High-Performance Computing Techniques for Record Linkage

Peter Christen* and Justin Zhu
Department of Computer Science

Markus Hegland, Stephen Roberts and Ole M. Nielsen
Mathematical Sciences Institute

ANU Data Mining Group
Australian National University
Canberra ACT 0200

Tim Churches and Kim Lim
Epidemiology and Surveillance Branch
New South Wales Department of Health
North Sydney NSW 2059

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

Abstract

Record linkage techniques are used to link together records from one or more data sets relating to the same entity, e.g. patient or customer. As data is often not primarily collected for data analysis purposes, a common unique identifier is missing in many cases, and probabilistic linkage techniques have to be applied. Historical collections of administrative and other (health) data may contain many tens or even hundreds of millions of records, with new data being added at the rate of millions of records per year. Although improvements in available computing power have to some extent mitigated against the effects of this accelerating growth in the size of the data sets to be linked, large-scale probabilistic record linkage is still a slow and resource-intensive process.

The *ANU Data Mining Group* is currently working in collaboration with the *Epidemiology and Surveillance Branch* of the *NSW Department of Health* on the development of improved techniques for probabilistic record linkage. Our main focus is the development of techniques that make good use of modern high-performance parallel computers, and the exploration of data mining and machine learning techniques to reduce the time consuming and tedious aspects of record linkage, such as the manual clerical review process of possible links. The software under development is published under an open source software license and can be downloaded for free from the project web page.

*Corresponding author, E-Mail: Peter.Christen@anu.edu.au

1 Introduction

Record linkage is a rapidly growing field with applications in many areas of health research [1, 7, 9]. It is an initial step in many epidemiological studies and data mining projects, which aim to analyse large and complex data sets to find patterns and rules, to detect outliers or to build predictive models of such data sets. Because data are often not primarily collected for data analysis purposes, a common unique identifier is missing in many cases, and probabilistic linkage techniques as developed by *Fellegi and Sunter* [5] have to be applied.

Historical collections of administrative and other health data nowadays contain many tens or even hundreds of millions of records, with new data being added at the rate of millions of records per annum. Although computing power has increased tremendously in the last decades, large-scale probabilistic record linkage is still a slow and resource-intensive process. There have been relatively few advances over the last decade in the way in which probabilistic record 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. Unlike computers, there has been no increase in the rate at which humans can undertake these clerical tasks.

This paper describes a project currently undertaken by the *ANU Data Mining Group* in collaboration with the *Epidemiology and Surveillance Branch* of the *NSW Department of Health.* The aim of the project is to develop improved techniques for probabilistic record linkage. Our main focus is on the development of techniques that make good use of modern high-performance parallel computing platforms, such as clusters of commodity PCs or workstations (which can be used as virtual parallel computers with some additional software installation), multiprocessor servers or supercomputers. To our knowledge, no probabilistic record linkage software is available for parallel computers. In a second step we will explore data mining and machine learning techniques to improve linkage quality and reduce the time consuming and tedious manual clerical review process for possible links.

The software under development is published under an *open source software license.* It will allow researchers and users in the health area to link much larger data sets. Additional benefits will be reduced costs in conducting such linkages, due to the reduction in human resources needed and the free availability of the software. The software will be made available in stages, with the first components scheduled for release in July 2002.

2 Record Linkage and Data Cleaning

Record linkage techniques are used to link together data records relating to the same entities, such as patients or customers. Record linkage can be used to improve data quality and integrity, to allow reuse of existing data sources for new studies, and to reduce costs and effort in data acquisition.

If no unique identifier is available in the data sets to be linked, probabilistic linkage techniques [5] have to be applied. Moreover, data may be recorded or captured in various formats, and data items may be missing or contain errors. A pre-processing phase that aims to clean and standardise the data is therefore an important first step in every linkage process. Data sets may also contain

duplicate entries, in which case linkage has to be applied within a data set to de-duplicate it before linkage with other files can be attempted.

The process of linking records has various names in different user communities. While epidemiologists and statisticians speak of *record* or *data linkage*, the same process is often referred to as *data scrubbing* or *data cleaning* by computer scientists and in the database community, whereas it is sometimes called *merge/purge processing* in commercial processing of customer databases or mailing lists. Historically, the statistical and the computer science community have developed their own techniques, and until recently few cross-references could be found. In this Section we give an overview and try to identify similarities in the developed methods.

Computer assisted record linkage goes back as far as the 1950s. At this time, most linkage projects were based on ad-hoc heuristic methods. The basic ideas of probabilistic record linkage were introduced by *Newcombe and Kennedy* [14] in 1962 while the theoretical foundation was provided by *Fellegi and Sunter* [5] in 1969. Using frequency counts [20], agreement and disagreement probabilities, each field of a record is assigned a match weight, and critical values of these match weights are used to designate a pair of records either as a link, a possible link or a non-link. Possible links are those pairs for which human oversight, also known as clerical review, is needed to decide their final linkage status. To reduce the number of comparisons (potentially each record in one data set has to be compared with every record in a second data set), *blocking techniques* are used. The data sets are split into smaller blocks using blocking variables, like the postcode or the *Soundex* encoding of surnames. Only records within the same blocks are then compared. To deal with typographical variations and data entry errors, approximate string comparisons [16] are often used for name and addresses. They usually return a score between 0.0 (two strings are completely different) and 1.0 (two strings are the same).

In recent years, researchers have been exploring the use of machine learning and data mining techniques [18] both to improve the linkage process and to allow linkage of larger data sets. For very large data sets, with hundreds of millions of records, special techniques have to be applied [19] to be able to handle such large volumes of data. Sorting large number of records becomes the main bottleneck so extracting possible links from an unsorted large data file [21] has to be done as a pre-processing step before the actual linkage can be done.

The terms *data cleaning*, *standardisation*, *data pre-processing* and *ETL* (extraction, transformation and loading) are used synonymously to refer to the general tasks of transforming the source data (often derived from operational, transactional information systems) into clean and consistent sets of records which are suitable for record linkage or for loading into a data warehouse [17]. The meaning of the term standardisation in this context is quite different from its use in epidemiology and statistics. The main task of standardisation in record linkage is the resolution of inconsistencies in the way information is represented or encoded in the data. Inconsistencies can arise through typographical or other data capture errors, the use of different code sets or abbreviations, and differences in record layouts. Once the data has been standardised, the central task of record linkage is to identify records in the source data sets that represent the same real-world entity. In the computer science literature, this process is also called the *object identity* or *merge/purge* problem [8].

Fuzzy techniques and methods from information retrieval have been applied to solve this problem. One approach is to represent text (or records) as document vectors and compute the *cosine distance* [4] between such vectors. Another possibility is to use an *SQL* like language [6] 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 [11] include statistical outlier identification, pattern matching, clustering and association rules based approaches. Sorting data sets (to group similar records together) and comparing records within a sliding window [8] is a technique similar to blocking as applied by traditional record linkage approaches. The accuracy of the matching can be improved by having smaller window sizes and performing several passes over the data using different keys, rather than having a large window size but only one pass. This corresponds to applying several blocking strategies in a record linkage process.

Even though most approaches described in the computer science literature use approximate string comparison operators and external lookup-tables to improve the matching quality, none considers the statistical theory of record linkage as developed by *Fellegi and Sunter* [5] and improved and extended by others.

The problem of finding similar entities not only applies to records of persons. Increasingly important is the removal of duplicates in web search engines and automatic text indexing systems, where copies of documents have to be identified and filtered out before being presented to the user.

3 Parallel Computing

While high-performance computing was historically restricted to science and engineering, technological advantages in the last decade allowed the dissemination into the commercial IT world. Multiprocessor servers, also called *symmetric multiprocessors* (SMP), are nowadays common in many organisations as compute, database or web servers. These are equipped with a number of processors (CPUs), usually numbering from two up to around 30, have a main memory size in the one to several Gigabytes[1] and they often have disk arrays (RAID) for improved availability with a capacity of several Terabytes[2]. These machines normally use a version of the *Unix* operating system that allows them to run a mixture of sequential as well as parallel jobs. Parallel applications use *threads* - pieces of program code that can run independently from others - for increased performance. For example, in a database server, each transaction can independently update records, or a web server can handle incoming requests simultaneously by processing them on different CPUs.

While multiprocessor servers are still fairly expensive, even ordinary personal computers (PCs) or workstations, connected by a local area network, can be used collaboratively as a (virtual) parallel computer using appropriate software packages. The computing power of a single PC nowadays is comparable to the capabilities of a supercomputer just a decade ago. Office computers can easily be left on all the time, and these idle resources can be used for compute intensive jobs overnight and on weekends.

[1] 1 Gigabyte = 1,024 Megabytes.
[2] 1 Terabyte = 1,024 Gigabytes.

Record linkage in general, and the standardisation process especially, have a good potential for parallelism. The standardisation of each record in a data set can be done independently from all others, which allows efficient parallelism. The blocking technique used in the traditional record linkage process can be used as a starting point for a parallel record linkage system. In Section 5.3 we will describe our approach to parallelisation in more details.

4 Open Source Software

Using *open source software*[3] instead of commercial software can have several advantages. Not only can people get the software at no cost, they can also access and modify the source code, and thus software can evolve and improve as a result of contributions from various parties. This is especially helpful for prototype software such as is the subject of this project. A rapid evolutionary development process often produces better software than the traditional closed model used in the development of commercial software. Examples of successful open source projects include the operating system *Linux*, the database server *MySQL*, the web server *Apache* (the most popular web server on the Internet), and the programming language *Python*.

For our record linkage project, we are using *Python*[4] as the primary programming platform. *Python* is open source software, it is available for many platforms (including *Windows*, *Macintosh* and *Unix*), and it has a strong and active user community. *Python* has been demonstrated to be robust and able to handle large amounts of data efficiently [3, 15]. It provides a very easily learned syntax while providing high-level object-oriented features which make it suitable for the construction of large and complex systems. *Python* provides a very flexible set of built-in data structures such as general lists as well as *dictionaries* (lookup-tables), which are implemented as very efficient hash-tables. Functions can be used as templates that can be changed and extended as needed by the user. *Python* is distributed with a number of extension libraries that contain a large collection of modules for all kinds of tasks, including regular expression parsing, array-based numerical computation, statistics, Internet and Web data handling and encryption. Additionally, many third party modules are available which allow accessing and controlling of other (open source) software through a *Python* interface. For example, interface modules are available for most database systems. It is possible to readily extend the capabilities of *Python* through extension modules written in the *C* programming language, as well as using the *Python* language itself. Thus, existing program libraries written in C can be seamlessly integrated into *Python*.

5 Prototype Software

In this Section we describe in more detail our approach to implement prototype software for parallel high-performance probabilistic record linkage. This software is freely available and can be downloaded from the project web page at:

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

[3] `http://www.opensource.org`
[4] `http://www.python.org`

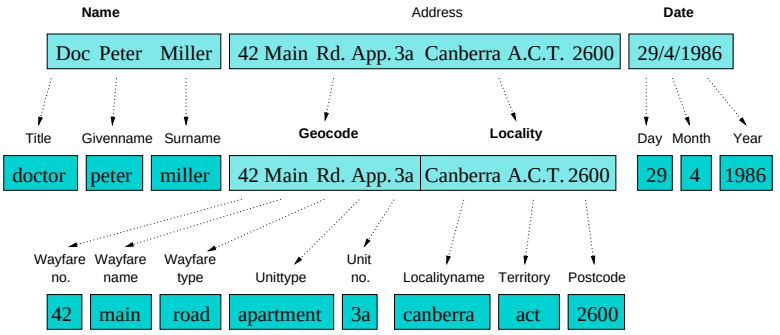

Figure 1: Example standardisation

Only standard *Python* (Version 2.2) is used, so with the exception of *Python* potential users will not have to install any other software packages. Portability of our software should therefore be possible to all platforms where *Python* is available. Data access will in a first version be limited to text files, but we are planning to include database (*SQL*) functionalities using the *Python* database interface later.

The prototype contains two main modules, **pyStandard.py**[5] for the standardisation process and **pyLinkage.py** for the actual record linkage. While the **pyStandard.py** module is available for downloading, the linkage module is still under development at the time of writing (June 2002). One aim of the software is to simplify the configuration process. Only one file (a module called **config.py**) will need to be edited and customised by the user. Within this file, data and lookup-table files, as well as standardisation and linkage parameters can be modified and adjusted to a user's needs. Instead of defining a new pattern matching language, which is the approach taken by the *AutoStan* [12] standardisation or data scrubbing program, only simple lookup-tables will be used. All other functionality, including sophisticated string handling and manipulation, will be implemented within the *Python* code. Due to the open source licensing of the software, users will be free to modify and enhance this functionality, or to request others to do so on their behalf. In time, we expect that different versions of the software will be developed for specific purposes or data sets.

5.1 Standardisation

Standardising a data set is an important first step for successful record linkage. The aim of standardisation is to clean the raw input data records and assign words and numbers to certain output fields. The standardisation module **pyStandard.py** can process four different components of a record, namely **Name**, **Geocode**, **Locality** and **Date**. Figure 1 shows an example standardisation. The module opens input file(s), loads records and splits them into their constituent components, handing each component off to their corresponding parsing routines and finally combining the results into a new standardised record that is then written into an output file. Additionally, log and error information can be saved into files.

[5] Python programs are normally given the file extension **.py**

Table 1: Supported output fields

Component	Name	Geocode	Locality	Date
Fields	Givenname	Wayfaretype	Postcode	Day
	Middlenames	Wayfarename	Localityname	Month
	Surname	Wayfarenumber	Localityqualifier	Year
	Title	Wayfareprefix	Territory	
	Altgivenname	Wayfaresuffix		
	Altsurname	Unittype		
	Gender	Unitnumber		

An input record is cleaned and parsed, and its words and numbers are assigned into the output fields shown in Table 1. It is assumed that the input data is a text file and contains one record per line, with fixed column width (i.e. each input field occupies a well defined range of columns) or comma or tabulator separated fields. Parsed and standardised records are written into a new text file in comma delimited or column wise format. Other modes of input and output, such as reading and writing from and to a database, will be added in later versions. As the standardisation process can be done in parallel, more than one output file can be written (see Section 5.3).

Lookup-tables are used to correct nicknames, expand abbreviations and handle word spelling variations and typographical errors. Figure 2 shows a lookup-table file with Australian state and territory words. Words in an input record that are listed in the right part of an entry (after the colon) are replaced with the corresponding word on the left. A user can easily edit these lookup-table files. Once loaded, they are converted into efficient *Python* mapping data structures known as *dictionaries*. Data for these lookup-tables can often be found on the Internet or purchased from third party suppliers. For example, *Australia Post* provides an updated list of all Australian postcode, suburb and state triplets[6], while the *Australian Whitepages* contain street- and surnames (which can also be used to build frequency distributions for the linkage process), and various other Web sites provide downloadable name and abbreviation lists.

```
# Australian state and territory words
          australia : a, aus, aust, austr
            capital : c, cap, capit, capitol, capt
               land : lnd
    new_south_wales : nsw, new-south-wales, n_s_w
         queensland : qld
           tasmania : tas
          territory : teritory
           victoria : vic, vict
              wales : wals, wal
```

Figure 2: Example lookup-table file for Australian state and territory words

A separate parsing routine handles each of the four input components. It is assumed that the input to a parsing routine is a string that contains the corresponding component of a record. The first step in parsing a component consists in cleaning the input string by converting all letters into lowercase, by removing

[6] `http://www.post.com.au/postcodes/`

unwanted characters and by replacing certain characters by others. For example, a vertical bar | replaces all forms of brackets. In a second step, lookup-tables are used to check for certain (component specific) abbreviations and misspellings, which are then replaced by expanded or corrected versions. For example, using the lookup table in Figure 2, each occurrence of **austr** will be replaced by **australia** in the *locality* component. In a third step, lookup-tables are used to label words, numbers and separators with one or more *tags*. According to these tags, input words are then assigned to the output fields. This can be done in two different ways. The first – traditional – approach is to use rules to decide to which output field a word or number is assigned to. This results in rather complex programs with many rules to cover the various special cases. The second approach is to use a probabilistic *Hidden Markov Model* (HMM). Using standardised and labeled training data the HMM assigns probabilities to a given input sequence. For example, a trained HMM for names would assign probabilities that a name starts in 30% of all cases with a title word, in 60% with a given name and in 10% with a surname. HMMs are successfully applied in speech recognitions and information retrieval, and first experiments with addresses standardisation have been presented in [2]. We will present our experiences using HMMs for name and address standardisation elsewhere.

5.2 Linkage

Once data is cleaned and standardised, the linkage process can be started using the module **pyLinkage.py**. Probabilistic linkage techniques as described in *Fellegi and Sunter* [5] will be implemented in this module. All linkage parameters (like cut-off scores, choice of blocking variables, etc.) can be adjusted by the user in the configuration module **config.py**. While some components of the linkage process, like several *phonetic name encodings* as well as different *approximate string comparators*, have been implemented and tested, the main part of this module is still in the early stages of development at the time of writing and details will be published elsewhere. Some general comments on the parallelisation approach of the linkage process are given in the next Section.

5.3 Parallelisation

Both the standardisation and linkage processes have good potential for parallelisation, as they both consist of smaller independent sub-processes. In this Section, we describe our approach to parallelising both processes. To our knowledge, no parallel record linkage software is currently available.

The standardisation of each record in the input data set(s) can be done independently of all other records. Thus, assuming P processors (or computing nodes) are available, each of these processors gets assigned $(1/P)$th of the input records. The **pyStandard.py** module gets as input arguments the range of records to standardise. For example, if 4 processors are available, and a data set with $100,000$ records has to be processed, the first processor gets records 1 to $25,000$, the second processor gets $25,001$ to $50,000$, the third gets $50,001$ to $75,000$ and the forth processor gets the remaining records. Each of the processors then opens the input file, skips to its first record and then loads and standardises its part of the input file, before writing it into separate output files as explained below.

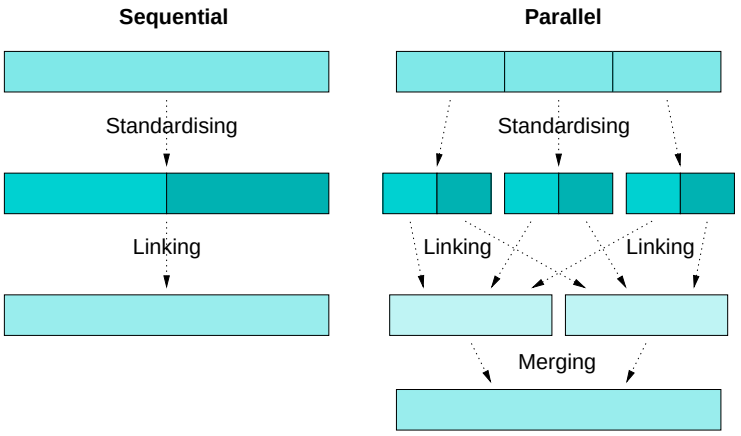

Figure 3: Sequential and parallel record linkage approach

In the linkage process, the blocking technique serves as an excellent starting point for parallelisation. Blocking is used to reduce the number of comparisons. Input data is split according to the values of one or more blocking variables. For example, records with the same year of birth values are moved into separate blocks, and only records with the same blocking values are then compared. As no comparisons are conducted between different blocks, each block can be processed independently from all others. For example, using year of birth as the blocking variable can result in up to around hundred blocks that can be processed independently. One problem with this approach is that blocks can contain different numbers of records, which results in varying processing times. A *dynamic load balancing* strategy has thus to be applied in order to distribute the computing loads onto the processors. A *master-worker* approach [10] will be developed for the parallel record linkage process, where a master *Python* process will coordinate the worker processes by sending them blocks of records to be linked.

For efficient processing, the parallel standardisation process has to write the processed records into files to facilitate the parallel linkage process. Assuming B different blocks for the linkage, each standardisation process writes B smaller files, one per linkage block. If there are P parallel standardisation processes, a total of $P \times B$ files will be written. A linkage process (linking one block) then has to read P of these files and concatenate them to get all the records for one block. This can be done efficiently in *Python*, and eliminates the need to sort or index the files, which is a time consuming process.

Figure 3 shows the sequential and parallel processes for record standardisation and linkage. The only additional step needed in a parallel record linkage system is to merge the final outcomes of the parallel linkage into one single file. The exact mechanism for doing this has yet to be developed.

5.4 Automating Clerical Review

One of the most tedious and time-consuming aspects of current record linkage practice is the *clerical review* process. In the probabilistic linkage model, each

pair of records that has been compared is given a *match weight*. Pairs whose match weight is above a certain threshold are declared *links*, and pairs whose match weight is below another, lower threshold are declared *non-links*. Records whose match weight fall in the *grey* zone between these two critical values are denoted as needing clerical review, i.e. review by a person who is assumed to either have access to additional information external to the files being linked which enables them to resolve whether the pair is a match or not, or who is able bring to bear the power of human reasoning and perception in order to extract any residual hints or clues contained in the pair of records which enable a decision to be made regarding their link status.

In many circumstances additional, external information is simply not available, and thus the clerical review process involves the application of human intuition to try to resolve the doubtful cases. This process is acceptable for small linkage projects, but in larger projects, thousands or even tens of thousands of pairs of records need to be reviewed in this manner. Apart from the tedium involved, it is often difficult to maintain consistency and repeatability when so many often arbitrary decisions need to be made regarding the link status of pairs of records.

An important aim of this project is to explore the utility of supervised machine learning algorithms [13] in the partial or total automation of the clerical review process. Supervised machine learning involves the use of examples in a training data set in order to instruct a learning algorithm to classify data – in this case pairs of clerical review records as either a link or as non-link. The results of these investigations will be reported elsewhere as this work progresses.

6 Outlook

In this paper a project currently undertaken by the *ANU Data Mining Group* in collaboration with the *Epidemiology and Surveillance Branch* of the *NSW Department of Health* has been presented. The prototype software currently under development has been described in some detail. This software is published under an open source software license. It allows researchers and users in the health area to link much larger data sets at reduced costs, due to the reduction in human resources needed and the free availability of the software.

Acknowledgments

This project is equally funded by the *Australian National University (ANU)* and the *NSW Department of Health* under an *AICS (ANU-Industry Collaboration Scheme) AICS #1-2001*. The authors would like to thank everybody who supported this project and helped to make it happen.

References

[1] G.B. Bell and A. Sethi, *Matching Records in a National Medical Patient Index*, Communications of the ACM, Vol. 44 No. 9, September 2001.

[2] V. Borkar, K. Deshmukh and S. Sarawagi, *Automatic segmentation of text into structured records*, in Proceedings of the 2001 ACM SIGMOD international conference on Management of data, Santa Barbara, California, 2001.

[3] P. Christen, O.M. Nielsen and M. Hegland, *DMtools - Open Source Software for Database Mining*, Workshop on Database Support for KDD (at the PKDD'2001 Conference), Freiburg, Germany, September 2001.
Available online at: `http://www.informatik.uni-freiburg.de/~ml/ecmlpkdd/WS-Proceedings/w09/index.html`

[4] W.W. Cohen, *The WHIRL Approach to Integration: An Overview*, in Proceedings of the AAAI-98 Workshop on AI and Information Integration. AAAI Press, 1998.

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

[6] H. Galhardas, D. Florescu, D. Shasha and E. Simon, *An Extensible Framework for Data Cleaning*, Technical Report 3742, INRIA, 1999.

[7] L. Gill, *Methods for Automatic Record Matching and Linking and their use in National Statistics*, National Statistics Methodology Series No. 25, London 2001.

[8] M.A. Hernandez and S.J. Stolfo, *The Merge/Purge Problem for Large Databases*, in Proceedings of the SIGMOD Conference, San Jose, 1995.

[9] C.W. Kelman, *Monitoring Health Care Using National Administrative Data Collections*, PhD thesis, Australian National University, Canberra, May 2000.

[10] V. Kumar, A. Grama, A. Gupta and G. Karypis, *Introduction to Parallel Computing: Design and Analysis of Algorithms*, Benjamin Cummings, Redwood City, 1994.

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

[12] *AutoStan and AutoMatch, User's Manuals*, MatchWare Technologies, Kennebunk, Maine, 1998.

[13] T.M. Mitchell, *Machine Learning*, McGraw-Hill, 1997.

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

[15] O.M. Nielsen, P. Christen, M. Hegland, T. Semenova and T. Hancock, *A Toolbox Approach to Flexible and Efficient Data Mining*, in Proceedings of the PAKDD-2001 Conference, Hong Kong, April 2001. Published in the Springer Lecture Notes in Computer Science, Artificial Intelligence series, LNAI2035.

[16] E.H. Porter and W.E. Winkler, *Approximate String Comparison and its Effect on an Advanced Record Linkage System*, Research Report RR97/02, US Bureau of the Census, 1997.

[17] E. Rahm and H.H. Do, *Data Cleaning: Problems and Current Approaches*, IEEE Bulletin of the Technical Committee on Data Engineering, Vol. 23 No. 4, December 2000.

[18] V.S. Verykios, A.K. Elmagarmid and E.N. Houstis, *Automating the Approximate Record-Matching Process*, Information Sciences, Vol. 126, July 2000.

[19] W.E. Winkler, *Quality of Very Large Databases*, Research Report RR2001/04, US Bureau of the Census, 2001.

[20] W.E. Yancey, *Frequency-Dependent Probability Measures for Record Linkage*, Research Report RR00/07, Statistical Research Division, US Bureau of the Census, July 2000.

[21] W.E. Yancey, *BigMatch: A Program for Extracting Probable Matches from a Large File for Record Linkage*, Research Report RR 2000-01, Statistical Research Division, US Bureau of the Census, March 2002.