</td>
<td>234-162-1234</td>
<td>Main St</td>
</tr>
<tr>
<td>Kevin Spacey</td>
<td>928-184-2813</td>
<td>Frost Blvd</td>
</tr>
<tr>
<td>Jack Lemon</td>
<td>430-817-8294</td>
<td>Maple Street</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

- Find records from different datasets that refer to same entity

Example of origin of duplicates

Original

Scanned
Ironically, “Duplicate Detection” has many Duplicates

- Household matching
- Mixed and split citation problem
- Match
- Deduplication
- Entity resolution
- Identity uncertainty
- Hardening soft databases
- Householding
- Doubles
- Duplicate detection
- Record linkage
- Object identification
- Object consolidation
- Entity clustering
- Reference reconciliation
- Reference matching

Data matching: 1st Challenge

- Match tuples accurately
  - Record-oriented matching: A pair of records with different fields is considered

- Difficult because matches often appear quite differently, due to typing errors, different formatting conventions, abbreviations, etc
- Use string matching algorithms
Apart from the string matching algorithms: domain-dependent similarity measures

- Data Types
  - Special similarity for dates
  - Special similarity for numerical attributes
  - ...

- Matching rules
  - [Hernandez Stolfo 1998]
  - Given two records, \( r_1 \) and \( r_2 \).
    
    \[
    \text{IF } \text{last name of } r_1 = \text{last name of } r_2, \\
    \text{AND first names differ slightly,} \\
    \text{AND address of } r_1 = \text{address of } r_2 \\
    \text{THEN } r_1 \text{ is equivalent to } r_2.
    \]

Record-oriented matching techniques

- Treat each tuple as a string and apply string matching algorithms
- Exploit the structured nature of data – hand-crafted matching rules
- Automatically discover matching rules from training data – supervised learning
- Iteratively assign tuples to clusters, no need of training data – clustering
- Model the matching domain with a probability distribution and reason with the distribution to take matching decisions – probabilistic approaches
- Exploit correlations among tuple pairs to match them all at once – collective matching
Record-oriented matching techniques

- Treat each tuple as a string and apply string matching algorithms
  - Exploit the structured nature of data – **hand-crafted matching rules**
  - Automatically discover matching rules from training data – **supervised learning**
- Iteratively assign tuples to clusters, no need of training data – **clustering**
- Model the matching domain with a probability distribution and reason with the distribution to take matching decisions – **probabilistic approaches**
- Exploit correlations among tuple pairs to match them all at once – **collective matching**

2nd Challenge

- **Efficiently** match a very large amount (tens of millions) of tuples
  - **Record-set oriented matching:** A potentially large set (or two sets) of records needs to be compared
  - Aims at minimizing the number of tuple pairs to be compared and do each of the comparisons efficiently
Record-set oriented matching techniques

- For minimizing the number of tuple pairs to be compared
  - Hashing the tuples into buckets and only match those within a bucket
  - Sorting the tuples using a key and then compare each tuple with only the previous (w-1) tuples, for a pre-defined window size w
  - Index tuples using an inverted index on one attribute, for instance
  - Use a cheap similarity measure to quickly group tuples into overlapping clusters called canopies
  - Use representatives: tuples that represent a cluster of matching tuples against which new tuples are matched
  - Combine the techniques: because using a single heuristic runs the risk of missing tuple pairs that should be matched but are not

- And for minimizing the time taken to match each pair
  - Short-circuiting the matching process – exit immediately if one pair of attributes doesn’t match

Outline

- Record-oriented matching approaches
  - Rule-based matching
  - Learning-based matching

- Record-set oriented matching: scaling up data matching
  - Sorting: Sorted Neighborhood Method (SNM)
  - Variant1: Clustering the data first
  - Variant2: Incremental Merge/Purge

- Measures and data sets
Outline

- Record-oriented matching approaches
  - Rule-based matching
  - Learning-based matching

- Record-set oriented matching: scaling up data matching
  - Sorting: Sorted Neighborhood Method (SNM)
  - Variant1: Clustering the data first
  - Variant2: Incremental Merge/Purge

- Measures and data sets

Rule-based matching

- Hand-crafted matching rules that can be (linearly weighted) combined through:

\[ \text{sim}(x, y) = \sum_{i=1}^{n} \alpha_i \cdot \text{sim}_i(x, y) \]

that returns the similarity score between two tuples \(x\) and \(y\), where:

- \(n\) is the nb attributes in each table \(X\) and \(Y\)
- \(\text{sim}_i(x, y)\) is the similarity score between the \(i\)-th attributes of \(x\) and \(y\)
- \(\alpha_i\) is a pre-specified weight indicating the importance of the \(i\)-th attribute to the total similarity score

- \(\alpha_i\) in \([0,1]\); \(\sum_{i=1}^{n} \alpha_i = 1\)

- If \(\text{sim}(x, y) \geq \beta\) we say tuples \(x\) and \(y\) match
To match names, define a similarity function $\text{sim}_{\text{Name}}(x,y)$ based on the Jaro-Winkler distance.

To match SSNs, define a function $\text{sim}_{\text{SSN}}(x,y)$ based on edit distance, etc.

$$\text{sim}(x,y) = 0.3*\text{sim}_{\text{Name}}(x,y) + 0.3*\text{sim}_{\text{SSN}}(x,y) + 0.2*\text{sim}_{\text{Addr}}(x,y)$$

Complex matching rules (1)

- Linearly weighted matching rules do not work well when encoding more complex matching knowledge.

  - Ex: two persons match if their names match approximately and either the SSN matches exactly or otherwise the addresses must match exactly.

- Modify the similarity functions.

  - Ex: $\text{sim}'_{\text{SSN}}(x,y)$ returns true only if the SSN match exactly; analogous with $\text{sim}'_{\text{Addr}}(x,y)$.

  - And then the matching rule would be:

    If $\text{sim}_{\text{Name}}(x,y) < 0.8$ then return “no match”
    Else if $\text{sim}'_{\text{SSN}}(x,y) = \text{true}$ then return “match”
    Else if $\text{sim}'_{\text{SSN}}(x,y) \geq 0.9$ and $\text{sim}'_{\text{Addr}}(x,y) = \text{true}$ then return “match”
    Else return “no match”
Complex matching rules (2)

- This kind of rules are often written in a **high-level declarative language**
  - Easier to understand, debug, modify and maintain

- Still, it is labor intensive to write good matching rules
- Or not clear at all how to write them
- Or difficult to set the parameters $\alpha$, $\beta$

Learning-based matching

- **Supervised learning**
  - can also be unsupervised (clustering)
- **Idea**: learn a matching model $M$ from the training data, then apply $M$ to match new tuple pairs.

- Training data has the form:
  $$T = \{(x_1, y_1, l_1), (x_2, y_2, l_2), \ldots, (x_n, y_n, l_n)\}$$
  where each triple $(x_i, y_i, l_i)$ consists of a tuple pair $(x_i, y_i)$ and a label $l_i$ with value “yes” if $x_i$ matches $y_i$ and “no” otherwise.
Training (1)

- Define a set of features $f_1, f_2, \ldots, f_m$ thought to be potentially relevant to matching
  - each $f_i$ quantifies one aspect of the domain judged possibly relevant to matching the tuples
  - Each feature $f_i$ is a function that takes a tuple pair $(x, y)$ and produces a numerical, categorical, or binary value.
- The learning algorithm will use the training data to decide which features are in fact relevant

Training (2)

- Convert each training example $(x_i, y_i, l_i)$ in the set $T$ into a pair:
  
  $<(f_1(x_i, y_i), f_2(x_i, y_i), \ldots, f_m(x_i, y_i)), c_i>$

  where $v_i = <f_1(x_i, y_i), f_2(x_i, y_i), \ldots, f_m(x_i, y_i)>$ is a feature vector that encodes the tuple pair $(x_i, y_i)$ in terms of the features and $c_i$ is an appropriately transformed version of label $l_i$
- Training set $T$ is converted into a new training set $T'$:
  
  $\{(v_1, c_1), (v_2, c_2), \ldots, (v_n, c_n)\}$

  and then we apply a learning algorithm such as SVM or Decision Trees to $T'$ to learn a matching model $M$
Matching

- Given a new pair \((x, y)\), transform it into a feature vector
  \[ v = <f_1(x, y), f_2(x, y), \ldots, f_m(x, y)> \]
- And then apply model \(M\) to predict whether \(x\) matches \(y\)

Table X

<table>
<thead>
<tr>
<th>Name</th>
<th>Phone</th>
<th>City</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dave Smith</td>
<td>(608) 395 9462</td>
<td>Madison</td>
<td>WI</td>
</tr>
<tr>
<td>Joe Wilson</td>
<td>(408) 123 4265</td>
<td>San Jose</td>
<td>CA</td>
</tr>
<tr>
<td>Dan Smith</td>
<td>(608) 256 1212</td>
<td>Middleton</td>
<td>WI</td>
</tr>
</tbody>
</table>

Table Y

<table>
<thead>
<tr>
<th>Name</th>
<th>Phone</th>
<th>City</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>David D. Smith</td>
<td>395 9462</td>
<td>Madison</td>
<td>WI</td>
</tr>
<tr>
<td>Daniel W. Smith</td>
<td>256 1212</td>
<td>Madison</td>
<td>WI</td>
</tr>
</tbody>
</table>

Goal: learn a linearly weighted rule to match \(x\) and \(y\)

\[
\text{sim}(x, y) = \sum_{i=1}^{n} \alpha_i \cdot \text{sim}_i(x, y)
\]
Training data

\[ \langle x_1 = (\text{Mike Williams}, (425) 247 4893, \text{Seattle, WA}), y_1 = (M. Williams, 247 4893, \text{Redmond, WA}), \text{yes} \rangle \]

\[ \langle x_2 = (\text{Richard Pike}, (414) 256 1257, \text{Milwaukee, WI}), y_2 = (R. Pike, 256 1237, \text{Milwaukee, WI}), \text{yes} \rangle \]

\[ \langle x_3 = (\text{Jane McCain}, (206) 111 4215, \text{Renton, WA}), y_3 = (J.M. McCain, 112 5200, \text{Renton, WA}), \text{no} \rangle \]

- Consider 6 possibly relevant features:
  - \( f_1(x, y) \) and \( f_2(x, y) \): Jaro-Winkler and edit distance between person names of tuples \( x \) and \( y \)
  - \( f_3(x, y) \): edit distance between phone numbers, ignoring the area code
  - \( f_4(x, y) \) and \( f_5(x, y) \): returns 1 if the city names and the state names match exactly
  - \( f_6(x, y) \) returns 1 if the area code of \( x \) is an area code of the city of \( y \)

Transforming training data and learn

\[ \langle v_1, c_1 \rangle = \langle [f_1(x_1, y_1), f_2(x_1, y_1), f_3(x_1, y_1), f_4(x_1, y_1), f_5(x_1, y_1), f_6(x_1, y_1)], 1 \rangle \]

\[ \langle v_2, c_2 \rangle = \langle [f_1(x_2, y_2), f_2(x_2, y_2), f_3(x_2, y_2), f_4(x_2, y_2), f_5(x_2, y_2), f_6(x_2, y_2)], 1 \rangle \]

\[ \langle v_3, c_3 \rangle = \langle [f_1(x_3, y_3), f_2(x_3, y_3), f_3(x_3, y_3), f_4(x_3, y_3), f_5(x_3, y_3), f_6(x_3, y_3)], 0 \rangle \]

- Goal: learn the weight \( \alpha_i \) with \( i \) in \([1, 6]\) that gives a linearly weighted matching rule of the form: \( \text{sim}(x, y) = \sum_{i=1}^{6} \alpha_i \cdot f_i(x, y) \)
- Perform a least-squares linear regression on the transformed data set for finding the weights \( \alpha_i \) that minimize the squared error:
  \[ \sum_{i=1}^{3} (c_i - \sum_{j=1}^{6} \alpha_j \cdot f_j(v_i))^2 \]
  where \( c_i \) is the label associated with feature vector \( v_i \) and \( f_j(v_i) \) is the \( j \)-th element of feature vector \( v_i \)
- Learn \( \beta \) from the training set by setting it to the value that lets us minimize the number of incorrect matching predictions.
Advantages/inconveniences supervised learning

**Advantages:**
- Can automatically examine a large set of features to select the most useful ones
- Can construct very complex rules, very difficult to construct in rule-based learning

**Inconveniences:**
- Requires a large number of training examples which can be labor intensive to obtain

Outline

- Matching approaches
  - Rule-based matching
  - Learning-based matching

  - **Scaling up data matching**
    - Sorting: Sorted Neighborhood Method (SNM)
    - Incremental Merge/Purge

- Measures and data sets
  - Recall, precision, F-measure
Record Pairs as Matrix

Number of comparisons: All pairs

400 comparisons
Reflexivity of Similarity

Symmetry of Similarity
Complexity

- **Problem**: Too many comparisons!
  - 10,000 customers => 49,995,000 comparisons
  - \((n^2 - n) / 2\)
  - Each comparison is already expensive.

- **Idea**: Avoid comparisons…
  - … by filtering out individual records.
  - … by partitioning the records and comparing only within a partition.

Partitioning / Blocking

- **Partition** the records (horizontally) and compare pairs of records only within a partition
  - Ex1: Partitioning by first two zip-digits
    - Ca. 100 partitions in Germany
    - Ca. 100 customers per partition
    - => 495,000 comparisons
  - Ex2: Partition by first letter of surname
  - …

- **Idea**: Partition multiple times by different criteria
  - Then apply **transitive closure** on discovered duplicates.
Records sorted by ZIP

Blocking by ZIP
Record-set oriented matching techniques (recap.)

- For minimizing the number of tuple pairs to be compared
  - Hashing: the tuples into buckets and only match those within a bucket
  - Sorting: the tuples using a key and then compare each tuple with only the previous \((w-1)\) tuples, for a pre-defined window size \(w\)
  - Index: tuples using an inverted index on one attribute, for instance
  - Use a cheap similarity measure to quickly group tuples into overlapping clusters called canopies
  - Use representatives: tuples that represent a cluster of matching tuples against which new tuples are matched
  - Combine the techniques: because using a single heuristic runs the risk of missing tuple pairs that should be matched but are not

- And for minimizing the time taken to match each pair
  - Short-circuiting: the matching process – exit immediately if one pair of attributes doesn’t match

Sorted Neighbourhood Method - SNM (or Windowing)

- **Concatenate** all records to be matched in a single file (or table)
- **Sort** the records using a pre-defined key based on the values of the attributes for each record
- **Move** a window of a specific size \(w\) over the file, comparing only the records that belong to this window
Sorted Neighborhood Method in detail

1. **Create Key**: Compute a key $K$ for each record in the list by extracting relevant fields or portions of fields.
   - Relevance is decided by experts.
2. **Sort Data**: Sort the records in the data list using $K$
3. **Merge**: Move a fixed size window through the sequential list of records limiting the comparisons for matching records to those records in the window. If the size of the window is $w$ records, then every new record entering the window is compared with the previous records to find “matching” records.

1. **Create key**

- Compute a key for each record by extracting relevant fields or portions of fields

Example:

<table>
<thead>
<tr>
<th>First</th>
<th>Last</th>
<th>Address</th>
<th>ID</th>
<th>Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sal</td>
<td>Stolfo</td>
<td>123 First Street</td>
<td>45678987</td>
<td>STLSAL123FRST456</td>
</tr>
</tbody>
</table>
2. Sort Data

- Sort the records in the data list using the key in step 1
- This can be very time consuming
  - O(NlogN) for a good algorithm,
  - O(N^2) for a bad algorithm

3. Merge records

- Move a fixed size window through the sequential list of records.
- This limits the comparisons to the records in the window
- To compare each pair of records, a set of complex rules (called equational theory) is applied
Considerations

- What is the **optimal window size** while
  - Maximizing accuracy
  - Minimizing computational cost
- The effectiveness of the SNM highly depends on the **key** selected to sort the records
  - A key is defined to be a sequence of a subset of attributes
  - Keys must provide sufficient discriminating power

---

Example of Records and Keys

<table>
<thead>
<tr>
<th>First</th>
<th>Last</th>
<th>Address</th>
<th>ID</th>
<th>Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sal</td>
<td>Stolfo</td>
<td>123 First Street</td>
<td>45678987</td>
<td>STLSAL123FRST456</td>
</tr>
<tr>
<td>Sal</td>
<td>Stolfo</td>
<td>123 First Street</td>
<td>45678987</td>
<td>STLSAL123FRST456</td>
</tr>
<tr>
<td>Sal</td>
<td>Stolfo</td>
<td>123 First Street</td>
<td>45678987</td>
<td>STLSAL123FRST456</td>
</tr>
<tr>
<td>Sal</td>
<td>Stiles</td>
<td>123 Forest Street</td>
<td>45654321</td>
<td>STLSAL123FRST456</td>
</tr>
</tbody>
</table>
Equational Theory (record matching rules)

- The comparison during the merge phase is an inferential process
- Compares much more information than simply the key
- The more information there is, the better inferences can be made

Equational Theory - Example

- Two names are spelled nearly identically and have the same address
  - It may be inferred that they are the same person
- Two social security numbers are the same but the names and addresses are totally different
  - Could be the same person who moved
  - Could be two different people and there is an error in the social security number
A simplified rule in English

Given two records, r1 and r2
IF the last name of r1 equals the last name of r2,
AND the first names differ slightly,
AND the address of r1 equals the address of r2
THEN
r1 is equivalent to r2

Examples of matched records

<table>
<thead>
<tr>
<th>SSN</th>
<th>Name (First, Initial, Last)</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>334600443</td>
<td>Lisa Boardman</td>
<td>144 Wars St.</td>
</tr>
<tr>
<td>334600443</td>
<td>Lisa Brown</td>
<td>144 Ward St.</td>
</tr>
<tr>
<td>525520001</td>
<td>Ramon Bonilla</td>
<td>38 Ward St.</td>
</tr>
<tr>
<td>525520001</td>
<td>Raymond Bonilla</td>
<td>38 Ward St.</td>
</tr>
<tr>
<td>0</td>
<td>Diana D. Ambrosion</td>
<td>40 Brik Church Av.</td>
</tr>
<tr>
<td>0</td>
<td>Diana A. Dambrosion</td>
<td>40 Brick Church Av.</td>
</tr>
<tr>
<td>789912345</td>
<td>Kathi Kason</td>
<td>48 North St.</td>
</tr>
<tr>
<td>879912345</td>
<td>Kathy Kason</td>
<td>48 North St.</td>
</tr>
<tr>
<td>879912345</td>
<td>Kathy Smith</td>
<td>48 North St.</td>
</tr>
</tbody>
</table>
Building an equational theory

- The process of creating a **good equational theory** is similar to the process of creating a good knowledge-base for an expert system.
- In complex problems, an **expert’s assistance** is needed to write the equational theory.

Looses some matching pairs

- In general, **no single pass** (i.e. no single key) will be sufficient to catch all matching records.
- An attribute that appears first in the key has higher discriminating power than those appearing after them.
  - If an employee has two records in a DB with SSN 193456782 and 913456782, it’s unlikely they will fall under the same window.
Possible solutions

- **Goal**: To increase the number of similar records being matched

- Widen the scanning window size, \( w \)
- Execute several independent runs of the SNM
  - Use a different key each time
  - Use a relatively small window
  - Call this the Multi-Pass approach

Multi-pass approach

- Each independent run of the Multi-Pass approach will produce a set of pairs of records
  - Although one field in a record may be in error, another field may not

- **Transitive closure** can be applied to those pairs to be merged
Example of multi-pass matches

**Pass 1 (Lastname discriminates)**
KSNKAT48NRTH789 (Kathi Kason 789912345)
KSNKAT48NRTH879 (Kathy Kason 879912345)

**Pass 2 (Firstname discriminates)**
KATKSN48NRTH789 (Kathi Kason 789912345)
KATKSN48NRTH879 (Kathy Kason 879912345)

**Pass 3 (Address discriminates)**
48NRTH879KSNKAT (Kathy Kason 879912345)
48NRTH879SMTKAT (Kathy Smith 879912345)

---

Transitive closure example

IF A similar to B
AND B similar to C
THEN A similar to C

From the example:

789912345 Kathi Kason 48 North St. (A)
879912345 Kathy Kason 48 North St. (B)
879912345 Kathy Smith 48 North St. (C)
Variant of SNM: Clustering the data first

- We do not need a completely sorted database

- Partition the data into independent subsets such that matching records appear in each cluster

- Apply the SNJ to each individual cluster independently

Clustering the data first in detail

1. **Cluster Data**: Extract an n-attribute key $K$ and map it into an n-dimensional cluster.

2. **Sorted-Neighborhood Method**: Apply the Sorted-Neighborhood Method independently on each cluster using $K$. 
1. Cluster Data

Example:

<table>
<thead>
<tr>
<th>First</th>
<th>Last</th>
<th>Address</th>
<th>ID</th>
<th>Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lisa</td>
<td>Board</td>
<td>11 Warsh Street</td>
<td>334600443</td>
<td>33LIBO11</td>
</tr>
<tr>
<td>Michael</td>
<td>Bond</td>
<td>38 Ward Street</td>
<td>525500443</td>
<td>52MIBO38</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
<tr>
<td>Lisa</td>
<td>Board</td>
<td>11 Wars Street</td>
<td>334600443</td>
<td>33LIBO11</td>
</tr>
<tr>
<td>Michele</td>
<td>Bond</td>
<td>38 Ward St.</td>
<td>525250001</td>
<td>52MIBO38</td>
</tr>
</tbody>
</table>

Cluster 2

<table>
<thead>
<tr>
<th>First</th>
<th>Last</th>
<th>Address</th>
<th>ID</th>
<th>Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>Michael</td>
<td>Bond</td>
<td>38 Ward Street</td>
<td>525520001</td>
<td>52MIBO38</td>
</tr>
<tr>
<td>Michele</td>
<td>Bond</td>
<td>38 Ward St.</td>
<td>525250001</td>
<td>52MIBO38</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>

2. Sorted-Neighborhood Method

- Apply the SNJ on each cluster independently

- The key extracted to cluster the data can be used to sort

- Ideally: a cluster must be main memory-based when analyzed
Another SNM variant: Incremental Merge/Purge

- Lists of records are concatenated for first time processing
- Concatenating new data records before reapplying the merge/purge process may be very expensive in both time and space
- An incremental merge/purge approach is needed: Prime Representatives method

Prime-Representative (PR)

- **Prime-Representative**: set of records extracted from each cluster of records to represent the information in that cluster
- Initially, no Prime-Representative exists
- After the execution of the first merge/purge, clusters of similar records are created
- Correct selection of PR from cluster impacts accuracy of results
Some strategies for choosing PR

- **Random Sample**
  - Select a sample of records at random from each cluster

- **N-Latest**
  - Most recent elements entered in DB

- **Syntactic**
  - Choose the largest or more complete record

Outline

- **Matching approaches**
  - Rule-based matching
  - Learning-based matching

- **Scaling up data matching**
  - Sorting: Sorted Neighborhood Method (SNM)
  - Incremental Merge/Purge

  - **Measures and data sets**
    - Recall, precision, F-measure
Precision & Recall

**True positives (TP):** Correctly declared duplicates
**False positives (FP):** Incorrectly declared duplicates
**True negatives (TN):** Correctly avoided pairs
**False negatives (FN):** Missed duplicates

- **Precision** = \( \frac{TP}{TP + FP} \)
  - = TP / declared duplicates
  - Proportion of found matches that are correct
  - Correctness
- **Recall** = \( \frac{TP}{TP + FN} \)
  - = TP / all duplicates
  - Proportion of correct matches that are found
  - Completeness

**F1-Measure** = \( \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \)
Typical recall-precision graph

Typical F1-measure Graph

**F1-measure**
- Harmonic mean of recall and precision
- Weighted mean of recall and precision

From Creating probabilistic databases from duplicated data
Oktie Hassanzadeh · Renée J. Miller (VLDBJ)
Data sets to evaluate deduplication

- **Requirements**
  - Real-world application
  - Interestingly large
  - Interestingly dirty
  - Gold standard
  - Publicly available
  - (Relational)
  - (Understandable)

- **Hardly available**
  - Privacy issues
  - Security issues
  - Embarrassment
  - Data fiefdoms

Small datasets with gold standard

- **CORA**
  - 1878 bibliographic references in XML format
    - [http://www.hpi.uni-potsdam.de/naumann/projekte/repeatability/datasets/cora_dataset.html](http://www.hpi.uni-potsdam.de/naumann/projekte/repeatability/datasets/cora_dataset.html)

- **DBLP**
  - 50,000 bibliographic references in XML format
    - [http://www.hpi.uni-potsdam.de/naumann/projekte/repeatability/datasets/dblp_dataset.html](http://www.hpi.uni-potsdam.de/naumann/projekte/repeatability/datasets/dblp_dataset.html)

- **Restaurants**
  - 864 restaurants with 112 duplicates
    - [http://www.cs.utexas.edu/users/ml/ridge/data.html](http://www.cs.utexas.edu/users/ml/ridge/data.html)

- **Whirl datasets**
  - 11 smaller datasets with a single string attribute
Large datasets without gold standard

- **Places**
  - 1.4 million POIs from Facebook, Gowalla, Foursquare

- **WheelMap**
  - 120,000 places/things in Germany

- **FreeDB**
  - 1.9 million CDs, dirty, some duplicate clusters quite large
  - derived: [http://www.hpi.uni-potsdam.de/naumann/projekte/repeatability/datasets/cd_datasets.html](http://www.hpi.uni-potsdam.de/naumann/projekte/repeatability/datasets/cd_datasets.html)

- **CITESEERX**
  - 1.3 million publications in CSV
  - [http://asterix.ics.ucr.edu/data/cs-x.raw.txt.gz](http://asterix.ics.ucr.edu/data/cs-x.raw.txt.gz)

http://hpi.de/naumann/projects/repeatability/datasets.html

Data generation

- For lack of gold standard: create one
- **Database**
  - Real-world data sets (without or without enough duplicates)
  - Real-world values (from dictionaries)
  - Synthetic strings
- **Data corruption**: Duplicate and modify some percentage of tuples
  - Duplication: Cluster sizes?
  - Data values
    - Insert/remove/transpose/change certain letters
    - Delete values
    - Swap values (within tuple, from dictionary, across tuples)
- General suspicion: Similarity measure and candidate selection is geared towards known types of errors.
Data generators

- **UIS Database Generator**
  - Generates a list of randomly perturbed names and US mailing addresses.
  - Written by Mauricio Hernández.

- **FEBRL-Generator**
  - Part of a cleansing suite
  - Dictionaries with frequencies

- **Dirty XML Generator**
  - [http://www.hpi.uni-potsdam.de/naumann/projekte/completed_projects/dirtyxml.html](http://www.hpi.uni-potsdam.de/naumann/projekte/completed_projects/dirtyxml.html)

Next Lecture

- **Data Fusion**