Blind data linkage using *n*-gram similarity comparisons

Tim Churches and *Peter Christen*

Data Mining Group, Australian National University

Centre for Epidemiology and Research, New South Wales Department of Health

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

Project web page: **http://datamining.anu.edu.au/linkage.html**

Outline

- Data linkage
- Privacy and confidentiality issues
- Methods and protocol
- Blind data linkage
- Outlook

Data linkage

- The task of linking together information from one or more data sources representing the same entity (patient, customer, business, gene sequence, etc.)
- If no *unique identifier* is available, *probabilistic linkage techniques* have to be applied
- Real world data is often *dirty*
  - Missing values
  - Typographical and other errors
  - Different coding schemes / formats
  - Out-of-date data
- Names and addresses are especially prone to data entry errors

Data linkage techniques

- Deterministic or exact linkage
  - A *unique identifier* is needed, which is of high quality (precise, robust, stable over time, highly available)
  - For example *Medicare* number (?)
- Probabilistic linkage (*Fellegi & Sunter*, 1969)
  - Apply linkage using available (personal) information
  - Examples: *name*, *address*, *date of birth*
- Other techniques (rule-based, fuzzy approach, information retrieval)

Privacy and confidentiality issues

- Traditionally data linkage requires that identified data is being given to the person or institution doing the linkage
- Privacy of individuals in data sets is invaded
  - Consent of individuals involved is needed
  - Alternatively, approval from ethics committees

Invasion of privacy could be avoided (or mitigated) if some method were available to determine which records in two data sets match without revealing any identifying information.

Methods

- Alice has database **A**, with attributes **A.a**, **A.b**, etc.
- Bob has database **B**, with attributes **B.a**, **B.b**, etc.
- Alice and Bob wish to determine whether any of the values in **A.a** match any of the values in **B.a**, without revealing the actual values in **A.a** and **B.a**
- Easy if only *exact matches* are considered
  (use one-way message authentication digests (HMAC) based on secure one-way hashing like SHA or MD5)
- More complicated if values contain errors or typographical variations
  (even a single character difference between two strings will result in very different message digest values)

Protocol – I

- A protocol is required which permits the *blind* calculation by a trusted third party (Carol) of a more general and robust measure of similarity between pairs of secret strings
- Proposed protocol is based on n-grams
  For example ($n = 2$ bigrams): *'peter'* $\rightarrow$ *('pe','et','te','er')*
- Protocol step 1
  - Alice and Bob agree on a secret random key
  - They also agree on a secure one-way message authentication algorithm (HMAC)
  - They also agree on a standard of preprocessing strings

Protocol – II

- Protocol step 2
  - Alice computes a sorted list of n-grams for each of her values in **A.a**
  - Next she calculates all possible sub-lists with length larger than 0 (power-set without empty set)
    For example: *'peter'* $\rightarrow$
    ('er'), ('et'), ('pe'), ('te'),
    ('er','et'), ('er','pe'), ('er','te'), ('et','pe'), ('et','te'), ('pe','te'),
    ('er','et','pe'), ('er','et','te'), ('er','pe','te'), ('et','pe','te'),
    ('er','et','pe','te')
  - Then she transforms each sub-list into a secure hash digest and stores these in **A.a_hash_bigr_comb**

Protocol – III

- Protocol step 2 (continued)
  - Alice computes encrypted version of the record identifier and stores it in **A.a_encrypt_rec_key**
  - Next she places the number of bigrams of each **A.a_hash_bigr_comb** into **A.a_hash_bigr_comb_len**
  - She then places the length (total number of bigrams) of each original string into **A.a_len**
  - Alice then sends the quadruplet [**A.a_encrypt_rec_key**, **A.a_hash_bigr_comb**, **A.a_hash_bigr_comb_len**, **A.a_len**] to Carol
- Protocol step 3
  - Bob carries out the same as in step 2 with his **B.a**

Protocol – IV

- Protocol step 4
  - For each value of **a_hash_bigr_comb** shared by **A** and **B**, for each unique pairing of [**A.a_encrypt_rec_key**, **B.a_encrypt_rec_key**], Carol calculates a *bigram score*

$$\textbf{bigr\_score} = \frac{2 \cdot \textbf{A.a\_hash\_bigr\_comb\_len}}{(\textbf{A.a\_len} + \textbf{B.a\_len})}$$

  - Carol then selects the maximum **bigr_score** for each pairing [**A.a_encrypt_rec_key**, **B.a_encrypt_rec_key**] and sends these results to Alice and Bob

Blind data linkage

- Several attributes **a**, **b**, **c**, etc. can be compared independently (by different Carols)
- Different Carols send their results to a third party (David), who forms a (sparse) matrix by joining the results
- The final *matching weight* for a record pair is calculated using individual **bigr_score**s
- David arrives at a set of *blindly linked records* (pairs of [**A.a_encrypt_rec_key**, **B.a_encrypt_rec_key**])

Outlook

- *Blind record linkage* – including matching of typographical variations – is possible
- Proof-of-concept implementation available from the authors (implemented in *Python*)
- Future work
  - Implementation within our data linkage system *Febrl* (Freely extensible biomedical record linkage)
  - Improvement of performance (reduction of data communication volume)
  - Extension to numerical and other data (times, dates)
  - Development of blind geocoding methods