# Probabilistic Deduplication, Data Linkage and Geocoding

Peter Christen

Data Mining Group, Australian National University
in collaboration with
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

Funded by the ANU, the NSW Department of Health, the Australian Research Council (ARC), and the Australian Partnership for Advanced Computing (APAC)



- Data cleaning and standardisation
- Data linkage and probabilistic linkage
- Privacy and ethics
- *Febrl* overview and open source tools
- Probabilistic data cleaning and standardisation
- Blocking and indexing
- Record pair classification
- Parallelisation in *Febrl*
- Data set generation
- Geocoding

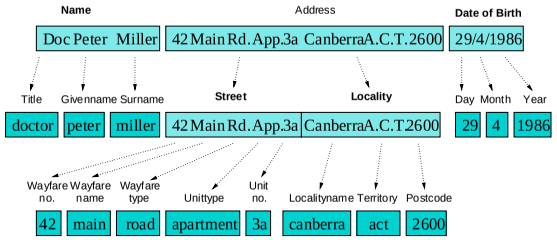


## Data cleaning and standardisation (1)

- Real world data is often *dirty*
  - Missing values, inconsistencies
  - Typographical and other errors
  - Different coding schemes / formats
  - Out-of-date data
- Names and addresses are especially prone to data entry errors
- Cleaned and standardised data is needed for
  - Loading into databases and data warehouses
  - Data mining and other data analysis studies
  - Data linkage and data integration



## Data cleaning and standardisation (2)




- Remove unwanted characters and words
- Expand abbreviations and correct misspellings
- Segment data into well defined *output fields*



## Data linkage

- Data (or record) linkage is the task of linking together information from one or more data sources representing the same entity
- Data linkage is also called *database matching*, *data integration*, *data scrubbing*, or *ETL (extraction, transformation and loading)*
- Three records, which represent the same person?

1. *Dr Smith, Peter; 42 Miller Street 2602 O'Connor*
2. *Pete Smith; 42 Miller St 2600 Canberra A.C.T.*
3. *P. Smithers, 24 Mill Street 2600 Canberra ACT*



## 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*, *ABN* or *Tax file* number (are they *really* unique, stable, trustworthy?)
- Probabilistic linkage (*Fellegi & Sunter*, 1969)
  - Apply linkage using available (personal) information (for example *names*, *addresses*, *dates of birth*, etc)
- Other techniques
  (rule-based, fuzzy approach, information retrieval, AI)



## Probabilistic linkage

- Computer assisted data 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)
  - Using matching weights based on frequency ratios (global or value specific ratios)
  - Compute matching weights for all fields used in linkage
  - Summation of matching weights is used to designate a pair of records as *link*, *possible-link* or *non-link*

## Approximate string comparison

- Account for partial agreement between strings
- Return score between *0.0* (totally different) and *1.0* (exactly the same)
- Examples:

| String_1 | String_2 | Winkler | Bigram | Edit-Dist |
|---|---|---|---|---|
| tanya | tonya | 0.880 | 0.500 | 0.800 |
| dwayne | duane | 0.840 | 0.222 | 0.667 |
| sean | susan | 0.805 | 0.286 | 0.600 |
| jon | john | 0.933 | 0.400 | 0.750 |
| itman | smith | 0.567 | 0.250 | 0.000 |
| 1st | ist | 0.778 | 0.500 | 0.667 |
| peter | pete | 0.911 | 0.800 | 0.800 |

- Bringing together spellings variations of the same name
- Examples:

| Name | Soundex | NYSIIS | Double-Metaphone |
|---|---|---|---|
| stephen | s315 | staf | stfn |
| steve | s310 | staf | stf |
| gail | g400 | gal | kl |
| gayle | g400 | gal | kl |
| christine | c623 | chra | krst |
| christina | c623 | chra | krst |
| kristina | k623 | cras | krst |



- Assume two data sets with a $3\%$ error in field *month of birth*
- Probability that two linked records (that represent the same person) have the same month value is $97\%$ *(L agreement)*
- Probability that two linked records do not have the same month value is $3\%$ *(L disagreement)*
- Probability that two (randomly picked) unlinked records have the same month value is $1/12 = 8.3\%$ *(U agreement)*
- Probability that two unlinked records do not have the same month value is $11/12 = 91.7\%$ *(U disagreement)*
- Agreement weight $(L_{ag}/U_{ag})$: $\log_2(0.97/0.083) = 3.54$
  Disagreement weight $(L_{di}/U_{di})$: $\log_2(0.03/0.917) = -4.92$



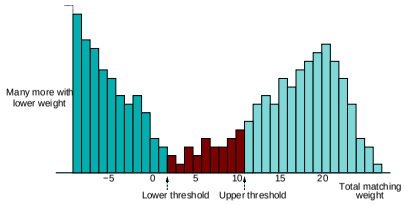
## *Value specific frequencies*

- Example: Surnames
  - Assume the frequency of *Smith* is higher than *Dijkstra* (*NSW Whitepages*: 25,425 *Smith*, only 3 *Dijkstra*)
  - Two records with surname *Dijkstra* are more likely to be the same person than with surname *Smith*
- The matching weights need to be adjusted
  - Difficulty: How to get value specific frequencies that are characteristic for a given data set
  - Earlier linkages done on same or similar data
  - Information from external data sets (e.g. *Australian Whitepages*)



## *Final linkage decision*

- The final weight is the sum of weights of all fields
  - Record pairs with a weight above an *upper threshold* are designated as a *link*
  - Record pairs with a weight below a *lower threshold* are designated as a *non-link*
  - Record pairs with a weight between are *possible link*






## *Applications and usage*

- Applications of data linkage
  - Remove duplicates in a data set (internal linkage)
  - Merge new records into a larger master data set
  - Create patient or customer oriented statistics
  - Compile data for longitudinal (over time) studies
  - Clean data sets for data analysis and mining projects
  - Geocode data
- Widespread use of data linkage
  - Census statistics
  - Business mailing lists
  - Health and biomedical research (epidemiology)
  - Fraud and crime detection



## *Privacy and ethics*

- For some applications, personal information is not of interest and is removed from the linked data set (for example epidemiology, census statistics, data mining)
- In other areas, the linked information is the aim (for example business mailing lists, crime and fraud detection, data surveillance)
- Personal privacy and ethics is most important
  - *Privacy Act*, 1988
  - *National Statement on Ethical Conduct in Research Involving Humans*, 1999



## *Febrl – Freely extensible biomedical record linkage*

- Commercial software for data linkage is often expensive and cumbersome to use
- Project aims
  - Allow linkage of larger data sets (high-performance and parallel computing techniques)
  - Reduce the amount of human resources needed (improve linkage quality by using machine learning)
  - Reduce costs (free open source software)
- Modules for data cleaning and standardisation, data linkage, deduplication and geocoding
- Free, open source **https://sourceforge.net/projects/febrl/**

## *Open source software tools*

- Scripting language *Python* **www.python.org**
  - Easy and rapid prototype software development
  - Object-oriented and cross-platform *(Unix, Win, Mac)*
  - Can handle large data sets stable and efficiently
  - Many external modules, easy to extend
  - Large user community
- Parallel libraries *MPI* and *OpenMP*
  - Widespread use in high-performance computing (quasi standards) $\Rightarrow$ Portability and availability
  - Parallel *Python* extensions: *PyRO* and *PyPar*

- Three step approach in *Febrl*
  1. Cleaning
     – Based on look-up tables and correction lists
     – Remove unwanted characters and words
     – Correct various misspellings and abbreviations
  2. Tagging
     – Split input into a list of words, numbers and separators
     – Assign one or more tags to each element of this list (using look-up tables and some hard-coded rules)
  3. Segmenting
     – Use either rules or a *hidden Markov model (HMM)* to assign list elements to *output fields*



- Assume the input *component* is one string (either name or address – dates are processed differently)
- Convert all letters into lower case
- Use *correction lists* which contain pairs of `(original,replacement)` strings
- An empty `replacement` string results in removing the original string
- Correction lists are stored in text files and can be modified by the user
- Different correction lists for names and addresses



## Step 2: Tagging

- Cleaned strings are split at whitespace boundaries into lists of words, numbers, characters, etc.
- Using *look-up tables* and some hard-coded rules, each element is tagged with one or more *tags*
- Example:
  - Uncleaned input string: *"Doc. peter Paul MILLER"*
  - Cleaned string: *"dr peter paul miller"*
  - Word and tag lists:
    `['dr', 'peter', 'paul', 'miller']`
    `['TI', 'GM/SN', 'GM', 'SN']`

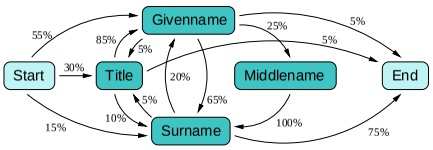


## Step 3: Segmenting

- Using the tag list, assign elements in the word list to the appropriate *output fields*
- Rules based approach (e.g. *AutoStan*)
  - Example: *"if an element has tag 'TI' then assign the corresponding word to the 'Title' output field"*
  - Hard to develop and maintain rules
  - Different sets of rules needed for different data sets
- Hidden Markov model (HMM) approach
  - A machine learning technique (supervised learning)
  - Training data is needed to build HMMs



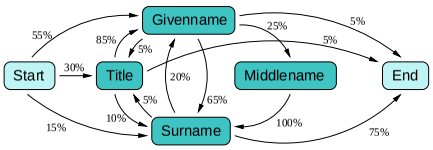
## Hidden Markov model (HMM)




- 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
  - In our approach, the observation symbols are *tags* and the states correspond to the *output fields*



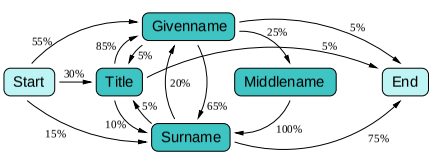
## HMM probability matrices




| | State | | | | | |
|---|---|---|---|---|---|---|
| **Observation** | *Start* | Title | Givenname | Middlename | Surname | *End* |
| TI | – | 96% | 1% | 1% | 1% | – |
| GM | – | 1% | 35% | 33% | 15% | – |
| GF | – | 1% | 35% | 27% | 14% | – |
| SN | – | 1% | 9% | 14% | 45% | – |
| UN | – | 1% | 20% | 25% | 25% | – |

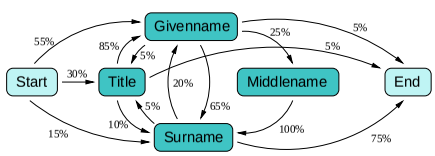


## HMM data segmentation

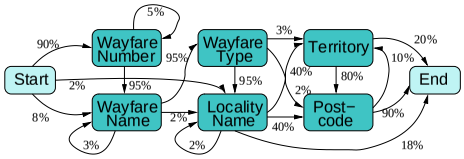



- For an observation sequence we are interested in the most likely path through a given HMM (in our case an observation sequence is a *tag list*)
- The *Viterbi* algorithm is used for this task (a dynamic programming approach)
- *Smoothing* is applied to account for unseen data (assign small probabilities for unseen observation symbols)

## HMM segmentation example




- Input word and tag list
  `['dr', 'peter', 'paul', 'miller']`
  `['TI', 'GM/SN', 'GM', 'SN']`
- Two example paths through the HMM
  1: `Start -> Title (TI) -> Givenname (GM) -> Middlename (GM) -> Surname (SN) -> End`
  2: `Start -> Title (TI) -> Surname (SN) -> Givenname (GM) -> Surname (SN) -> End`




1. Raw input: *'73 Miller St, NORTH SYDENY 2060'*
   Cleaned into: *'73 miller street north sydney 2060'*
2. Word and tag lists:
   `['73', 'miller', 'street', 'north_sydney', '2060']`
   `['NU', 'UN', 'WT', 'LN', 'PC' ]`
3. Example path through HMM
   `Start -> Wayfare Number (NU) -> Wayfare Name (UN) -> Wayfare Type (WT) -> Locality Name (LN) -> Postcode (PC) -> End`



- Both transition and observation probabilities need to be trained using *training data*
  (maximum likelihood estimates (MLE) are derived by accumulating frequency counts for transitions and observations)
- Training data consists of records, each being a sequence of `tag:hmm_state` pairs
- Example (2 training records):

```
# '42 / 131 miller place manly 2095 new_south_wales'
NU:unnu,SL:sla,NU:wfnu,UN:wfna1,WT:wfty,LN:loc1,PC:pc,TR:ter1

# '2 richard street lewisham 2049 new_south_wales'
NU:wfnu,UN:wfna1,WT:wfty,LN:loc1,PC:pc,TR:ter1
```



## HMM training (2)

- A *bootstrapping* approach is applied for semi-automatic training
  1. Manually edit a small number of training records and train a first rough HMM
  2. Use this first HMM to segment and tag a larger number of training records
  3. Manually check a second set of training records, then train an improved HMM
- Only a few person days are needed to get a HMM that results in an accurate standardisation
  (instead of weeks or even months to develop rules)



## Address standardisation results

- Various NSW Health data sets
  - *HMM1* trained on 1,450 Death Certificate records
  - *HMM2* contains *HMM1* plus 1,000 Midwifes Data Collection training records
  - *HMM3* is *HMM2* plus 60 unusual training records
- *AutoStan* rules (for ISC) developed over years

| | HMM/Method | | | |
|---|---|---|---|---|
| **Test Data Set** | HMM | HMM | HMM | Auto |
| (1,000 records each) | 1 | 2 | 3 | Stan |
| Death Certificates | 95.7% | 96.8% | 97.6% | 91.5% |
| Inpatient Statistics Collection | 95.7% | 95.9% | 97.4% | 95.3% |



## Name standardisation results

- NSW Midwifes Data Collection (1990 - 2000)
  (around 963,000 records, no medical information)
- 10-fold cross-validation study with 10,000 random records (9,000 training and 1,000 test records)
- Both *Febrl* rule based and HMM data cleaning and standardisation
  - Rules were better because most names were simple (not much structure to learn for HMM)

| | Min | Max | Average | StdDev |
|---|---|---|---|---|
| HMM | 83.1% | 97.0% | 92.0% | $\pm$4.7% |
| Rules | 97.1% | 99.7% | 98.2% | $\pm$0.7% |



## Blocking and indexing

- Number of possible links equals the product of the sizes of the two data sets to be linked
  (two databases with 1,000,000 and 5,000,000 records will result in 1,000,000 $\times$ 5,000,000 = $5 \times 10^{12}$ = 5 Trillion record pair comparisons)
- Performance bottleneck is the (expensive) comparison of field values (similarity measures) between record pairs
- Blocking / indexing / filtering techniques are used to reduce the large amount of record comparisons
  (for example, only compare records which have the same *postcode* value)



## Field comparison functions in Febrl

- Exact string
- Truncated string (only consider beginning of strings)
- Approximate string (using *Winkler*, *Edit dist*, *Bigram* etc.)
- Encoded string (using *Soundex*, *NYSIIS*, etc.)
- Keying difference (allow a number of different characters)
- Numeric percentage (allowing percentage tolerance)
- Numeric absolute (allow absolute tolerance)
- Date (allow day tolerance)
- Age (allow percentage tolerance)
- Time (allow minute tolerance)

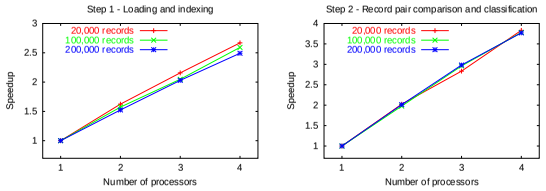
## Record pair classification

- For each record pair compared a vector containing *matching weights* is calculated

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

- Matching weights are used to classify record pairs as *links*, *non-links*, or *possible links*
- *Fellegi & Sunter* classifier simply sums all the weights, then uses two thresholds to classify
- Improved classifiers are possible
  (for example using machine learning techniques)

- Implemented transparently to the user
- Currently using *MPI* via Python module *PyPar*
- Use of super-computing centres is problematic (privacy) → Alternative: *In-house office clusters*
- Some initial performance results (on *Sun SMP*)






- Difficult to acquire data for testing and evaluation (as data linkage deals with names and addresses)
- Also, linkage status is often not known (hard to evaluate and test new algorithms)
- *Febrl* contains a data set generator
  - Uses frequency tables for given- and surnames, street names and types, suburbs, postcodes, etc.
  - *Duplicate records* are created via random introduction of modifications (like insert/delete/transpose characters, swap field values, delete values, etc.)
  - Also uses lists of known misspellings



## Data set generation – Example

- Data set with 4 original and 6 duplicate records

| REC_ID | ADDRESS1 | ADDRESS2 | SUBURB |
|---|---|---|---|
| rec-0-org, | wylly place, | pine ret vill, | taree |
| rec-0-dup-0, | wyllyplace, | pine ret vill, | taree |
| rec-0-dup-1, | pine ret vill, | wylly place, | taree |
| rec-0-dup-2, | wylly place, | pine ret vill, | tared |
| rec-0-dup-3, | wylly parade, | pine ret vill, | taree |
| rec-1-org, | stuart street, | hartford, | menton |
| rec-2-org, | griffiths street, | myross, | kilda |
| rec-2-dup-0, | griffith sstreet, | myross, | kilda |
| rec-2-dup-1, | griffith street, | mycross, | kilda |
| rec-3-org, | ellenborough place, | kalkite homestead, | sydney |

- Each record is given a unique identifier, which allows the evaluation of accuracy and error rates for data linkage

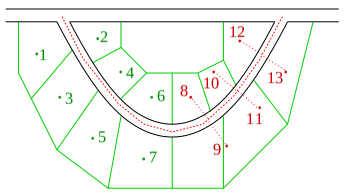


## Geocoding

- The process of matching addresses with geographic locations (longitude and latitude)
- It is estimated that 80% to 90% of governmental and business data contain address information *(US Federal Geographic Data Committee)*
- Geocoding tasks
  - Pre-process the geocoded reference data (cleaning, standardisation and indexing)
  - Clean and standardise the user addresses
  - (Approximate) matching of user addresses with the reference data



## Geocoding techniques

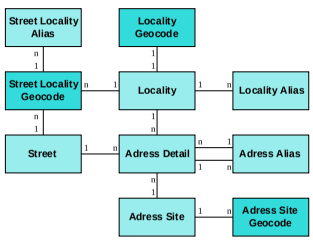



- Street centreline based (many commercial systems)
- Property parcel centre based (our approach)
- A recent study found substantial differences (specially in rural areas)
  *Cayo and Talbot; Int. Journal of Health Geographics, 2003*



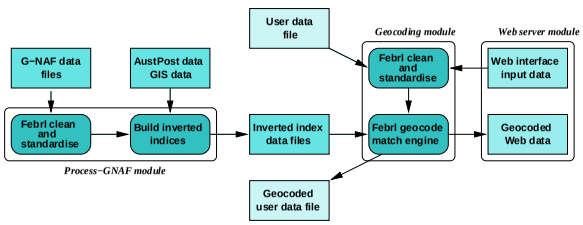
## Geocoded national address file

- G-NAF: Available since early 2004 (PSMA, `http://www.g-naf.com.au/`)
- Source data from 13 organisations (around 32 million source records)
- Processed into 22 normalised database tables






## Febrl geocoding system



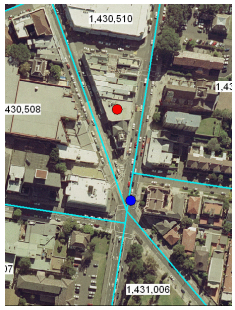

- Only NSW G-NAF data available (around 4 million address, 58,000 street and 5,000 locality records)
- Additional Australia Post and GIS data used

## Additional data files

- Use external *Australia Post* postcode and suburb look-up tables for correcting and imputing (e.g. if a suburb has a unique postcode this value can be imputed if missing, or corrected if wrong)
- Use boundary files for postcodes and suburbs to build *neighbouring region* lists
  - Idea: People often record neighbouring suburb or postcode if it has a higher perceived social status
  - Create lists for direct and indirect neighbours (neighbouring levels 1 and 2)

- Uses cleaned and standardised user address(es) and G-NAF inverted index data
- Fuzzy rule based approach
  1. Find street match set (street name, type and number)
  2. Find postcode and locality match set (with *no*, then *direct*, then *indirect* neighbour levels)
  3. Intersect postcode and locality sets with street match set (if no match increase neighbour level and go back to 2.)
  4. Refine with unit, property, and building match sets
  5. Retrieve corresponding location (or locations)
  6. Return location and match status (address, street or locality level match; none, one or many matches)




- Red dots: *Febrl* geocoding (G-NAF based)
- Blue dots: Street centreline based geocoding



## *Outlook*

- Several research areas
  - Improving probabilistic data standardisation
  - New and improved blocking / indexing methods
  - Apply machine learning techniques for record pair classification
  - Improve performances (scalability and parallelism)
- Project web page

  **http://datamining.anu.edu.au/linkage.html**

Febrl *is an ideal experimental platform to develop, implement and evaluate new data standardisation and data linkage algorithms and techniques*



## *Contributions / Acknowledgements*

- Dr Tim Churches (New South Wales Health Department, Centre for Epidemiology and Research)
- Dr Markus Hegland (ANU Mathematical Sciences Institute)
- Dr Lee Taylor (New South Wales Health Department, Centre for Epidemiology and Research)
- Ms Kim Lim (New South Wales Health Department, Centre for Epidemiology and Research)
- Mr Alan Willmore (New South Wales Health Department, Centre for Epidemiology and Research)
- Mr Karl Goiser (ANU Computer Science PhD student)
- Mr Puthick Hok (ANU Computer Science honours student, 2004)
- Mr Justin Zhu (ANU Computer Science honours student, 2002)
- Mr David Horgan (ANU Computer Science summer student, 2003/2004)