Data Matching – An Overview, Recent Advances, and Research at the ANU

Peter Christen

School of Computer Science,
ANU College of Engineering and Computer Science,
The Australian National University,
Canberra, ACT 0200

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

- Overview of data matching
  - Applications and challenges
  - The matching process and matching techniques
- Recent advances in data matching research
- Data matching research at the ANU
  - The *Febrl* project
  - Matching historical census data
  - Matching bibliographic data
- Outlook
- Additional material (probabilistic data cleaning, privacy-preserving matching, geocoding, measuring quality and complexity)
Short introduction to data matching

The process of linking/matching records from one or more data sources that represent the same entity (such as a patient, customer, publication, etc.)

Also called record or data linkage, entity resolution, data scrubbing, object identification, merge-purge, etc.

Challenging if no unique entity identifiers available

For example, which of these three records refer to the same person?

<table>
<thead>
<tr>
<th>Dr Smith, Peter</th>
<th>42 Miller Street 2602 O'Connor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pete Smith</td>
<td>42 Miller St, 2600 Canberra A.C.T.</td>
</tr>
<tr>
<td>P. Smithers</td>
<td>24 Mill Street; Canberra ACT 2600</td>
</tr>
</tbody>
</table>
Recent interest in data matching

Traditionally, data matching has been used in health (epidemiology) and statistics (census)

In recent years, increased interest from businesses and governments

- Increased computing power and storage capacities
- A lot of data is being collected by many organisations
- Data warehousing and distributed databases
- Need for data sharing between organisations
- Data mining of large data collections
- E-Commerce and Web applications
- Geocode matching and spatial data analysis
Applications of data matching

- Remove duplicates in one data set (internal linkage)
- 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 data matching
  - Immigration, taxation, social security, census
  - Fraud, crime and terrorism intelligence
  - Business mailing lists, exchange of customer data
  - Social, health and biomedical research
Data matching challenges

- Often no unique entity identifiers are available
- Real world data is dirty
  (typographical errors and variations, missing and out-of-date values, different coding schemes, etc.)
- Scalability
  - Naïve comparison of all record pairs is $O(|A| \times |B|)$
  - Some form of blocking, indexing or filtering is required
- Privacy and confidentiality
  (because personal information, like names and addresses, are commonly required for matching)
- No training data in many matching applications
  (no record pairs with known true match status)
The data matching process

- Database A
  - Cleaning and standardisation
  - Blocking / Indexing
  - Weight vector classification
    - Matches
    - Non-matches
    - Possible matches
  - Clerical review
  - Field comparison
  - Evaluation
Data matching techniques

- Deterministic matching
  - Exact matching (if a unique identifier of high quality is available: precise, robust, stable over time)
  - Examples: Medicare, ABN or Tax file number (?)
  - Rules based matching (complex to build and maintain)

- Probabilistic linkage
  - Use common attributes for matching (often personal information, like names, addresses, dates of birth, etc.)
  - Can be wrong, missing, coded differently, or out-of-date

- Modern approaches
  (based on machine learning, data mining, AI, database, or 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*.

Many more with lower weights...
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 ($L_{di} / U_{di}$): $\log_2(0.03 / 0.917) = -4.92$
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 $1,000,000 \times 5,000,000 = 5 \times 10^{12}$ record pairs)

Performance bottleneck in a data matching 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 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 be uniformly distributed (as the most frequent values determine the size of the largest blocks).

Example: Frequency of ‘Smith’ in NSW: 25,425
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
Recent indexing approaches (2)

- **StringMap based blocking**
  - Map strings into a multi-dimensional space 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
    (suffix array: ‘peter’ → ‘eter’, ‘ter’, ‘er’)

- **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.
Indexing for real-time matching (1)

- Traditional approaches are aimed at matching two static databases (only one approach for query-time entity resolution: 31 sec for matching a query record with 831,000 records)

- Today, many applications require real-time matching
  - Identity verification during credit application, government services and benefits, e-Health, etc.
  - Crime detection and terrorism prevention systems
  - Health surveillance systems (disease outbreaks)

- A task similar to large-scale Web search (match a record to a large database, return most similar results)
**Real-time entity resolution (2)**

**Objectives:**
- Process a stream of incoming query records with one or several large databases
- Match these query records as quickly as possible
- Generate a match-score (allows setting a threshold)

**Challenges:**
- Large databases with many million records
- Dynamic database updates
- User constraints (like *black-lists*, or known name variations of people who have changed names)
- Multiple databases with different information content
Similarity-aware inverted indexing
Classification challenges

- In many cases there is no training data available
  - Possible to use results of earlier matching projects?
    - Or from manual *clerical review* process?
  - How confident can we be about correct manual classification of *possible matches*?

- Often there is no *gold standard* available
  - (no data sets with true known match status)

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

- Recent small repository: *RIDDLE*
    - (Repository of Information on Duplicate Detection, Record Linkage, and Identity Uncertainty)
Improved record pair classification

Fellegi and Sunter summing of weights results in loss of information.

View record pair classification as a multi-dimensional binary classification problem (use weight vectors to classify record pairs as matches or non-matches, but not possible matches).

Many machine learning techniques can be used:

- Supervised: Decision trees, SVMs, 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).
Collective matching example

(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, 31(2), 2006
Collective matching issues

- Several approaches have been developed (by machine learning, data mining and database communities).

- Combine graph and clustering based techniques (iteratively refine connection weights).

- Generally achieve much improved matching quality (compared to traditional matching based only on attribute similarities between two records).

- However, the computational complexity of these approaches is generally high:
  - For matching two databases with \( n \) records each, \( n \times n \) calculation steps (or more) are required.
  - Not scalable to large databases.
Outline

Overview of data matching
- Applications and challenges
- The matching process and matching techniques

Recent advances in data matching research

Data matching research at the ANU
- The Febrl project
- Matching historical census data
- Matching bibliographic data

Outlook

Additional material (probabilistic data cleaning, privacy-preserving matching, geocoding, measuring quality and complexity)
The Febrl project

Freely Extensible Biomedical Record Linkage

A collaboration with the NSW Department of Health (ARC Linkage Project 2004–2008)

Aim was to develop new and improved techniques for parallel large scale data matching

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
Overview of Febrl software

- Is implemented in *Python* (open source, object oriented, good for rapid prototype development)
- Source code is available (easy to extend and modify)
- Includes many recently developed data matching algorithms and techniques
- A tool to experiment with and learn about data matching (facilitated by a graphical user interface)
- *Is a prototype tool, not production software!*
- Freely available at: 
  
  [https://sourceforge.net/projects/febrl/](https://sourceforge.net/projects/febrl/)
Main Febrl features

Three main functionalities
- Cleaning and standardisation (of names, addresses, dates, and phone numbers)
- Deduplication of one data set
- Matching of two data sets

A variety of data matching techniques
- Seven blocking / indexing methods
- Twenty-six similarity functions (mainly for strings)
- Six record pair classifiers

Includes a data generator and various test data sets
Example Febrl GUI screen-shot

- Showing comparison function definitions
Matching historical census data

- Work done with the ANU Australian Demographic and Social Research Institute (CASS)

- Aim: Reconstruct families and households across historical census data sets that were collected at different points in time

- We have access to a data collection from the UK made of six data sets from 1851 to 1901 (around 30,000 records each)

- Basic idea is to apply novel backwards-forwards matching across time (starting with individual records, then families and households)

- We submitted an ARC Discovery Project grant earlier this year
ANU Research Office data matching

- For ERA (Excellence in Research for Australia), match Thompson ISI / Elsevier Scopus with ANU ARIES database

- ANU RO has conducted SQL based matching
  - Different match criteria (‘rule based’)
  - Author names so far not considered
  - Successfully matched around 74% of ARIES publications with ISI

- Apply more sophisticated data matching
  - Deal with cases that have typographical errors and variations in authors, journals and articles
  - Combine article and author matches
Example chemistry article titles

‘Undecacarbonyl(methylcyclopentadienyl)-tetrahedro-triiridiummolybdenum, undecacarbonyl(tetramethylcyclopentadienyl)-tetrahedro-triiridiummolybdenum and undecacarbonyl(pentamethylcyclopentadienyl)-tetrahedro-triiridiummolybdenum’

‘Fused supracyclopentadienyl ligand precursors. Synthesis, structure, and some reactions of 1,3-diphenylcyclopenta[l]phenanthrene-2-one, 1,2,3-triphenylcyclopenta[l]phenanthrene-2-ol, 1-chloro-1,2,3-triphenylcyclopenta[l]phenanthrene, 1-bromo-1,2,3-triphenylcyclopenta[l]phenanthrene, and 1,2,3-triphenyl-1H-cyclopenta[l] phenanthrene’
ANU RO data matching challenges

- Only author surnames and initials in both ARIES and ISI (many records with ‘M Smith’ or ‘J Williams’)
- Journal abbreviations and name changes
- Domain specific article titles (very similar when seen as text strings – such as examples on previous slide)
- What relative matching weights to give to journals, articles and authors?
- Different number of authors (have to normalise number of matched authors by number of listed authors)
- Initial matching using Febrl found all but 7 of the RO matches (and many thousand more new potential matches, including many false positives)
Outlook

Recent interest in data matching
- Data mining and data warehousing, e-Commerce and Web applications
- Health, census, crime/fraud detection, social security, immigration, intelligence/surveillance

Main future challenges
- Automatic and accurate matching (reduce manual effort)
- Higher performance (matching very large data sets)
- Secure and privacy-preserving data matching

For more information see our project Web site
(publications, talks, software, Web resources / links)
http://datamining.anu.edu.au/linkage.html
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’)

Peter Christen, August 2009 – p.34/66
**Address standardisation tasks**

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

- **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), alphanumeric (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):
  [‘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)
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

<table>
<thead>
<tr>
<th></th>
<th>F</th>
<th>LT</th>
<th>LT&amp;F</th>
<th>Febrl</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easy addresses</td>
<td>89.0%</td>
<td>87.6%</td>
<td>89.2%</td>
<td>82.0%</td>
</tr>
<tr>
<td>Accuracy</td>
<td>96.6%</td>
<td>95.4%</td>
<td>97.4%</td>
<td>96.8%</td>
</tr>
<tr>
<td>Close accuracy</td>
<td>97.0%</td>
<td>97.4%</td>
<td>98.0%</td>
<td>97.6%</td>
</tr>
<tr>
<td>Time per record</td>
<td>6 ms</td>
<td>11 ms</td>
<td>92 ms</td>
<td>7 ms</td>
</tr>
</tbody>
</table>
## Results for Nursing homes data

<table>
<thead>
<tr>
<th></th>
<th>F</th>
<th>LT</th>
<th>LT&amp;F</th>
<th>Febrl</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easy addresses</td>
<td>90.3%</td>
<td>89.7%</td>
<td>90.3%</td>
<td>88.2%</td>
</tr>
<tr>
<td>Accuracy</td>
<td>92.7%</td>
<td>98.5%</td>
<td>96.7%</td>
<td>96.0%</td>
</tr>
<tr>
<td>Close accuracy</td>
<td>96.5%</td>
<td>98.5%</td>
<td>97.8%</td>
<td>98.3%</td>
</tr>
<tr>
<td>Time per record</td>
<td>7 ms</td>
<td>18 ms</td>
<td>445 ms</td>
<td>9 ms</td>
</tr>
</tbody>
</table>
## Results for unusual addresses

<table>
<thead>
<tr>
<th></th>
<th>F</th>
<th>LT</th>
<th>LT&amp;F</th>
<th>Febrl</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easy addresses</td>
<td>20.6%</td>
<td>18.0%</td>
<td>20.6%</td>
<td>14.7%</td>
</tr>
<tr>
<td>Accuracy</td>
<td>79.3%</td>
<td>72.7%</td>
<td>92.7%</td>
<td>96.0%</td>
</tr>
<tr>
<td>Close accuracy</td>
<td>80.7%</td>
<td>80.0%</td>
<td>94.7%</td>
<td>96.0%</td>
</tr>
<tr>
<td>Time per record</td>
<td>7 ms</td>
<td>37 ms</td>
<td>720 ms</td>
<td>10 ms</td>
</tr>
</tbody>
</table>

150 manually selected unusual address records
(like rural addresses, corner addresses, building and institution addresses, etc.)
Privacy-preserving sharing and matching approaches

Based on cryptographic techniques
(secure multi-party computations – more on next slide)

Assume two data sources, and possibly a third (trusted) party to conduct the matching

Objective: No party learns about the other parties’ private data, only matched records are released

Various approaches with different assumptions about threats, what can be inferred by parties, and what is being released
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 [Yao 1982; Goldreich 1998/2002]

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

Step 0: $Z=999$

- **Party 1**: $x_1=55$

Step 1: $Z+x_1=1054$

- **Party 2**: $x_2=73$

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

- **Party 3**: $x_3=42$

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

Step 4: $s = 1169-Z = 170$
Privacy-preserving matching techniques

- Pioneered by French researchers for exact matching [Dusserre et al. 1995; Quantin et al. 1998]
  - Using one-way hash-encoding (‘tim’ → ‘51d3a6a70’)
- Secure and private sequence comparisons (edit distance) [Atallah et al. WPES’03]
- Blindfolded record linkage (details on following slides) [Churches and Christen, BioMed Central 2004]
- Secure protocol for computing string distance metrics (TF-IDF and Euclidean distance) [Ravikumar et al. PSDM’04]
- Privacy-preserving blocking [Al-Lawati et al. IQIS’05]
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
Uses *Febrl* address cleaning and standardisation routines

Aim: To transform user addresses into the same format as G-NAF addresses → Higher matching quality
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
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

<table>
<thead>
<tr>
<th>Match status</th>
<th>Number of records</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exact address level match</td>
<td>7,288</td>
<td>72.87 %</td>
</tr>
<tr>
<td>Average address level match</td>
<td>213</td>
<td>2.13 %</td>
</tr>
<tr>
<td>Exact street level match</td>
<td>1,290</td>
<td>12.90 %</td>
</tr>
<tr>
<td>Many street level match</td>
<td>154</td>
<td>1.54 %</td>
</tr>
<tr>
<td>Exact locality level match</td>
<td>917</td>
<td>9.17 %</td>
</tr>
<tr>
<td>Many locality level match</td>
<td>135</td>
<td>1.35 %</td>
</tr>
<tr>
<td>No match</td>
<td>3</td>
<td>0.03 %</td>
</tr>
</tbody>
</table>

10,000 NSW *Land and Property Information* records

Average 143 milliseconds for geocoding one record on a 480 MHz UltraSPARC II
Red dots: Febrl geocoding (G-NAF based)

Blue dots: Street centreline based geocoding
Measuring data matching 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 (*|·| = 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 matching complexity

Recently proposed measures on blocking performance

- Reduction ratio: \( 1 - \frac{N_b}{|A| \times |B|} \)
  (with \( N_b \leq |A| \times |B| \) 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