Web Data Integration

Identity Resolution
Outline

1. Introduction
2. Entity Matching
3. Blocking
4. Evaluation
5. Similarity Measures – In Detail
6. Learning Matching Models
7. References
1. Introduction

**Goal of Identity Resolution:** Find all records in all data sources that refer to the same real-world entity („synonymous“ records).

- The problem appears whenever
  1. data from multiple sources is combined.
  2. a single data source should be cleaned (de-duplicated).

- Various commercial tools are available, especially for CRM domain.
Example Use Case: E-Commerce

[Image: Google search results for iPhone 5 smartphone]
Negative Effects of Duplicates within a Data Source

1. Unnecessary memory and processing power consumption

2. Queries give you wrong results
   - Number of customers ≠ SELECT COUNT(*) FROM customer
   - turnover ≠ SELECT SUM(sales value) FROM sales

3. You just see parts and not the whole
   - wrong assessment of customer value for CRM
   - customers that exceed credit limits are not recognized
   - multiple mailings of same catalog to same household
   - quantity discounts are not used when ordering from a supplier
   - …
Ironically, “Identity Resolution” has many Synonyms

- Duplicate detection
- Record linkage
- Household matching
- Mixed and split citation problem
- Object identification
- Match
- Deduplication
- Reference matching
- Object consolidation
- Entity resolution
- Fuzzy match
- Object consolidation
- Identity uncertainty
- Approximate match
- Entity clustering
- Hardening soft databases
- Merge/purge
- Reference reconciliation
- Household matching
- Doubles
The Two Central Challenges of Identity Resolution

- **Challenge 1**: Representations of the same real-world entity are not identical.
  - *Fuzzy duplicates*
- **Solution**: Similarity measures
  - Compare multiple attributes of the records using attribute-specific similarity measures.

<table>
<thead>
<tr>
<th>Name</th>
<th>Date</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chris Miller</td>
<td>12/20/1982</td>
<td>Bardon Street, Melville</td>
</tr>
<tr>
<td>Christian Miller</td>
<td>2/20/1982</td>
<td>7 Bardon St., Melville</td>
</tr>
<tr>
<td>Chris Miller</td>
<td>12/14/1973</td>
<td>Bardon St., Madison</td>
</tr>
</tbody>
</table>

- **Questions**:
  - What is the right similarity measure for each attribute?
  - How should the similarity values of multiple attributes be combined/weighted?
A Wide Range of Similarity Measures exists

- **Edit-based**
  - Damerau-Levenshtein
  - Levenshtein
  - Smith-Waterman
  - Smith-Waterman-Gotoh

- **Token-based**
  - Jaccard
  - Dice
  - Cosine Similarity
  - Soft TF-IDF
  - Monge-Elkan

- **Hybrid**
  - Monge-Elkan
  - Soft TF-IDF

- **Datatype-specific**
  - Numerical Attributes
  - Dates Times
  - Geo Coordinates
  - Soundex
  - Kölner Phonetik

- **Phonetic**
  - Metaphone
  - Double Metaphone
The Two central Challenges of Identity Resolution

- **Challenge 2**: Data sets are large.
  - Quadratic complexity: Comparing every pair of records is too expensive.
- **Solution**: Blocking Algorithms
  - E.g., avoid “unnecessary” comparisons by partitioning.
2. Entity Matching

Challenge 1: Representations of the same real-world entity are not identical.
2.1 Linearly Weighted Combination Rules

- Compute the sim score between tuples x and y as a linearly weighted combination of individual sim scores
  \[ \text{sim}(x,y) = \sum_{i=1}^{n} \alpha_i * \text{sim}_i(x, y) \]
  - n is number of attributes in each table
  - \( \text{sim}_i(x,y) \) is a sim score between the i-th attributes of x and y
  - \( \alpha_i \in [0,1] \) is a pre-specified weight that indicates the important of the i-th attribute to \( \text{sim}(x,y) \), such that \( \sum_{i=1}^{n} \alpha_i = 1 \)

- We declare x and y matched if \( \text{sim}(x,y) \geq \beta \) for a pre-specified \( \beta \), and not matched otherwise
  - in another variation: declare x and y matched if \( \text{sim}(x,y) \geq \beta \), not matched if \( \text{sim}(x,y) < \gamma \) and subject to human review if \( \text{sim}(x,y) \) in between.
Example

\[ \text{sim}(x,y) = 0.3s_{\text{name}}(x,y) + 0.3s_{\text{phone}}(x,y) + 0.1s_{\text{city}}(x,y) + 0.3s_{\text{state}}(x,y) \]

- \( s_{\text{name}}(x,y) \): using the Jaro-Winkler similarity function
- \( s_{\text{phone}}(x,y) \): based on edit distance between x’s phone (after removing area code) and y’s phone
- \( s_{\text{city}}(x,y) \): based on edit distance
- \( s_{\text{state}}(x,y) \): based on exact match; yes \( \Rightarrow 1 \), no \( \Rightarrow 0 \)
2.2 More Complex Rules

- Appropriate when we have and want to encode more complex matching knowledge.

- Example 1: Two persons match if names match approximately and either phones match exactly or addresses match exactly
  1. If $s_{\text{name}}(x,y) < 0.8$ then return “not matched”
  2. Otherwise if $e_{\text{phone}}(x,y) = \text{true}$ then return “matched”
  3. Otherwise if $e_{\text{city}}(x,y) = \text{true}$ and $e_{\text{state}}(x,y) = \text{true}$ then return “matched”
  4. Otherwise return “not matched”

- Example 2: Two genes match if their names match approximately and any of the different gene identifiers matches closely (deals with missing values)
  - If $s_{\text{name}}(x,y) > 0.7$ and
  - $\max (s_{\text{genID}}(x,y), s_{\text{componentID}}(x,y), s_{\text{structureID}}(x,y)) > 0.9$
  - then return “matched”
2.3 Data Gathering for Matching

- Idea: Not only values of the records, but values of related records are relevant for the similarity computation.
  - Movies: actors
  - CDs: songs
  - Persons: spouse, children, employer
  - Customers: orders, addresses

<table>
<thead>
<tr>
<th>ID</th>
<th>Country</th>
<th>Street</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>USA</td>
<td>First Ave.</td>
</tr>
<tr>
<td>2</td>
<td>United States</td>
<td>High St.</td>
</tr>
<tr>
<td>3</td>
<td>United States</td>
<td>Broadway</td>
</tr>
<tr>
<td>4</td>
<td>Los Angeles</td>
<td>Embarcadero</td>
</tr>
<tr>
<td>5</td>
<td>New York</td>
<td>Broadway</td>
</tr>
<tr>
<td>6</td>
<td>Los Angeles</td>
<td>Second St.</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>P St.</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>Pennsylvania Av</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>Sunset Blvd</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>Santa Monica St</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td>Ocean Ave.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>City</th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>New York</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Los Angeles</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>New York</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>Los Angeles</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>New York</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>Los Angeles</td>
<td>3</td>
</tr>
</tbody>
</table>
Example: Matching Films

![Graph showing precision against recall for matching films with and without actors. The graph has two lines: one in red marked "without actors" and one in green marked "with actors." The graph shows a steep decrease in precision for both lines as recall increases, with the line for "with actors" being consistently above the line for "without actors."]
2.4 Data Preprocessing for Matching

In order to enable similarity measures to compute reliable scores, the data needs to be normalized.

- Normalize spelling
  - lower case everything: Müller and müller \(\rightarrow\) müller
  - remove punctuation: U.S.A \(\rightarrow\) usa

- Remove stopwords
  - The Netherlands \(\rightarrow\) netherlands

- Normalize value formats and units of measurement
  - +49 621 181 2677 and (621) 181 2677 \(\rightarrow\) 496211812677
  - 1000 MB and 1 GB \(\rightarrow\) 1000 MB

- Normalize abbreviations and synonyms/surface forms
  - Inc. \(\rightarrow\) Incorporated, Mr. \(\rightarrow\) Mister, USA \(\rightarrow\) United State of America
  - using domain-specific lists of abbreviations and synonyms/surface forms
Parsing and Normalization Services

- **Parsing**
  - Extract attribute values from compound descriptions or titles and compare the values afterwards.
    - Using regular expressions or attribute specific extractors (e.g. using list of all brands)
  - Often required for product data or postal addresses:
    - Apple MacBook Air MC968/A 11.6-Inch Laptop
    - Apple MacBook Air 11-in, Intel Core i5 1.60GHz, 64 GB, Lion 10.7

- **Normalization using External Services**
  - **Geocoding**
    - Translate addresses into geo-coordinates and compare coordinates afterwards.
    - e.g. using Google Geocoding API
  - **Translation into Target Language**
    - Mannheim vs. מנהיים
    - e.g. using Google Translate API or other translation software

Example: Complex Matching Rule including Preprocessing
2.5 Local versus Global Matching

- **Input:** A matrix containing entity similarities
- **Output:** A set of entity pairs that should be considered as matches

- **Local Matching**
  - Consider all pairs above threshold as matches.
  - Implies: One entity can be matched with several other entities.
  - Makes sense for duplicate detection within single data source.

- **Global Matching**
  - Enforce constraint that one entity in data set A should only be matched to one entity in data set B.
  - Makes sense for distinct data sources that do not contain duplicates.
  - Approaches:
    1. Bipartite pairs with the maximal sum of their similarity values
    2. Stable marriage (see Chapter: Schema Mapping)
Summary: The Entity Matching Process

1. Gather Data for Matching
2. Normalize Attribute Values
3. Apply Attribute-specific Similarity Measures
4. Combine Similarity Scores
5. Decide Match/Non-Match
3. Blocking

- Real world data sets are often large.
- **Problem**: Quadratic complexity of matching process.
  - Comparison of every pair of records is too expensive:
    - 100 customers $\Rightarrow$ 10,000 comparisons
    - 10,000 customers $\Rightarrow$ 100 million comparisons
    - 100,000 customers $\Rightarrow$ 10 billion comparisons
  - Each comparison itself is also expensive as it involves calculating various similarity scores
    - Calculation of a string similarity score often has quadratic complexity itself.
- **Solution**: Reduce number of comparisons by
  - avoiding unnecessary comparisons (next 3 slides)
  - applying Blocking Methods that further reduce the number
    - Negative effect: matches might be overlooked 😞
Number of comparisons: All pairs

Complexity: $n^2$

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

20 records ➞ 400 comparisons
Reflexivity of Similarity

Complexity: $n^2 - n$

Similarity is reflexive: $\text{sim}(x, x) = 1$

380 comparisons
Symmetry of Similarity

Complexity: \( \frac{n^2 - n}{2} \)

|   | 1   | 2   | 3   | 4   | 5   | 6   | 7   | 8   | 9   | 10  | 11  | 12  | 13  | 14  | 15  | 16  | 17  | 18  | 19  | 20  |
|---|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 1 |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 2 |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 3 |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 4 |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 5 |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 6 |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 7 |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 8 |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 9 |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 10|     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 11|     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 12|     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 13|     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 14|     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 15|     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 16|     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 17|     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 18|     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 19|     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 20|     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |

Similarity is symmetric:
\( \text{sim}(x,y) = \text{sim}(y,x) \)

190 comparisons

Still quadratic 😞
3.1 Standard Blocking

Idea: Reduce number of comparisons by partitioning the records into buckets and compare only records within each bucket.

- Examples:
  - Partition customers by first two digits of zip code
    - results in about 100 partitions for Germany
    - given about 100 customers per partition
    - \( \Rightarrow \) 495,000 comparisons instead of 49,995,000
  - Partition books by publisher
  - Partition people by first two characters of surname
- Blocking is also called hashing or partitioning

Source: wikipedia.de
### Blocking by ZIP

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

32 comparisons
Choice of the Blocking Key

- The reduction ratio depends on effectiveness of blocking key.
  - High: If records are equally distributed over buckets.
  - Low: If majority of the records end up in one bucket.
    - Example: 90% of all customers are from Mannheim.

- Possible Solution:
  - Use combination of different attributes as blocking key.
  - Example, based on the assumption that there are less errors in first letters:

<table>
<thead>
<tr>
<th>FirstName</th>
<th>Name</th>
<th>Adresse</th>
<th>ID</th>
<th>Blocking Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sal</td>
<td>Stolpho</td>
<td>123 First St.</td>
<td>456780</td>
<td>STOSAL</td>
</tr>
<tr>
<td>Mauricio</td>
<td>Hernandez</td>
<td>321 Second Ave</td>
<td>123456</td>
<td>HERMAU</td>
</tr>
</tbody>
</table>
Problems of Blocking

1. **Recall:** Blocking might miss matches because actually matching pairs are not compared as their blocking key values differ.
   - Partition by city: What if customer has moved?
   - Partition by first letter of name: What if already typo in this letter?
   - Possible solution: Use phonetic encoding like Soundex to deal with typos
     - Example: PAUL $\rightarrow$ P400 and PUAL $\rightarrow$ P400
     - Phonetic encodings will be covered later in Section on Similarity Measures

2. **Efficiency:** The size of the buckets may vary widely.
   - Reduction ratio depends on value distribution within the data set.
   - Potential solution: Build sub-buckets using a second blocking attribute
     - e.g., hash houses into buckets using zip codes,
     - then hash houses within each bucket using street names

   - Both problems are eased by the Sorted Neighborhood Method.
3.2 The Sorted Neighborhood Method

Idea: Sort records so that similar records are close to each other. Only compare records within a small neighborhood (window).

1. Generate Key
   - E.g.: Social Security Number + First 3 letters of name + ...

2. Sort by Key
   - So that similar records end up close to each other.

3. Slide window over sorted records
   - match each record with only the previous \((w-1)\) records, where \(w\) is a pre-specified window size.

<table>
<thead>
<tr>
<th>FirstName</th>
<th>Surname</th>
<th>Address</th>
<th>SSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mauricio</td>
<td>Hernandez</td>
<td>321 Second Ave</td>
<td>123456</td>
</tr>
<tr>
<td>All</td>
<td>Stolpho</td>
<td>123 First St.</td>
<td>456780</td>
</tr>
<tr>
<td>Sal</td>
<td>Stolpho</td>
<td>123 First St.</td>
<td>456780</td>
</tr>
<tr>
<td>Sal</td>
<td>Stolfo</td>
<td>123 First Street</td>
<td>456789</td>
</tr>
</tbody>
</table>
Window size = 4

Complexity:
1. Key generation: $O(n)$
2. Sorting: $O(n \cdot \log(n))$
3. Comparisons: $O(n \cdot w)$

54 comparisons
Challenges when applying the SNM

- **Choice of the Blocking Key**
  - SNM assumes that records that are likely to match fall within the window
  - Thus, key should be strongly “discriminative” and bring together records that are likely to match, and pushes apart records that are not
    - example keys: social sec, student ID, last name, soundex value of last name.

- **Choice of Window Size**
  - Depends on the types and frequency of the errors/typos in the data.
  - Practical experience: $w = 20$ often a good compromise.

- **Workaround: Use Multi-Pass Approach**
  1. Run SNM several times with different keys.
    - Use simple keys and a small $w$
  2. Merge sets of matches found in each run.
    - Less efficient, but much more effective than single-pass.
4. Evaluation
Precision & Recall

- **Ground Truth** for the evaluation: Manually label a set of pairs as matches or non-matches.
  - True positives (TP): Correctly discovered duplicates
  - False positives (FP): Incorrectly discovered duplicates
  - True negatives (TN): Correctly avoided pairs
  - False negatives (FN): Missed duplicates

- **Precision** = $\frac{TP}{TP + FP}$
  - discovered correct matches / declared matches
  - fraction of declared matches that are correct
  - Correctness

- **Recall** = $\frac{TP}{TP + FN}$
  - discovered correct matches / all matches
  - fraction of all correct matches that are found
  - Completeness
Precision & Recall

True matches

Declared matches

All pairs

False negatives

True positives

False positives

True negatives

Accuracy is not a good measure true negatives usually dominate overall result.

Precision = \frac{\text{True positives}}{\text{Declared matches}}

Recall = \frac{\text{True positives}}{\text{True matches}}

F-Measure = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}
Harmonic mean („F-Measure“) emphasizes the importance of small values, whereas arithmetic mean („Average“) is affected more by outliers that are unusually large.

\[ z = \frac{1}{2} (x + y) \]

\[ z = \frac{2 (x \cdot y)}{(x + y)} \]
F-Measure Graph

Optimal Threshold
Efficiency Measures

- Besides of the quality of the similarity measure, the quality of the blocking method is also important.

- Solution 1: Runtime measurements
  - But: Different hardware, difficult repeatability

- Solution 2: Measure how well/poor the blocking method filters the candidates.
  - How many true positives are missed?

- Reduction Ratio = \( \frac{\text{pairs}_{\text{withoutBlocking}}}{\text{pairs}_{\text{afterBlocking}}} \)

- Pairs Completeness = \( \frac{\text{TP}}{\text{FN} + \text{TP}} = \frac{\text{matches}_{\text{afterBlocking}}}{\text{all matches}} \)

- Pairs Quality = \( \frac{\text{TP}}{\text{TP} + \text{TN}} = \frac{\text{matches}_{\text{afterBlocking}}}{\text{all pairs}_{\text{selectedByBlocking}}} \)
Evaluating Identity Resolution

Precision → similarity threshold → Recall

similarity measure

Efficiency

partition/window size
Standard Evaluation Datasets

Different matching methods should be evaluated using the same datasets in order to make the results comparable.

1. DBLP-ACM-Scholar, Amazon-Google Products Datasets

<table>
<thead>
<tr>
<th>Match task</th>
<th>Source size (#entities)</th>
<th>Mapping size (#correspondences)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Source 1</td>
<td>Source 2</td>
</tr>
<tr>
<td>Bibliographic</td>
<td>DBLP-ACM</td>
<td>2,616</td>
</tr>
<tr>
<td></td>
<td>DBLP-Scholar</td>
<td>2,616</td>
</tr>
<tr>
<td>E-commerce</td>
<td>Amazon- GoogleProducts</td>
<td>1,363</td>
</tr>
<tr>
<td></td>
<td>Abt-Buy</td>
<td>1,081</td>
</tr>
</tbody>
</table>


2. Further datasets: CORA (bibliographic), Restaurants, CDs, Movies

5. Similarity Measures – In Detail

- **Edit-based**
  - Levenshtein
  - Damerau-Levenshtein
  - Smith-Waterman
  - Smith-Waterman-Gotoh

- **Token-based**
  - Jaro
  - Jaro-Winkler
  - Words / n-grams
  - Dice

- **Datatype-specific**
  - Numerical Attributes
  - Dates Times
  - Geo Coordinates

- **Phonetic**
  - Soundex
  - Kölner Phonetik
  - Metaphone
  - Double Metaphone

- **Hybrid**
  - Monge-Elkan
  - Soft TF-IDF
  - Cosine Similarity
The Entity Matching Process

- Gather Data for Matching
- Normalize Attribute Values
- Apply Attribute-specific Similarity Measures
- Combine Similarity Scores
- Decide Match/Non-Match

But do not forget about the importance of the first two steps!
Similarity and Distance Measures

- sim(x,y)
  - x and y can be strings, numbers, dates, geo coordinates, records, images, ...

- Normalized: sim(x,y) ∈ [0,1]
  - sim(x,y) = 1 for exact match
  - sim(x,y) = 0 for “completely different“ x and y.
  - 0 < sim(x,y) < 1 for some approximate similarity

- Distance function / distance metric
  - Reflexive: dist(x,x) = 0
  - Positive: dist(x,y) ≥ 0
  - Symmetric: dist(x,y) = dist(y,x)
  - Triangular inequation: dist(x,z) ≤ dist(x,y) + dist(y,z)

- sim(x,y) = 1 − dist(x,y) if dist(x,y) ∈ [0,1]
- sim(x,y) = 1/(dist(x,y)+1) if dist(x,y) ∈ [0,∞]
5.1 Edit-based Similarity Measures

Similarity Measures

- **Edit-based**
  - Damerau-Levenshtein
  - Levenshtein
  - Smith-Waterman
  - Smith-Waterman-Gotoh

- **Datatype-specific**
  - Numerical attributes
  - Dates Times
  - Geo Coordinates

- **Token-based**
  - Jaccard
  - Dice
  - Monge-Elkan
  - Soft TF-IDF

- **Phonetic**
  - Words / n-grams
  - Cosine Similarity
  - Metaphone
  - Double Metaphone

- **Hybrid**
  - Monge-Elkan
  - Soft TF-IDF
Levenshtein Distance

- Measures the dissimilarity of two strings.
- Measures the **minimum number of edits** needed to transform one string into the other.

- **Allowed edit operations:**
  - insert a character into the string
  - delete a character from the string
  - replace one character with a different character

- **Examples:**
  - `levensthein('Table', 'Cable') = 1`  (1 Substitution)
  - `levensthein('Table', 'able') = 1`  (1 Deletion)
**Levenshtein Similarity**

\[ sim_{Levenshtein} = 1 - \frac{Levenshtein\ nDist}{\max(|s_1|, |s_2|)} \]

<table>
<thead>
<tr>
<th>(s_1)</th>
<th>(s_2)</th>
<th>Levenshtein Distance</th>
<th>(sim_{Levenshtein})</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jones</td>
<td>Johnson</td>
<td>4</td>
<td>0.43</td>
</tr>
<tr>
<td>Paul</td>
<td>Pual</td>
<td>2</td>
<td>0.5</td>
</tr>
<tr>
<td>Paul Jones</td>
<td>Jones, Paul</td>
<td>11</td>
<td>0</td>
</tr>
</tbody>
</table>
Levenshtein Discussion

- Good general purpose string similarity measure.
  - Can deal with typos.
  - Does not work if parts of string (words) have different order.
    - ‘Firstname Surname’ vs. ‘Surname, Firstname’
    - Other similarity measures are optimized for specific strings like names.

- Has quadratic complexity 😞
  - Levenshtein distance is calculated using dynamic programming
  - Time Complexity $O(|x| \cdot |y|)$
Jaro Similarity

- Specifically designed for matching names at US Census Bureau.
- Applies heuristics that empirically proofed to work for names.
  - first names, surnames, street/town names, country names

1. Search for matching characters within a specific distance
   - \( m \): number of matching characters
   - Search range for matching characters: \( \frac{\max(|x|,|y|)}{2} - 1 \)
   - Division by 2 as names often have two parts.

2. Look for swapped adjacent characters
   - transposition: ‘pe’ vs. ‘ep’
   - \( t \): number of transpositions
     \[
     \text{sim}_{jaro} = \frac{1}{3} \left( \frac{m}{|x|} + \frac{m}{|y|} + \frac{m - t}{m} \right)
     \]
Jaro Similarity – Example

\[ sim_{jaro} = \frac{1}{3} \left( \frac{m}{|x|} + \frac{m}{|y|} + \frac{m - t}{m} \right) \]

\( m = 4 \quad t = 1 \)

\[ sim_{jaro} = \frac{1}{3} \cdot \left( \frac{4}{4} + \frac{4}{4} + \frac{4 - 1}{4} \right) \approx 0.92 \]

\( m = 4 \quad t = 0 \)

\[ sim_{jaro} = \frac{1}{3} \cdot \left( \frac{4}{5} + \frac{4}{7} + \frac{4 - 0}{4} \right) \approx 0.79 \]
Winkler Similarity

- Intuition: Similarity of first few letters is most important.
  - Less typos in first letters
  - Dealing with abbreviations
    - ‘Apple Corp.’ vs. ‘Apple Cooperation’
    - ‘Bizer, Christian’ vs. ‘Bizer, Chris’

- Let $p$ be the length of the common prefix of $x$ and $y$.
  
  $$
  sim_{winkler}(x, y) = sim_{jaro}(x, y) + (1 - sim_{jaro}(x, y)) \frac{p}{10}
  $$
  
  - $= 1$ if common prefix is $\geq 10$
Jaro-Winkler Similarity

- Extension of Jaro similarity, considering a common prefix

\[
\text{if } \text{sim}_{jaro} \leq 0.7: \text{sim}_{jaro \text{inkle}} = \text{sim}_{jaro}
\]

\[
\text{otherwise: } \text{sim}_{jaro \text{inkle}} = \text{sim}_{jaro} + l \cdot p \cdot (1 - \text{sim}_{jaro})
\]

- \(l\): length of common prefix up to a maximum of 4 characters
- \(p\): constant scaling factor for how much the score is adjusted upwards for having common prefixes (typically \(p=0.1\))

- Examples:

  \[s_1 = PAUL \quad s_2 = PUAL\]
  \[
  \text{sim}_{jaro} = 0.92
  \]
  \[
  l = 1
  \]
  \[
  p = 0.1
  \]
  \[
  \text{sim}_{jaro \text{inkle}} = 0.92 + 1 \cdot 0.1 \cdot (1 - 0.92) = 0.928
  \]

  \[s_1 = JONES \quad s_2 = JOHNSON\]
  \[
  \text{sim}_{jaro} = 0.79
  \]
  \[
  l = 2
  \]
  \[
  p = 0.1
  \]
  \[
  \text{sim}_{jaro \text{inkle}} = 0.79 + 2 \cdot 0.1 \cdot (1 - 0.79) = 0.832
  \]
5.2 Token-based Similarity Measures

- Edit-based
  - Damerau-Levenshtein
  - Levenshtein
  - Smith-Waterman
  - Smith-Waterman-Gotoh

- Token-based
  - Jaccard
  - Dice
  - Cosine Similarity
  - Monge-Elkan
  - Soft TF-IDF

- Datatype-specific
  - Numerical attributes
  - Dates Times

- Phonetic
  - Geo Coordinates
  - Soundex
  - Kölner Phonetik

- Hybrid
  - Monge-Elkan
  - Soft TF-IDF
Token-based Similarity

Token-based measures can deal with the different order of words in longer strings.

- ‘Chris Bizer’ and ‘Bizer, Chris’ do not look similar to edit-based measures
- ‘Processor: Intel Xeon E5620’ vs. ‘Intel Xeon E5620 processor’ vs. ‘Intel Xeon E5620’
- Tokenization
  - Forming words from sequence of characters
- General idea: Separate string into tokens using some separator
  - Possible separators: Space, hyphen, punctuation, special characters
- Alternative: Split string into short substrings
  - n-grams: next slide
**n-grams (aka q-grams)**

- Split string into short substrings of length $n$.
  - Sliding window over string
  - $n=2$: Bigrams
  - $n=3$: Trigrams
  - Variation: Pad with $n-1$ special characters
    - Emphasizes beginning and end of string
    - Variation: Include positional information to weight similarities

- Goals:
  1. Deal with typos and different order of words
  2. Reduce the time complexity compared to Levenshtein

<table>
<thead>
<tr>
<th>String</th>
<th>Bigrams</th>
<th>Padded bigrams</th>
<th>Positional bigrams</th>
<th>Trigrams</th>
</tr>
</thead>
<tbody>
<tr>
<td>gail</td>
<td>ga, ai, il</td>
<td>☯g, ga, ai, il, l⊗</td>
<td>(ga,1), (ai,2), (il,3)</td>
<td>gai, ail</td>
</tr>
<tr>
<td>gayle</td>
<td>ga, ay, yl, le</td>
<td>☯g, ga, ay, yl, le, e⊗</td>
<td>(ga,1), (ay,2), (yl,3), (le,4)</td>
<td>gay, ayl, yle</td>
</tr>
<tr>
<td>peter</td>
<td>pe, et, te, er</td>
<td>☯p, pe, et, te, er, r⊗</td>
<td>(pe,1), (et,2), (te,3), (er,4)</td>
<td>pet, ete, ter</td>
</tr>
<tr>
<td>pedro</td>
<td>pe, ed, dr, ro</td>
<td>☯p, pe, ed, dr, ro, o⊗</td>
<td>(pe,1), (ed,2), (dr,3), (ro,4)</td>
<td>ped, edr, dro</td>
</tr>
</tbody>
</table>
Token-based Similarity Measures

- Can be applied to words or n-grams

- **Overlap coefficient**: \( sim_{overlap}(x, y) = \frac{|tok(x) \cap tok(y)|}{\min(|tok(x)|, |tok(y)|)} \)
  
  - Example: Useful for attribute label matching where attribute labels might contain units of measurements or years

- **Jaccard coefficient**:
  
  \[
  sim_{jaccard}(x, y) = \frac{|tok(x) \cap tok(y)|}{|tok(x)| + |tok(y)| - |tok(x) \cap tok(y)|} = \frac{|tok(x) \cap tok(y)|}{|tok(x) \cup tok(y)|}
  \]
  
  - Widely used general purpose similarity measure for tokens.

- **Dice's coefficient**: \( sim_{dice}(x, y) = \frac{2 \cdot |tok(x) \cap tok(y)|}{|tok(x)| + |tok(y)|} \)
5.3 Phonetic Similarity Measures

- **Edit-based**
  - Damerau-Levenshtein
  - Smith-Waterman
  - Smith-Waterman-Gotoh
  - Levenshtein
  - Hamming
  - Jaro
  - Jaro-Winkler

- **Token-based**
  - Words / n-grams
  - Jaccard
  - Dice
  - Cosine Similarity

- **Hybrid**
  - Monge-Elkan
  - Soft TF-IDF

- **Datatype-specific**
  - Numerical attributes
  - Dates Times
  - Geo Coordinates

- **Phonetic**
  - Soundex
  - Kölner Phonetik
  - Metaphone
  - Double Metaphone
  - Soft TF-IDF
Soundex

- Soundex codes a last name based on the way a last name sounds.

- Algorithm:
  1. Retain first letter of the name and drop all other occurrences of A, E, H, I, O, U, W, Y
  2. Replace consonants with digits
  3. Two adjacent letters with the same number are coded as a single number
  4. Continue until you have one letter and three numbers. If you run out of letters, pad with 0s.

- If a surname has a prefix, such as Van, Con, De, Di, La, or Le, code both with and without the prefix.

- Rules have been generated empirically.

<table>
<thead>
<tr>
<th>Digit</th>
<th>Letters</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>B, F, P, V</td>
</tr>
<tr>
<td>2</td>
<td>C, G, J, K, Q, S, X, Z</td>
</tr>
<tr>
<td>3</td>
<td>D, T</td>
</tr>
<tr>
<td>4</td>
<td>L</td>
</tr>
<tr>
<td>5</td>
<td>M, N</td>
</tr>
<tr>
<td>6</td>
<td>R</td>
</tr>
</tbody>
</table>

- Example
  - PAUL: P400
  - PUAL: P400
  - JONES: J520
  - JOHNSON: J525
  - J525 also: Jenkins, Jansen, Jameson
Soundex on WolframAlpha

Input interpretation:

Soundex Levenshtein

Soundex code:

L152

Soundex-close English words:

Livingstone | lebensraum | Livingston | lovemaking
Like Soundex, but aimed at German last names

Letters get different codes based on the context

Code length is not restricted

Multiple occurrences of the same code and "0" are removed

Examples:
- PAUL: 15
- PUAL: 15
- JONES: 68
- JOHNSON: 686

<table>
<thead>
<tr>
<th>Letter</th>
<th>Context</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>A, E, I, J, O, U, Y</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>H</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>B</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>P</td>
<td>not before H</td>
<td>1</td>
</tr>
<tr>
<td>D, T</td>
<td>not before C, S, Z</td>
<td>2</td>
</tr>
<tr>
<td>F, V, W</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>P</td>
<td>before H</td>
<td>3</td>
</tr>
<tr>
<td>G, K, Q</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>C</td>
<td>in the initial sound before A, H, K, L, O, Q, R, U, X</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>before A, H, K, O, Q, U, X but not after S, Z</td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>not after C, K, Q</td>
<td>48</td>
</tr>
<tr>
<td>L</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>M, N</td>
<td></td>
<td>6</td>
</tr>
<tr>
<td>R</td>
<td></td>
<td>7</td>
</tr>
<tr>
<td>S, Z</td>
<td>after S, Z</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>in the initial sound, but not before A, H, K, L, O, Q, R, U, X</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>not before A, H, K, O, Q, U, X</td>
<td></td>
</tr>
<tr>
<td>D, T</td>
<td>before C, S, Z</td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>after C, K, Q</td>
<td></td>
</tr>
</tbody>
</table>
5.4 Hybrid Similarity Measures

- **Edit-based**
  - Damerau-Levenshtein
  - Hamming
  - Levenshtein
  - Jaro-Winkler
  - Smith-Waterman
  - Smith-Waterman-Gotoh

- **Token-based**
  - Jaccard
  - Dice
  - Words / n-grams
  - Cosine Similarity

- **Datatype-specific**
  - Numerical attributes
  - Dates
  - Times
  - Geo Coordinates

- **Phonetic**
  - Soundex
  - Kölner Phonetik
  - Metaphone
  - Double Metaphone

- **Hybrid**
  - Monge-Elkan
  - Soft TF-IDF
Monge-Elkan Similarity

− Hybrid: Token-based and internal similarity function for tokens
  • Find best match for each token

− Goal: Deal with typos and different order of words.

$$sim_{MongeElkan}(x, y) = \frac{1}{|x|} \sum_{i=1}^{|x|} \max_{j=1,|y|} sim'(x[i], y[j])$$

• $|x|$ is number of tokens in $x$
• $sim'$ is internal similarity function (e.g. Levenshtein or Jaro)

− If strings contain just one token each
  • $sim_{MongeElkan}(x, y) = sim'(x, y)$

− Complexity: Quadratic in number of tokens 😐
Monge-Elkan – Example

\[
sim_{\text{MongeElkan}}(x, y) = \frac{1}{|x|} \sum_{i=1}^{|x|} \max_{j=1,|y|} \ sim'(x[i], y[j])
\]

− Peter Christen vs. Christian Pedro
  • \( \text{sim}_{\text{Jaro}}(\text{peter, christian}) = 0.3741 \)
  • \( \text{sim}_{\text{Jaro}}(\text{peter, pedro}) = 0.7333 \)
  • \( \text{sim}_{\text{Jaro}}(\text{christen, christian}) = 0.8843 \)
  • \( \text{sim}_{\text{Jaro}}(\text{christen, pedro}) = 0.4417 \)

− \( \text{sim}_{\text{MongeElkan}}(\text{'peter christen', 'christian pedro'}) = \frac{1}{2} (0.7333 + 0.8843) = 0.8088 \)
Extended Jaccard Similarity

**Standard Jaccard**

\[
    \text{sim}_{\text{jaccard}}(x, y) = \frac{|\text{tok}(x) \cap \text{tok}(y)|}{|\text{tok}(x)| + |\text{tok}(y)| - |\text{tok}(x) \cap \text{tok}(y)|} = \frac{|\text{tok}(x) \cap \text{tok}(y)|}{|\text{tok}(x) \cup \text{tok}(y)|}
\]

- If strings contain multiple words, choose words as tokens.

**Extended Jaccard**

- Use internal similarity function (e.g. Levenshtein or Jaro) to calculate similarity between all pairs of tokens.
- Consider all tokens as shared if similarity is above threshold.
- Shared tokens: \( S = \{(x_i, y_j) | x_i \in \text{tok}(x) \land y_j \in \text{tok}(y): \text{sim}'(x_i, y_j) \geq \theta\} \)
- Unique tokens: \( U_{\text{tok}(x)} = \{x_i | x_i \in \text{tok}(x) \land y_j \in \text{tok}(y) \land (x_i, y_j) \notin S\} \)

\[
    \text{sim}_{\text{jaccad\_ext}}(x, y) = \frac{|S|}{|S| + |U_{\text{tok}(x)}| + |U_{\text{tok}(y)}|}
\]
5.5 Datatype-specific Similarity Measures

- **Edit-based**
  - Damerau-Levenshtein
  - Levenshtein
  - Smith-Waterman
  - Smith-Waterman-Gotoh

- **Token-based**
  - Jaccard
  - Dice
  - Cosine Similarity
  - Monge-Elkan
  - Soft TF-IDF

- **Phonetic**
  - Soundex
  - Kölner Phonetik
  - Metaphone
  - Double Metaphone

- **Datatype-specific**
  - Dates
  - Times
  - Geo Coordinates
  - Numerical attributes

- **Hybrid**
  - Monge-Elkan
Numerical Comparison

Approach 1: Tolerate absolute difference between values, independently of absolute values.

- \( \text{sim}_{\text{num.abs}}(n, m) = \begin{cases} 
1 - \left( \frac{|n-m|}{d_{\text{max}}} \right) & \text{if } |n-m| < d_{\text{max}} \\
0 & \text{else}
\end{cases} \)

  - Linear extrapolation between 0 and \( d_{\text{max}} \)
  - \( d_{\text{max}} = \) maximal numeric distance in which numbers should be considered similar

- Example:
  - \( d_{\text{max}} = \$1,000 \)
  - \( \text{sim}_{\text{num.abs}}(2,000, 2,500) = 1 - \frac{500}{1,000} = 0.5 \)
  - \( \text{sim}_{\text{num.abs}}(200,000, 200,500) = 1 - \frac{500}{1,000} = 0.5 \)

Approach 2: Tolerate difference up to a certain percentage of the absolute values.

- \( \text{sim}_{\text{num.perc}}(n, m) = \begin{cases} 
1 - \left( \frac{pc}{pc_{\text{max}}} \right) & \text{if } pc < pc_{\text{max}} \\
0 & \text{else}
\end{cases} \)

  - \( pc = \frac{|n-m|}{\max(|n|, |m|)} \cdot 100 \) is percentage difference
  - \( pc_{\text{max}} = 33\% \) is the maximal percentage that should be tolerated

- Example:
  - \( pc = \frac{2,000 - 2,500}{2,500} \cdot 100 = 20\% \)
  - \( \text{sim}_{\text{num.perc}}(2,000, 2,500) = 1 - \frac{20}{33} = 0.394 \)
  - \( \text{sim}_{\text{num.perc}}(200,000, 200,500) = 1 - \frac{0.25}{33} = 0.993 \)
Time and Space Comparisons

- Dates
  - Convert dates into days after year 0 → integer
  - Afterwards use $\text{sim}_{\text{num\_abs}}$

- Geographic Coordinates
  - Distance is measured along the surface of the Earth in kilometers or miles.
  - Compute distance based on geographic projection of coordinates.
  - Java package for calculating geographic distances: Geographiclib
4.6 Implementations of Similarity Measures

- **SecondString Library**
  - Java package that we will use in the exercise
  - Supports all basic string comparisons
  - MongeElkan, SoftTFIDF
  - [http://sourceforge.net/projects/secondstring/](http://sourceforge.net/projects/secondstring/)

- **SimMetrics Library**
  - Alternative Java package
  - Supports all basic string comparisons
6. Learning Matching Models

- **Problem:**
  It is hard for humans to write good matching rules, as this requires a lot of knowledge about the data set and matching techniques.
  - What kind of typos and other errors are contained in the data?
  - Which string similarity measure fits which attribute?
  - How to set similarity thresholds?
  - How to weight different attributes?

- **Solution:**
  1. Manually label a certain amount of pairs as matches and non-matches.
  2. Use machine learning to generate a matching model from this training data.

- **Advantage:**
  - The human does what she is good at: Understand the data.
  - The computer does what it is good at: Learn detailed model from examples.
Training Data and Feature Generation

- **Training data:** \( T = \{(x_1, y_1, l_1), \ldots, (x_n, y_n, l_n)\} \), where
  - each \((x_i, y_i)\) is a record pair and
  - \(l_i\) is a label: “yes” if \(x_i\) matches \(y_i\) and “no” otherwise

- **Feature Generation**
  - define a set of features \( f_1, \ldots, f_m \), each quantifying one aspect of the domain judged possibly relevant to matching the records
  - Feature = Similarity measure applied to attribute pair
  - Feature Value = Similarity score
  - If you want the learning algorithm to decides which similarity metric fits best for a specific attribute pair, you generate multiple features for the pair.
    - Levenshtein(\(\text{Name1}, \text{Name2}\))
    - Jaro(\(\text{Name1}, \text{Name2}\))
    - Jaro-Winkler(\(\text{Name1}, \text{Name2}\))
Example: Feature Generation

- \( s_1 \) and \( s_2 \) use Jaro-Winkler and edit distance
- \( s_3 \) uses edit distance (ignoring area code of \( a \))
- \( s_4 \) and \( s_5 \) return 1 if exact match, 0 otherwise
- \( s_6 \) encodes a heuristic constraint (using a lookup table)
Learn Matching Model M

1. convert each training example \( (x_i, y_i, l_i) \) in T to a pair \( (v_i, c_i) \)
   
   - \( v_i = f_1(x_i, y_i), \ldots, f_m(x_i, y_i) \) is a feature vector that encodes \( (x_i, y_i) \) in terms of the features (list of similarity values)
   
   - \( c_i \) is an appropriately transformed version of label \( l_i \) (e.g., yes/no or 1/0, depending on learning algorithm used afterwards)
   
   - thus T is transformed into \( T' = \{(v_1, c_1), \ldots, (v_n, c_n)\} \)

2. apply a learning algorithm to \( T' \) to learn a matching model M
   
   - e.g. linear regression, SVMs, decision trees, …
Example: Learning a Linearly Weighted Rule

- Goal: learn rule \( s(a,b) = \sum_{i=1}^{6} \alpha_i s_i(a, b) \)
- Perform a least-squares linear regression on training data

\[
\begin{align*}
v_1 &= \langle s_1(a_1,b_1), s_2(a_1,b_1), s_3(a_1,b_1), s_4(a_1,b_1), s_5(a_1,b_1), s_6(a_1,b_1) \rangle, 1> \\
v_2 &= \langle s_1(a_2,b_2), s_2(a_2,b_2), s_3(a_2,b_2), s_4(a_2,b_2), s_5(a_2,b_2), s_6(a_2,b_2) \rangle, 1> \\
v_3 &= \langle s_1(a_3,b_3), s_2(a_3,b_3), s_3(a_3,b_3), s_4(a_3,b_3), s_5(a_3,b_3), s_6(a_3,b_3) \rangle, 0>
\end{align*}
\]

to find weights \( \alpha_i \) that minimizes the squared error

\[
\sum_{i=1}^{3} (c_i - \sum_{j=1}^{6} \alpha_j s_j(v_i))^2
\]
Example: Learning a Decision Tree

Positive:
The decision tree learning algorithm automatically selects the features that are useful.
Discussion Learning-based Approach

- **Pros compared to writing rules by hand**
  - when writing rules by hand, you must manually decide if a particular feature is useful → labor intensive and limits the number of features we can consider.
  - learning-based approaches can automatically examine a large number of features.

- **Cons**
  - you need to label training examples.
  - you don’t know which examples matter to the algorithm and thus might label an unnecessary large number of examples in order to cover all relevant corner-cases.

- **Solution**
  - Use Active Learning in order to let the algorithm decide which examples matter.
  - Practical experience: F1 > 0.95 after labeling less 10-50 pairs.

Learning Linkage Rules within the Silk Workbench

1. Labeling Pairs of Movies

2. Learned Linkage Rule

http://wifo5-03.informatik.uni-mannheim.de/bizer/silk/
References

- **All Aspects of Identity Resolution**
  - Peter Christen: Data Matching. Springer 2012.

- **Blocking**
References

- **Similarity Measures**

- **Matching Systems and their Evaluation**

- **Learning Matching Rules**