A Probabilistic Geocoding System based on a National Address File

Peter Christen[1]*, Tim Churches[2], and Alan Willmore[2]

[1] Department of Computer Science, Australian National University,
Canberra ACT 0200, Australia, peter.christen@anu.edu.au

[2] Centre for Epidemiology and Research, New South Wales Department of Health,
Locked Mail Bag 961, North Sydney NSW 2059, Australia,
{tchur,awill}@doh.health.nsw.gov.au

Abstract. It is estimated that between 80% and 90% of governmental and business data collections contain address information. Geocoding – the process of assigning geographic coordinates to addresses – is becoming increasingly important in many application areas that involve the analysis and mining of such data. In many cases, address records are captured and/or stored in a free-form or inconsistent manner. This fact complicates the task of robustly matching such addresses to spatially-annotated reference data. In this paper we describe a geocoding system that is based on a comprehensive high-quality geocoded national address database. It uses a learning address parser based on hidden Markov models to separate free-form addresses into components, and a rule-based matching engine to determine the best set of candidate matches to a reference file. The geocoding software modules are implemented (as part of the *Febrl* open source data linkage system) in the object-oriented language Python, which allows rapid prototype development and testing.

Keywords: Data mining preprocessing, geocoding, spatial data analysis, data linkage, data cleaning, indexing, G-NAF, hidden Markov model.

1 Geocoding

Increasingly, many data mining and data analysis projects need information from multiple data sources to be integrated, matched, combined or linked in order to enrich the available data and to allow more detailed analysis. The aim of such linkages is to merge all records relating to the same entity, such as a patient, customer or business. Most of the time the linkage (or matching) process is challenged by the lack of a common unique entity identifier, and thus becomes non-trivial [3, 8, 15]. In such cases, the available partially identifying information – like names, addresses, and dates of birth – is used to decide if two (or more) records correspond to the same entity. This process is compute intensive, and linking todays large data collections becomes increasingly difficult using traditional linkage techniques.

* Corresponding author

A special case of linkage is *geocoding*, the matching of a data source with geocoded reference data (which is made of cleaned and standardised records containing address information plus their geographical location). The US Federal Geographic Data Committee estimates that geographic location is a key feature in 80% to 90% of governmental data collections [14]. In many cases, addresses are the key to spatially enable data. The aim of geocoding is to generate a geographical location (longitude and latitude) from street address information in the data source. Once geocoded, the data can be used for further processing, in spatial data mining [6] projects, and it can be visualised and combined with other data using Geographical Information Systems (GIS).

The applications of spatial data analysis and mining are widespread. In the health sector, for example, geocoded data can be used to find local clusters of disease. Environmental health studies often rely on GIS and geocoding software to map areas of potential exposure and to locate where people live in relation to these areas. Geocoded data can also help in the planning of new health resources, e.g. additional health care providers can be allocated close to where there is an increased need for services. An overview of geographical health issues is given in [1]. When combined with a street navigation system, accurate geocoded data can assist emergency services find the location of a reported emergency (for example, if a caller reports an incomplete or unclear address).

Geocoded customer data, combined with additional demographic data, can help businesses to better plan marketing and future expansion, and the analysis of historical geocoded data, for example, can show changes in their customer base. Within census, geocoding can be used to assign people or households to small area units, for example census collection districts, which are then the basis of further statistical analysis.

There are two basic scenarios for geocoding user data. In the first, a user wants to automatically geocode a data set. The geocoding system should find the *best possible* match for each record in the user data set without human intervention. Each record needs to be attributed with the corresponding location plus a *match status* which indicates the accuracy of the match obtained (for example an exact address match, or a street level match, or a postcode level match). This scenario might become problematic if the user data is not of high quality, and contains records with missing, incorrect or out-of-date address information. Typographical errors are common with addresses, especially when they are recorded over the telephone or from hand-written forms. As reported in [11], a match rate of 70% successfully geocoded records is often considered an acceptable result. In the second scenario a user wants to geocode a single address that may be incomplete, erroneous or unformatted. The system should return the location if an exact match can be found, or alternatively a list of possible matches, together with a matching status and a likelihood rating. This geocoding of a single record should be done in (near) real time (i.e. less than a couple of seconds response time) and be available via a suitable user interface (e.g. a Web site).

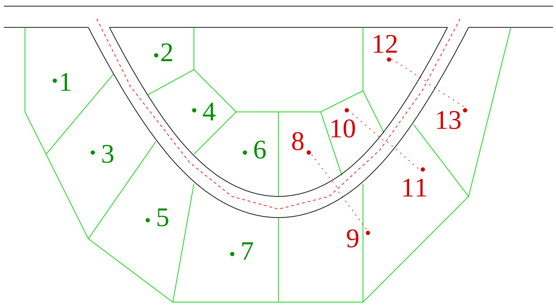

Fig. 1. Example geocoding using property parcel centres (numbers 1 to 7) and street reference file centreline (dashed line and numbers 8 to 13, with the dotted lines corresponding to a global street offset).

Standard data (or record) linkage techniques [3, 8, 15], where the aim is to link (or match) together all records belonging to the same entity, normally classify compared record pairs into one of the three classes *links*, *non-links* and *possible links*, with the latter class containing those record pairs for which human oversight, also known as *clerical review*, is needed to decide their final linkage status. Often no additional information is available so the clerical review process becomes one of applying human intuition, experience or common sense to the decision based on available data. This is similar to the second geocoding scenario described above, where the user is presented with a selection of possible matches (sorted according to their matching status and likelihood rating).

Many GIS software packages provide for street level geocoding. As a recent study shows [2], substantial differences in positional error exist between addresses which are geocoded using street reference files (containing geographic centreline coordinates, street numbers and names, and postcodes) and the corresponding true locations. The use of point property parcel coordinates (i.e. the centres or centroids of properties), derived from cadastral data, is expected to significantly reduce these positional errors. Figure 1 gives an illustrative example. Even small discrepancies in geocoding can result in addresses being assigned to, for example, different census collection districts, which can have huge implications when doing small area analysis. A comprehensive property based database is now available for Australia: the Geocoded National Address File (G-NAF). It is presented in details in Section 1.1.

We give an overview of our geocoding system in Section 2. The two central technical issues for a geocoding system are (1) the accurate and efficient matching of user input addresses with the address information stored in the geocoded reference data, and (2) the efficient retrieval of the address location (longitude and latitude) of the matched geocoded records. In order to achieve accurate match results, addresses both in the user data set and the geocoded reference data need to be cleaned and standardised in the same way. We cover this issue in more details in Section 2.1. Address locations can efficiently be retrieved from

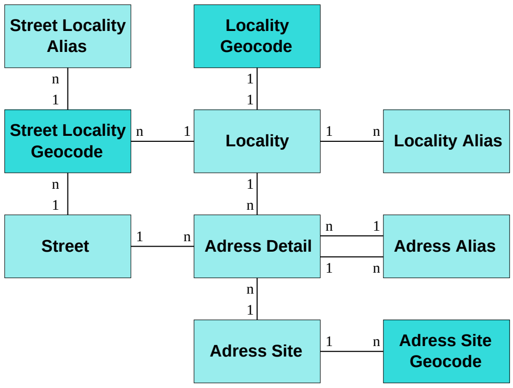

Fig. 2. Simplified G-NAF data model (10 main files only). Links *1–n* denote one-to-many, and links *1–1* denote one-to-one relationships.

the geocoded reference data by converting the traditional database tables (or files) into inverted indexes, as presented in Section 2.2. The geocode matching engine is the topic of Section 3, with some initial experimental results presented in Section 4, and conclusions and an outlook to future work is given in Section 5.

1.1 G-NAF – A Geocoded National Address File

In many countries geographical data is collected by various state and territory agencies. In Australia, for example, each state and territory have their own governmental agency that collect data to be used for land planning, as well as property, infrastructure or resource management. Additionally, national organisations like post and telecommunications, electoral rolls and statistics bureaus collect their own data. All these data sets are collected for specific purposes, have varying content and are stored in different formats.

The need for a nation-wide, standardised and high-quality geocoded data set has been recognised in Australia since 1990 [11], and after years of planning, collaborations and development the G-NAF was first released in March 2004. Approximately 32 million address records from 13 organisations were used in a five-phase cleaning and integration process, resulting in a database consisting of 22 normalised files (or tables). Figure 2 shows the simplified data model of the 10 main G-NAF files.

G-NAF is based on a hierarchical model, which stores information about address sites separately from locations and streets. It is possible to have multiple geocoded locations for a single address, and vice versa, and aliases are available at various levels. Three geocode files contain location (longitude and latitude) information for different levels. If an exact address match can be found, its location can be retrieved from the `ADDRESS_SITE_GEOCODE` file. If there is only a match on street level (but not street number), the `STREET_LOCALITY_GEOCODE` file will

Table 1. Characteristics of the 10 main G-NAF files (NSW data only).

G-NAF data file	Numbers of records and attributes	Keys (persistent identifiers)
ADDRESS_ALIAS	289,788 / 6	PRINCIPAL_PID ALIAS_PID
ADDRESS_DETAIL	4,145,365 / 28	GNAF_PID LOCALITY_PID STREET_PID ADDRESS_SITE_PID
ADDRESS_SITE	4,096,507 / 6	ADDRESS_SITE_PID
ADDRESS_SITE_GEOCODE	3,336,778 / 12	ADDRESS_SITE_PID
LOCALITY	5,017 / 7	LOCALITY_PID
LOCALITY_ALIAS	700 / 5	LOCALITY_PID ALIAS_PID
LOCALITY_GEOCODE	4,978 / 11	LOCALITY_PID
STREET	58,083 / 6	STREET_PID
STREET_LOCALITY_ALIAS	5,584 / 6	STREET_PID LOCALITY_PID
STREET_LOCALITY_GEOCODE	128,609 / 13	STREET_PID LOCALITY_PID

provide an overall street geocode. Finally, if no street level match can be found the `LOCALITY_GEOCODE` file contains geocode information for localities (e.g. towns and suburbs). Both the `STREET_LOCALITY_GEOCODE` and `LOCALITY_GEOCODE` files also contain information about the extent of streets and localities.

For our project we only used the G-NAF records covering the Australian state of New South Wales (NSW), containing around 4 million address, 60,000 street and 5,000 locality records. Table 1 gives an overview of the size and content of the 10 main G-NAF data files used.

2 System Overview

The geocoding system presented in this paper is part of the *Febrl* (Freely Extensible Biomedical Record Linkage) data linkage system [3, 7], that contains modules to clean and standardise data sets which can contain names, addresses and dates; and link and deduplicate such cleaned data. An overview of the *Febrl* geocoding system is shown in Figure 3. The geocoding process can be split into the preprocessing of the G-NAF data files (which is described in detail in Sections 2.1 and 2.2), and the matching with user-supplied addresses as presented in Section 3.

The preprocessing step takes the G-NAF data files and uses the *Febrl* address cleaning and standardisation routines to convert the detailed address values (like street names, types and suffixes, house numbers and suffixes, flat types

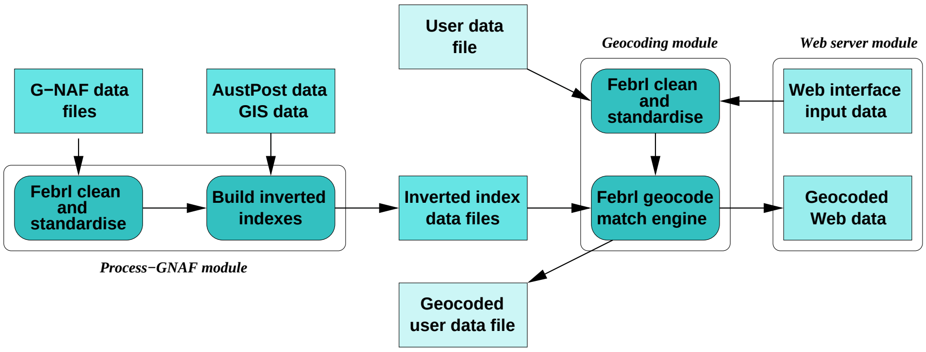

Fig. 3. Overview of the *Febrl* geocoding system.

and numbers, locality names, postcodes, etc.) into a form which makes them consistent with the user data after *Febrl* standardisation. Note that the G-NAF data files already come in a highly standardised form, but the finer details, for example how whitespaces within locality names are treated, make the difference between successful or failed matching. The cleaned and standardised reference records are then inserted into a number of inverted index data structures.

Additional data used in the preprocessing step are a postcode-suburb look-up table which is publicly available, and which can be used to impute missing postcodes or suburb values in the G-NAF locality files; and a table extracted from a commercial GIS system containing postcode and suburb boundary information, which is used to create *neighbouring region* look-up tables.

The geocode matching engine takes as input the inverted indexes and the raw user data, which is cleaned and standardised before geocoding is attempted. As shown in Figure 3, the user data can either be loaded from a data file, geocoded and then stored back into a data file, or it can be passed as one or more address(es) to the geocoding system and returned via a Web interface.

The complete *Febrl* system, including the geocoding and Web server modules, is implemented in the object-oriented open source language *Python*[1], which allows rapid prototype development and testing.

2.1 Probabilistic Address Cleaning and Standardisation

The first crucial step when processing both the geocoded reference files and the user data is the cleaning and standardisation of the data (i.e. addresses) used for geocoding. It is commonly accepted that real world data collections contain erroneous, incomplete and incorrectly formatted information. Data cleaning and standardisation are important preprocessing steps for successful data linkage and before including such data in a data warehouse for further analysis [13]. Data may be recorded or captured in various, possibly obsolete, formats and data items may

[1] `http://www.python.org`

be missing, out-of-date, or contain errors. The cleaning and standardisation of addresses is especially important for data linkage and geocoding so that accurate matching results can be achieved.

The main task of cleaning and standardising addresses is the conversion of the raw input data into well defined, consistent forms and the resolution of inconsistencies in the way address values are represented or encoded. Rule-based data cleaning and standardisation is currently used by many commercial systems and is cumbersome to set up and maintain, and often needs adjustments for new data sets. We have recently developed (and implemented within *Febrl*) new probabilistic techniques [4] based on hidden Markov models (HMMs) [12] which achieved better address standardisation accuracy and are easier to set-up and maintain compared to popular commercial linkage software.

A HMM is a probabilistic finite-state machine consisting of a set of observation or output symbols, a finite set of discrete, hidden (unobserved) states, a matrix of transition probabilities between those hidden states, and a matrix of probabilities with which each hidden state emits an observation symbol [12] (this *emission matrix* is also called an *observation matrix*). In our case, the hidden states of the HMM correspond to the output fields of the standardised addresses.

The *Febrl* approach to address cleaning and standardisation consist of the following three steps.

1. The user input addresses are *cleaned*. This involves converting all letters to lower-case, removing certain characters (like punctuations), and converting various sub-strings into their canonical form, for example *'c/-'*, *'c/o'* and *'c.of'* would all be replaced with *'care_of'*. These replacements are based on user-specified and domain specific substitution tables. Note that these substitution tables can also contain common misspellings for street and locality names, for example, and thus help to increase the matching quality.
2. The cleaned input strings are split into a list of words, numbers and characters, using whitespace marks as delimiters. Look-up tables and some hard-coded rules are then used to assign one or more tags to the elements in this list. These tags will be the observation symbols in the HMM used in the next step.
3. The list of tags is given to a HMM, and assuming that each tag (observation symbol) has been emitted by one of the hidden states, the *Viterbi* algorithm [12] will find the most likely path through the HMM, and the corresponding hidden states will give the assignment of the elements from the input list to the output fields.

Consider for example the address `'73 Miller St, NORTH SYDENY 2060'`, which will be cleaned (`SYDENY` corrected to `sydney`), split into a list of words and numbers, and tagged in steps one and two. The resulting lists of words/numbers and tags looks as follows.

```
['73', 'miller', 'street', 'north_sydney', '2060']
['NU', 'UN',     'WT',     'LN'          , 'PC'  ]
```

with `'NU'` being the tag for numbers, `'UN'` the tag for unknown words (not found in any look-up table or covered by any rule), `'WT'` the tag for a word found in the wayfare (street) type look-up table, `'LN'` the tag for a sequence of words found to be a locality name, and `'PC'` the tag for a known postcode.

In the third step the tag list is given to a HMM (which has previously been trained using similar address training data), and the *Viterbi* algorithm will return the most likely path through the HMM which will correspond to the following sequence of output fields.

```
'street number': '73'
  'street name': 'miller'
  'street type': 'street'
'locality name': 'north_sydney'
     'postcode': '2060'
```

Details about how to efficiently train the HMMs for address (as well as name) standardisation, and experiments with real-world data are given in [4]. Training of the HMMs is quick and does not require any specialised skills. For addresses, our HMM approach produced equal or better standardisation accuracies than a widely-used rule-based system.

2.2 Processing the G-NAF Files

Processing the G-NAF data files consists of two steps, the first being the cleaning and standardisation as described above, and the second being the building of inverted indexes. Such an inverted index is a keyed hash-table in which the keys are the values from the cleaned G-NAF data files, and the entries in the hash-table are sets with the corresponding PIDs (persistent identifiers) of the values. For example, assume there are four records in the `LOCALITY` file with the following content (the first line is a header-line with the attribute names).

```
locality_pid,  locality_name,  state_abbrev,  postcode
60310919,      sydney,         nsw,           2000
60709845,      north_sydney,   nsw,           2059
60309156,      north_sydney,   nsw,           2060
61560124,      the_rocks,      nsw,           2000
```

The inverted indexes for the three attributes `locality_name`, `state_abbrev` and `postcode` then are (square brackets denote lists and round brackets denote sets):

```
locality_name_index = ['north_sydney':(60709845,60309156),
                             'sydney':(60310919),
                          'the_rocks':(61560124)]

state_abbrev_index = ['nsw':(60310919,60709845,60309156,61560124)]

postcode_index = ['2000':(60310919,61560124),
                  '2059':(60709845),
                  '2060':(60309156)]
```

The matching engine then finds intersections of the inverted index sets for the values in a given record. For example, a postcode value `'2000'` would result in a set of PIDs `(60310919,61560124)`, and when intersected with the PIDs for

Table 2. G-NAF attributes used for geocode matching.

G-NAF data file	Attributes used
ADDRESS_DETAIL	flat_number_prefix, flat_number, flat_number_suffix, flat_type, level_number, level_type, building_name, location_description, number_first_prefix, number_first, number_first_suffix, number_last_prefix, number_last, number_last_suffix, lot_number_prefix, lot_number, lot_number_suffix
LOCALITY_ALIAS	locality_name, postcode, state_abbrev
LOCALITY	locality_name, postcode, state_abbrev
STREET	street_name, street_type, street_suffix
STREET_LOCALITY_ALIAS	street_name, street_type, street_suffix

locality name value `'the_rocks'`, would result in the single PID set `(61560124)` which corresponds to the original record. The location of this PID can then be look-up in the corresponding G-NAF geocode index. Table 2 shows the 23 attributes for which inverted indexes are built.

2.3 Additional Data Files

Additional information is used in the *Febrl* geocoding system during the preprocessing step to verify and correct (if possible) postcode and locality name values, and in the matching engine to enable searching for matches in neighbouring regions (postcodes and suburbs) if no exact match can be found.

Australia Post publishes a look-up table containing postcode and suburb information[2], which can be used when processing the G-NAF locality files to verify and even correct wrong or missing postcodes and suburb names. For example, if a postcode is missing in a record, the Australia Post look-up table can be used to find the official postcode(s) of the suburb in this record, and if this is a unique postcode it can be safely imputed into the record. Similarly, missing suburb names can be imputed if they correspond to a unique postcode.

Other look-up tables are used to find *neighbouring* regions for postcodes and suburbs, i.e. for a given region these tables contain all its neighbours. These look-up tables are created using geographical data extracted from a commercial GIS system, and integrated into the *Febrl* geocode matching engine.

Look-up tables of both direct and indirect neighbours (i.e. neighbours of direct neighbours) are used in the geocode matching engine to find matches in addresses where no exact postcode or suburb match can be found. Experience shows that people often record different postcode or suburb values if a neighbouring postcode or suburb has a higher perceived social status (e.g. *'Double Bay'* and *'Edgecliff'*), or if they live close to the border of such regions.

[2] `http://www.auspost.com.au/postcodes/`

3 Geocode Matching Engine

Febrl's geocode matching engine is based on the G-NAF inverted index data, and takes a rule-based approach to find an exact match or alternatively one or more approximate matches. Its input is a cleaned and standardised user record.

The matching engine tries to find an exact match first, but if none can be found it extends its search to neighbouring postcode and suburb regions. First direct neighbouring regions (level 1) are searched, then direct and indirect neighbouring regions (level 2), until either an exact match or a set of approximate matches can been found. In the latter case, either a weighted average location over all the found matches is returned, or a ranked (according to a likelihood rating) list of possible matches. The following steps explain in more detail (but still on a high conceptual level) how the matching engine works.

1. Find the set of address level matches (using street number and suffix) and the set of street level matches (using street name and type).
2. Find common matches between street and address levels (using set intersection).
3. Set the neighbour search level to 0 (no neighbouring regions are searched).
4. Find the locality match set (using locality name, qualifier and postcode) according to the current value of the neighbour search level. Postcode information is only used if no other locality information is available.
5. Find common matches between locality and address level, and between locality and street level (using set intersections).
6. If no matches between locality and address, and locality and street were found, increase the neighbour level (up to a maximum of 2) and jump back to step 4.
7. If matching records have been found, try to refine the match set using the postcode value (only if the postcode has not been used for the locality matches in step 4), as well as unit, flat and building (or property) information (if such information is available in the record).
8. If matches between street and address, or locality and address have been found, get their coordinates from the address geocode index. If only one match has been found, or if all found matches have the same location (this might be due to several G-NAF records corresponding to the same building) return the found location (longitude and latitude) together with an `'exact address match'` status. If more than one match with different locations have been found then calculate the average location and return it together with an `'average address match'` status.
9. If no address level match has been found use the street level match set. If only one match has been found or if all matches have the same location return the found location together with an `'exact street match'` status. If several street matches with different location were found return a `'many street match'` status and the list of found PIDs.
10. If no street level match has been found use the locality level match set. If only one match has been found or if all matches have the same location return the

Table 3. Matching results for geocoding 10,000 free-form LPI address records.

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 %

found location together with an `'exact locality match'` status. If several locality matches with different location were found return a `'many locality match'` status and the list of found PIDs.

11. If no match was found return a `'no match'` status.

Geocoding of multiple addresses is an iterative process where each record is first cleaned and standardised, then geocoded and written into an output data set with coordinates and a match status added to each record.

4 Experimental Results

We have run experiments with geocoding various data sets. In this section we present initial results of geocoding a NSW *Land and Property Information* data set containing 10,000 randomly selected free-form addresses (from a data set containing around 2.7 million records). Table 3 shows the matching results. A total of 94.94% exact matches could be found at different levels. A closer analysis of the results showed that for 456 records no exact address match was found due to missing coordinates in the `ADDRESS_SITE_GEOCODE` file (i.e. our G-NAF data set did not have coordinates for these addresses). With better quality of future G-NAF releases we can therefore expect improved matching qualities.

Using a *SUN Enterprise 450* shared memory (SMP) server with four 480 MHz *Ultra-SPARC II* processors and 4 Giga Bytes of main memory, it took 23 minutes and 50 seconds to geocode the 10,000 address records, which is an average of 143 milli-seconds per record.

5 Conclusions and Future Work

In this paper we have described a geocoding system based on a geocoded national address file. We are currently evaluating and improving this system using raw uncleaned addresses taken from various administrative health related data sets. We are also planning to compare the accuracy of our geocoding system with commercial street level based GIS systems, and similar to [2] we expect more accurate results. We are also fully integrating our geocoding system into the *Febrl* data linkage system [3, 7] and will publish it under an open source software license later this year.

Our main future efforts will be directed towards the refinement of the geocode matching engine to achieve more accurate matching results, as well as improving the performance of the matching engine (i.e. reducing the time needed to match a record). Three other areas of future work include:

- The *Febrl* standardisation routines currently return fields (or attributes) which are different from the ones available in G-NAF. This makes it necessary to map *Febrl* fields to G-NAF fields within the geocode matching engine. Better would be if the *Febrl* standardisation returns the same fields as the ones available in G-NAF, resulting in explicit field by field comparisons. We are planning to modify the necessary *Febrl* standardisation routines.
- Currently both the G-NAF preprocessing and indexing, as well as the geocode matching engine work in a sequential fashion only. Due to the large data files involved parallel processing becomes desirable. In the preprocessing step, the G-NAF data files can be processed independently or in a blocking fashion distributed over a number of processors, with only the final inverted indexes that need to be merged. Geocoding of a large user data file can easily be done in parallel as the cleaning, standardisation and matching of each record is independent from all others. An additional advantage of parallelisation is the increased amount of main memory available on many parallel platforms. We are planning to explore such parallelisation techniques and implement them into the *Febrl* system to allow faster geocoding of larger data sets. Additional performance improvements can be achieved by profiling and then replacing the core computational routines in the matching engine with C or C++ code.
- Geocoding uses identifying information (i.e. addresses) which raises privacy and confidentiality issues. Organisations that collect sensitive health data (e.g. cancer registries) cannot send their data to a geocoding service as this results in the loss of privacy for individuals involved. Methods are desirable which allow for privacy preserving geocoding of addresses. We aim to develop such methods based on techniques recently developed for blindfolded data linkage [5, 9, 10].

Acknowledgements

This work is funded by the NSW Department of Health, Centre for Epidemiology and Research. The authors would like to thank David Horgan (student at the University of Queensland) who worked on a first version of this system while he was a summer student at the ANU. The authors also wish to thank PSMA for providing the G-NAF data files.

References

1. Boulos, M.N.K.: Towards evidence-based, GIS-driven national spatial health information infrastructure and surveillance services in the United Kingdom. International Journal of Health Geographics 2004, 3:1. Available online at: `http://www.ij-healthgeographics.com/content/3/1/1`
2. Cayo, M.R. and Talbot, T.O.: Positional error in automated geocoding of residential addresses. International Journal of Health Geographics 2003, 2:10. Available online at: `http://www.ij-healthgeographics.com/content/2/1/10`
3. Christen, P., Churches, T. and Hegland, M.: A Parallel Open Source Data Linkage System. Proceedings of the 8th PAKDD'04 (Pacific-Asia Conference on Knowledge Discovery and Data Mining), Sydney. Springer LNAI-3056, pp. 638–647, May 2004.
4. Churches, T., Christen, P., Lim, K. and Zhu, J.X.: Preparation of name and address data for record linkage using hidden Markov models. BioMed Central Medical Informatics and Decision Making 2002, 2:9, Dec. 2002. Available online at: `http://www.biomedcentral.com/1472-6947/2/9/`
5. Churches, T. and Christen, P.: Some methods for blindfolded record linkage. BioMed Central Medical Informatics and Decision Making 2004, 4:9, June 2004. Available online at: `http://www.biomedcentral.com/1472-6947/4/9/`
6. Ester, M., Kriegel, H.-P. and Sander, J.: Spatial Data Mining: A Database Approach, Fifth Symposium on Large Spatial Databases (SSD'97). Springer LNCS 1262, pp. 48–66, 1997.
7. Freely extensible biomedical record linkage (Febrl) project web page, URL: `http://datamining.anu.edu.au/linkage.html`
8. Fellegi, I. and Sunter, A.: A Theory for Record Linkage. Journal of the American Statistical Society, 1969.
9. Hok, P.: Development of a Blind Geocoding System. Honours thesis, Department of Computer Science, Australian National University, Canberra, November 2004.
10. O'Keefe, C.M., Yung, M., Gu, L. and Baxter, R.: Privacy-Preserving Data Linkage Protocols. Proceedings of the Workshop on Privacy in the Electronic Society (WPES'04). Washington, DC, October 2004.
11. Paull, D.L.: A geocoded National Address File for Australia: The G-NAF What, Why, Who and When? PSMA Australia Limited, Griffith, ACT, Australia, 2003. Available online at: `http://www.g-naf.com.au/`
12. Rabiner, L.R.: A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. Proceedings of the IEEE, vol. 77, no. 2, Feb. 1989.
13. Rahm, E. and Do, H.H.: Data Cleaning: Problems and Current Approaches. IEEE Data Engineering Bulletin, 2000.
14. US Federal Geographic Data Committee. Homeland Security and Geographic Information Systems – How GIS and mapping technology can save lives and protect property in post-September 11th America. Public Health GIS News and Information, no. 52, pp. 21–23, May 2003.
15. Winkler, W.E.: The State of Record Linkage and Current Research Problems. RR99/03, US Bureau of the Census, 1999.