A Probabilistic Deduplication, Record Linkage and Geocoding System

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

Peter Christen[1] and Tim Churches[2]

[1]*Department of Computer Science, Australian National University, Canberra ACT 0200; peter.christen@anu.edu.au*
[2]*Centre for Epidemiology and Research, NSW Department of Health, North Sydney NSW 2059; tchur@doh.health.nsw.gov.au*

Abstract

In many data mining projects in the health sector information from multiple data sources needs to be cleaned, deduplicated and linked in order to allow more detailed analysis. The aim of such linkages is to merge all records relating to the same entity, such as a patient. Most of the time the linkage process is challenged by the lack of a common unique entity identifier. Additionally, personal information, like names and addresses, are frequently recorded with typographical errors, can be formatted differently, and parts can even be missing or swapped, making the duplication or linkage task non-trivial. A special case of linkage is geocoding, the process of matching user records with geocoded reference data, allowing spatial data analysis and mining, for example of disease outbreaks, or correlations with environmental factors.

In this paper we present an overview of the *Febrl* (Freely extensible biomedical record linkage) project, which aims at developing improved algorithms and techniques for large scale data cleaning and standardisation, record linkage, deduplication and geocoding. We discuss new probabilistic techniques for data cleaning and standardisation, approximate geocode matching, parallelisation of blocking and linkage algorithms, as well as a probabilistic data set generator.

Record Linkage and Geocoding in Health

The health sector produces and collects massive amounts of data on a daily basis, including administrative Medicare and PBS data, emergency and hospital admission data, clinical data, as well as data collected in special databases like cancer registries. The mining of such data has attracted interest both from academia and governmental organisation. Often data from various sources needs to be integrated and linked in order to allow more detailed analysis. In health surveillance systems linked data can also help to enrich data that is used for pattern detection in data mining systems. Linked data also allows re-using of existing data sources for new studies, and to reduce costs and efforts in data acquisition for research studies. Linked data might contain information which is needed to improve health policies, and which traditionally has been collected with time consuming and expensive survey methods.

Of increasing interest in the health sector is geocoding, the linking 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 [29]. 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 user data. Once geocoded, the data can be used for further processing, in spatial data mining projects, and it can be visualised and combined with other data using geographical information systems (GIS). The applications of spatial data analysis and mining in the health sector are widespread. 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 [4]. When combined with a street navigation system, accurate geocoded data can assist emergency services find the location of a reported emergency.

In this paper we present an overview of the *Febrl* (Freely extensible biomedical record linkage) project, and we discuss our future research plans. *Febrl* is implemented in the object-oriented open source language *Python*[1] (which is open source itself) and available from the project web page. Due to the availability of its source code, *Febrl* is an ideal platform for the rapid development, implementation, and testing of new and improved record linkage algorithms and techniques.

A Short Overview of Record Linkage

If unique entity identifiers or keys are available in all the data sets to be linked, then the problem of linking or deduplication at the entity level becomes trivial, a simple *join* operation in *SQL* or its equivalent is all that is required. However, in most cases no unique identifiers are shared by all of the data sets, and more sophisticated linkage techniques need to be applied. These techniques can be broadly classified into *deterministic* or rules-based approaches (in which sets of often very complex rules are used to classify pairs of records as *links*, i.e. relating to the same entity, or as *non-links*), and *probabilistic* approaches (in which statistical models are used to classify record pairs). Probabilistic methods can be further divided into those based on *classical* probabilistic record linkage theory as developed by *Fellegi & Sunter* [11], and newer approaches using machine learning techniques [6, 9, 10, 13, 15, 19, 21, 28, 30].

Computer-assisted record linkage goes back as far as the 1950s, when most linkage projects were based on *ad hoc* heuristic methods. The basic ideas of probabilistic record linkage were introduced by *Newcombe & Kennedy* [22] in 1962 while the theoretical foundation was provided by *Fellegi & Sunter* [11] in 1969. The basic idea is to link records by comparing common attributes, which include person identifiers (like names and dates of birth) and demographic information. Pairs of records are classified as *links* if their common attributes predominantly agree, or as *non-links* if they predominantly disagree. If two data sets $\mathbf{A}$ and $\mathbf{B}$ are to be linked, record pairs are classified in a product space $\mathbf{A} \times \mathbf{B}$ into M, the set of true matches, and U, the set of true non-matches. *Fellegi &*

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

Sunter [11] considered ratios of probabilities of the form

$$R = \frac{P(\gamma \ \varepsilon \ \Gamma|M)}{P(\gamma \ \varepsilon \ \Gamma|U)}$$

where γ is an arbitrary agreement pattern in a comparison space Γ. For example, Γ might consist of six patterns representing simple agreement or disagreement on (1) given name, (2) surname, (3) date of birth, (4) street address, (5) suburb and (6) postcode. Alternatively, some of the γ might additionally consider typographical errors, or account for the relative frequency with which specific values occur. For example, a surname value `'Miller'` is much more common in Australia than a value `'Dijkstra'`, resulting in a smaller agreement value. The ratio R or any monotonically increasing function of it (such as its logarithm) is referred to as a *matching weight*. A decision rule is then given by

if $R > t_{upper}$, then	designate a record pair as *link*
if $t_{lower} \leq R \leq t_{upper}$, then	designate a record pair as *possible link*
if $R < t_{lower}$, then	designate a record pair as *non-link*

The thresholds t_{lower} and t_{upper} are determined by a-priori error bounds on false links and false non-links. The class of *possible links* are those record pairs for which human oversight, also known as *clerical review*, is needed to decide their final linkage status (as often no additional information is available the clerical review process becomes one of applying human intuition, experience or common sense to the decision based on available data).

Probabilistic Data Cleaning and Standardisation

The cleaning and standardisation of raw input data is important for record linkage, as data can be encoded in different ways in the various data sources. Most real world data can contain noisy, incomplete, out-of-date and incorrectly formatted information. Data cleaning and standardisation are important preprocessing steps for successful record linkage, and before such data can be loaded into data warehouses or used for further analysis [27].

The main task of data cleaning and standardisation is the conversion of the raw input data into well defined, consistent forms, and the resolution of inconsistencies in the way names and addresses are represented or encoded. *Febrl* includes a probabilistic data standardisation technique [8] based on hidden Markov models (HMMs) [26]. 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. We use one HMM for names and one for addresses, and the hidden states of the HMMs correspond to the output fields of the standardised names and addresses.

Our approach to data cleaning and standardisation for names and addresses consist of the following three steps[2].

[2] *Febrl* also contains rules-based standardisation methods for dates and telephone numbers.

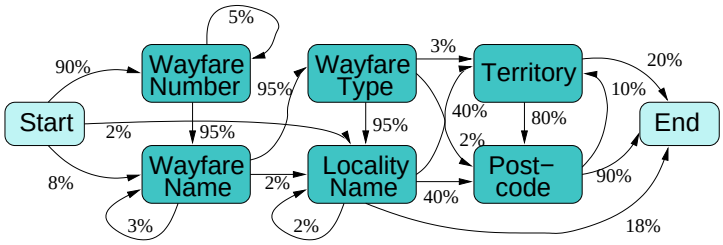

Figure 1: **Simple example address hidden Markov model.**

1. The user input records are *cleaned*. This involves converting all letters to lowercase, removing certain characters (like punctuations), and converting various substrings 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, which can also contain common misspellings of names and address words, and thus help to increase the linkage 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 HMMs used in the next step.
3. The list of tags is given to a HMM (either name or address), and assuming that each tag (observation symbol) has been emitted by one of the hidden states, the *Viterbi* algorithm [26] will find the most likely path through the HMM. The corresponding sequence of 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 1 and 2. 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 step 3 the tag list is given to a HMM (like the simple example shown in figure 1), which has previously been trained using similar address training data. The *Viterbi* algorithm will then return the most likely path through the HMM which will correspond to the following sequence of output fields.

```
'wayfare_number':  '73'
  'wayfare_name':  'miller'
  'wayfare_type':  'street'
 'locality_name':  'north_sydney'
      'postcode':  '2060'
```

Details about how to efficiently train the HMMs, and experiments with real-world data are given in [8]. 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 popular commercial rules-based system. However, accuracies were slightly worse when used with simpler name data [8].

We are planning to investigate the use of the *Baum-Welch* forward-backward algorithm [26] to re-estimate the probabilities in the HMMs, and to explore techniques that can be used for developing HMMs without explicitly specifying the hidden states. We are also planning to modify our address standardisation so the same output fields as used by G-NAF [7, 23] are created, which should result in improved geocode matching accuracy (see section on geocoding below).

Blocking

If two data sets **A** and **B** are to be linked, the number of possible comparisons equals the product of the number of records in the two data sets $|\mathbf{A}| \times |\mathbf{B}|$. As the performance bottleneck in a record linkage system is usually the expensive evaluation of the similarity measures between record pairs [1], it is computationally not feasible to consider all pairs when the data sets are large. Linking two data sets with $100,000$ records each would result in ten billion possible comparisons. On the other hand, the maximum number of linked record pairs that are possible corresponds to $min(|\mathbf{A}|, |\mathbf{B}|)$, assuming a record can only be linked to one other record. Thus, the space of potential links becomes sparser when linking larger data sets, while the computational efforts increase exponentially.

To reduce the large amount of possible record pair comparisons, traditional record linkage techniques [11, 30] work in a blocking fashion, i.e. they use one or a combination of record attributes to split the data sets into blocks. Only records having the same value in such a *blocking variable* are then compared (as they will be in the same block). This technique becomes problematic if a value in a blocking variable is recorded wrongly, as the corresponding record is inserted into a different block. To overcome this problem, several passes (iterations) with different blocking variables are normally performed. While the aim of blocking is to reduce the number of comparisons made as much as possible (by eliminating comparisons between records that obviously are not links), it is important that no potential link is overlooked because of the blocking process. There is a trade-off between the reduction in number of record pair comparisons and the number of missed true matches (accuracy) [1].

Febrl currently contains three different blocking methods, with more to be included in the future. The first method is the standard blocking [11, 30] applied in traditional record linkage systems. The second method is based on the *sorted neighbourhood* [15] approach, where records are sorted alphabetically according to the values of the blocking variable, then a sliding window is moved over the sorted records, and record pairs are formed using all records within the window. The third method uses *bigrams* (sub-strings of length 2) and allows for fuzzy blocking. The values in the blocking variable are converted into lists of bigrams, and permutations of bigram sub-lists are used as keys in an inverted index, which is then used to retrieve the records in a block [1].

Experiments [1] showed that innovative blocking methods can improve upon the traditional method used in record linkage, but further research needs to be conducted. The exploration of improved blocking methods is one of our major research areas. We aim to further explore alternatives, in terms of their applicability as well as their scalability both in data size and parallelism. Techniques include, for example, high-dimensional approximate distance metrics to form overlapping clusters [19], inverted indices, and improved fuzzy *n-gram* indices [1, 6].

Table 1: **Available field comparison functions.**

Exact string	(either field value strings are the same or not)
Truncated string	(only consider beginning of strings)
Approximate string	(using *Jaro*, *Winkler*, *Edit distance*, *Bigram* etc. algorithm [25])
Encoded string	(using *Soundex*, *NYSIIS*, *Phonex* etc. algorithm [17])
Keying difference	(allow a certain number of different characters)
Numeric percentage	(allowing percentage tolerance)
Numeric absolute	(allow absolute tolerance)
Date	(allow day tolerance)
Age	(allow percentage tolerance)
Time	(allow minute tolerance)
Distance	(allow kilometre tolerance, for example for postcode centroids)

Record Pair Classification and Assignment Restrictions
Each record pair produced in the blocking process is compared using a variety of field (or attribute) comparison functions (which are shown in table 1), resulting in a vector of *matching weights*. Frequency based weight calculation is currently supported for all string and the keying difference comparison functions. The weight vectors are then used to classify record pairs as either a *link*, *non-link*, or *possible link* (in which case the decision should be done by a human review). Classifiers currently implemented in *Febrl* are the classical *Fellegi & Sunter* [11] classifier (which sums all weights in a vector into one final matching weight), and a *flexible classifier* that allows the calculation of the final matching weight using various functions.

The original *Fellegi & Sunter* approach is closely related to a *Naive Bayes* classifier, and it assumes independence of the attributes. The conditional independent assumption efficiently deals with the *curse of dimensionality*, which becomes a major computational challenge when conditional dependencies are considered. In real world data, attributes are often dependent on each other (e.g. a change of address often results in changed street name, street number, postcode and suburb name). We aim to improve upon the classical probabilistic linkage method by combining them with deterministic and, in particular, machine learning and data mining techniques. Improvements in the linkage quality are paramount in order to reduce the time consuming and labour intensive clerical review process for possible links. Techniques like clustering [13] and active learning [28] have shown to be promising for this task.

In many linkage projects one is often only interested in the best linked record pairs, and one-to-one assignments need to be enforced. The simplest way to do this would be to use a greedy algorithm working on the sorted linked record pairs, but this would result in some assignments being not optimal due to the transitive closure problem. A linear sum assignment procedure based on the *Auction* algorithm [2] is thus used in *Febrl* to produce an optimal one-to-one assignment of linked record pairs.

Geocoding
Many commercial GIS software packages provide for street level geocoding. As a recent study shows [5], substantial differences in positional error exist between addresses which

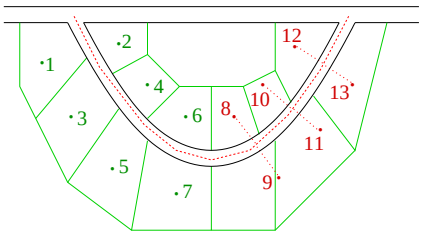

Figure 2: **Example geocoding using property parcel centres (numbers 1 to 7) and street reference file centreline (dashed line and numbers 8 to 13, with the dotted lines corresponding to a global street offset).**

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 2 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) [23]. 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. 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 (address, street and locality).

The geocoding process can be split into the preprocessing of the reference data files and the matching with user-supplied addresses. The preprocessing step takes the G-NAF data files and uses the *Febrl* address cleaning and standardisation routines to convert the detailed address values into a form which makes them consistent with the user data after *Febrl* standardisation. The cleaned and standardised reference records are inserted into a number of inverted index data structures. Positional *n-gram* indices can be built for attributes like street and locality names, allowing for approximate matching when the user data contains typographical errors (which are common in health data sets). Using auxiliary data with postcode and suburb boundary information, look-up tables with *neighbouring region* are built for suburbs and postcodes, allowing to search for addresses in adjacent regions if no exact match can be found. Experience shows that people often record a neighbouring postcode or suburb value if it has a higher perceived social status (e.g. `'Double Bay'` and `'Edgecliff'`), or if they live close to the border of such regions.

The *Febrl* geocode matching engine [7] is based on the G-NAF inverted index data and takes a rule-based approach. It tries to find an exact match first. If none can be found it uses approximate matching, and if still no match can be found it extends its search to neighbouring suburb and postcode regions. First direct neighbouring regions are searched, then direct and indirect neighbouring regions, until either an exact match or a set of approximate matches can been found. In the latter case, either a weighted

Table 2: **Geocoding results for 1,000 NSW LPI addresses.**

Match type	Number of matches	Percentage
Exact address	759	75.9%
Average address	19	1.9%
Many address	11	1.1%
Exact street	125	12.5%
Many street	9	0.9%
Exact locality	68	6.8%
Many locality	9	0.9%

average location (if the matches are within a small area) is returned, or a ranked (according to a matching weight) list of the found matches. User input records are cleaned and standardised before geocoding is attempted. Table 2 shows some matching results for addresses from a NSW Land and Property Information data set. Our future efforts will be directed towards the refinement of the geocode matching engine to achieve more accurate matching results.

Parallelisation

Although computing power has increased tremendously in the last few decades, large-scale data cleaning, standardisation and record linkage are still resource-intensive processes. In order to be able to process massive data sets, parallel processing becomes essential. Issues that have to be addressed are efficient data distribution, fault tolerance, dynamic load balancing, portability and scalability (both with the data size and the number of processors used).

Confidentiality and privacy have to be considered as record linkage often deals with partially identified data, and access restrictions are required. The use of high-performance computing centres (which traditionally are multi-user environments) or grid computing becomes problematic. An attractive alternative are networked personal computers or workstations which are available in large numbers in many businesses and organisations. Such office based clusters can be used as virtual parallel computing platforms to run large scale linkage tasks over night or on weekends.

Parallelism within *Febrl* is currently implemented based on the *Message Passing Interface* (MPI) [20] standard. Cleaning and standardisation, as well as geocoding is embarrassingly parallel (assuming the data is available distributed), as each record can be processed independently. The main parallel bottleneck is the blocking process which is not scalable in the number of processors, as access to the complete data sets is needed to build the blocking indices. The comparison of record pairs (which is the most compute intensive step) and classification steps are again scalable.

As an example, deduplication of $200,000$ records from a real world health data set on a *SUN Enterprise 450* shared memory (SMP) server with four 480 MHz *Ultra-SPARC II* processors and 4 Giga Bytes of main memory resulted in run times of 106 hours on one and 29 hours on four processors (speedup of 3.66). More than 95% of the time was spent in the comparison of record pairs. Communication times for this experiment were less than 0.35% of the total run times.

We will continue to improve upon the parallel processing functionalities of *Febrl* with an emphasis on running large linkage processes on clusters of personal computers (PCs) and workstations. Confidentiality and privacy aspects will need to be considered as well, as record linkage in many cases deals with identified data.

Probabilistic Data Generation
As record linkage and deduplication is often dealing with data sets that contain partially identified data (like names and addresses) it can be difficult to acquire data for testing and evaluation of new linkage algorithms and techniques. It is also hard to learn how to use and customise record linkage systems effectively without data sets where the linkage or deduplication status of record pairs is known.

In recent record linkage literature, a variety of data sets were used for experimental studies, some publicly available [3, 9, 19, 28], others proprietary [3, 10, 28]. This makes it difficult to validate the presented results, as well as to compare newly developed linkage algorithms.

What is needed is a collection of publicly available real test data sets for deduplication and record linkage, which can be used as a standard test bed for developing and comparing algorithms (similar to standard data sets used in information retrieval or machine learning). However, due to privacy and confidentiality issues it is unlikely that such data will ever become publicly available. De-identified data unfortunately cannot be used as the real values of names and addresses, for example, are at the core of many linkage algorithms.

An alternative is the use of artificially generated data sets. They have the advantages that the amount of errors introduced, as well as the linkage status of record pairs, are known. Controlled experiments can be performed and replicated easily. A first such data set generator (*DBGen*) was presented by [14] and has been used by others in a number of studies. This generator allows the creation of databases containing duplicates records. It uses lists of names, cities, states, and postcodes (all from the USA), and provides a large number of parameters, including size of the database to be generated, percentage and distribution of duplicates, and the amount and types of errors introduced.

We have improved upon *DBGen* by using frequency tables for name and address values taken from Australian telephone directories, and dictionary look-up tables with real world spelling variations of a large number of words, as well as user controlled maximum number of errors introduced per attribute and per record. User provided parameters also include the number of *original* and *duplicate* records to be created, the maximum number of duplicates for one original record, and the probabilities for introducing various errors to create the duplicate records (like inserting, deleting, transposing and substituting characters; swapping an attribute value with another value from the same look-up table; inserting or deleting spaces; setting an attribute value to missing; or swapping the values of two attributes). The position of where errors are introduced, as well as the types of errors introduced, are modelled according to studies on typographical and related errors [16, 24]. Each created record is given a unique identifier, which allows the evaluation of error rates (false linked non-duplicates and non-linked true duplicates).

Related Work

The processes of data cleaning, standardisation and record linkage have various names in different user communities. While statisticians and epidemiologists speak of *record* or *data linkage* [11], the same process is often referred to as *data* or *field matching*, *data scrubbing*, *data cleaning*, *preprocessing*, or as the *object identity problem* [12, 18, 27] by computer scientists and in the database community, whereas it is sometimes called *merge/purge processing* [14], *data integration* [9], *list washing* or *ETL* (extraction, transformation and loading) in commercial processing of customer databases or business mailing lists. Historically, the statistical and the computer science community have developed their own techniques, and until recently few cross-references could be found.

Improvements [30] upon the classical *Fellegi & Sunter* [11] approach include the application of the expectation-maximisation (EM) algorithm for improved parameter estimation [31], and the use of approximate string comparisons [25] to calculate partial agreements when attribute values have typographical errors. Fuzzy techniques and methods from information retrieval have recently been used to address the record linkage problem [6]. One approach is to represent records as document vectors and to compute the *cosine distance* [9] between such vectors. Another possibility is to use an *SQL* like language [12] that allows approximate joins and cluster building of similar records, as well as decision functions that decide if two records represent the same entity. Other methods [18] include statistical outlier identification, pattern matching, clustering and association rules based approaches.

In recent years, researchers have also started to explore the use of machine learning and data mining techniques to improve the linkage process. The authors of [10] describe a hybrid system that in a first step uses unsupervised clustering on a small sample data set to create data that can be used in the second step to classify record pairs into links or non-links. Learning field specific string-edit distance weights [21] and using a binary classifier based on support vector machines (SVM) is another approach. A system that is capable to link very large data sets with hundreds of millions of records – using special sorting and preprocessing techniques – is presented in [32].

Conclusions and Future Work

Written in an object-oriented open source scripting language, the *Febrl* record linkage system is an ideal experimental platform for researchers to develop, implement and evaluate new record linkage algorithms and techniques. While the current system can be used to perform smaller data cleaning, standardisation, linkage and geocoding tasks (up to several thousand records), further work needs to be done to allow the efficient processing of very large data sets.

Acknowledgements

This work is supported by an Australian Research Council (ARC) Linkage Grant LP0453463 and partially funded by the NSW Department of Health.

References

[1] Baxter, R., Christen, P. and Churches, T.: A Comparison of Fast Blocking Methods for Record Linkage. ACM SIGKDD '03 Workshop on Data Cleaning, Record

Linkage, and Object Consolidation, August 27, 2003, Washington, DC, pp. 25-27.

[2] Bertsekas, D.P.: Auction Algorithms for Network Flow Problems: A Tutorial Introduction. Computational Optimization and Applications, vol. 1, pp. 7-66, 1992.

[3] Bilenko, M. and Mooney, R.J.: Adaptive duplicate detection using learnable string similarity measures. Proceedings of the 9th ACM SIGKDD conference, Washington DC, August 2003.

[4] 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

[5] 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

[6] Chaudhuri, S., Ganjam, K., Ganti, V. and Motwani, R.: Robust and efficient fuzzy match for online data cleaning. Proceedings of the 2003 ACM SIGMOD International Conference on on Management of Data, San Diego, USA, 2003, pp. 313-324.

[7] Christen, P., Churches, T. and Willmore, A.: A Probabilistic Geocoding System based on a National Address File. Proceedings of the 3rd Australasian Data Mining Conference, Cairns, December 2004.

[8] 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, Dec. 2002. Available online at: http://www.biomedcentral.com/1472-6947/2/9/

[9] Cohen, W.: Integration of heterogeneous databases without common domains using queries based on textual similarity. Proceedings of SIGMOD, Seattle, 1998.

[10] Elfeky, M.G., Verykios, V.S. and Elmagarmid, A.K.: TAILOR: A Record Linkage Toolbox. Proceedings of the ICDE' 2002, San Jose, USA, 2002.

[11] Fellegi, I. and Sunter, A.: A theory for record linkage. In Journal of the American Statistical Society, 1969.

[12] Galhardas, H., Florescu, D., Shasha, D. and Simon, E.: An Extensible Framework for Data Cleaning. Proceedings of the Inter. Conference on Data Engineering, 2000.

[13] Gu, L. and Baxter, R.: Decision models for record linkage. Proceedings of the 3rd Australasian Data Mining Conference, Cairns, December 2004.

[14] Hernandez, M.A. and Stolfo, S.J.: The Merge/Purge Problem for Large Databases. Proceedings of the ACM-SIGMOD Conference, 1995.

[15] Hernandez, M.A. and Stolfo, S.J.: Real-world data is dirty: Data cleansing and the merge/purge problem. In Data Mining and Knowledge Discovery 2, Kluwer Academic Publishers, 1998.

[16] Kukich, K.: Techniques for automatically correcting words in text. ACM computing surveys, vol. 24, no. 4, pp. 377–439, December 1992.

[17] Lait, A.J. and Randell, B.: An Assessment of Name Matching Algorithms. Technical Report, Department of Computing Science, University of Newcastle upon Tyne, UK 1993.

[18] Maletic, J.I. and Marcus, A.: Data Cleansing: Beyond Integrity Analysis. Proceedings of the Conference on Information Quality (IQ2000), Boston, October 2000.

[19] McCallum, A., Nigam, K. and Ungar, L.H.: Efficient clustering of high-dimensional data sets with application to reference matching. Knowledge Discovery and Data Mining, pp. 169-178, 2000.

[20] Gropp, W., Lusk, E. and Skjellum, A.: Using MPI – 2nd Edition, Portable Parallel Programming with the Message Passing Interface, MIT Press, 1999.

[21] Nahm, U.Y, Bilenko M. and Mooney, R.J.: Two Approaches to Handling Noisy Variation in Text Mining. Proceedings of the ICML-2002 Workshop on Text Learning (TextML'2002), pp. 18-27, Sydney, Australia, July 2002.

[22] Newcombe, H.B. and Kennedy, J.M.: Record Linkage: Making Maximum Use of the Discriminating Power of Identifying Information. Communications of the ACM, vol. 5, no. 11, 1962.

[23] 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/`

[24] Pollock, J.J. and Zamora, A.: Automatic spelling correction in scientific and scholarly text. Communications of the ACM, vol. 27, no. 4, pp. 358–368, April 1984.

[25] Porter, E. and Winkler, W.E.: Approximate String Comparison and its Effect on an Advanced Record Linkage System. RR 1997-02, US Bureau of the Census, 1997.

[26] 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.

[27] Rahm, E. and Do, H.H.: Data Cleaning: Problems and Current Approaches. IEEE Data Engineering Bulletin, 2000.

[28] Sarawagi, S. and Bhamidipaty, A.: Interactive deduplication using active learning. Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 269–278, Edmonton, 2002.

[29] 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.

[30] Winkler, W.E.: The State of Record Linkage and Current Research Problems. RR 1999-04, US Bureau of the Census, 1999.

[31] Winkler, W.E.: Using the EM algorithm for weight computation in the Fellegi-Sunter model of record linkage. RR 2000-05, US Bureau of the Census, 2000.

[32] Yancey, W.E.: BigMatch: A Program for Extracting Probable Matches from a Large File for Record Linkage. RR 2002-01, US Bureau of the Census, March 2002.