Parallel Computing Techniques for High-Performance Probabilistic Record Linkage

Peter Christen*
Department of Computer Science

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

ANU Data Mining Group
Australian National University
Canberra ACT 0200

Tim Churches and Kim Lim
Epidemiology and Surveillance Branch
New South Wales Health Department
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 nowadays contain tens 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 *Epidemiology and Surveillance Branch* of the *NSW Health Department* 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 manual clerical review process for possible links. The developed software will be published under an open source software license. We hope to have prototype software available early in the second half of 2002.

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

1 Introduction

Record linkage is a rapidly growing field with applications in many areas, including health research [1, 6, 8]. As data is often not primarily collected for data analysis purposes, a common unique identifier is missing in many cases, and probabilistic linkage techniques [4] have to be applied. Record linkage 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.

Historical collections of administrative and other health data nowadays contain tens or even hundreds of millions of records, with new data being added at the rate of millions of records per annum. 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. 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 Health Department*. The aim of the project is to develop improved techniques for probabilistic record linkage. The main interests are developing techniques for high-performance linkage on parallel computers, and the exploration of 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 developed software will be 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, which is due to the reduction in human resources needed and the free availability of the software. We hope to have prototype software available early in the second half of 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 [4] have to be applied. Moreover, data can be entered in various formats, and data items can be missing or they contain errors. A preprocessing 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, so linkage has to be applied within a data set to deduplicate 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 linkage* or *data linkage*, the same process is often referred to as *data scrubbing* or *data cleaning* by computer scientists and in the database community. 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* [13] in 1962 while the theoretical foundation was provided by *Fellegi and Sunter* [4] 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 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 preprocessing step before the actual linkage can be applied.

The terms *data cleaning*, *standardisation* and *data preprocessing* 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 which represent the same real-world entity. In the computer science literature, this process is also called the *object identity* or *merge/purge* problem [7].

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* [3] between such vectors. Another possibility is to use an *SQL* like language [5] 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 [10] 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 [7] 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* [4] 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 usually use a version of the *Unix* operating system which 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.

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.

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

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 5.5 we will describe our approach to parallelisation in more details.

4 Open Source Software

Using open source[3] software 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 specially 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 will be 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 [2, 14]. It provides a very easily learnt 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 which can be changed and extended as needed by the user. *Python* is distributed with a number of extension libraries which 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 will be made available (in the second half of 2002) on the project Web page at:

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

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

Only standard *Python* (Version 2.1) will be used for the first version, so 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 (including *Windows*, *Macintosh* and *Unix*). 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 will contain two main modules, **PYstandard.py** for the standardisation process and **PYlinkage.py** for the actual record linkage. One aim 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* [11] 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 standardisation module **PYstandard.py** can process four different components of a record, namely *name*, *address*, *locality* and *date*. 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 which is then written into an output file. Additionally, log and error information can be saved into files.

Table 1: Supported output fields

Component	*name*	*address*	*locality*	*date*
Fields	givenname	wayfaretype	postcode	day
	middlenames	wayfarename	locality	month
	surname	wayfarenumber	localityqualifier	year
	title	wayfareprefix	territory	
	altsurname	wayfaresuffix		
	altsur2name	unittype		
	altgivname	unitnumber		

An input record is parsed and split into different output fields, as 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 5.5).

Lookup-tables are used to correct nicknames, expand abbreviations and handle word spelling variations and typographical errors. Figure 1 shows an excerpt from a lookup-table for titles. If a word is found in the left column of the table, it is replaced with the corresponding word on the right column. These lookup-tables are implemented using 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[5], 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.

dr	dr
doc	dr
phd	dr
doctor	dr
miss	ms
mrs	ms
ms	ms
mr	mr
mister	mr

Figure 1: Example lookup-table for titles

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 unwanted characters and by replacing certain characters by others. For example, all forms of brackets are replaced by a vertical bar |. In a second step, lookup-tables are used to check for certain (component specific) abbreviations, which are then replaced by corrected or expanded versions. For example, **a.k.a.** is replaced by **known as** in the name component, and the word **also** is removed (as it is not a necessary qualifier for alternative names). Next, the input string is split into words and separators. The resulting list then only contains words and certain separator characters.

It is now easy to check if a word in this list is stored in a lookup-table. If so, it can be replaced with the corresponding corrected word and it's type (e.g. title, surname, given name, etc.) can be determined. Separators mark the beginning and end of alternative names, and using the possible structure of an input component (e.g. titles are usually written before given names, which are then written before surnames), rules can be used to extract words into the appropriate output fields.

A new field is added to each standardised output record, which contains status information for each of the processed input fields. A status code (in form of one character) is used per input field. This status code can for example indicate if a postcode has been found in a postcode lookup-table, or if a given name has not been found in a given name lookup-table, but is assumed to be a given name

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

because of its position in the input. The status field can be used by the linkage process to get reliability measures for each field of a record.

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* [4] 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**.

This module was 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 given in Section 5.5.

5.3 Phonetic Encoding

Phonetic encoding is often used to create blocking variables. Several algorithms for phonetic encoding are available, the most populars being *Soundex* and *NYSIIS*. A third, more recently developed algorithm [15] is called *Double-Metaphone*. It accounts for non-english words, like European and Asian names. Similar to *NYSIIS*, *Double-Metaphone* returns a code only consisting of letters, while *Soundex* returns an alpha-numerical code with a fixed length of four. The following Table 2 shows some example encodings.

Table 2: Phonetic word encodings

Word	**Soundex**	**NYSIIS**	**Double-Metaphone**
peter	p360	patar	ptr
christen	c623	chrastan	krstn
ole	o400	ol	al
nielsen	n425	nalsan	nlsn
markus	m622	marc	mrks
hegland	h245	haglan	hklnt
stephen	s315	stafan	stfn
steve	s310	staf	stf
roberts	r163	rabad	rprts
tim	t500	tan	tm
churches	c622	carc	xrxs, xrks
kim	k500	can	km
lim	l500	lan	lm

All three algorithms have been implemented in the prototype software. Note that in some cases *Double-Metaphone* returns two codes, according to two different variations in pronunciation. In general, *Double-Metaphone* seems to be closer to the correct pronunciation of names than *NYSIIS*. All of these phonetic codes are particularly sensitive to errors in the first letter of a name. Therefore, the software includes the ability to compute *reverse* codes from a reversed version of a string.

5.4 Approximate String Comparison

Algorithms for approximate string comparisons are important for good linkage results, as the numerical value they return is used to compute matching weights for string fields like names. Various algorithms for approximate string comparisons have been developed, both in the record linkage [16] and in the computer science and language processing communities.

The *Jaro* string comparator and its modification due to *Winkler* [16] are commonly used in record linkage software. They compute the number of common characters in two strings, the lengths of both strings, and the number of transpositions to compute a similarity measure between 0.0 (two strings are completely different) and 1.0 (both strings are the same). The *Winkler* comparator takes into account that typographical errors occur more often towards the end of words, and thus gives an increased value to agreeing characters at the beginning of the strings.

In the *Bigram* algorithm, the number of common bigrams in the two strings is counted and divided by the average number of bigrams in the two strings. Bigrams are the two-character substrings in a word, e.g. *peter* contains the bigrams *pe*, *et*, *te* and *er*. The *Edit distance* algorithm counts the minimum number of deletions, transpositions and insertions that have to be made to transform one string into the other. The following Table 3 shows various examples and the resulting – normalised – comparison results.

Table 3: Approximate string comparators

Word A	**Word B**	**Jaro**	**Winkler**	**Bigram**	**Edit distance**
shackleford	shackelford	0.970	0.982	0.700	0.818
dunningham	cunnigham	0.896	0.896	0.706	0.800
nichleson	nichulson	0.926	0.956	0.625	0.778
jones	johnson	0.790	0.832	0.400	0.429
massey	massie	0.889	0.933	0.600	0.667
abroms	abrams	0.889	0.922	0.600	0.833
hardin	martinez	0.722	0.722	0.333	0.500
itman	smith	0.567	0.567	0.250	0.000
jeraldine	geraldine	0.926	0.926	0.875	0.889
marhta	martha	0.944	0.961	0.400	0.667
michelle	michael	0.869	0.921	0.615	0.625
julies	julius	0.889	0.933	0.600	0.833
tanya	tonya	0.867	0.880	0.500	0.800
dwayne	duane	0.822	0.840	0.222	0.667
sean	susan	0.783	0.805	0.286	0.600
jon	john	0.917	0.933	0.400	0.750
jon	jan	0.778	0.800	0.000	0.667
peter	ole	0.511	0.511	0.000	0.200

5.5 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.

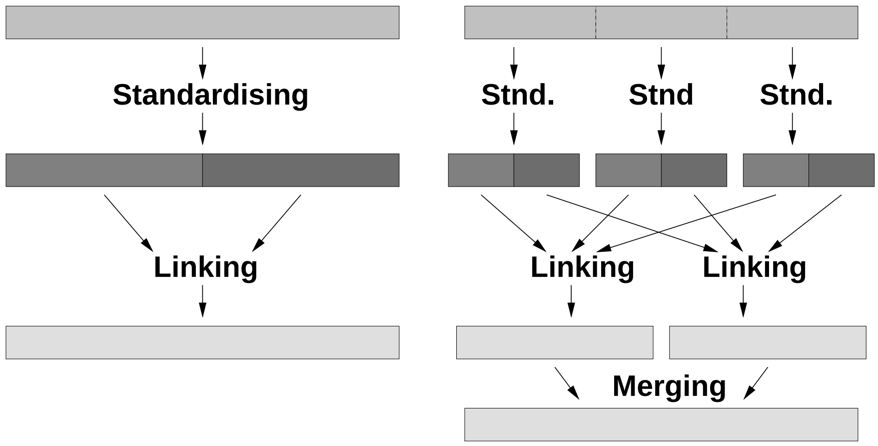

Figure 2: Sequential and parallel record linkage approach.

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 processors gets the remaining records. Each of the processors then opens the input file, skips to its first record and loads and standardises its part of the input file, before writing it into separate output files as explained below.

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 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 [9] 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 2 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.

5.6 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 which have 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 [12] 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 Health Department* has been presented. The prototype software currently under development has been described in some detail. This software will be published later this year under an open source software license. It will allow 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 Health Department* 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] 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`

[3] 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.

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

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

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

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

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

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

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

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

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

[13] 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.

[14] 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.

[15] L. Philips, *The Double-Metaphone Search Algorithm*, C/C++ User's Journal, Vol. 18 No. 6, June 2000.

[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.