A Parallel Open Source Data Linkage System

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

Peter Christen[1], Tim Churches[2] and Markus Hegland[3]

[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@doh.health.nsw.gov.au
[3] Centre for Mathematics and its Applications, Mathematical Sciences Institute, Australian National University, Canberra ACT 0200, Australia, markus.hegland@anu.edu.au

Abstract. In many data mining projects information from multiple data sources needs to be integrated, combined or 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 or a customer. Most of the time the linkage process is challenged by the lack of a common unique entity identifier, and thus becomes non-trivial. Linking todays large data collections becomes increasingly difficult using traditional linkage techniques. In this paper we present an innovating data linkage system called *Febrl*, which includes a new probabilistic approach for improved data cleaning and standardisation, innovative indexing methods, a parallelisation approach which is implemented transparently to the user, and a data set generator which allows the random creation of records containing names and addresses. Implemented as open source software, *Febrl* is an ideal experimental platform for new linkage algorithms and techniques.

Keywords: record linkage, data matching, data cleaning and standardisation, parallel processing, data mining preprocessing.

1 Introduction

Data linkage can be used to improve data quality and integrity, to allow re-use of existing data sources for new studies, and to reduce costs and efforts in data acquisition for research studies. In the health sector, for example, linked data might contain information which is needed to improve health policies, information that is traditionally collected with time consuming and expensive survey methods. Linked data can also help in health surveillance systems to enrich data that is used for pattern detection in data mining systems. Businesses routinely deduplicate and link their data sets to compile mailing lists, while in taxation offices and departments of social security data linkage can be used to catch people who register for benefits multiple times or who work and collect unemployment money. Another application of current interest is the use of data linkage in crime and terror detection. Security agencies and crime investigators increasingly rely on the ability to quickly bring up files for a particular individual which may help to prevent crimes or terror by early intervention. As such, data linkage is an important preprocessing step for further data analysis and mining.

If a unique entity identifier or key is available in all the data sets to be linked, then the problem of linking at the entity level becomes trivial, a simple *join* operation in *SQL* or its equivalent in other data management systems is all that is required. However, in most cases no unique key is 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 person or 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* [8], and newer approaches using maximum entropy, clustering and other machine learning techniques [3, 6, 7, 13, 15, 17, 22, 23].

Computer-assisted data linkage goes back as far as the 1950s. At that time, most linkage projects were based on *ad hoc* heuristic methods. The basic ideas of probabilistic data linkage were introduced by *Newcombe & Kennedy* [18] in 1962 while the theoretical foundation was provided by *Fellegi & Sunter* [8] in 1969. The basic idea is to link records by comparing common attributes, which include person identifiers (like names, dates of birth, etc.) 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 & Sunter* [8] 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 account for the relative frequency with which specific values occur. For example, a surname value *"Miller"* is normally much more common 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. If $\gamma \,\varepsilon\, \Gamma$ mainly consists of agreements then the ratio R would be large and thus the record pair would more likely to be designated as a link. On the other hand for a $\gamma \,\varepsilon\, \Gamma$ that primarily consists of disagreements the ratio R would be small. 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. In theory, the person undertaking this clerical review has access to additional data (or may be able to seek it out) which enables them to resolve the linkage status. In practice, often no additional data is available and the clerical review process becomes one of applying human intuition, experience or common sense to the decision based on available data.

In this paper we present some key aspects of our parallel open source data linkage system *Febrl* (for "Freely extensible biomedical record linkage"), which is implemented in the object-oriented open source scripting language *Python*[1] and freely available from the project web page (see URL on the title page). Due to the availability of its source code, *Febrl* is an ideal platform for the rapid development and implementation of new and improved data linkage algorithms and techniques.

After an overview of related work in Section 2, we discuss in Section 3 a newly developed probabilistic data cleaning and standardisation method based on hidden Markov models (HMM). The important task of blocking (or indexing), which aims in reducing the computational complexity of the linkage process, is the topic of Section 4, and in Section 5 we present the parallelisation techniques used within *Febrl*. A random data set generator is described in Section 6, and we conclude this paper by giving an outlook on future work in Section 7.

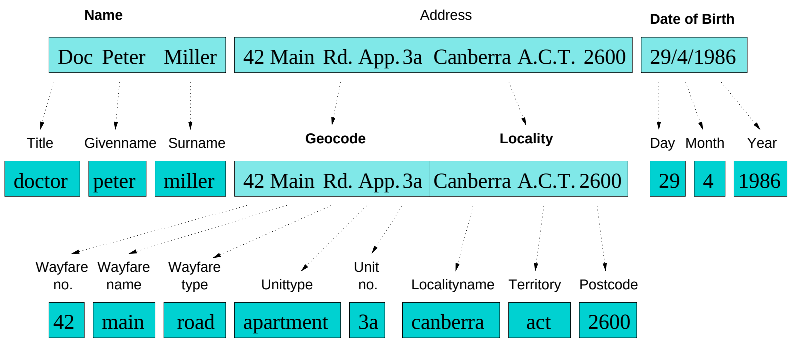
2 Related Work

The processes of data cleaning, standardisation and data linkage have various names in different user communities. While statisticians and epidemiologists speak of *record* or *data linkage* [8, 10], the same process is often referred to as *data* or *field matching*, *data scrubbing*, *data cleaning*, *preprocessing*, or as the *object identity problem* [9, 14, 21] by computer scientists and in the database community, whereas it is sometimes called *merge/purge processing* [12], *data integration* [6], *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 [23] upon the classical *Fellegi & Sunter* [8] approach include the application of the expectation-maximisation (EM) algorithm for improved parameter estimation [24], and the use of approximate string comparisons [19] to calculate partial agreements when attribute values have typographical errors. Fuzzy techniques and methods from information retrieval have recently been

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

Fig. 1. Example name and address standardisation.



used to address the data linkage problem [3]. One approach is to represent records as document vectors and compute the *cosine distance* [6] between such vectors. Another possibility is to use an *SQL* like language [9] 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 [14] include statistical outlier identification, pattern matching, clustering and association rules based approaches.

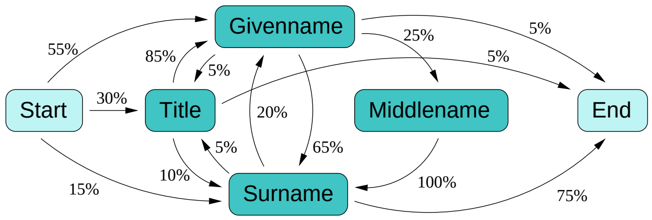
In recent years, researchers have also started to explore the use of machine learning and data mining techniques to improve the linkage process. The authors of [7] describe a hybrid system that in a first step uses unsupervised clustering on a small sample data set to create data that can be used in the second step to classify record pairs into links or non-links. Learning field specific string-edit distance weights [17] 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 billions of records – using special sorting and preprocessing techniques – is presented in [25].

3 Probabilistic Data Cleaning and Standardisation

As most real world data collections contain noisy, incomplete and incorrectly formatted information, data cleaning and standardisation are important pre-processing steps for successful data linkage, and before such data can be loaded into data warehouses or used for further analysis [21]. Data may be recorded or captured in various, possibly obsolete, formats and data items may be missing, out of date, or contain errors. The cleaning and standardisation of names and addresses is especially important for data linkage, to make sure that no misleading or redundant information is introduced (e.g. duplicate records).

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 information is represented or encoded. The example record shown in Figure 1 consisting of three input components is cleaned and

Fig. 2. Simple example hidden Markov model for names.



standardised into 14 output fields (the dark coloured boxes). Comparing these output fields individually with the corresponding output fields of other records results in a much better linkage quality than just comparing the whole name or the whole address as a string with the name or address of other records.

Rule-based data cleaning and standardisation as currently done by many commercial systems 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, 5] based on hidden Markov models (HMMs) [20] which showed to achieve better standardisation accuracy and are easier to set-up and maintain compared to popular commercial linkage software.

Our approach is based on the following three steps. First, the input strings are *cleaned*. This involves converting input into lower-case characters, and replacing certain words and abbreviations with others (these replacements are listed in look-up tables that can be edited by the user). In the second step, the input strings are split into a list of words, numbers and characters, which are then *tagged* using look-up tables (mainly for names and addresses) and some hard-coded rules (e.g. for numbers, hyphens or commas). Thirdly, these tagged lists are *segmented* into output fields using a HMM (like the one shown in Figure 2).

Let's assume for example an input record contains the name component `'Doc Peter Paul MILLER'`. In the cleaning step the input is converted into lower-case and the title `'doc'` is replaced with the standard word `'doctor'` (as found in a title replacement look-up table). Then, using tagging look-up tables, the input is split into a list and each element (word) is tagged:

```
['doctor', 'peter', 'paul', 'miller']
['TI',     'GM',    'GM',   'SN'    ]
```

with `'TI'` being the tag for title words, `'GM'` the tag for male given names and `'SN'`the tag for surnames. In the third step the tag list is given to a HMM (like the one shown in Figure 2) and using the *Viterbi* algorithm [20] the most likely path through the HMM gives us the corresponding output fields (the states of the HMM):

```
['doctor', 'peter',     'paul',       'miller' ]
['Title',  'Givenname', 'Middlename', 'Surname']
```

Details about how to efficiently train the HMMs for name and address standardisation, and experiments with real-world data are given in [4, 5]. 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. However, accuracies were worse when used with simpler name data [4, 5].

4 Blocking, Indexing and Classification

Data linkage considers the distribution of record pairs in the product space $\mathbf{A} \times \mathbf{B}$ and determines which of these pairs are links. The number of possible pairs equals the product of the sizes of the two data sets $\mathbf{A}$ and $\mathbf{B}$. The straight-forward approach would consider all pairs and model their distribution. As the performance bottleneck in a data linkage system is usually the expensive evaluation of a similarity measure between pairs of records [1], this approach is computationally not feasible for large data sets, it is *non-scalable*. Linking two data sets each with $100,000$ records would result in ten billion potential links (and thus comparisons). On the other hand, the maximum number of links that are possible corresponds to the number of records in the smaller data set (assuming a record can be linked to only one other record). Thus, the space of potential links becomes sparser with increasing number of records, while the computational efforts increase exponentially.

To reduce the huge amount of possible record comparisons, traditional data linkage techniques [8, 10, 23] work in a blocking fashion, i.e. they use one or more record attributes (e.g. the postcode and the year of birth) 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 iterations with different blocking variables are normally performed.

While such blocking (or indexing) techniques should 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 indexing process. Thus there is a trade-off between (a) the reduction in number of record pair comparisons and (b) the number of missed true matches (accuracy).

Febrl currently contains three different indexing methods, with more to be included in the future. In fact, the exploration of improved indexing methods is one of our major research areas [1]. The first indexing method is the standard blocking method [8, 10, 23] applied in traditional data linkage systems. The second indexing method is based on the *sorted neighbourhood* [13] 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 *q-grams* (sub-strings of length q) and allows for fuzzy blocking (in the current implementation we use bigrams, i.e. $q = 2$). The values of the blocking variable are converted into lists of bigrams, and permutations of sub-lists are built using a threshold (a value between 0.0 and 1.0) of all possible permutations. The resulting bigram sub-lists are converted back into strings and used as keys in an inverted index (into which record numbers are stored). Such an inverted index will then be used to retrieve the blocks. Let's assume for example a blocking variable value *'peter'*, and a bigram threshold set to 0.8. The bigram list will be *['pe','et','te','er']* with four elements, and using the 0.8 threshold results in $4 \times 0.8 = 3.2$, rounded to 3, which means all sub-list permutations of length 3 are calculated, which are: *['pe','et','te']*, *['pe','et','er']*, *['pe','te','er']*, and *['et','te','er']*. The record numbers of all records with a blocking variable value *'peter'* will therefore be inserted into the four inverted index blocks with keys *'peette'*, *'peeter'*, *'peteer*, and *'etteer'*.

Initial experiments [1] showed that innovative indexing methods can improved upon the traditional blocking used in data linkage, but further research needs to be conducted. Other research groups have also investigated the use of *q-grams* [3] as well as high-dimensional approximate distance metrics to form overlapping clusters [15].

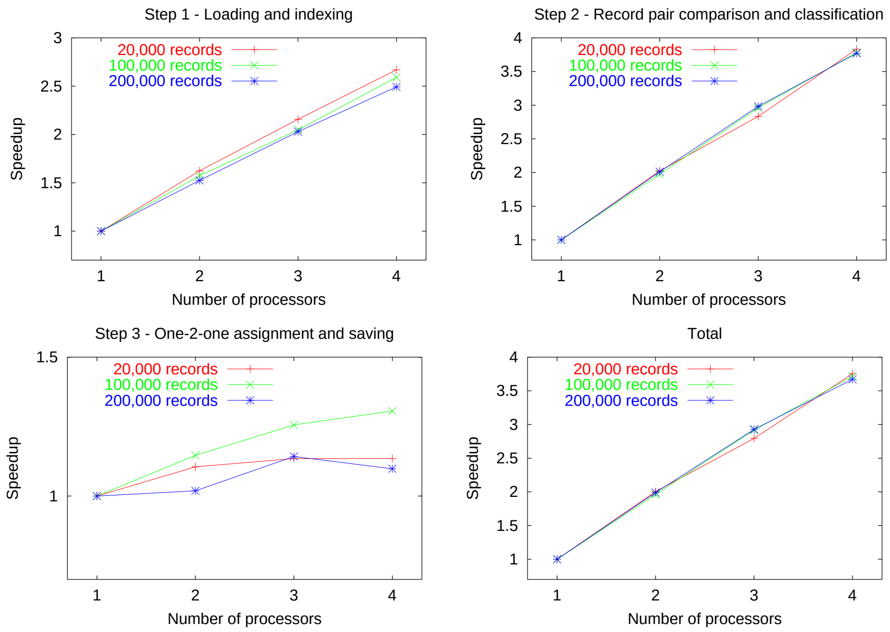
For each record pair in the index a vector containing *matching weights* is calculated using field comparison functions. This vector is then used to classify the pair as either a *link*, *non-link*, or *possible link* (in which case the decision should be done by a human review). While the classical *Fellegi & Sunter* [8] simply sums all the weights in the vector into one matching weight, alternative classifiers are possible which improve upon this. For example, separate weights can be calculated for names and addresses [10], or machine learning techniques can be used for the classification task. In the *Febrl* system currently two classifiers are implemented. The first is the classical *Fellegi & Sunter* [8] classifier described earlier, and the second is a *flexible classifier* that allows a flexible calculation of the matching weights using various functions.

5 Parallelisation

Although computing power has increased tremendously in the last few decades, large-scale data cleaning, standardisation and data linkage are still resource-intensive processes. There have been relatively few advances over the last decade in the way in which probabilistic data linkage is undertaken, particularly with respect to the tedious clerical review process which is still needed to make decisions about pairs of records whose linkage status is doubtful. In order to be able to link large 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 data linkage deals with partially identified data, and access restrictions are required. The use of high-performance computing centers (which traditionally are multi-user environments) becomes problematic. An attractive alternative are networked personal comput-

Fig. 3. Speedups for parallel internal linkage processes.



ers 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 in its initial stages. Based on the well known *Message Passing Interface* (MPI) [16] standard, and the Python module *Pypar*[2] which provides bindings to an important subset of the MPI routines, parallelism is implemented transparently to the user of *Febrl*.

To give an idea on the parallel performance of *Febrl* some initial timing results of experiments made on a parallel computing platform (a *SUN Enterprise 450* shared memory (SMP) server with four 480 MHz *Ultra-SPARC II* processors and 4 Giga Bytes of main memory) are presented in this section. Three internal linkage (deduplication) processes were performed with 20,000, 100,000 and 200,000 records, respectively, from a health data set containing midwife data records provided by the *NSW Department of Health*. This data set had earlier been standardised into a clean form and was stored as a CSV (comma separated values) text file. Six field comparison functions were used and the classical blocking index technique with three indexes (passes) was applied. The standard *Fellegi & Sunter* [8] classifier was used to classify the record pairs, and finally a one-to-one assignment procedure [2] was applied, in order to restrict a record in the data set to be linked to maximal one other record.

[2] `http://datamining.anu.edu.au/pypar/`

Table 1. Memory usage for internal linkage processes (in Mega Bytes).

Number of processors	1	2	3	4
20, 000 records	60	134	188	242
100, 000 records	238	603	820	1,005
200, 000 records	906	2,130	2,829	3,495

These internal linkage processes were run using 1, 2, 3 and 4 processors, respectively. In Figure 3 *speedup* (which is defined as the time on one processor divided by the time on 2, 3, or 4 processors, respectively) results are shown scaled by the number of records (i.e. the total elapsed run times divided by the number of records were used to calculate the speedup). The results show that the record comparison and classification step (which takes between 94% and 98% of the total run times) is scalable, while building the blocking indexes and the one-to-one assignment steps result in low speedup values and are not scalable. As most time is spent in the record pair comparison and classification (step 2), the overall parallel performance is quite scalable. Communication times can be neglected as they were less than 0.35% of the total run times in all experiments.

Table 1 shows the maximal amount of memory used by these internal linkage experiments. The amount of memory used increases more than linearly compared with the number of records (a ten-fold increase in the number of records results in an around fifteen-fold increase in the amount of memory needed). Additionally, parallel processing in *Febrl* also results in an increased amount of memory needed, which is due to the replication of various data structures on the parallel *Febrl* processes.

6 Data Set Generation

As data linkage is dealing with data sets that contain partially identified data, like names and addresses, it can be very difficult to acquire data sets for testing and evaluating newly developed linkage algorithms and techniques. For the user it can be difficult to learn how to apply, evaluate and customise data linkage algorithms effectively without example data sets where the linkage status of record pairs is known.

To overcome this we have developed a database generator based on ideas by *Hernandez & Stolfo* [12]. This generator can create data sets that contain names (based on frequency look-up tables for surnames and given names), addresses (based on frequency look-up tables for suburbs, postcodes, street numbers and names, and state/territory names), dates (like dates of birth), and identifier numbers (to randomly create for example social security numbers).

To generate a data set, a user must provide the number of *original* and *duplicate* records to be created, the maximal number of duplicates that can be created for one original record, a probability distribution of how duplicates are created (possible are *uniform*, *Poisson* and *zipf*), and the probabilities that one

Fig. 4. Randomly generated example data set.

```
     rec_id, streetnum,           address1,           address2, suburb, postcode
  rec-0-org,        11,        wylly place, inverpine ret vill,  taree,     4860
rec-0-dup-0,        11,         wyllyplace, inverpine ret vill,  taree,     4860
rec-0-dup-1,        11, inverpine ret vill,        wylly place,  taree,     4860
rec-0-dup-2,         3,        wylly place, inverpine ret vill,  tared,     4860
rec-0-dup-3,        11,       wylly parade, inverpine ret vill,  taree,     4680
  rec-1-org,        15,      stuart street,           hartford, menton,     3135
  rec-2-org,        20,   griffiths street,             myross,  kilda,     2756
rec-2-dup-0,        20,   griffith sstreet,             myross,  kilda,     2576
rec-2-dup-1,        20,    griffith street,            mycross,  kilda,
  rec-3-org,         5, ellenborough place,    elura homestead,  sydney,    2212
```

or more of the following random modifications are introduced into duplicate records:

- Insert a new character at a random position into a field (attribute).
- Delete a character at a random position from a field.
- Substitute a character in a field with another character (with the possibility to set probabilities for randomly choosing a neighbouring key in the same keyboard row or column).
- Transpose two characters at a random position in a field.
- Swap (replace) the value in a field with another value (randomly selected from a look-up table of possible values).
- Insert a space into a field and splitting a word.
- Delete a space (if available) in a field and merging two words.
- Set a field value to missing (with a user definable missing value).
- Swap the values of two fields (e.g. surname with given name) in a record.

In a first step the original records are generated, and in a second step duplicates of these original records are created using randomly introduced modifications. Each of these records is given a unique identifier, which allows the evaluation of accuracy and error rates (false linked record pairs and un-linked true matches) for data linkage procedures.

Figure 4 shows a small example data set containing 10 records (4 originals and 6 duplicates), randomly created using Australian address frequency look-up tables.

7 Conclusions and Future Work

Written in an object-oriented open source scripting language, the *Febrl* data linkage system is an ideal experimental platform for researchers to develop, implement and evaluate new data linkage algorithms and techniques. While the current system can be used to perform simple data cleaning, standardisation and linkage tasks, further work needs to be done to allow the efficient linkage of large data sets. We plan to improve *Febrl* in several areas.

For data cleaning and standardisation, we will be improving upon our recently developed probabilistic techniques [4, 5] based on hidden Markov models (HMMs), by using the *Baum-Welch* forward-backward algorithm [20] to re-estimate the probabilities in the HMMs, and we will explore techniques that can be used for developing HMMs without explicitly specifying the hidden states.

We aim to further explore alternative techniques for indexing based on high-dimensional clustering [15], inverted indexes, or fuzzy *q-gram* indexes [1, 3], in terms of their applicability for indexing as well as their scalability both in data size and parallelism.

Current methods for record pair classification based on the traditional *Fellegi & Sunter* [8] approach apply the semi-parametric mixture models and the EM algorithm [24] for the estimation of the underlying densities and the clustering and classification of links and non-links [23]. We will investigate the performance of data mining or non-parametric techniques including tree-based classifiers, sparse grids [11] and Bayesian Nets. An advantage of these methods is that they allow adaptive modelling of correlated data and can deal with complex data and the curse of dimensionality. For all these methods current fitting algorithms will be adapted and new ones developed.

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 and workstations as available in many businesses and organisations. Such office based PC clusters (with some additional software installed) can be used as virtual parallel computing platforms to run large scale linkage tasks over night or on weekends. Confidentiality and privacy aspects will need to be considered as well, as data linkage in many cases deals with identified data.

Acknowledgments

This project is funded by the *Australian National University (ANU)* and the *New South Wales Department of Health* under an *AICS (ANU-Industry Collaboration Scheme)* grant *AICS #1-2001*. Additional funding is provided by the *Australian Partnership for Advanced Computing (APAC)*.

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. Chaudhuri, S., Ganjam, K., Ganti, V. and Motwani, R.: Robust and efficient fuzzy match for online data cleaning. Proceedings of the 2003 ACM SIGMOD intern. conference on on Management of Data, ASan Diego, USA, 2003, pp. 313-324.
4. Christen, P., Churches, T. and Zhu, J.X.: Probabilistic Name and Address Cleaning and Standardisation. Proceedings of the Australasian Data Mining Workshop, Canberra, Dec. 2002.

5. 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/
6. Cohen, W.: Integration of heterogeneous databases without common domains using queries based on textual similarity. Proceedings of SIGMOD, Seattle, 1998.
7. Elfeky, M.G., Verykios, V.S. and Elmagarmid, A.K.: TAILOR: A Record Linkage Toolbox. Proceedings of the ICDE' 2002, San Jose, USA, 2002.
8. Fellegi, I. and Sunter, A.: A theory for record linkage. In Journal of the American Statistical Society, 1969.
9. Galhardas, H., Florescu, D., Shasha, D. and Simon, E.: An Extensible Framework for Data Cleaning. Proceedings of the Inter. Conference on Data Engineering, 2000.
10. Gill, L.: Methods for Automatic Record Matching and Linking and their use in National Statistics. National Statistics Methodology Series No. 25, London, 2001.
11. Hegland, M.: Adaptive sparse grids. ANZIAM J., vol. 44, 2003, pp. C335-C353.
12. Hernandez, M.A. and Stolfo, S.J.: The Merge/Purge Problem for Large Databases. Proceedings of the ACM-SIGMOD Conference, 1995.
13. 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.
14. Maletic, J.I. and Marcus, A.: Data Cleansing: Beyond Integrity Analysis. Proceedings of the Conference on Information Quality (IQ2000), Boston, October 2000.
15. 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.
16. Gropp, W., Lusk, E. and Skjellum, A.: Using MPI – 2nd Edition, Portable Parallel Programming with the Message Passing Interface, MIT Press, 1999.
17. 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.
18. 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.
19. Porter, E.H. and Winkler, W.E.: Approximate String Comparison and its Effect on an Advanced Record Linkage System. Research Report RR 1997-02, US Bureau of the Census, 1997.
20. 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.
21. Rahm, E. and Do, H.H.: Data Cleaning: Problems and Current Approaches. IEEE Data Engineering Bulletin, 2000.
22. Verykios, V.S., Elmagarmid, A.K., Elfeky, M.G., Cochinwala, M. and Dalal, S.: On the Completeness and Accuracy of the Record Matching Process. Proceedings of the MIT Conference on Information Quality, Boston, MA, October 2000.
23. Winkler, W.E.: The State of Record Linkage and Current Research Problems. Research Report RR 1999-04, US Bureau of the Census, 1999.
24. Winkler, W.E.: Using the EM algorithm for weight computation in the Fellegi-Sunter model of record linkage. Research Report RR 2000-05, US Bureau of the Census, 2000.
25. Yancey, W.E.: BigMatch: A Program for Extracting Probable Matches from a Large File for Record Linkage. Research Report RR 2002-01, Statistical Research Division, US Bureau of the Census, March 2002.