Data Linkage Research at the ANU

Peter Christen

Department of Computer Science,
Faculty of Engineering and Information Technology,
ANU College of Engineering and Computer Science,
The Australian National University

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

Project Web site: **http://datamining.anu.edu.au/linkage.html**

Funded by the Australian National University, the NSW Department of Health,
and the Australian Research Council (ARC) under Linkage Project 0453463.

Outline

- Short introduction to data linkage
- Improving indexing and classification
- Probabilistic name and address cleaning and standardisation
- Privacy preserving data linkage
- Our project: *Febrl*
  (Freely extensible biomedical record linkage)
- Outlook
- Additional material:
  - Measures for linkage quality and complexity
  - Geocoding

What is data (or record) linkage?

- The process of linking and aggregating records from one or more data sources representing the same entity (patient, customer, business name, etc.)
- Also called *data matching*, *data integration*, *data scrubbing*, *ETL (extraction, transformation and loading)*, *object identification*, *merge-purge*, etc.
- Challenging if no unique entity identifiers available
  E.g., which of these records represent the same person?

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

Data linkage techniques

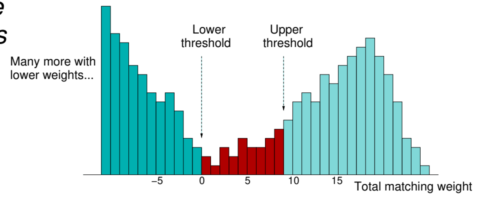
- Deterministic linkage
  - Exact linkage (if a *unique identifier* of high quality is available: precise, robust, stable over time)
    Examples: *Medicare*, *ABN* or *Tax file* number (??)
  - Rules based linkage (complex to build and maintain)
- Probabilistic linkage
  Use available (personal) information for linkage (which can be missing, wrong, coded differently, out-of-date, etc.)
  Examples: *names*, *addresses*, *dates of birth*, etc.
- Modern approaches
  Based on machine learning, data mining, AI and information retrieval techniques

Probabilistic data 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*
  - Compare common record attributes (or fields)
  - Compute 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 *match*, *non-match*, or *possible match*
  - Problems: Estimating errors, find optimal thresholds, assumption of independence, and manual *clerical review*

Fellegi and Sunter classification

- For each compared record pair a vector containing *matching weights* is calculated
  Record A: `['dr', 'peter', 'paul', 'miller']`
  Record B: `['mr', 'john', '', 'miller']`
  Matching weights: `[0.2, -3.2, 0.0, 2.4 ]`
- Sum weights in vector, then use two thresholds to classify record pairs as *matches*, *non-matches*, or *possible matches*



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% (L agreement)*
- Probability that two matched records do not have the same month value is *3% (L disagreement)*
- Probability that two (randomly picked) un-matched records have the same month value is *1/12 = 8.3% (U agreement)*
- Probability that two un-matched 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}): $\log_2(0.03 / 0.917) = -4.92$

Why blocking / indexing / filtering?

- Number of record pair comparisons equals the product of the sizes of the two data sets
  (linking two data sets with 1 and 5 million records will result in $1{,}000{,}000 \times 5{,}000{,}000 = 5 \times 10^{12}$ record pairs)
- Performance bottleneck in a data linkage system is usually the (expensive) comparison of field values between record pairs
  (similarity measures or field 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 which 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 be uniformly distributed (as the most frequent values determine the size of the largest blocks)
    Example: Frequency of *'Smith'* in NSW: *25,425*

Outline: Improved techniques

- Short introduction to data linkage
- Improving indexing and classification
- Probabilistic name and address cleaning and standardisation
- Privacy preserving data linkage
- Our project: *Febrl*
  (Freely extensible biomedical record linkage)
- Outlook
- Additional material:
  - Measures for linkage quality and complexity
  - Geocoding

Recent indexing approaches (1)

- Sorted neighbourhood approach
  Sliding window over sorted blocking variable
  Use several passes with different blocking variables
- *Q*-gram based blocking (e.g. *2-grams / bigrams*)
  Convert values into *q*-gram lists, then generate sub-lists
  'peter' → *['pe','et','te','er']*, ***['pe','et','te']***, *['pe','et','er']*, *...*
  'pete' → ***['pe','et','te']***, *['pe','et']*, *['pe','te']*, *['et','te']*, *...*
  Each record will be inserted into several blocks
- Overlapping *canopy* clustering
  Based on *q*-grams and a 'cheap' similarity measure, such as Jaccard or TF-IDF/cosine
  Records will be inserted into several clusters, use global thresholds for cluster similarities

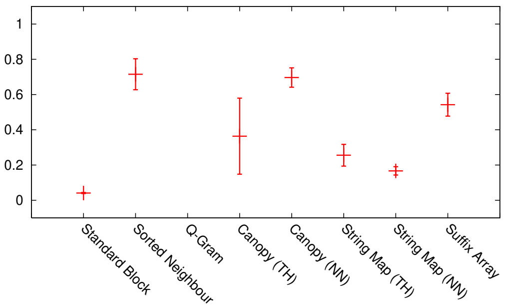
Recent indexing approaches (2)

- *StringMap* based blocking
  Map strings into a multi-dimensional space (d = *15...20*) such that distances between pairs of strings are preserved
  Use similarity join to find similar pairs
- Suffix array based blocking
  Generate suffix array based inverted index
  Only use values longer than minimum length
  Suffix array: *'peter'* → *'eter', 'ter', 'er', 'r'*
- Post-blocking filtering
  For example, string length or *q*-grams count differences
- US Census Bureau: *BigMatch*
  Pre-process 'smaller' data set so its values can be directly accessed; with all blocking passes in one go

How good are recent approaches?

- No experimental comparisons of recent indexing techniques have so far been published

Pairs completeness for dirty data sets and concatenated blocking key.



Improved record pair classification

- *Fellegi & Sunter* summing of weights results in loss of information
- View record pair classification as a *multi-dimensional binary classification* problem
  (use weight vector to classify record pairs a *matches* or *non-matches*, but no *possible matches*)
- Many machine learning techniques can be used
  - Supervised: *Decision trees*, *neural networks*, *learnable string comparisons*, *active learning*, etc.
  - Un-supervised: Various *clustering* algorithms
- Recently, *collective* entity resolution techniques have been investigated (rather than classifying each record pair independently)

Classification challenges

- In many cases there is no training data available
  - Possible to use results of earlier linkage projects? Or from manual *clerical review* process?
  - How confident can we be about correct manual classification of *possible links*?
- Often there is no *gold standard* available
  (no data sets with true known linkage status)
- No large test data set collection available
  (like in information retrieval or machine learning)
- Recent small repository: *RIDDLE*
  http://www.cs.utexas.edu/users/ml/riddle/
  (Repository of Information on Duplicate Detection, Record Linkage, and Identity Uncertainty)

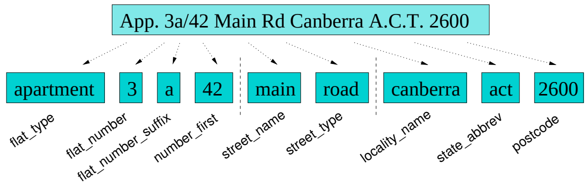
Outline: Probabilistic data cleaning

- Short introduction to data linkage
- Improving indexing and classification
- Probabilistic name and address cleaning and standardisation
- Privacy preserving data linkage
- Our project: *Febrl*
  (Freely extensible biomedical record linkage)
- Outlook
- Additional material:
  - Measures for linkage quality and complexity
  - Geocoding

Why cleaning and standardisation?

- Real world data is 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'*)

Address standardisation tasks



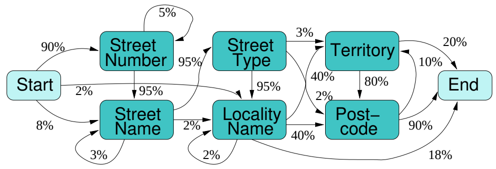
- Clean input
  - Remove unwanted characters and words
  - Expand abbreviations and correct misspellings
- Segment address into well defined *output fields*
- Verify if address (or parts of it) exists in reality

Address standardisation approaches

- Traditionally: Rules based
  - Manually developed parsing and transformation rules
  - Time consuming and complex to develop and maintain
- Recently: Probabilistic methods
  - Mainly based on *hidden Markov models* (HMMs)
  - More flexible and robust with regard to new unseen data
  - Drawback: Training data needed for most methods

HMMs are widely used in natural language processing and speech recognition, as well as for text segmentation and information extraction.

What is a Hidden Markov model?



- 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
  - In our approach, the states correspond to *output fields*

Probabilistic address standardisation

- Segmentation of Indian and US addresses *(Borkar, Deshmukh & Sarawagi, 2001)*
  - Hierarchical features and nested HMMs
  - Allow the integration of external hierarchical databases for improved segmentation
  - Presented results better than rules-based system *Rapier*
- Attribute recognition models *(Agichtein & Ganti, 2004)*
  - Automatic system only using an external database
  - Based on HMMs, capture the characteristics of values in database
  - Feature hierarchies are used to learn the HMM topology and probabilities

Our standardisation approach

- Based on our previous work *(BioMed Central 2002)*
  - Uses lexicon-based tokenisation rather than original values as HMM observation symbols
  - Manually compiled look-up tables
  - Manual preparation of training data needed
  - Better results than rule-based system *AutoStan*
- More recent contributions *(AusDM 2005)*
  - Build initial HMM structure from postal guidelines
  - Automatically create HMM training data using initial HMM structure and a national address database
  - Automatically create look-up tables from address database

Address standardisation steps

- Three step approach
  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/or features)
  3. Segmenting
     – Use a trained HMM to assign list elements to *output fields*

Tagging step

- Tags are based on look-up tables and features
  - If found in look-up tables for street name (`SN`), street type (`ST`), locality name (`LN`), postcode (`PC`), etc.
  - Otherwise according to more general features
- Features characterise values
  - If a value contains letters (`L`), numbers (`N`), alpha-numerics (`A`), or is mixed (`M`)
  - The length of a value (`1`, `2`, ... , `6_8`, `9_11`, `12_15`, `16+`)
- Examples:

`'avenue'` will be tagged with `'ST'` and `'L6_8'`
`'2602'` will be tagged with `'PC'` and `'N4'`
`'12b'` will be tagged with `'A3'`

Example address standardisation

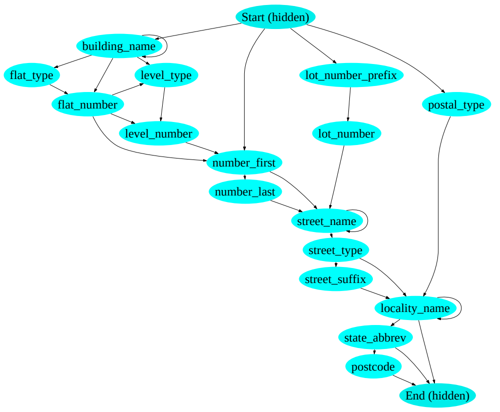
- Raw input address: `'42 meyer Rd COOMA 2371'`
- Cleaned into: `'42 meyer road cooma 2371'`
- Tagged (both look-up tables and feature tags):
  `['42','meyer','road', 'cooma', '2371' ]`
  `['N2','SN/L5','ST/L4','LN/SN/L5','PC/N4']`
- Segmented by HMM into *output fields*:

```
 number_first :  '42'
  street_name :  'meyer'
  street_type :  'road'
locality_name :  'cooma'
     postcode :  '2371'
```


Preparation and training phase

- Initial HMM structure is built using national postal guidelines *(Australia Post, AS4212-1994, AS4590-1999)*
  - Currently manual, in future XML scheme likely
- Records from a comprehensive address database are used as HMM training records
  - We use G-NAF (Geocoded National Address File) with around 4.5 million addresses from NSW
  - Contains clean and segmented records (26 attributes)
  - Missing are postal addresses and many postcodes, as well as characters like slash (/) and hyphen (–)

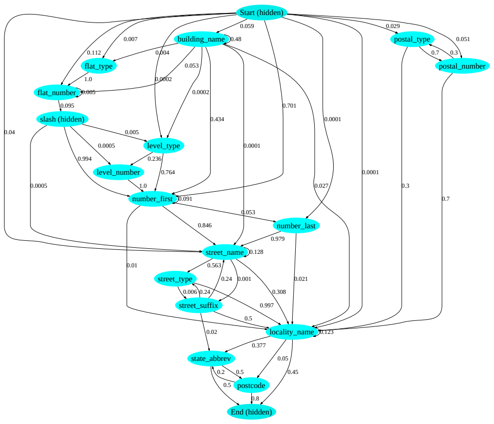
Initial HMM structure (simplified)



Automated HMM training

- Address records are re-ordered according to topologically sorted initial HMM structure
- Various tweaks need to be done
  Insert postcodes, postal addresses, slash, hyphen, etc.
- HMM observation symbols are tags
  Either features only (**F**), look-ups only (**LT**) or both look-ups and features (**LT&F**)
- Processed records are then used for HMM training
  Smoothing is used to make HMM more robust towards unseen data in the standardisation phase
- Look-up tables are built for name attributes
  (and merged into existing tables)

Final HMM (simplified)



Some experiments (AusDM 2005)

- Three smaller data sets
  - NSW Midwives data (500 records, randomly selected)
  - Nursing homes (600 records, randomly selected)
  - Unusual addresses (150 records, manually selected)
- HMMs generated for **F**, **LT**, and **LT&F**
- Compared to manually generated HMM using earlier *Febrl* approach *(BioMed Central 2002)*
- Measurements
  - Correctness: Exact and *close* standardisation accuracy
  - Number of *easy* addresses (with simple structure, like `[street num, name, type; loc, state, pc]`)

Results for Midwives data collection

	F	LT	LT&F	*Febrl*
Easy addresses	89.0%	87.6%	89.2%	82.0%
Accuracy	96.6%	95.4%	97.4%	96.8%
Close accuracy	97.0%	97.4%	98.0%	97.6%
Time per record	6 ms	11 ms	92 ms	7 ms

Results for Nursing homes data

	F	LT	LT&F	*Febrl*
Easy addresses	90.3%	89.7%	90.3%	88.2%
Accuracy	92.7%	98.5%	96.7%	96.0%
Close accuracy	96.5%	98.5%	97.8%	98.3%
Time per record	7 ms	18 ms	445 ms	9 ms

Results for unusual addresses

	F	LT	LT&F	*Febrl*
Easy addresses	20.6%	18.0%	20.6%	14.7%
Accuracy	79.3%	72.7%	92.7%	96.0%
Close accuracy	80.7%	80.0%	94.7%	96.0%
Time per record	7 ms	37 ms	720 ms	10 ms

- 150 manually selected unusual address records
  (like rural addresses, corner addresses, building and institution addresses, etc.)

Outline: Privacy preservation

- Short introduction to data linkage
- Improving indexing and classification
- Probabilistic name and address cleaning and standardisation
- Privacy preserving data linkage
- Our project: *Febrl*
  (Freely extensible biomedical record linkage)
- Outlook
- Additional material:
  - Measures for linkage quality and complexity
  - Geocoding

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, seek approval from ethics committees

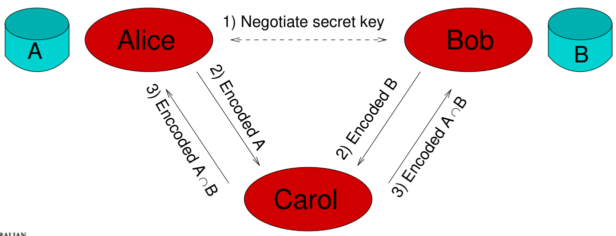
Invasion of privacy could be avoided (or mitigated) if some method were available to determine which records in two data sets match, without revealing any identifying information.

Privacy preserving approach

- *Alice* has a database **A** she wants to link with *Bob*
  (without revealing the actual values in **A**)
- *Bob* has a database **B** he wants to link with *Alice*
  (without revealing the actual values in **B**)
- Easy if only *exact matches* are considered
  - Encode data using one-way hashing (like *SHA*)
  - Example: *'tim'* → *'51ddc7d3a611eeba6ca770'*
- More complicated if values contain errors or typographical variations
  (even a single character difference between two strings will result in very different hash encodings)

Third party linkage protocol

- *Alice* and *Bob* negotiate a shared secret key
  (for example a 160 bit long SHA hash code)
- A third party (*Carol*) performs the actual linkage
- Only encrypted data is transmitted



Privacy preserving research

- Pioneered by French researchers in mid-to-late 1990s (for situations where de-identified data needs to be centralised and linked for follow-up studies)
- *Blindfolded record linkage*
  Approximate linkage of strings with typographical errors, based on *n*-gram techniques *(Churches & Christen, 2004)*
- *Privacy-preserving data linkage protocols*
  Several protocols with improved security and less information leakage *(O'Keefe et al., 2004)*
- *Blocking aware private record linkage*
  Approximate linkage based on tokens and TF-IDF, and three blocking approaches *(Al-Lawati et al., 2005)*

Outline: Our project

- Short introduction to data linkage
- Improving indexing and classification
- Probabilistic name and address cleaning and standardisation
- Privacy preserving data linkage
- Our project: *Febrl*
  (Freely extensible biomedical record linkage)
- Outlook
- Additional material:
  - Measures for linkage quality and complexity
  - Geocoding

Our project: Febrl

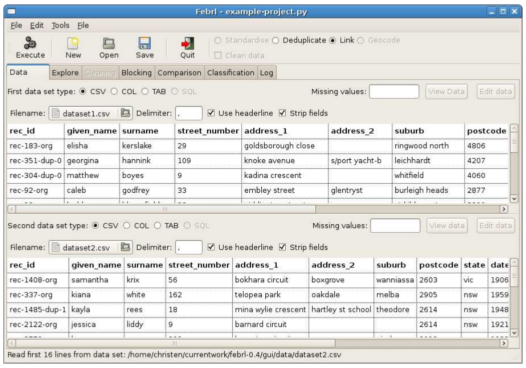
- Supported by and in collaboration with NSW Health (Australian Research Council (ARC) Linkage grant)
- Aim is to develop new and improved techniques for parallel large scale data linkage
- Main research areas
  - Probabilistic techniques for automated data cleaning and standardisation (mainly on addresses, using *G-NAF*)
  - New and improved blocking and indexing techniques
  - Improved record pair classification using (un-supervised) machine learning techniques (reduce clerical review)
  - Improved performance (scalability and parallelism)

Febrl prototype software

- An experimental platform for new and improved data linkage algorithms
- Modules for data cleaning and standardisation, data linkage, deduplication, geocoding, and generation of synthetic data sets, GUI (soon!)
- Open source https://sourceforge.net/projects/febrl/
- Implemented in *Python* http://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 community

Febrl graphical user interface

- Currently under development (published later this year)



Outlook

- Recent interest in data linkage / matching
  - Data mining and data warehousing, e-Commerce and Web applications
  - Health, census, crime/fraud detection, social security, immigration, intelligence/surveillance
- Main future challenges
  - Automated and accurate linkage
  - Higher performance (linking very large data sets)
  - Secure and privacy-preserving data linkage
- For more information see our project Web site (publications, talks, software, Web resources / links)
  http://datamining.anu.edu.au/linkage.html

Contributions / Acknowledgements

- Dr Tim Churches (New South Wales Health Department, Centre for Epidemiology and Research)
- Mr Alan Willmore (New South Wales Health Department)
- Dr Lee Taylor (New South Wales Health Department)
- Ms Kim Lim (New South Wales Health Department)
- Dr Markus Hegland (ANU Mathematical Sciences Institute)
- Mr Karl Goiser (ANU Computer Science PhD student)
- Mr Joseph Guillaume (ANU BPhilHon student, 2007)
- Ms Li Xiong (Cathy) and Mr Changyang Li (ANU eScience Masters students, 2007)
- Mr Yinghua Zheng (ANU Computer Science honours student, 2006–2007)
- Mr Daniel Belacic (ANU Computer Science honours student, 2005)
- Mr Puthick Hok (ANU Computer Science honours student, 2004)
- Mr David Horgan (ANU Computer Science summer student, 2003/2004)
- Mr Justin Zhu (ANU Computer Science honours student, 2002)

Outline: Measuring linkage quality

- Our project: *Febrl* (Freely extensible biomedical record linkage)
- Short introduction to data linkage
- Improving indexing and classification
- Probabilistic name and address cleaning and standardisation
- Privacy preserving data linkage
- Outlook
- Additional material:
  - Measures for linkage quality and complexity
  - Geocoding

Measuring data linkage quality

- Classifying record pairs results in four outcomes
  1. True matches classified as matches *(True Pos)*
  2. True matches classified as non-matches *(False Neg)*
  3. True non-matches classified as matches *(False Pos)*
  4. True non-matches classified as non-matches *(True Neg)*
- Various quality measures ($|\cdot| =$ *number of*)
  - Accuracy: $\frac{|TP|+|TN|}{|TP|+|FP|+|TN|+|FN|}$
  - Precision (or positive predictor value): $\frac{|TP|}{|TP|+|FP|}$
  - Recall (or sensitivity): $\frac{|TP|}{|TP|+|FN|}$
  - Specificity (or true negative rate): $\frac{|TN|}{|TN|+|FP|}$

Measuring quality issues

- Big question: *What to count?*
  - Actually compared record pairs (after blocking)?
  - All possible record pairs (full comparison space)?
  - Matched and non-matched *entities*?
- When counting record pairs, the number of TN will be increased quadratically (but not the numbers of TP, FN and FP)
  - Quality measures which include the number of TN can produce deceptive accuracy results
- Blocking also affects quality measures (aim of blocking is to remove as many TN and FP as possible, without removing any TP and FN)

Measuring data linkage complexity

- Recently proposed measures on blocking performance
  - Reduction ratio: $1 - \frac{N_b}{|\mathbf{A}| \times |\mathbf{B}|}$ (with $N_b \leq |\mathbf{A}| \times |\mathbf{B}|$ being the number of record pairs produced by a blocking algorithm)
  - Pairs completeness: $\frac{N_m}{|M|}$ (with N_m being the number of correctly classified true matched record pairs (TP) in the blocked comparison space, and $|M|$ total number of true matches)
- There is a trade-off between the reduction ratio and pairs completeness
- For more on this topic: *Christen & Goiser, 2007*

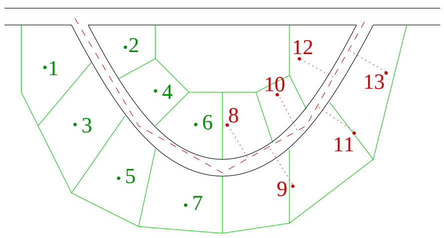
Outline: Geocoding

- Our project: *Febrl*
  (Freely extensible biomedical record linkage)
- Short introduction to data linkage
- Improving indexing and classification
- Probabilistic name and address cleaning and standardisation
- Privacy preserving data linkage
- Outlook
- Additional material:
  - Measures for linkage quality and complexity
  - Geocoding

What is geocoding?

- The process of assigning geographical coordinates (longitude and latitude) to addresses
- It is estimated that 80% to 90% of governmental and business data contain address information
  (US Federal Geographic Data Committee)
- Useful in many application areas
  - GIS, spatial data mining
  - Health, epidemiology
  - Business, census, taxation
- Various commercial systems available
  (e.g. MapInfo, www.geocode.com)

Geocoding techniques

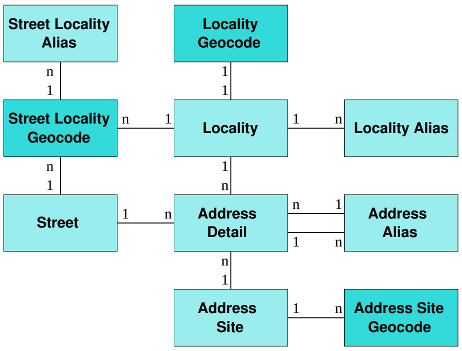


- Street centreline based (many commercial systems)
- Property parcel centre based (our approach)
- A recent study found substantial differences
  (specially in rural areas) *(Cayo & Talbot, 2003)*

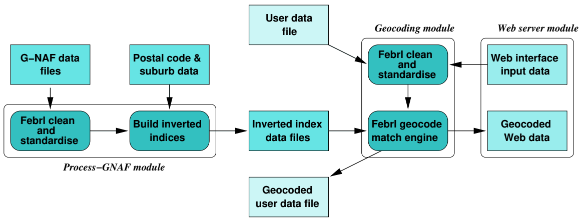
Geocoded National Address File

- Need for a national address file recognised in 1990
- 32 million source addresses from 13 organisations
- 5-phase cleaning and integration process
- Resulting database consists of 22 files or tables
- Hierarchical model (separate geocodes for each)
  - Address sites
  - Streets
  - Localities (towns and suburbs)
- Aliases and multiple locations possible

Simplified G-NAF data model



Febrl geocoding system



- Uses *Febrl* address cleaning and standardisation routines
- Aim: To transform user addresses into the same format as G-NAF addresses → Higher matching quality

Processing G-NAF

- Two step process
  1. Do cleaning and standardisation as discussed earlier
     (to make user data similar to G-NAF data)
  2. Build inverted indices (sets, implemented as keyed hash tables with field values as keys)
     Example (postcode): `'2000':(60310919,61560124)`
- Within geocode matching engine, intersections are used to find matching records
- Inverted indices are built for 23 G-NAF fields

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)

Geocode matching engine

- Rules based approach for exact or approximate matching
- Start with address and street level matching set intersection
- Intersect with locality matching set (start with neighbouring level 0, if no match increase to 1, finally 2)
- Refine with postcode, unit, property matches
- Return best possible match coordinates
  - Exact / average address
  - Exact / many street
  - Exact / many locality / no match

Some results

Match status	Number of records	Percentage
Exact address level match	7,288	72.87 %
Average address level match	213	2.13 %
Exact street level match	1,290	12.90 %
Many street level match	154	1.54 %
Exact locality level match	917	9.17 %
Many locality level match	135	1.35 %
No match	3	0.03 %

- 10,000 NSW *Land and Property Information* records
- Average 143 milliseconds for geocoding one record on a 480 MHz UltraSPARC II

Geocoding examples

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