Febrl – A Freely Available Record Linkage System with a Graphical User Interface

Peter Christen

Department of Computer Science, The Australian National University
Canberra ACT 0200, Australia
Email: peter.christen@anu.edu.au

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

Abstract

Record or data linkage is an important enabling technology in the health sector, as linked data is a cost-effective resource that can help to improve research into health policies, detect adverse drug reactions, reduce costs, and uncover fraud within the health system. Significant advances, mostly originating from data mining and machine learning, have been made in recent years in many areas of record linkage techniques. Most of these new methods are not yet implemented in current record linkage systems, or are hidden within 'black box' commercial software. This makes it difficult for users to learn about new record linkage techniques, as well as to compare existing linkage techniques with new ones. What is required are flexible tools that enable users to experiment with new record linkage techniques at low costs.

This paper describes the *Febrl* (Freely Extensible Biomedical Record Linkage) system, which is available under an open source software licence. It contains many recently developed advanced techniques for data cleaning and standardisation, indexing (blocking), field comparison, and record pair classification, and encapsulates them into a graphical user interface. *Febrl* can be seen as a training tool suitable for users to learn and experiment with both traditional and new record linkage techniques, as well as for practitioners to conduct linkages with data sets containing up to several hundred thousand records.

Keywords: Health data linkage, data matching, data integration, deduplication, data cleaning, open source software, record linkage software, GUI.

1 Introduction

Many private and public organisations in the health sector are collecting, storing, processing and analysing fast-growing amounts of data with millions of records. Most of this data is about people (such as GP or hospital patients, or members of a private health insurance company) and contains names, addresses, and other health related personal details. Linking and aggregating records that relate to the same person from several databases is becoming increasingly important, as linked data can contain information that is not available otherwise, and thus allows studies that would otherwise not have been possible, or only using expensive and time consuming survey methods.

In the health sector, record linkage is of prime interest as linked data can help to improve health policies, uncover fraud, reduce costs, detect adverse drug reactions, and be used instead of expensive survey data in epidemiological studies (Brook et al. 2005, Clarke 2004). For example, research in Western Australia based on an ambulance cardiac arrest database linked with hospital data and death registers led to the installation of defibrillators in ambulances and hospital wards, thereby saving many lives.

In recent years, the importance of health record linkage has been recognised in Australia with the inclusion of *Population Health and Clinical Data Linkage* as one of twelve capability areas within the federal government's *National Collaborative Research Infrastructure Strategy* (NCRIS),[1] and the establishment in 2006 of the NSW and ACT *Centre for Health Record Linkage* (CHeReL),[2] which is based on the methodology developed by the Western Australian *Data Linkage Unit* (Kelman et al. 2002).

While there are many commercial data integration and linkage systems available, most of them are a 'black box' from the user's perspective, in that the details of the technology implemented within the linkage engine are not available. Additionally, many of these systems are specialised to a certain domain, such as integration of business data or cleaning and deduplication of customer mailing lists. In the health sector, however, linkages are often more complex and involve data from many disparate sources, possibly including population data collected outside of the health system (such as electoral rolls or police accident databases), as well as historical data like patients' medical histories. While commercial linkage systems are used in the health sector as core linkage engines, a significant amount of additional programming is often required by a health linkage unit to integrate a linkage engine into a specific domain. This often means that the linkage environment is limited by the functionality of the commercial linkage engine at its core.

Record linkage is a complex process and requires the user to understand, up to a certain degree, many technical details. For example, it is important that a user understands how approximate string comparisons work on name and address strings, as this will influence the way the matching weights will be calculated. Similarly, understanding the trade-off of using certain record fields (or attributes) for blocking or indexing is crucial, as on one hand a certain choice of blocking keys will result in poor linkage quality, while on the other hand another choice of blocking keys will generate too many record pairs and make a linkage computationally infeasible.

While there are several affordable smaller commercial record linkage systems available, they are often limited in their ability to deal with different types of

[1] http://www.ncris.dest.gov.au
[2] http://www.cherel.org.au

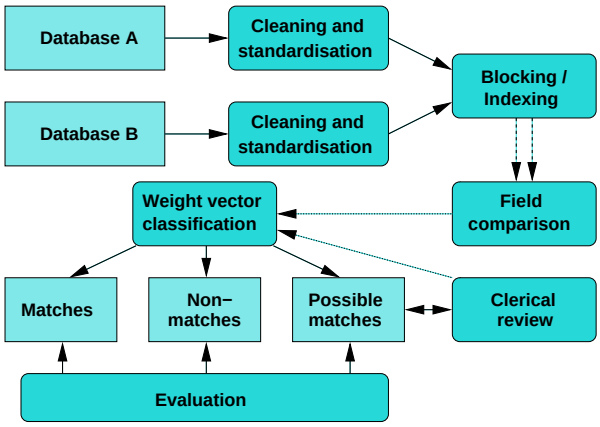

Figure 1: General record linkage process. The output of the blocking step are candidate record pairs, while the comparison step produces weight vectors with numerical similarity weights, that are then classified into matches, non-match and possible matches.

data, only contain a limited amount of functionality (for example implement only certain commonly used string comparison methods), or can only link small data sets. Large scale record linkage system, on the other hand, usually have a very high price tag and are therefore only affordable by large organisations. Almost all commercial systems are a 'black box' from the user's perspective, as the source code of their linkage engines is not available for inspection.

It is therefore important to have tools available that allow record linkage practitioners to experiment with the traditional as well as new advanced record linkage techniques, in order to understand their advantages and their limitations. Such tools should be flexible and contain many different linkage methods, and allow a multitude of configuration options for a user to conduct a variety of experimental linkages. Additionally, as most record linkage users in the health sector do not have extensive experience in programming, an intuitive graphical user interface should provide a well structured and logical way on how to setup and run record linkage projects.

This lack of flexible record linkage systems that allow access to source code and include a large number of linkage techniques is addressed by the freely available *Febrl* linkage system presented in this paper (Christen et al. 2002, 2004). To the best of the author's knowledge, *Febrl* is the only free linkage system with a graphical user interface. First, in the next section, a short overview of the general record linkage process is provided, and in Section 3 the structure and functionality of the *Febrl* user interface is described in detail. This is followed in Section 4 with a discussion of the use of *Febrl* in the health sector, and the paper is then concluded with Section 5 containing an outlook to future work.

2 Record Linkage Process

A general schematic outline of the record linkage process is given in Figure 1. As most real-world data collections contain noisy, incomplete and incorrectly formatted information, data cleaning and standardisation are important pre-processing steps for successful record linkage, and also before data can be loaded into data warehouses or used for further analysis or data mining (Rahm and Do 2000). A lack of good quality data can be one of the biggest obstacles to successful record linkage and deduplication (Clarke 2004). The main task of data cleaning and standardisation is the conversion of the raw input data into well defined, consistent forms, as well as the resolution of inconsistencies in the way information is represented and encoded (Churches et al. 2002).

If two databases, **A** and **B**, are to be linked, potentially each record from **A** has to be compared with all records from **B**. The total number of potential record pair comparisons thus equals the product of the size of the two databases, $|\mathbf{A}| \times |\mathbf{B}|$, with $|\cdot|$ denoting the number of records in a database. Similarly, when deduplicating a database, **A**, the total number of potential record pair comparisons is $|\mathbf{A}| \times (|\mathbf{A}| - 1)/2$, as each record potentially has to be compared with all others. The performance bottleneck in a record linkage or deduplication system is usually the expensive detailed comparison of record fields (or attributes) between pairs of records (Baxter et al. 2003, Christen and Goiser 2007), making it unfeasible to compare all pairs when the databases are large. Assuming there are no duplicate records in the databases (i.e. one record in database **A** can only match to one record in database **B**, and vice versa), then the maximum number of true matches corresponds to the number of records in the smaller database. Therefore, while the computational efforts increase quadratically, the number of potential true matches only increases linearly when linking larger databases. This also holds for deduplication, where the number of duplicate records is always less than the total number of records in a database.

To reduce the large amount of potential record pair comparisons, record linkage methods employ some form of indexing or filtering techniques, collectively known as *blocking* (Baxter et al. 2003): a single record field (attribute) or a combination of fields, often called the *blocking key*, is used to split the databases into blocks. All records that have the same value in the blocking key will be inserted into one block, and candidate record pairs are then generated only from records within the same block. These candidate pairs are compared using a variety of comparison functions applied to one or more (or a combination of) record fields. These functions can be as simple as an exact string or a numerical comparison, can take variations and typographical errors into account (Cohen et al. 2003, Christen 2006), or can be as complex as a distance comparison based on look-up tables of geographic locations (longitudes and latitudes).

Each field comparison returns a numerical similarity value, called a *matching weight*, often in normalised form. Two field values that are equal, therefore, will have a matching weight of 1, while the matching weight of two completely different field values will be 0. Field values that are somewhat similar will have a matching weight somewhere between 0 and 1. A *weight vector* is formed for each compared record pair containing all the matching weights calculated by the different comparison functions. These weight vectors are then used to classify record pairs into *matches*, *non-matches*, and *possible matches*, depending upon the decision model used (Christen and Goiser 2007, Fellegi and Sunter 1969, Gu and Baxter 2006). Record pairs that were removed by the blocking process are classified as non-matches without being compared explicitly. A variety of evaluation measures can then be used to assess the quality of the linked record pairs (Christen and Goiser 2007).

The class of possible matches are those record pairs for which human oversight, also known as *clerical review*, is needed to decide their final linkage status. In theory, it is assumed that the person undertaking this clerical review has access to additional data (or may be able to seek it out) which enables her or him to resolve the linkage status. In practice, however, often

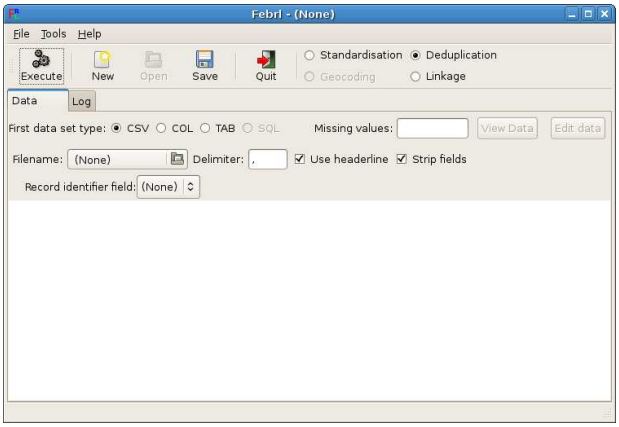

Figure 2: Initial *Febrl* user interface after start-up.

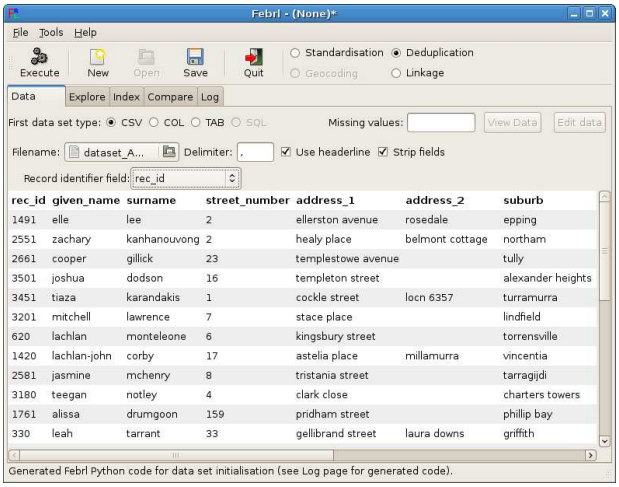

Figure 3: *Febrl* user interface for a deduplication project after the input data set has been initialised. The shown values have been randomly generated.

no additional data is available and the clerical review process becomes one of applying experience, common sense or human intuition to make the decision. The reviewed and manually classified record pairs can also be used as training data for improving the classification quality of subsequent linkages.

3 *Febrl* GUI Structure and Functionality

Febrl is implemented in Python,[3] a free, object oriented programming language that is available on all major computing platforms and operating systems. Originally developed as scripting language, Python is now used in a large number of applications, ranging from Internet search engines and Web applications to steering of computer graphics for Hollywood movies and large scientific simulation codes. Many organisations use Python, including Google and NASA, and due to its clear structure and syntax it is also used by various universities for undergraduate teaching in introductory programming courses.

Python is an ideal platform for rapid prototype development as it provides data structures such as sets, lists and dictionaries (associative arrays) that allow efficient handling of very large data sets, and includes many modules offering a large variety of functionalities. For example, it has excellent built-in string handling capabilities, and the large number of extension modules facilitate, for example, database access and graphical user interface (GUI) development. For the *Febrl* user interface, the PyGTK[4] library and the Glade[5] toolkit were use, which, combined, allow rapid platform independent GUI development.

Febrl is published for free under an open source software licence.[6] Due to the availability of its source code, *Febrl* is suitable for the rapid development, implementation, and testing of new and improved record linkage algorithms and techniques, as well as for both new and experienced users to learn about and experiment with various record linkage techniques.

In previous *Febrl* versions (Christen et al. 2004) the user had to implement his or her linkage or deduplication project by manually developing or modifying a Python program. This required the user to be familiar with both the syntax of the Python programming language, as well as with the many configuration options available in *Febrl*. This limited the application of *Febrl* to technically experienced users. To improve upon this, a GUI has been developed by the author, with the main aim to improve the accessibility of *Febrl* to non-technical record linkage practitioners. The *Febrl* GUI follows the structure provided by the *Rattle* open source data mining tool (Williams 2007). The basic idea is to have a window that contains one page (or tab, similar to tabs in modern Web browsers) per major step of the record linkage process (as shown in Figure 1). The initial *Febrl* GUI after start-up is shown in Figure 2. Only the 'Data' and 'Log' pages are visibly initially, additional pages will appear once the input data has been initialised. Note that *Febrl*'s geocoding functionalities (Christen et al. 2006) have not yet been incorporated into the GUI.

On each page, the user can select a method and its corresponding parameters, and then confirm these setting by clicking on the 'Execute' button. Appropriate Python code will be generated for this step of the record linkage process and shown in the 'Log' page. Once all necessary steps are set up, the generated Python code can be saved and run outside of the GUI, or a standardisation, linkage or deduplication project can be started and its results can be evaluated from within the GUI. All major steps will be described in more detail and illustrated with corresponding screenshots in the following sections.

3.1 Input Data Initialisation

In a first step, a user has to select if she or he wishes to conduct a project for (a) cleaning and standardisation of a data set, (b) deduplication of a data set, or (c) linkage of two data sets. The 'Data' page of the *Febrl* GUI will change accordingly and either show one or two data set selection areas. Several text based data set types are currently supported, including the most commonly used comma separated values (CSV) file format. SQL database access will be added in the near future. As shown in Figure 3, various settings can be selected, such as if a data set file contains a header line with field names (if not these field names can be entered manually); if one of the fields contains unique record identifiers; a list of missing values can be given (like 'missing' or 'n/a') that automatically will be removed when the data is loaded; and there are data type specific parameters to be set as well (such as the delimiter for CSV data sets).

When a user selects an input data set, the first few lines from the corresponding file will be shown, as illustrated in Figure 3. This allows the user to visually verify the chosen settings and change them

[3] http://www.python.org
[4] http://www.pygtk.org
[5] http://glade.gnome.org
[6] https://sourceforge.net/projects/febrl/

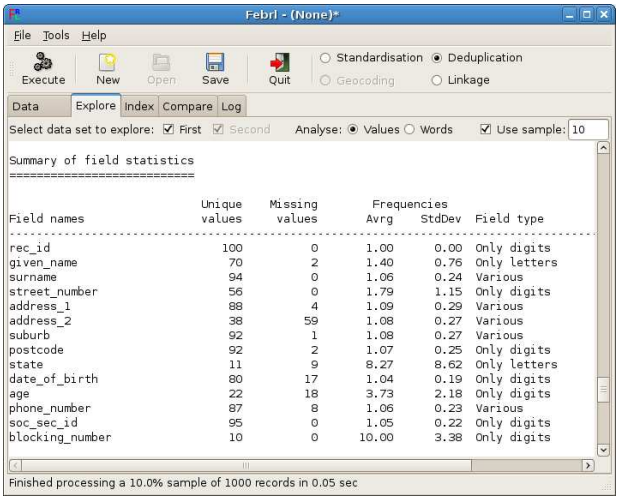

Figure 4: Data exploration page showing summary analysis of record fields (or attributes, columns).

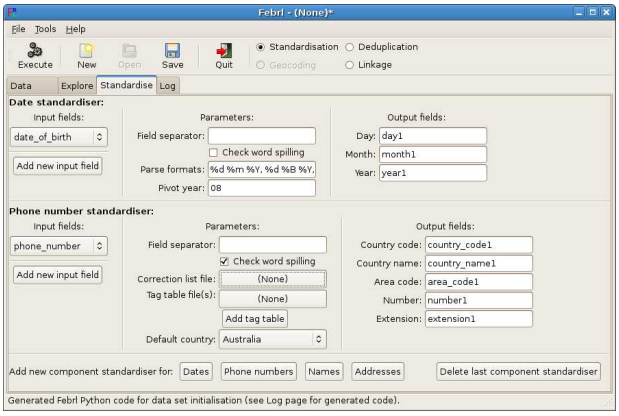

Figure 5: Example date and telephone number standardisers.

if required. When satisfied, a click on 'Execute' will confirm the settings and generate the corresponding Python code segment (which can be viewed in the 'Log' page). The tabs for data exploration and, depending upon the project type selected, standardisation, or indexing, comparison and classification will then become visible.

3.2 Data Exploration

The 'Explore' page allows the user to analyse the selected input data set(s) in order to get a better understanding of the content and quality of the data to be used for a standardisation, deduplication or linkage project. In order to speed up exploration of large data sets, it is possible to select a sampling rate as percentage of the number of records in a data set.

When the user clicks on 'Execute', the data set(s) will be read and all the fields (or attributes, columns) will be analysed. When finished, a report will be displayed in the *Febrl* GUI that for each field provides information about the number of different values in it, the smallest and largest values, the most and least frequent values, the quantiles distribution of the values, the number of records with missing (i.e. empty) value, as well as a guess of the type of the field (if it contains only digits, only letters, or is of mixed type). A summary table of the analysis of all record fields is then reported, as shown in Figure 4.

This report is finalised by a table that calculates the suitability of each field in the data set(s) for indexing (blocking) according to the number and distribution of the field values and number of records with missing (or empty) value. For a deduplication, for example, a field value that occurs c times in a data set will result in $c(c-1)/2$ record pairs being generated in the comparison step. Fields that have a high percentage of records with missing value will not be suitable for use in the indexing (blocking) step.

3.3 Data Cleaning and Standardisation

The cleaning and standardisation of a data set using the *Febrl* GUI is currently done separately from a linkage or deduplication project, rather than as a first step as shown in Figure 1. A data set can be cleaned and standardised and is written into a new data set, which in turn can then be deduplicated or used for a linkage. When a user selects the 'Standardisation' project type, and has initialised a data set on the 'Data' page, she or he can define one or more component standardisers on the 'Standardise' page, as shown in Figure 5.

Currently, component standardisers are available in *Febrl* for names, addresses, dates, and telephone numbers. The name standardiser uses a rule-based approach for simple names (such as those made of one given- and one surname only) in combination with a probabilistic hidden Markov model (HMM) approach for more complex names (Churches et al. 2002), while address standardisation is fully based on a HMM approach (Christen and Belacic 2005). These HMMs currently have to be trained outside of the *Febrl* GUI, using separate *Febrl* modules. Dates are standardised using a list of format strings that provide the expected formats of the dates likely to be found in the uncleaned input data set. Telephone numbers are also standardised using a rules based approach.

Each standardiser requires one or several input fields from the input data set (shown on the left side of a standardiser in the GUI), and cleans and segments a component into a number of output fields (three for dates, five for phone numbers, six for names, and 27 for addresses), shown on the right side in the GUI. Various parameters can be set for each component standardiser in the middle area of the GUI.

It is possible to define more than one standardiser for a component type (i.e. name, address, date or phone number). For example, if a midwives data set contains dates of birth of both mothers and babies, two date standardiser are required. Once initialised and confirmed with a click on 'Execute', on the 'Output/Run' page the details of the standardised output file can be set, such as its file name and a list of fields to pass directly from the input to the output data set without standardisation. A standardisation project can then be started by clicking 'Execute' on the 'Output/Run' page.

3.4 Indexing (Blocking) Definition

Once the required input data sets have been initialised, the indexing method and its corresponding index (blocking) keys have to be defined for a deduplication or linkage. The 'Index' page allows the user to select one of seven possible indexing methods (shown in part (a) of Figure 7). Besides the 'FullIndex' (which will compare all record pairs and thus has a quadratic complexity) and the standard 'BlockingIndex' approach (Baxter et al. 2003) as implemented in many record linkage systems, *Febrl* contains five recently developed indexing methods (Christen 2007): 'SortingIndex', which is based on the sorted neighbourhood approach (Hernandez and Stolfo 1995);

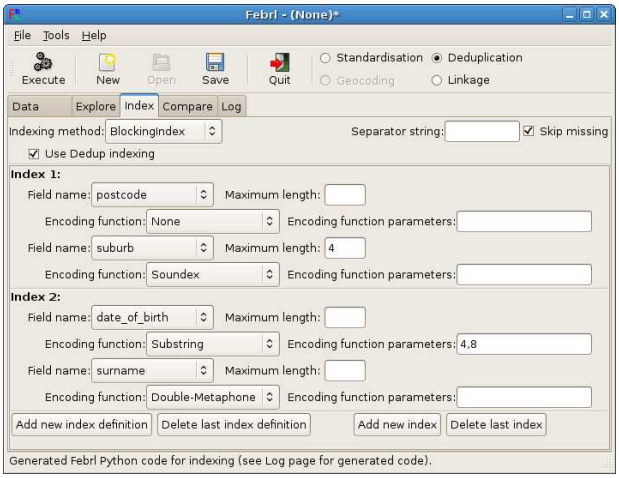

Figure 6: Example indexing definition using the 'BlockingIndex' method and two index definitions.

'QGramIndex', which uses sub-strings of length q (for example bigrams, where $q = 2$) to allow fuzzy blocking (Baxter et al. 2003); 'CanopyIndex', which employs overlapping canopy clustering using TF-IDF or Jaccard similarity (Cohen and Richman 2002); 'StringMapIndex', which maps the index key values into a multi-dimensional space and performs canopy clustering on these multi-dimensional objects (Jin et al. 2003); and 'SuffixArrayIndex', which generates all suffixes of the index key values and inserts them into a sorted array to enable efficient access to the index key values and generation of the corresponding blocks (Aizawa and Oyama 2005).

For deduplication using 'BlockingIndex', 'SortingIndex' or 'QGramIndex', the indexing step can be performed in an overlapping fashion with the field comparison step, by building an inverted index data structure while records are read from the input data set and their blocking key values are extracted and inserted into the index. The current record is compared with all previously read and indexed records having the same blocking key value. This approach can be selected by the user by ticking the 'Dedup' indexing box. For a linkage, and using one of the three indexing methods mentioned above, the *BigMatch* (Yancey 2002) approach can be selected, where first the smaller input data set is loaded and the inverted index data structures are built in main memory, including all record attribute values required in the comparison step. Each record of the larger input data set is then read, its blocking key values are extracted, and all records in the same block from the smaller data set are retrieved from the index data structure and compared with the current record. This approach performs only one single pass over the large data set and does not require indexing, sorting or storing of any of its records. The user can tick the corresponding 'BigMatch' indexing box when conducting a linkage project.

Once an index method has been chosen, the actual index key definitions have to be selected and their various parameters have to be set. Index keys are made of one field value, or a concatenation of several field values, that are often phonetically encoded to group similar sounding values into the same block. Part (b) of Figure 7 lists the encoding functions implemented in *Febrl* (Christen 2006), and Figure 6 shows an example index definition. In this example, a user has selected the standard 'BlockingIndex' method and has defined two index keys. The first index key will be generated by concatenating the values from the 'postcode' field with the first four characters of the 'Soundex' encoded values taken from the 'suburb' field, and the second index key will be generated by taking the fourth to the eighth digit of the 'date_of_birth' field (assumed to be the year of birth) concatenated with the 'Double-Metaphone' encoded values taken from the 'surname' field. Records that have the same values in either of the two index key definitions will be inserted into the same block and compared in the record pair comparison step.

Figure 7: Available methods for indexing (a), string encoding (b), field comparison (c), and weight vector classification (d). These are the pull-down menus from the corresponding *Febrl* GUI pages.

3.5 Field Comparison Functions

The comparison functions to be used to compare the field values of record pairs can be selected and setup on the 'Comparison' page, as shown in Figure 8. Each field comparison requires the user to select one of the many available comparison functions (shown in part (c) of Figure 7), as well as the two record fields that will be compared. While one normally would select fields with the same content from the two data sets (for example, to compare suburb names with suburb names), it is feasible to select different fields (for example to accommodate for swapped given- and surname values). Most of the comparison functions implemented in *Febrl* are variations of approximate string comparisons (Christen 2006, Cohen et al. 2003), while others are special functions that allow the user to compare numerical values, or fields that contain date, age or time values.

All the comparison functions return a raw similarity value between 0 (for total dissimilarity) and 1 (for an exact match). It is possible to adjust these values by setting agreement and disagreement weights,

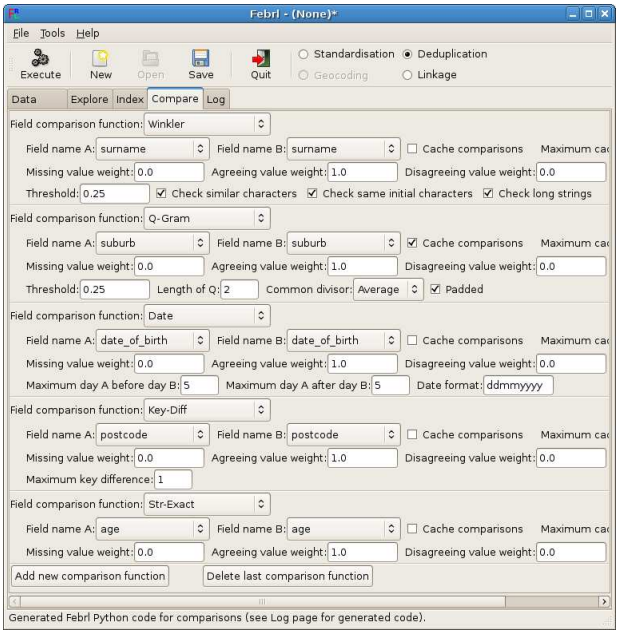

Figure 8: An example of five field comparison function definitions. The comparisons for the 'suburb' field will be cached to improve performance.

as well as a special value that will be returned if one or both of the compared field values are missing (i.e. are an empty value). There is no limit to the number of comparison functions that can be initialised.

As approximate similarity calculations are often computationally quite expensive, it is possible to cache the compared field values and their similarity value to allow fast retrieval of the similarity value for all subsequent comparisons of the same two field values. This will be especially useful for fields that contain a small number of longer values, such as suburb names or business and company names.

3.6 Weight Vector Classification

The last major step required is the selection of the method used for weight vector classification and setting of its parameters. Currently, *Febrl* offers six different classification techniques (listed in part (d) of Figure 7). The simple 'FellegiSunter' classifier allows manual setting of two thresholds (Fellegi and Sunter 1969). With this classifier, the similarity weights of the weight vector of each compared record pair are summed into one matching weight, and record pairs that have a summed weight above the upper classification threshold are classified as matches, pairs with a matching weight below the lower threshold are classified as non-matches, and those record pairs that have a matching weight between the two classification thresholds are classified as possible matches.

With the 'OptimalThreshold' classifier it is assumed that the true match status for all compared record pairs is known (i.e. supervised classification), and thus an optimal threshold can be calculated based on the corresponding summed weight vectors. The match status is assumed to have been generated by an exact comparison of one of the fields in the data set(s). For example, if a field 'entity_id' contains the entity identifiers, an exact match of two records that refer to the same entity will result in a similarity value 1, while all comparisons of records that refer to two different entities will result in a similarity value of 0. Thus these similarity values can be used to determine the true match and non-match status of all

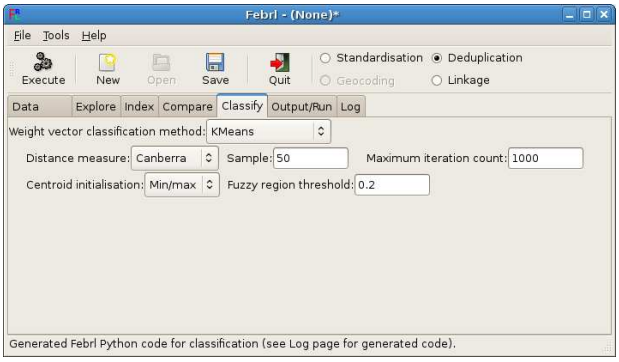

Figure 9: 'KMeans' weight vector classifier.

weight vectors, which in turn can be used to find one optimal classification threshold (i.e. no record pairs will be classified as possible matches).

Both the 'KMeans' and 'FarthestFirst' (Goiser and Christen 2006) classifiers are based on unsupervised clustering approaches, and group the weight vectors into a match and a non-match cluster. Several methods for centroid initialisation and different distance measures can be selected. It is also possible to use only a fraction of the weight vectors when calculating the clusters (using sampling), which will be useful when deduplicating or linking large data sets that have resulted in a large number of weight vectors. These two classifiers also allow the selection of a 'fuzzy region', as described in (Gu and Baxter 2006), which will classify the weight vectors, and thus the corresponding record pairs, in the area half-way between the match and non-match centroids as possible matches. The *Febrl* user interface with the 'KMeans' classifier selected is shown in Figure 9.

The 'SuppVecMachine' classifier uses a supervised support vector machine (SVM) and thus requires the user to provide the true match status (as described above for the 'OptimalThreshold' classifier) of weight vectors in order to be able to train this classifier. It is based on the *libsvm* library (Chang and Lin 2001), and the most important parameters of the SVM classifier can be set in the *Febrl* GUI.

Finally, the 'TwoStep' classifier is an unsupervised approach which in a first step selects weight vectors from the compared record pairs that with high likelihood correspond to true matches and true non-matches, and in a second step uses these vectors as training examples for a binary classifier (Christen 2007). Several methods are implemented on how to select the training examples in the first step, and for the second step a SVM classifier or k-means clustering can be used. Experimental results have shown that this unsupervised approach to weight vector classification can a achieve linkage quality almost as good as fully supervised classification (Christen 2007).

3.7 Output Files and Running a Project

Before a linkage or deduplication project can be started, the way the match status and the matched record pairs will be saved into files have to be set on the 'Output/Run' page. There are also three further output options that can be modified. The first is a percentage value according to which a progress report for the various steps in the linkage process will be reported. In the example shown in Figure 10, for each 10% of record pairs compared a progress message will be given. The second option allows for length filtering in the field comparison step as described in (Gu and Baxter 2004). If this parameter is set to a per-

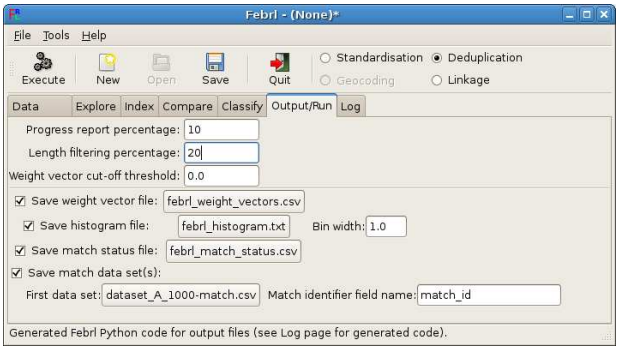

Figure 10: Setting of output options and files for a deduplication project.

centage value, then for all record pairs the length difference of their concatenated field values will be calculated, and if this difference is larger than the given percentage value, then the record pair will not compared using the field comparison functions but directly classified as a non-match. This allows efficient filtering of record pairs that with high likelihood are non-matches. The third parameter allows setting of a weight vector cut-off threshold. If set, all weight vectors with a summed matching weight below this cut-off threshold will not be saved as they will likely correspond to non-matches, thus saving both memory as well as time in the classification step.

There are four output file options available. First, the user can select to save the raw weight vectors produced in the record pair comparison step into a comma separated values (CSV) text file, with the first two columns holding the identifiers of the two compared records, and all following columns the similarity values of the field comparisons conducted on this record pair. The second option is to generate and save into a file a simple text based histogram which shows how many of the compared record pairs have resulted in a certain summed matching weight.

For the third and fourth options, for each of the compared record pairs that was classified as a match a unique match identifier will be generated. In the third option, this match identifier will be saved into a file together with the two record identifiers of the corresponding pair and its summed matching weight, while for the fourth output option the input data set(s) will be copied into a new file (or files) with an additional field (or attribute) that will contain the match identifiers of all classified matches a record was involved with (this can be one, many, or no match at all).

Once the output options have been defined, with a click on the 'Execute' button the complete *Febrl* project code required to run a deduplication or linkage project will be generated and can be saved into a Python file so it can be run later outside of the GUI. It is now also possible to run the defined project, and once the record pairs have been compared and classified their quality can be evaluated within the GUI.

3.8 Evaluation and Clerical Review

As shown in Figure 11, on the 'Evaluation' page the results of a deduplication or linkage project are visualised as a histogram of the summed matching weights of all compared record pairs. If the true match and non-match status of record pairs is available (as discussed earlier in Section 3.6), the quality of the conducted linkage will be shown using the measurements of accuracy, precision, recall and F-measure (or F-score). Note that the accuracy measure commonly used in machine learning applications is often misleading in record linkage applications due to the large number of non-matches compared to matches (Christen and Goiser 2007), i.e. classifying weight vectors is often a very imbalanced classification problem. Accuracy is shown on the *Febrl* GUI to allow users to learn about its misleading characteristic, as well as to compare quality measure results. Measures that allow the evaluation of the complexity of a deduplication or linkage project (i.e. the number of record pairs generated by the indexing step and their quality) are the reduction ratio, pairs completeness and pairs quality (Christen and Goiser 2007), and are shown in the lower right part of the 'Evaluate' page.

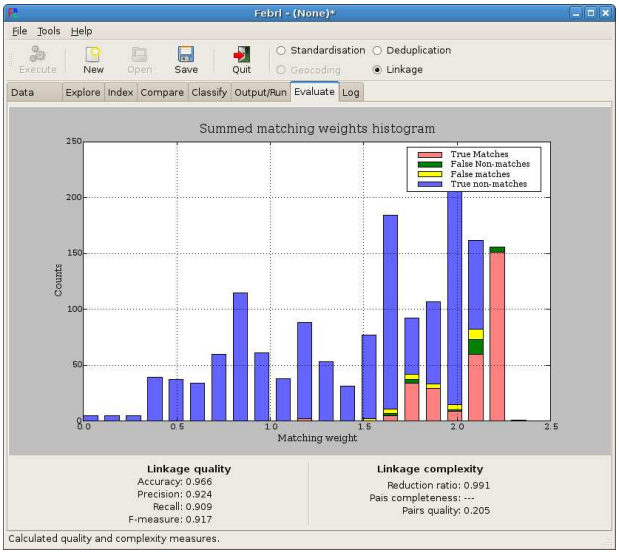

Figure 11: Evaluation page showing the matching weight histogram and quality and complexity measures for a linkage.

Additional evaluation measures, such as the ROC curve or AOC, will be implemented in the near future. An interactive feature that allows manipulation of a classification threshold on the shown histogram will also be added. This will enable a user to move a vertical line, that designates the classification threshold, horizontally over the histogram, with constant updates of the corresponding quality measures shown.

The 'Review' page, which will also be added in the near future, will allow a user to view and manually designate record pairs as matches or non-matches that were originally classified as possibly matches. Note that not all classifiers generate possible matches. The manually classified record pairs can then also be used as training examples, by feeding their match status back to the classifier, as shown in Figure 1, allowing to improve the deduplication or linkage quality.

3.9 Log Page

This page shows the *Febrl* Python code generated when clicking 'Execute' on other GUI pages, as can be see in the example shown in Figure 12. This allows an experienced user to verify the correctness of the generated code, and enables copying of this code into her or his own *Febrl* Python modules. After a deduplication or linkage project has been conducted, a simple textual histogram representing the summed matching weights can be viewed on the 'Log' page.

4 Discussion

For application in the health sector, the *Febrl* record linkage system is a useful tool for several reasons.

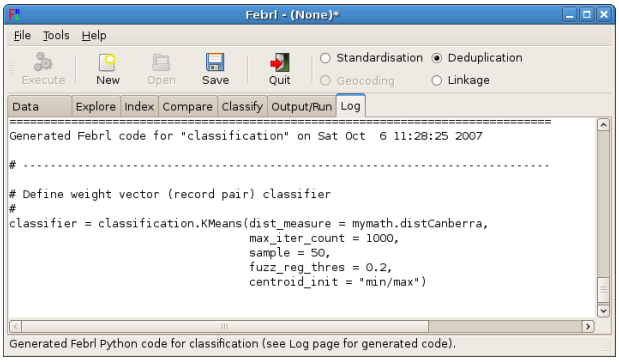

Figure 12: Log page showing the *Febrl* Python code generated for the 'KMeans' classifier from Figure 9.

First, it is freely available and can be installed on all major computing platforms and operating systems. It is based only on software components and libraries that are freely available themselves, thus no costs are involved for the user. This will allow even small organisations or private users to install the *Febrl* software and experiment with it.

Second, it is a flexible tool that contains many recently developed record linkage techniques, including several advanced indexing methods, many different field comparison functions, several phonetic string encoding methods, and both traditional as well as advanced experimental weight vector classifiers. This will allow record linkage users to conduct various experiments combining these techniques and evaluating their suitability for linking their data.

Third, due to the availability of its source code, users can extend the *Febrl* system with their own record linkage techniques, or can adjust them to their needs. For example, a user can add code that allows connection to certain database types, or saving the linkage results into a format specific to a certain domain. The availability of the source code will also allow users to better understand the many algorithms employed within modern record linkage systems.

It is envisaged that *Febrl* can be used in the field of health data linkage in two main areas. First, it is a tool suitable to train new users in record linkage, as it allows them to experiment with both traditional and new advanced methods that are either not yet available in commercial systems or hidden within their 'black box' linkage engines. This will enable record linkage practitioners to extend and deepen their practical knowledge of the many new linkage techniques that have been developed in the past few years.

Second, *Febrl* can be used alongside commercial linkage systems for comparative studies of the linkage results generated. This will allow record linkage practitioners to better evaluate commercial linkage systems, which are often a 'black box' from a user's perspective. It will also allow validation of the linkage results generated by commercial systems, and the many included techniques in *Febrl*, and the availability of its source code, will facilitate fine-tuning of the many different linkage settings.

5 Conclusion and Future Work

In this paper the freely available *Febrl* record linkage system has been presented and its graphical user interface (GUI) has been described in detail. *Febrl* is an training tool suitable for new record linkage users and practitioners, and to conduct small to medium sized experimental linkages and deduplications with up to several hundred thousand records.

Within the health sector, it can be used alongside commercial linkage systems for comparative linkage studies; and for both new and experienced record linkage practitioners to learn about the many advanced linkage techniques that have been developed in recent years and that are implemented in *Febrl*.

The current GUI for *Febrl* does not support *Febrl*'s geocoding functionalities (Christen et al. 2006), and neither does it allow hidden Markov model (HMM) training (Christen 2005, Churches et al. 2002) for name and address standardisation within the GUI. These features will be added in the future. It is also planned to integrate the *Febrl* data set generator (Christen et al. 2006) into the GUI.

Additionally, new indexing, comparisons and classification techniques will be added as they are being published in the record linkage research community. One classification method that will be added soon is the expectation-maximisation (EM) method that has traditionally be used in record linkage for estimation of the classification thresholds in the Fellegi and Sunter approach (Winkler 2000).

Given the importance of privacy and confidentiality in health data linkage (Christen and Churches 2006, Christen 2006), it is aimed to also include privacy-preserving record linkage techniques (Churches and Christen 2004) into a future version of *Febrl*, thus allowing experimental secure linkages of data between health organisations.

6 Acknowledgements

This work is supported by an Australian Research Council (ARC) Linkage Grant LP0453463 and partially funded by the New South Wales Department of Health. The author would like to thank Paul Thomas for proof-reading.

References

Aizawa, A. & Oyama, K. (2005), A fast linkage detection scheme for multi-source information integration, *in* 'Web Information Retrieval and Integration' (WIRI'05), Tokyo, pp. 30–39.

Baxter, R., Christen, P. & Churches, T. (2003), A comparison of fast blocking methods for record linkage, *in* 'ACM SIGKDD Workshop on Data Cleaning, Record Linkage and Object Consolidation', Washington DC, pp. 25–27.

Brook, E.L., Rosman, D.L., Holman, C.D.J. & Trutwein, B. (2005), 'Summary report: Research outputs project, WA Data Linkage Unit (1995-2003)', Western Australian Data Linkage Unit Perth.

Chang, C.-C. & Lin, C.-J. (2001), LIBSVM: A library for support vector machines, manual. Department of Computer Science, National Taiwan University. Software available at: `http://www.csie.ntu.edu.tw/~cjlin/libsvm`

Christen, P., Zhu, J.X., Hegland, M., Roberts, S., Nielsen, O.M., Churches, T. & Lim, K. (2002), High-performance computing techniques for record linkage, *in* 'Australian Health Outcomes Conference' (AHOC'02), Canberra.

Christen, P., Churches, T. & Hegland, M. (2004), Febrl – A parallel open source data linkage system, *in* 'Pacific-Asia Conference on Knowledge Discovery and Data Mining' (PAKDD'04), Sydney, Springer LNAI 3056, pp. 638–647.

Christen, P. (2005), Probabilistic data generation for deduplication and data linkage, *in* 'International Conference on Intelligent Data Engineering and Automated Learning' (IDEAL'05), Brisbane, Springer LNCS 3578, pp. 109–116.

Christen, P. & Belacic, D. (2005), Automated probabilistic address standardisation and verification, *in* 'Australasian Data Mining Conference' (AusDM'05), Sydney.

Christen, P., Willmore, A. & Churches, T. (2006), A probabilistic geocoding system utilising a parcel based address file, *in* 'Selected Papers from AusDM', Springer LNCS 3755, pp. 130–145.

Christen, P. (2006), A comparison of personal name matching: Techniques and practical issues, *in* 'Workshop on Mining Complex Data' (MCD'06), held at IEEE ICDM'06, Hong Kong.

Christen, P. (2006), Privacy-preserving data linkage and geocoding: Current approaches and research directions, *in* 'Workshop on Privacy Aspects of Data Mining' (PADM'06), held at IEEE ICDM'06, Hong Kong.

Christen, P. & Churches, T. (2006), Secure health data linkage and geocoding: Current approaches and research directions, *in* 'National e-Health Privacy and Security Symposium' (ehPASS'06), Brisbane, Australia.

Christen, P. & Goiser, K. (2007), Quality and complexity measures for data linkage and deduplication, *in* F. Guillet & H. Hamilton, eds, 'Quality Measures in Data Mining', Springer Studies in Computational Intelligence, vol. 43, pp. 127–151.

Christen, P. (2007), 'Towards parameter-free blocking for scalable record linkage', *Technical Report TR-CS-07-03, ANU Joint Computer Science Technical Report Series*, The Australian National University, Canberra.

Christen, P. (2007), A two-step classification approach to unsupervised record linkage, *in* 'Australasian Data Mining Conference' (AusDM'07), Gold Coast, Conferences in Research and Practice in Information Technology (CRPIT), vol. 70.

Churches, T., Christen, P., Lim, K. & Zhu, J.X. (2002), 'Preparation of name and address data for record linkage using hidden Markov models', *BioMed Central Medical Informatics and Decision Making*, vol. 2, no. 9.

Churches, T. & Christen, P. (2004), 'Some methods for blindfolded record linkage', *BioMed Central Medical Informatics and Decision Making*, vol. 4, no. 9.

Clarke, D.E. (2004), 'Practical introduction to record linkage for injury research', *Injury Prevention*, vol. 10, pp. 186–191.

Cohen, W.W. & Richman, J. (2002), Learning to match and cluster large high-dimensional data sets for data integration, *in* 'ACM International Conference on Knowledge Discovery and Data Mining' (SIGKDD'02), Edmonton, pp. 475–480.

Cohen W.W., Ravikumar P. & Fienberg S.E. (2003), A comparison of string distance metrics for name-matching tasks, *in* 'IJCAI-03 Workshop on Information Integration on the Web' (IIWeb-03), Acapulco, pp. 73–78.

Fellegi, I.P. & Sunter, A.B. (1969), 'A theory for record linkage', *Journal of the American Statistical Society*, vol. 64, no. 328, pp. 1183–1210.

Goiser K. & Christen, P. (2006), Towards automated record linkage, *in* 'Australasian Data Mining Conference' (AusDM'06), Sydney, Conferences in Research and Practice in Information Technology (CRPIT), vol. 61, pp. 23–31.

Gu, L. & Baxter, R. (2004), Adaptive filtering for efficient record linkage, *in* 'SIAM international conference on data mining' (SDM'04), Lake Buena Vista, Florida.

Gu, L. & Baxter, R. (2006), Decision models for record linkage, *in* 'Selected Papers from AusDM', Springer LNCS 3755, pp. 146–160.

Hernandez, M.A. & Stolfo, S.J. (1995), The merge/purge problem for large databases, *in* 'ACM international conference on management of data' (SIGMOD'95), San Jose, pp. 127–138.

Jin, L., Li, C. & Mehrotra, S. (2003), Efficient record linkage in large data sets, *in* 'International Conference on Database Systems for Advanced Applications' (DASFAA'03), Tokyo, pp. 137–146.

Kelman, C.W., Bass, J. & Holman, C.D.J. (2002), 'Research use of linked health data – A best practice protocol', *Aust NZ Journal of Public Health*, vol. 26, pp. 251–255.

Rahm, E. & Do, H.H. (2000), 'Data cleaning: Problems and current approaches', *IEEE Data Engineering Bulletin*, vol. 23, no. 4, pp. 3–13.

Williams, G.J. (2007), 'Data Mining with Rattle and R', Togaware, Canberra. Software available at: `http://datamining.togaware.com/survivor/`

Winkler, W.E. (2000), 'Using the EM algorithm for weight computation in the Fellegi-Sunter model of record linkage', *Technical report RR2000/05*, US Bureau of the Census.

Yancey, W.E. (2002),'BigMatch: A program for extracting probable matches from a large file for record linkage', *Technical report RR2002/01*, US Bureau of the Census.