Record Linkage: Introduction, Recent Advances, and Privacy Issues

Peter Christen

Research School of Computer Science,
ANU College of Engineering and Computer Science,
The Australian National University

Contact: peter.christen@anu.edu.au
Motivation

Large amounts of data are being collected both by organisations in the private and public sectors, as well as by researchers and individuals.

Much of these data are about people, or they are generated by people:
- Financial, shopping, and travel transactions
- Electronic health records
- Tax, social security, and census records
- Vital events data (births, marriages, deaths)
- Emails, tweets, SMSs, Facebook posts, etc.

Analysing such data can provide huge benefits to businesses, governments and researchers.
Motivation (continued)

- Often data from different sources need to be integrated and linked
  - To allow data analyses that are impossible on individual databases
  - To improve data quality
  - To enrich data with additional information

- Lack of unique *entity identifiers* means that linking is often based on personal information

- When databases are linked across organisations, maintaining privacy and confidentiality is vital

- The linking of databases is challenged by **data quality**, **database size**, and **privacy concerns**
Motivating example: Health surveillance (1)
Motivating example: Health surveillance (2)

- Preventing the outbreak of epidemics requires monitoring of occurrences of unusual patterns of symptoms, ideally in real time.

- Data from many different sources will need to be collected (including travel and immigration records; doctors, emergency and hospital admissions; drug purchases; social network and location data; and possibly even animal health data).

- Privacy and confidentiality concerns arise if such data are stored and linked at a central location.

- Such data sets are large, dynamic, complex, heterogeneous and distributed, and they require linking and analysis in near real time.
Objective of this tutorial

- Provide an understanding of record linkage applications, challenges, and techniques
- Understand the record linkage process, and key techniques employed in each step of this process
- Have a basic understanding of advanced techniques for scalable indexing and machine-learning based classification for record linkage
- Appreciate the privacy and confidentiality challenges that record linkage poses
- Have a basic understanding of privacy-preserving record linkage
Content is loosely based on ‘Data Matching’ (Springer, 2012)

The book is very well organized and exceptionally well written. Because of the depth, amount, and quality of the material that is covered, I would expect this book to be one of the standard references in future years.

William E. Winkler, U.S. Bureau of the Census.
Outline

Part 1: Introduction
- Applications, history, challenges, and examples

Part 2: Record linkage process
- Key techniques used in record linkage

Part 3: Advanced record linkage techniques
- Indexing and blocking for scalable record linkage
- Learning, collective, and graph based techniques

Part 4: Privacy aspects in record linkage
- Motivating scenario
- Privacy-preserving record linkage

Conclusions and research directions
What is record linkage?

The process of linking records that represent the same entity in one or more databases (patients, customers, businesses, consumer products, publications, etc.)

Also known as data linkage, data matching, entity resolution, duplicate detection, etc.

Major challenge is that unique entity identifiers are not available in the databases to be linked (or if available, they are not consistent or change over time)

E.g., which of these records represent the same person?

<table>
<thead>
<tr>
<th>Name</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dr Smith, Peter</td>
<td>42 Miller Street 2602 O’Connor</td>
</tr>
<tr>
<td>Pete Smith</td>
<td>42 Miller St 2600 Canberra A.C.T.</td>
</tr>
<tr>
<td>P. Smithers</td>
<td>24 Mill Rd 2600 Canberra ACT</td>
</tr>
</tbody>
</table>
Applications of record linkage

- Remove duplicates in one data set (deduplication)
- Merge new records into a larger master data set
- Create patient or customer oriented statistics (for example for longitudinal studies)
- Clean and enrich data for analysis and mining
- Geocode matching (with reference address data)
- Widespread use of record linkage
  - Immigration, taxation, social security, census
  - Fraud, crime, and terrorism intelligence
  - Business mailing lists, exchange of customer data
  - Health and social science research
Recent interest in record linkage

- Traditionally, record linkage has been used in statistics (census) and health (epidemiology)
  - First computer based techniques developed in 1960s
- In recent years, increased interest from businesses and governments
  - Massive amounts of data are being collected, and increased computing power and storage capacities
  - Often data from different sources need to be integrated
  - Need for data sharing between organisations
  - Data mining (analysis) of large data collections
  - E-Commerce and Web services (comparison shopping)
  - Spatial data analysis and online map applications
A brief history of record linkage (1)

- Computer assisted record linkage goes back as far as the 1950s (based on ad-hoc heuristic methods)
- Basic ideas of probabilistic linkage were introduced by Newcombe & Kennedy (1962)
- Theoretical foundation by Fellegi & Sunter (1969)
  - No unique entity identifiers available
  - Compare common record attributes (or fields)
  - Compute matching weights based on frequency ratios (global or value specific) and error estimates
  - Sum of the matching weights is used to classify a pair of records as a match, non-match, or potential match
  - Problems: Estimating errors and thresholds, assumption of independence, and clerical review
A short history of record linkage (2)

- Strong interest in the last decade from computer science (from many research fields, including data mining, AI, knowledge engineering, information retrieval, information systems, databases, and digital libraries)
- Many different techniques have been developed
- Major focus has been on scalability to large databases, and linkage quality
  - Various indexing/blocking techniques to efficiently and effectively generate candidate record pairs
  - Various machine learning-based classification techniques, both supervised and unsupervised, as well as active learning based
The record linkage process

Database A

- Data pre-processing

Database B

- Data pre-processing

Indexing / Searching

Comparison

Classification

Clerical Review

Evaluation

Matches

Non-matches

Potential Matches

CSIC, July 2019 – p. 14/110
Record linkage techniques

- Deterministic matching
  - Rule-based matching (complex to build and maintain)
- Probabilistic record linkage (Fellegi and Sunter, 1969)
  - Use available attributes for linking (often personal information, like names, addresses, dates of birth, etc.)
  - Calculate match weights for attributes
- “Computer science” approaches
  - Based on machine learning, data mining, database, or information retrieval techniques
  - Supervised classification: Requires training data (true matches)
  - Unsupervised: Clustering, collective, and graph based
Major record linkage challenges

- No unique entity identifiers available
- Real world data are dirty
  (typographical errors and variations, missing and out-of-date values, different coding schemes, etc.)
- Scalability
  - Naïve comparison of all record pairs is quadratic
  - Remove likely non-matches as efficiently as possible
- No training data in many linkage applications
  - No record pairs with known true match status
- Privacy and confidentiality
  (because personal information, like names and addresses, are commonly required for linking)
Example 1: Web of Object (WOO)
(based on slides by Hye-Chung Kum, Texas A&M)

- **Goal**: To enable various products in Yahoo! to synthesise knowledge-bases of entities relevant to their domains (Bellare et al., VLDB, 2013)

- **Desiderata**:
  - *Coverage*: the fraction of real-world entities
  - *Accuracy*: information must be accurate
  - *Linkage*: the level of connectivity of entities
  - *Identifiability*: one and only one identifier for a real-world entity
  - *Persistence/content continuity*: variants of the same entity across time must be linked
  - *Multi-tenant*: be useful to multiple portals
Knowledge base synthesis is the process of ingestion, disambiguation, and enrichment of entities from a variety of structured and unstructured data sources.

- Sheer scale of the data
  - Hundreds of millions of entities daily
- Diverse domains
  - From hundreds of data sources
- Diverse requirements
  - Multiple tenants, such as Locals, Movies, Deals, and Events in (for example) the Yahoo! website
The WOO architecture (1)

Source: Bellare et al., VLDB, 2013
**Importer** takes a collection of data sources as input (like XML feeds, RDF content, Relational Databases, or other custom formats)
The WOO architecture (3)

Each data source is converted into a common format called the WOO schema.

The WOO Parcel, containing only the attributes needed for matching, is pushed to the Builder.
**Builder** performs the entity deduplication and produces a clustering decision, including (1) *blocker*, (2) *matcher*, (3) *connected component generator*, and (4) *group refiner*. 
Finalizer is responsible for handling the persistence of object identifiers and the blending of the attributes of the (potentially many) entities that are being merged.
Exporter generates a fully integrated and de-duplicated knowledge-base, both in a format consistent with the WOO schema and in any custom format.
**Curation** enables domain experts to influence the system behaviour through a set of graphical user interfaces (GUIs), such as: forcing or disallowing certain matches between entities, or by editing attribute values.
Example 2: Linking ‘big’ social science data

- Increasing use of large databases in social science research
- Often the aim is to create ‘social genomes’ for individuals by linking population databases (Population Informatics, Kum et al. IEEE Computer, 2013)
- Knowing how individuals and families change over time allows for a diverse range of studies (fertility, employment, education, health, crime, etc.)
- Different challenges for historical data compared to contemporary data, but some are common
  - Database sizes (computational aspects)
  - Accurate match classification (data quality)
Challenges for historical data

- Low literacy (recording errors and unknown exact values), no address or occupation standards
- Large percentage of a population had one of just a few common names (‘John’ or ‘Mary’)
- Households and families change over time
- Immigration and emigration, birth and death
- Scanning, OCR, and transcription errors
Challenges for present-day data

- These data are about living people, and privacy is therefore a major concern when such data are linked between organisations.
  - Linked data allow analyses not possible on individual databases (potentially revealing highly sensitive information).

- Modern databases contain more details and more complex types of data (free-format text or multimedia).

- Data are available from different sources (governments, businesses, social network sites, the Web).

- Major questions: Which data are suitable? Which can we get access to?
Outline

Part 1: Introduction
  - Applications, history, challenges, and examples

Part 2: record linkage process
  - Key techniques used in record linkage

Part 3: Advanced record linkage techniques
  - Indexing and blocking for scalable record linkage
  - Learning, collective, and graph based techniques

Part 4: Privacy aspects in record linkage
  - Motivating scenario
  - Privacy-preserving record linkage

Conclusions and research directions
The record linkage process

- Database A
  - Data pre-processing
  - Indexing / Searching
  - Comparison
  - Classification
  - Clerical Review
  - Potential Matches

- Database B
  - Data pre-processing
  - Matches
  - Non-matches
  - Evaluation
  - Matches

Potential
Matches
Why cleaning and standardisation?

- Real world data are often *dirty*
  - Typographical and other errors
  - Different coding schemes
  - Missing values
  - Data changing over time

- Name and addresses are especially prone to data entry errors
  - Scanned, hand-written, over telephone, hand-typed
  - Same person often provides her/his details differently
  - Different correct spelling variations for proper names (e.g. ‘Gail’ and ‘Gayle’, or ‘Dixon’ and ‘Dickson’)
Example: Address standardisation

App. 3a/42 Main Rd Canberra A.C.T. 2600

1. Clean input
   - Remove unwanted characters and words
   - Expand abbreviations and correct misspellings

2. Segment address into well defined output fields

3. Verify if address (or parts of it) exists in reality
Standardisation approaches

- Rules based
  - Manually developed parsing and transformation rules
  - Time consuming and complex to develop and maintain

- Probabilistic methods
  - Based for example on hidden Markov models (HMMs)
  - More flexible and robust with regard to new unseen data
  - Drawback: Training data needed for most methods (for example, sets of correctly standardised addresses)

HMMs have been widely used in natural language processing and speech recognition, as well as for text segmentation and information extraction.
A HMM is a **probabilistic** finite state machine

- Made of a set of *states* and *transition probabilities* between these states
- In each state an *observation* symbol is emitted with a certain probability distribution
- For data segmentation, the observation symbols are *tags* and the states correspond to the *output fields*
Standardisation steps

- **Cleaning**
  - Based on look-up tables and correction lists
  - Remove unwanted characters and words
  - Correct various misspellings and abbreviations

- **Tagging**
  - Split input into a list of *tokens* (words, characters, numbers, and separators)
  - Assign one or more *tags* to each token using look-up tables and/or features

- **Segmenting**
  - Use for example a trained HMM to assign list elements into *output fields*
Data tagging example

- Tags provide information about the category / type of a token, such as:
  - **TI** Name title words (‘ms’, ‘mr’, ‘dr’, etc.)
  - **GM** Male given names (‘thomas’, ‘paul’, etc.)
  - **SN** Surnames (‘smith’, ‘miller’, ‘thomas’, etc.)
  - **N4** Four-digit numbers (’2602’, ‘3000’, etc.)

- Specific tags for names, addresses, and other domains (some overlapping, like street names)

- Example tagging:
  - Uncleaned input string: ‘Doc. Thomas Paul MILLER’
  - Cleaned string: ‘dr thomas paul miller’
  - Token and tag lists:
    ```
    ['dr', 'thomas', 'paul', 'miller']
    ['TI', 'GM/SN', 'GM', 'SN']
    ```
**Blocking / indexing / filtering**

- Number of record pair comparisons equals the product of the sizes of the two data sets
  
  (matching two data sets containing 1 and 5 million records will result in $1,000,000 \times 5,000,000$ record pairs)

- Performance bottleneck in a record linkage system is usually the (expensive) detailed comparison of field values between record pairs
  
  (such as approximate string comparison functions)

- Blocking / indexing / filtering techniques are used to reduce the large amount of comparisons

- Aim of blocking: Cheaply remove candidate record pairs which are obviously not matches
Traditional blocking

Traditional blocking works by only comparing record pairs that have the same value for a blocking variable (for example, only compare records that have the same postcode value).

Problems with traditional blocking

- An erroneous value in a blocking variable results in a record being inserted into the wrong block (several passes with different blocking variables can solve this).
- Values of blocking variable should have uniform frequencies (as the most frequent values determine the size of the largest blocks).

Example: Frequency of ‘Smith’ in NSW: 25,425
  Frequency of ‘Dijkstra’ in NSW: 4
Phonetic encoding

- Bringing together spellings variations of the same name for improved blocking
- Techniques such as *Soundex*, *NYSIIS*, or *Double-Metaphone*

**Examples:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Soundex</th>
<th>NYSIIS</th>
<th>Double-Metaphone</th>
</tr>
</thead>
<tbody>
<tr>
<td>stephen</td>
<td>s315</td>
<td>staf</td>
<td>stfn</td>
</tr>
<tr>
<td>steve</td>
<td>s310</td>
<td>staf</td>
<td>stf</td>
</tr>
<tr>
<td>gail</td>
<td>g400</td>
<td>gal</td>
<td>kl</td>
</tr>
<tr>
<td>gayle</td>
<td>g400</td>
<td>gal</td>
<td>kl</td>
</tr>
<tr>
<td>christine</td>
<td>c623</td>
<td>chra</td>
<td>krst</td>
</tr>
<tr>
<td>christina</td>
<td>c623</td>
<td>chra</td>
<td>krst</td>
</tr>
<tr>
<td>kristina</td>
<td>k623</td>
<td>cras</td>
<td>krst</td>
</tr>
</tbody>
</table>
Soundex algorithm

- Keep first letter of a string (name), and remove all following occurrences of a, e, i, o, u, y, h, w
- Replace all consonants from position 2 onwards with digits using these rules:
  - b, f, p, v → 1
  - c, g, j, k, q, s, x, z → 2
  - d, t → 3
  - l → 4
  - m, n → 5
  - r → 6
- Only keep unique adjacent digits
- If length of code is less than 4 add zeros, if longer truncate at length 4
The record linkage process

Database A

Data pre-processing

Indexing / Searching

Comparison

Classification

Clerical Review

Potential Matches

Database B

Data pre-processing

Evaluation

Matches

Non-matches
Approximate string comparison

Aim: Calculate a normalised similarity between two strings \(0 \leq sim_{approx} \leq 1\)

- \(sim_{approx} = 1\) → Same (‘peter’, ‘peter’)
- \(sim_{approx} = 0\) → Totally different (‘peter’, ‘david’)
- \(0 < sim_{approx} < 1\) → Somewhat similar (‘peter’, ‘pedro’)

Many different techniques available, some generic, others specific for certain types of strings

- Edit-distance based (number of character edits)
- Set-based (Jaccard, Dice, and Overlap coefficients)
- Jaro-Winkler (specific for personal names)
- Monge-Elkan and Soft-TFIDF (specific for strings that contain several words)
Convert a string into q-grams (sub-strings of length $q$)
- For example, for $q = 2$: ‘peter’ $\rightarrow$ [‘pe’, ‘et’, ‘te’, ‘er’]

Find q-grams that occur in two strings, for example using the Dice coefficient:

$$sim_{Dice} = \frac{2 \times c_c}{c_1 + c_2}$$

where $c_c$ is number of common $q$-grams, and $c_1$ and $c_2$ the number of $q$-grams in string $s_1$ and $s_2$

- With $s_1 = ‘peter’$ and $s_2 = ‘pete’$: $c_1 = 4$, $c_2 = 3$, and $c_c = 3$ (‘pe’, ‘et’, ‘te’):

$$sim_{Dice}(‘peter’, ‘pete’) = \frac{2 \times 3}{4 + 3} = \frac{6}{7} = 0.86$$
Edit-distance based string comparisons

- The number of character edits needed to convert one string into another (insert, delete, substitute)
- Can be calculated using a dynamic programming algorithm (of quadratic complexity in length of strings)
- Convert distance into a similarity as:

\[ \text{sim}_{ED} = 1 - \frac{\text{dist}_{ED}}{\max(l_1, l_2)} \]

where \( l_1 \) and \( l_2 \) are the lengths of strings \( s_1 \) and \( s_2 \)

- With \( s_1 = \text{‘peter’} \) and \( s_2 = \text{‘pete’} \): \( l_1 = 5, l_2 = 4, \text{dist}_{ED} = 1 \) (delete ‘r’): \( \text{sim}_{ED} = 1 - 1/5 = 4/5 = 0.8 \)

- Variations consider transposition of two adjacent characters, allow for gaps, or different edit costs (learned from training data)
**Edit distance calculation example**

Matrix $D$ shows number of edits between substrings (for example, ‘ga’ and ‘gayle’ -> 3 inserts)

<table>
<thead>
<tr>
<th></th>
<th>g</th>
<th>a</th>
<th>y</th>
<th>l</th>
<th>e</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>g</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>a</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>i</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>l</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

- If $s_1[i] = s_2[j]$, then $D[i, j] = D[i - 1, j - 1]$
- If $s_1[i] \neq s_2[j]$, then $D[i, j] = \min \begin{cases} 
D[i - 1, j] + 1 & \text{del} \\
D[i, j - 1] + 1 & \text{ins} \\
D[i - 1, j - 1] + 1 & \text{subst} 
\end{cases}$

Edit path: ‘gail’ → substitute ‘i’ with ‘y’ → insert ‘e’ → ‘gayle’
(final edit distance $dist_{ED}('gail', 'gayle') = 2$)
Probabilistic record linkage

- Basic ideas of probabilistic linkage were introduced by Newcombe & Kennedy, 1962
- Theoretical foundation by Fellegi & Sunter, 1969
  - Compare common record attributes (or fields) using approximate (string) comparison functions
  - Calculate matching weights based on frequency ratios (global or value specific ratios) and error estimates
  - Sum of the matching weights is used to classify a pair of records as a match, non-match, or potential match
  - Problems: Estimating errors, find optimal thresholds, assumption of independence, and manual clerical review
Fellegi and Sunter classification (1)

For each compared record pair a vector of matching weights is calculated:

Record A: ['dr', 'thomas', 'paul', 'miller']
Record B: ['mr', 'john', '', 'miller']
Matching weights: [0.2, -3.2, 0.0, 2.4]

A ratio $R$ is calculated for each compared record pair $r = (a,b)$ in the product space $A \times B$:

$$R = \frac{P(\gamma \in \Gamma \mid r \in M)}{P(\gamma \in \Gamma \mid r \in U)},$$

where $M$ and $U$ are the sets of true matches and true non-matches, and $\gamma$ is an agreement pattern in the comparison space $\Gamma$, with:

- $A \times B = \{(a, b) : a \in A, b \in B\}$ for files $A$ and $B$
- $M = \{(a, b) : a = b, a \in A, b \in B\}$
- $U = \{(a, b) : a \neq b, a \in A, b \in B\}$
Fellegi and Sunter proposed the following decision rule:

\[ R \geq t_u \Rightarrow r \rightarrow \text{Match} \]
\[ t_l < R < t_u \Rightarrow r \rightarrow \text{Potential Match} \]
\[ R \leq t_l \Rightarrow r \rightarrow \text{Non-Match} \]
Fellegi and Sunter classification (3)

Assuming conditional independence between attributes allows to calculate individual attribute-wise probabilities

\[ m_i = P([a_i = b_i, a \in A, b \in B] \mid r \in M) \quad \text{and} \]
\[ u_i = P([a_i \neq b_i, a \in A, b \in B] \mid r \in U), \]

where \( a_i \) and \( b_i \) are the values of attribute \( i \) being compared

Based on these \( m \)- and \( u \)-probabilities, we calculate a matching weight \( w_i \) for attribute \( i \) as:

\[
w_i = \begin{cases} 
\log_2 \left( \frac{m_i}{u_i} \right) & \text{if } a_i = b_i \quad \text{(agreement weight)} \\
\log_2 \left( \frac{1 - m_i}{1 - u_i} \right) & \text{if } a_i \neq b_i \quad \text{(disagreement weight)}
\end{cases}
\]
Weight calculation: Month of birth

Assume two data sets with a 3% error in field *month of birth*

Probability that two matched records (representing the same person) have the same month value is 97% \( (m_i) \)

Probability that two matched records do not have the same month value is 3% \( (1-m_i) \)

Probability that two (randomly picked) un-matched records have the same month value is \( 1/12 = 8.3\% \) \( (u_i) \)

Probability that two un-matched records do not have the same month value is \( 11/12 = 91.7\% \) \( (1-u_i) \)

Agreement weight \( \log_2 \left( m_i / u_i \right) \): \( \log_2 \left( 0.97 / 0.083 \right) = 3.54 \)

Disagreement weight \( \log_2 \left( (1-m_i) / (1-u_i) \right) \): \( \log_2 \left( 0.03 / 0.917 \right) = -4.92 \)
Record linkage evaluation (1)

- At the end we need to evaluate how good the results of a record linkage project are.

Main measures for linkage complexity

- **Reduction ratio**: How many candidate record pairs were generated by blocking, compared to all pairs?

  \[
  rr = 1 - \left( \frac{\text{number of candidate pairs}}{\text{number of all record pairs}} \right)
  \]

- **Pairs completeness**: How many true matches were generated by blocking, divided by all true matches?

  \[
  pc = \frac{\text{number of true matching candidate pairs}}{\text{number of all true matching pairs}}
  \]
Record linkage evaluation (2)

To evaluate linkage quality, ground truth data (gold standard) in the form of known true matches and known true non-matches are required.

- True matches: Pairs of records that refer to the same real-world entity
- True non-matches: Pairs of records that refer to two different entities

In practical applications it is often difficult to get ground truth data (might need to be created using manual assessment of record pairs)
Binary classification outcomes

Four possible outcomes:

- A true matching record pair is correctly classified as matching (a *true match* / *true positive*)
- A true matching record pair is wrongly classified as non-matching (a *false non-match* / *false negative*)
- A true non-matching record pair is wrongly classified as matching (a *false match* / *false positive*)
- A true non-matching record pair is correctly classified as non-matching (a *true non-match* / *true negative*)
In record linkage, the number of true matches ($|TP| + |FN|$) is generally much lower than the number of true non-matches ($|TN| + |FP|$).

Without blocking / indexing, the number of record pair comparisons grows quadratic in the size of the databases to be linked (even with blocking / indexing this number usually grows more than linear).

Assuming no duplicates in the databases $D_A$ and $D_B$ to be linked (one record per entity), the maximum number of true matches is:

$$|TP| + |FN| \leq \min(|D_A|, |D_B|)$$

$\cdot$ represents the number of elements in a set
Calculating quality measures (1)

<table>
<thead>
<tr>
<th>True link status</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (match)</td>
<td>d</td>
<td>b</td>
</tr>
<tr>
<td>0 (non-match)</td>
<td>c</td>
<td>a</td>
</tr>
</tbody>
</table>

Predicted link status

| Predicted link status | d = |TP| | b = |FP| |
|-----------------------|----|---|----|----|
| 1 (match)             |    |   |    |    |
| 0 (non-match)         |    |   |    |    |

Accuracy $A = (a+d) / (a+b+c+d)$ is commonly used in classification problems to assess quality.

Due to the large number of $a$ (TN), accuracy is however not meaningful for record linkage (very high linkage accuracy is achieved if all record pairs are classified as non-matches because: $a \gg b, c, \text{ or } d$)
Calculating quality measures (2)

<table>
<thead>
<tr>
<th>True link status</th>
<th>1 (match)</th>
<th>0 (non-match)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicted link status</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 (match)</td>
<td>$d =</td>
<td>TP</td>
</tr>
<tr>
<td>0 (non-match)</td>
<td>$c =</td>
<td>FN</td>
</tr>
</tbody>
</table>

**Precision** $P = \frac{d}{b+d}$ is the proportion of compared record pairs classified as matches that are true matches (also known as *positive predictive value*).

**Recall** $R = \frac{d}{c+d}$ is the proportion of true matching record pairs that are classified as matches (also known as *sensitivity* or *true positive rate*).
The F-measure (1)

<table>
<thead>
<tr>
<th>Predicted link status</th>
<th>True link status</th>
<th>Precision and recall</th>
<th>F-measure formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (match)</td>
<td>1 (match)</td>
<td>$d =</td>
<td>TP</td>
</tr>
<tr>
<td>0 (non-match)</td>
<td>0 (non-match)</td>
<td>$b =</td>
<td>FP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$c =</td>
<td>FN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$a =</td>
<td>TN</td>
</tr>
</tbody>
</table>

- Precision and recall are commonly combined into one value, the F-measure:

- The harmonic mean of precision and recall
- Often used to compare different binary classifiers
The F-measure (2)

From the above, we see that the F-measure can be rewritten as

\[
F = \frac{c + d}{c + b + 2d} \times \frac{d}{c + d} + \frac{b + d}{c + b + 2d} \times \frac{d}{b + d} \\
= pR + (1 - p)P
\]

where

\[
p = \frac{c + d}{c + b + 2d} = \frac{|FN| + |TP|}{|FN| + |FP| + 2|TP|}
\]

As well as being the harmonic mean, the F-measure is also a weighted arithmetic mean with weight \(p\) given to recall and weight \((1-p)\) given to precision.
The F-measure – Some observations

Using a weighted arithmetic mean has a sensible justification: the weights would be the relative importance assigned to precision and recall.

However, the weights $p$ and $(1-p)$ are not chosen on the grounds of relative importance of precision and recall, but will vary based on the counts of $FP$, $FN$ and $TP$.

The measure being used to evaluate classification performance therefore depends on the thing being evaluated!

(for more see Hand and Christen, *A Note on using the F-measure*, Statistics and Computing, 2018)
Outline

Part 1: Introduction
  - Applications, history, challenges, and examples

Part 2: record linkage process
  - Key techniques used in record linkage

Part 3: Advanced record linkage techniques
  - Indexing and blocking for scalable record linkage
  - Learning, collective, and graph based techniques

Part 4: Privacy aspects in record linkage
  - Motivating scenario
  - Privacy-preserving record linkage

Conclusions and research directions
Advanced indexing approaches (1)

Sorted neighbourhood approach

- Sliding window over sorted databases
- Use several passes with different sorting criteria
- Window size can be fixed or adaptive (based on similarities between records)

For example, database sorted using first and last name:

<table>
<thead>
<tr>
<th>Name</th>
<th>Record</th>
</tr>
</thead>
<tbody>
<tr>
<td>abbybond</td>
<td>r5</td>
</tr>
<tr>
<td>paulsmith</td>
<td>r2</td>
</tr>
<tr>
<td>pedrosmith</td>
<td>r4</td>
</tr>
<tr>
<td>pedrosmith</td>
<td>r9</td>
</tr>
<tr>
<td>percysmith</td>
<td>r1</td>
</tr>
<tr>
<td>petersmith</td>
<td>r7</td>
</tr>
<tr>
<td>petersmith</td>
<td>r10</td>
</tr>
<tr>
<td>robinstevens</td>
<td>r3</td>
</tr>
<tr>
<td>sallytaylor</td>
<td>r6</td>
</tr>
<tr>
<td>sallytaylor</td>
<td>r8</td>
</tr>
</tbody>
</table>

- First window of records
- Second window of records
- Third window of records
- Fourth window of records
- Fifth window of records
- Last window of records
**Advanced indexing approaches (2)**

- **Canopy clustering**
  - Based on a computationally ‘cheap’ similarity measure such as Jaccard (set intersection based on q-grams)
  - Records will be inserted into several clusters / blocks
  - Algorithm steps:
    1) Randomly select a record in data set $D$ as cluster centroid $c_i$, $i = 1, 2, \ldots$
    2) Insert all records that have a similarity of at least $s_{loose}$ with $c_i$ into cluster $C_i$
    3) Remove all records $r_j \in C_i$ (including $c_i$) that have a similarity of at least $s_{tight}$ with $c_i$ from $D$, with $s_{tight} \geq s_{loose}$
    4) If data set $D$ not empty go back to step 1
**Advanced indexing approaches (3)**

- **Q-gram based blocking** (e.g. 2-grams / bigrams)
  - Convert values into q-gram lists, then generate sub-lists
    - ‘peter’ $\rightarrow$ ['pe', 'et', 'te', 'er'], ['pe', 'et', 'te'], ['pe', 'et', 'er'], ..
    - ‘pete’ $\rightarrow$ ['pe', 'et', 'te'], ['pe', 'et'], ['pe', 'te'], ['et', 'te'], ...
  - Records with the same sub-list value are inserted into the same block
  - Each record will be inserted into several blocks
  - Works well for ‘dirty’ data but has high computational costs

- **Mapping-based blocking**
  - Map strings into a multi-dimensional space such that distances between strings are preserved
Controlling block sizes

- Important for real-time and privacy-preserving linkage, and with certain machine learning algorithms (that have a quadratic or higher complexity)
- Use for example an iterative split-merge clustering approach

Original data set from Table 1

<table>
<thead>
<tr>
<th>Name</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>John, Smith</td>
<td>2000</td>
</tr>
<tr>
<td>Johnathon, Smith</td>
<td>2009</td>
</tr>
<tr>
<td>Joey, Schmidt</td>
<td>2009</td>
</tr>
<tr>
<td>Joe, Miller</td>
<td>2902</td>
</tr>
<tr>
<td>Joseph, Milne</td>
<td>2902</td>
</tr>
<tr>
<td>Paul</td>
<td>3000</td>
</tr>
<tr>
<td>Peter, Jones</td>
<td>3000</td>
</tr>
</tbody>
</table>

Split using <FN, F2>

<table>
<thead>
<tr>
<th>Name</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>John, Smith</td>
<td>2000</td>
</tr>
<tr>
<td>Johnathon, Smith</td>
<td>2009</td>
</tr>
<tr>
<td>Joey, Schmidt</td>
<td>2009</td>
</tr>
<tr>
<td>Joe, Miller</td>
<td>2902</td>
</tr>
<tr>
<td>Joseph, Milne</td>
<td>2902</td>
</tr>
</tbody>
</table>

Merge

<table>
<thead>
<tr>
<th>Name</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>John, Smith</td>
<td>2000</td>
</tr>
<tr>
<td>Johnathon, Smith</td>
<td>2009</td>
</tr>
<tr>
<td>Joey, Schmidt</td>
<td>2009</td>
</tr>
<tr>
<td>Joe, Miller</td>
<td>2902</td>
</tr>
<tr>
<td>Joseph, Milne</td>
<td>2902</td>
</tr>
</tbody>
</table>

Split using <SN, Sdx>

<table>
<thead>
<tr>
<th>Name</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>John, Smith</td>
<td>2000</td>
</tr>
<tr>
<td>Johnathon, Smith</td>
<td>2009</td>
</tr>
<tr>
<td>Joey, Schmidt</td>
<td>2009</td>
</tr>
<tr>
<td>Joe, Miller</td>
<td>2902</td>
</tr>
<tr>
<td>Joseph, Milne</td>
<td>2902</td>
</tr>
</tbody>
</table>

Split using <SN, Sdx>

<table>
<thead>
<tr>
<th>Name</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>John, Smith</td>
<td>2000</td>
</tr>
<tr>
<td>Johnathon, Smith</td>
<td>2009</td>
</tr>
<tr>
<td>Joey, Schmidt</td>
<td>2009</td>
</tr>
<tr>
<td>Joe, Miller</td>
<td>2902</td>
</tr>
<tr>
<td>Joseph, Milne</td>
<td>2902</td>
</tr>
</tbody>
</table>

Blocking Keys = <FN, F2>, <SN, Sdx>

\[ S_{\text{min}} = 2, S_{\text{max}} = 3 \]
Advanced classification techniques

- View record pair classification as a *multi-dimensional binary classification* problem
  - Use all attribute similarities to classify record pairs
  - Only classify into *matches* and *non-matches*

- Many machine learning techniques can be used
  - Supervised: Requires training data (record pairs with known true match and non-match status)
  - Different supervised techniques have been used: *Decision trees, support vector machines, neural networks, learnable string comparisons*, etc.
  - Active and semi-supervised learning
  - Unsupervised: *Clustering*
Classification challenges

In many cases there are no training data available (no data sets with known true match status)

- Possible to use results of earlier matching projects? Or from manual *clerical review* process?
- How confident can we be about correct manual classification of *potential matches*?

No large test data set collection available (like in information retrieval or machine learning)

- Due to privacy and confidentiality concerns
- Therefore much research (in computer science) has been using bibliographic data (author disambiguation)
Advanced classification: Active learning and group linkage

**Active learning**
- Semi-supervised by human-machine interaction
- Overcomes the problem of supervised learning that requires training data
- Selects a sample of record pairs to be manually classified (budget constraints)
- Trains and improves a classification model using manually labelled data

**Group linkage**
- First conduct pair-wise linking of individual records
- Then calculate group similarities using Jaccard or weighted similarities (based on pair-wise similarities)
Advanced classification: Graph-based linkage

- Based on structure between groups of records (for example linking households from different censuses)
  - One graph per household, finds best matching graphs using both record attribute and structural similarities
  - Edge attributes are information that does not change over time (like age differences)

![Diagrams showing graph-based linkage with attribute similarities](attr_sim = 0.63, 0.79, 0.84)
Advanced classification: Collective entity resolution

Considers **relational similarities** not just attribute similarities

(A1, Dave White, Intel)  
(A2, Don White, CMU)  
(A3, Susan Grey, MIT)  
(A4, John Black, MIT)  
(A5, Joe Brown, unknown)  
(A6, Liz Pink, unknown)  

(P1, John Black / Don White)  
(P2, Sue Grey / D. White)  
(P3, Dave White)  
(P4, Don White / Joe Brown)  
(P5, Joe Brown / Liz Pink)  
(P6, Liz Pink / D. White)

Adapted from: [Kalashnikov and Mehrotra, ACM TODS, 2006]
Managing transitive closure

If record $a1$ is classified as matching with record $a2$, and record $a2$ as matching with record $a3$, then records $a1$ and $a3$ must also be matching.

Possibility of chains of linked records occurring

Various algorithms have been developed to find optimal solutions (special clustering algorithms)

Collective classification and clustering approaches deal with this problem by default.
Generating and using synthetic data

- Privacy issues prohibit publication of real personal information
- De-identified or encrypted data cannot be used for record linkage research (as real name and address values are required)

Several advantages of synthetic data

- Volume and characteristics can be controlled (errors and variations in records, number of duplicates, etc.)
- It is known which records are duplicates of each other, and so matching quality can be calculated
- Data and the data generator program can be published (allowing others to repeat experiments)
Modelling of variations and errors

Abbreviations:
cc : character change
wc : word change
subs : substitution
ins : insertion
del : deletion
trans : transpose
repl : replace
ty : typographic
ph : phonetic
attr : attribute
### Example of generated data

<table>
<thead>
<tr>
<th>RecID</th>
<th>Age</th>
<th>FirstName</th>
<th>Surname</th>
<th>Street</th>
<th>Town</th>
</tr>
</thead>
<tbody>
<tr>
<td>rec-1-org</td>
<td>33</td>
<td>Madison</td>
<td>Solomon</td>
<td>Tazewell Circuit</td>
<td>Beechboro</td>
</tr>
<tr>
<td>rec-1-dup-0</td>
<td>33</td>
<td>Madison</td>
<td>Solomon</td>
<td>Tazewell Circ</td>
<td>Beech Boro</td>
</tr>
<tr>
<td>rec-1-dup-1</td>
<td></td>
<td>Madison</td>
<td>Solomon</td>
<td>Tazewell Crct</td>
<td>Bechboro</td>
</tr>
<tr>
<td>rec-2-org</td>
<td>39</td>
<td>Desirae</td>
<td>Contreras</td>
<td>Maltby Street</td>
<td>Burrawang</td>
</tr>
<tr>
<td>rec-2-dup-0</td>
<td>39</td>
<td>Desirae</td>
<td>Contreras</td>
<td>Maltby Street</td>
<td>Burrawang</td>
</tr>
<tr>
<td>rec-2-dup-1</td>
<td>39</td>
<td>Desire</td>
<td>Contreras</td>
<td>Maltby Street</td>
<td>Buahrawang</td>
</tr>
<tr>
<td>rec-3-org</td>
<td>81</td>
<td>Madisyn</td>
<td>Sergeant</td>
<td>Howitt Street</td>
<td>Nangilloc</td>
</tr>
<tr>
<td>rec-3-dup-0</td>
<td>87</td>
<td>Madisyn</td>
<td>Sergeant</td>
<td>Howvitt Street</td>
<td>Nangilloc</td>
</tr>
</tbody>
</table>

- rec-1: typing/abbreviations; rec-2: phonetic; rec-3: OCR

Generated using the *Febrl* and *GeCo* data generators
(see: [https://dmm.anu.edu.au/geco/](https://dmm.anu.edu.au/geco/))
Outline

Part 1: Introduction
  - Applications, history, challenges, and examples

Part 2: record linkage process
  - Key techniques used in record linkage

Part 3: Advanced record linkage techniques
  - Indexing and blocking for scalable record linkage
  - Learning, collective, and graph based techniques

Part 4: Privacy aspects in record linkage
  - Motivating scenario
  - Privacy-preserving record linkage

Conclusions and research directions
Privacy aspects in record linkage

Objective: To link data across organisations such that besides the linked records (the ones classified to refer to the same entities) no information about the sensitive source data can be learned by any party involved in the linking, or any external party.

Main challenges

- Allow for approximate linking of values
- Being able to assess linkage quality and completeness
- Have techniques that are not vulnerable to any kind of attack (frequency, dictionary, crypt-analysis, etc.)
- Have techniques that are scalable to linking large databases across multiple parties
Privacy and record linkage: A motivating scenario

A demographer who aims to investigate how mortgage stress is affecting different people with regard to their mental and physical health.

She will need data from financial institutions, government agencies (social security, health, and education), and private sector providers (such as health insurers).

It is unlikely she will get access to all these databases (for commercial or legal reasons).

She only requires access to some attributes of the records that are linked, but not the actual identities of the linked individuals (but personal details are needed to conduct the actual linkage).
Current best practice approach used in the health domain (1)

- Linking of health data is common in public health (epidemiological) research
- Data are sourced from hospitals, doctors, health insurers, police, governments, etc
- Only identifying data are given to a trusted linkage unit, together with an encrypted identifier
- Once linked, encrypted identifiers are given back to the sources, which ‘attach’ payload data to identifiers and send them to researchers
- Linkage unit does never see payload data
- Researchers do not see personal details
- All communication is encrypted
Current best practice approach used in the health domain (2)

- Step 1: Database owners send partially identifying data to linkage unit
- Step 2: Linkage unit sends linked record identifiers back
- Step 3: Database owners send ‘payload’ data to researchers

Current best practice approach used in the health domain (3)

- Problem with this approach is that the linkage unit needs access to personal details (metadata might also reveal sensitive information)
- Collusion between parties, and internal and external attacks, make these data vulnerable
- Privacy-preserving record linkage (PPRL) aims to overcome these drawbacks
  - No unencoded data ever leave a data source
  - Only some details about matched records are revealed
  - Provable security against different attacks

PPRL is challenging (employs techniques from cryptography, databases, data mining, etc.)
The PPRL process

Database A
Data pre-processing

Database B
Data pre-processing

Privacy-preserving context

Indexing / Searching

Comparison

Classification

Evaluation

Matches
Non-matches

Potential Matches

Encrypted data

Clerical Review
Basic PPRL protocols

Two basic types of protocols

- **Two-party**: Only the two database owners who wish to link their data
- **Three-party**: Use a (trusted) third party (linkage unit) to conduct the linkage (this party will never see any unencoded values, but collusion is possible)

**Multi-party protocols**: Linking records from more than two databases (with or without a linkage unit)
Adversary models

- **Honest-but-curious (HBC) model** assumes that parties follow the protocol while being curious to find about another party’s data
  - HBC model does not prevent collusion
  - Most existing PPRL protocols assume HBC model

- **Malicious model** assumes that parties behave arbitrarily (do not follow the protocol)
  - Protocols under this model often have high complexity

- **Accountable computing and covert model**
  - Allow for proofs if a party has followed the protocol or the misbehaviour can be detected with high probability
  - Lower complexity than malicious and more secure than HBC
**Attack methods**

- **Dictionary attacks**
  An adversary encodes a list of known values using existing encoding functions until a matching encoded value is identified (a keyed encoding approach, like HMAC, can help prevent this attack through a secret password)

- **Frequency attacks**
  Frequency distribution of encoded values is matched with the distribution of known values

- **Cryptanalysis attack**
  A special category of frequency attack applicable to Bloom filter based encoding

- **Collusion**
  A set of parties (in three- or multi-party protocols) collude with the aim to learn about another party’s data
Frequency attack example

If frequency distribution of hash-encoded values closely matches the distribution of values in a (public) database, then ‘re-identification’ of values might be possible.
PPRL techniques

- First generation (mid 1990s): exact matching only using simple hash encoding
- Second generation (early 2000s): approximate matching but not scalable (PP versions of edit distance and other string comparison functions)
- Third generation (mid 2000s): take scalability into account (often a compromise between PP and scalability, some information leakage accepted)
- Different approaches have been developed for PPRL, so far no clear best technique

For example based on Bloom filters, embedding space, generalisation, noise addition, differential privacy, or secure multi-party computation (SMC)
Hash-encoding for PPRL

- A basic building block of many PPRL protocols
- Idea: Use a one-way hash function (like SHA) to encode values, then compare hash-codes
  - Having only access to hash-codes will make it nearly impossible to learn their original input values
  - But dictionary and frequency attacks are possible
- Single character difference between two input values results in completely different hash codes
  - For example:
    - ‘peter’ → ‘101010...100101’ or ‘4R#x+Y4i9!e@t4o’
    - ‘pete’ → ‘011101...011010’ or ‘Z5%o-(7Tq1@?7iE/’
  - Only exact matching is possible
Bloom filter based PPRL (1)

- Proposed by Schnell et al. (Biomed Central, 2009)

- A Bloom filter is a bit-array, where a bit is set to 1 if a hash-function \( H_k(x) \) maps an element \( x \) of a set into this bit (elements in our case are q-grams)

  \[ 0 \leq H_k(x) < l \], with \( l \) the number of bits in Bloom filter

  - Many hash functions can be used (Schnell: \( k = 30 \))
  - Number of bits can be large (Schnell: \( l = 1000 \) bits)

- Basic idea: Map q-grams into Bloom filters using hash functions only known to database owners, send Bloom filters to a third party which calculates Dice coefficient (number of 1-bits in Bloom filters)
Bloom filter based PPRL (2)

1-bits for string ‘peter’: 7, 1-bits for ‘pete’: 5, common 1-bits: 5, therefore $sim_{Dice} = \frac{2 \times 5}{7+5} = \frac{10}{12} = 0.83$

Collisions will effect the calculated similarity values

Number of hash functions and length of Bloom filter need to be carefully chosen
Bloom filters are vulnerable to attacks

<table>
<thead>
<tr>
<th>Plain-text database V</th>
<th>Encoded Bloom filter database B</th>
</tr>
</thead>
<tbody>
<tr>
<td>maude</td>
<td>0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0</td>
</tr>
<tr>
<td>mary</td>
<td>1 0 1 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 1</td>
</tr>
<tr>
<td>max</td>
<td>0 0 0 0 1 0 1 0 1 0 1 0 1 0 0 0 1 0 0 0</td>
</tr>
<tr>
<td>joan</td>
<td>0 0 0 0 1 1 1 0 0 0 0 1 1 0 0 1 0 1 1 0</td>
</tr>
<tr>
<td>john</td>
<td>1 1 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0</td>
</tr>
</tbody>
</table>

Q-gram counts:

- 3: ma
- 2: jo
- 1: an, ar, au, ax, de, hn, oa, oh, ry, ud

(only shown for illustration, but not known to the attacker)

- Based on identifying commonly co-occurring 1-bits
- If \( k \) 1-bit positions co-occur in \( x \) BF\( s \), then they must encode a q-gram that occurs in \( x \) plain-text values
- This attack can be successful even if each Bloom filter in an encoded database is unique
- Ongoing research is developing more resilient encoding techniques as well as new attack methods
Secure multi-party computation

- Compute a function across several parties, such that no party learns the information from the other parties, but all receive the final results
- Simple example: Secure summation $s = \sum_i x_i$.

**Simple example: Secure summation $s = \sum_i x_i$.**

**Step 0:**
- Party 1
  - $x_1 = 55$
  - $Z = 999$

**Step 1:**
- $(Z + x_1) = 1054$

**Step 2:**
- $(Z + x_1) + x_2 = 1127$

**Step 3:**
- $((Z + x_1) + x_2) + x_3 = 1169$

**Step 4:**
- $s = 1169 - Z = 170$

Party 1
- $x_1 = 55$

Party 2
- $x_2 = 73$

Party 3
- $x_3 = 42$
Outline

Part 1: Introduction
- Applications, history, challenges, and examples

Part 2: record linkage process
- Key techniques used in record linkage

Part 3: Advanced record linkage techniques
- Indexing and blocking for scalable record linkage
- Learning, collective, and graph based techniques

Part 4: Privacy aspects in record linkage
- Motivating scenario
- Privacy-preserving record linkage

Conclusions and research directions
Conclusions and research directions (1)

- For historical data, a major challenge is data quality (need for (semi-) automatic data cleaning and standardisation techniques)

- How to employ collective classification techniques for data with personal information?

- No training data available in many applications
  - Employ active learning approaches
  - Visualisation for improved manual clerical review

- Linking data from many sources (significant challenge in PPRL, due to issue of collusion)

- Frameworks for record linkage that allow comparative experimental studies
Conclusions and research directions (2)

- Collections of test data sets which can be used by researchers
  - Challenging (impossible?) to have true match status
  - Challenging because most databases are proprietary and/or sensitive

- Develop provably secure PPRL techniques

- Develop practical PPRL techniques
  - A standard measures for privacy is needed
  - Improved advanced classification techniques for PPRL
  - Methods to assess accuracy and completeness

- Pragmatic challenge: Collaborations across multiple research disciplines
The book details the possibilities and limitations of information technology with respect to reasoning for population reconstruction.

Follows the three main processing phases from handwritten registers to a reconstructed digitized population.

Combines research from historians, social scientists, linguists, and computer scientists.
References (1)


Bouzelat H, Quantin C, and Dusserre L: *Extraction and anonymity protocol of medical file*. AMIA Fall Symposium, 1996.


Christen P: *Privacy-preserving data linkage and geocoding: Current approaches and research directions*. PADM held at IEEE ICDM, Hong Kong, 2006.


References (4)


Churches T: *A proposed architecture and method of operation for improving the protection of privacy and confidentiality in disease registers*. BMC Medical Research Methodology, 3(1), 2003.


Dusserre L, Quantin C and Bouzelat H: *A one way public key cryptosystem for the linkage of nominal files in epidemiological studies*. Medinfo, 8:644-7, 1995.


References (6)


References (7)


References (8)

References (9)

References (10)


References (12)

References (13)


References (14)


Tran KN, Vatsalan D and Christen P: GeCo: an online personal data generator and corruptor. CIKM, 2013.


References (15)

References (16)