A Comparison of Fast Blocking Methods for Record Linkage *

Rohan Baxter
CSIRO Mathematical and Information Sciences
GPO Box 664
Canberra ACT 2601, Australia
Rohan.Baxter@csiro.au

Peter Christen
Dept. of Computer Science
Australian National University
Canberra ACT 0200, Australia
christen@cs.anu.edu.au

Tim Churches
Centre for Epidemiology and Research
NSW Department of Health
Locked Bag 961
North Sydney 2059, Australia
tchur@doh.health.nsw.gov.au

ABSTRACT

Record linkage of millions of individual health records for ethically-approved research purposes is a computationally expensive task. Blocking methods are used in record linkage systems to reduce the number of candidate record comparison pairs to a feasible number whilst still maintaining linkage accuracy. New blocking methods have been implemented recently using high-dimensional indexing or clustering algorithms.

We compare two new blocking methods, bigram indexing and canopy clustering with TFIDF (Term Frequency/Inverse Document Frequency), with two older methods of standard traditional blocking and sorted neighbourhood blocking. The results show that recently blocking methods such as bigram indexing and canopy clustering provide scalable blocking methods while maintaining or improving upon record linkage accuracy. There is a potential for large performance speed-ups and better accuracy to be achieved by these new blocking methods.

Categories and Subject Descriptors

I.5.3 [**Pattern Recognition**]: Clustering; H.3.3 [**Information Storage and Retrieval**]: Information Storage and Retrieval—*clustering*

General Terms

Performance Evaluation

Keywords

Record linkage, object reconciliation, data integration

*CMIS Technical Report 03/139. This is the 6-page version of the paper published in the First Workshop on Data Cleaning, Record Linkage and Object Consolidation, KDD 2003, Washington DC, August 24-27 2003.

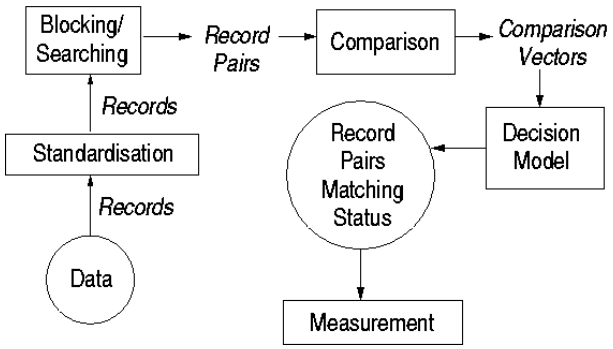

Figure 1: Process diagram of a record linkage system

1. INTRODUCTION

Record linkage techniques are used to link together records which relate to the same entity (e.g. patient or customer) in one or more data sets where a unique identifier is not available. Record linkage is an important initial step in many research and data mining projects in the biomedical and other sectors, where it is used to improve data quality and to assemble longitudinal or other data sets which would not otherwise be available.

Figure 1 shows the processing view of a standard record linkage system architecture as implemented in *TAILOR* [4], *Febrl* [2] or *AutoMatch* [7]. The major challenges in record linkage are computational complexity and linkage accuracy. Linking data sets with millions of records can take from hours to days on modern computing systems. Recent developments in information retrieval, database systems, machine learning and data mining have the potential to improve the efficiency and accuracy of record linkage system components. These developments include efficient blocking methods, adaptive distance metrics for evaluation of record pair similarity and learning methods for the classification task of deciding whether a record pair is a match, non-match or possible match.

As potentially each record in one data set has to be compared to all records in a second data set, the number of record pair comparisons grows quadratically with the number of records to be matched. This approach is computation-

ally infeasible for large data sets. To reduce the huge number of possible record pair comparisons, traditional record linkage techniques work in a *blocking* fashion, i.e. they use a record attribute (or sub-set of attributes) to split the data sets into blocks. Record pairs for detailed comparison are then generated from all the records in the same block (i.e. records with the same value in a blocking attribute). Such detailed comparison functions include approximate string comparisons for names and addresses, and date or age comparisons (e.g. for date of birth) [1].

In this paper, we focus on comparing the speed and accuracy of new blocking methods with established blocking method implementations. The performance bottleneck in a record linkage system is usually the evaluation of a similarity measure between pairs of records. The choice of a good blocking method can greatly reduce the number of record pair evaluations to be performed and so achieve significant performance speed-ups. We consider alternative clustering methods for forming blocks in the record linkage process. Recent work by McCallum, Nigam and Unger[11], Cohen and Richman[3], and others [9] have proposed the use of high-dimensional similarity indexing to improve the efficiency of blocking methods. The similarity of blocking to clustering has previously been observed [3, 11].

We compare *Standard Blocking* [8], the *Sorted Neighbourhood* method [6], *Bigram Indexing* [1] and *Canopy Clustering with TFIDF* [11]. This paper's contribution is to empirically compare the speed-up and accuracy (sensitivity and specificity) performance of these blocking methods. Blocking methods directly affect sensitivity (if record pairs of true matches are not in the same block, they will not be compared and can never be matched) and indirectly affect specificity (as a better reduction ratio of the number of record pair comparisons allows more computationally intensive comparators to be employed).

Our particular purpose is record linkage of public health records for ethically-approved research purposes. For this linkage accuracy is of critical importance in avoiding the introduction of unnecessary noise into the linkage results. This is needed to enable the reliable testing of the public health hypotheses addressed in the research projects. We note that linkage accuracy in other applications, such as mail-merge, bibliographic matching and web-site source integration, is desirable but not as critical. A decision model incorporating costs for a record linkage system has been proposed by Verykios et al [12].

This paper is structured as follows. In the next section we describe the details of the four blocking methods under investigation. Section 3 presents the experimental protocol, evaluation metrics and data sets, and in Section 4 we discuss the results of various experiments. Conclusions and future work are then presented in Section 5.

2. BLOCKING METHODS

2.1 Standard Blocking

The *Standard Blocking* (SB) method clusters records into blocks where they share the identical *blocking key* [8]. A blocking key is defined to be composed from the record attributes in each data set. An example of a blocking key is the first four characters of a *surname* attribute. A blocking key can also be composed of more than than one attribute, for example, a *postcode* attribute could be combined with an *age category* attribute.

There is a cost-benefit trade-off to be considered in choosing the blocking keys [10]. If the resulting blocks contain a large number of records, then more record pairs than necessary will be generated, leading to an inefficiently large number of comparisons. For example, using a *gender* attribute as blocking key puts all the available records into two very large blocks. On the other hand, if the blocks of records are too small, then true record pairs may be missed, therefore reducing linkage accuracy (sensitivity). For example, blocking on a *social security number (SSN)* attribute leads to almost as many small blocks as there are individuals in the data sets. Errors in the SSN may also mean true record pair matches are not included in the same blocks.

A further consideration in the choice of blocking keys is the error characteristics of the record attributes used. To achieve good linkage accuracy, it is preferable to use the least error-prone attributes available. Multiple blocking keys are also used as a way to mitigate the effects of errors in blocking keys. Several passes (iterations) with different blocking keys are performed (for example in *AutoMatch*), resulting in different blocks and different record pair comparisons. Multiple passes improve linkage accuracy but efficient implementation and tuning of the multiple blocks and multiple sets of record comparisons can be dfficult to achieve.

Another strategy used to reduce the effects of spelling and transcription errors [4] in record attributes used as blocking keys (typically name and address attributes) is to use phonetic encodings.

Assuming two data sets with n records each are to be linked, and the blocking method resulted in b blocks (all of the same size containing n/b records), the resulting number of record pair comparisons is $O(\frac{n^2}{b})$ [4]. This is of course the ideal case, hardly ever achievable with real data. Thus, the number of record pair comparisons can be dominated by the the largest block.

2.2 Sorted Neighbourhood

The *Sorted Neighbourhood* (SN) method [6] sorts the records based on a sorting key and then moves a window of fixed size w sequentially over the sorted records. Records within the window are then paired with each other and included in the candidate record pair list. The use of the window limits the number of possible record pair comparisons for each record to $2w - 1$. The resulting total number of record pair comparisons (assuming two data sets with n records each) of the sorted neighbourhood method is $O(wn)$ [4].

Similar to standard blocking, it is advantageous to do several passes (iterations) with different sorting keys and a smaller window size than one pass only with a large window size [5].

One problem with the sorted neighbourhood method arises if a number of records larger than the window size have the same value in a sorting key. For example, having a sorting key *surname*, hundreds of records can have a value of *'smith'*,

Blocking method, parameter	Number of blocks, $n = 9974$
Standard Blocking, $l = 1$	26
Standard Blocking, $l = 3$	1941
Standard Blocking, $l = 6$	5664
Bigram Indexing, $t = 0.2$	23695
Bigram Indexing, $t = 0.6$	48786
Bigram Indexing, $t = 0.9$	5663

Table 1: Number of blocks produced by Standard Blocking and Bigram Indexing with $n = 9974$. l is the length of the blocking key values, and t the bigram threshold.

and if the window size is small not all records with a surname value *'smith'* will be compared.

2.3 Bigram Indexing

The *Bigram Indexing* (BI) method [1] as implemented in the *Febrl* [2] record linkage system allows for *fuzzy blocking*. The basic idea is that the blocking key values are converted into a list of bigrams (sub-strings containing two characters) and sub-lists of all possible permutations will be built using a threshold (between 0.0 and 1.0). The resulting bigram lists are sorted and inserted into an inverted index, which will be used to retrieve the corresponding record numbers in a block.

For example, a blocking key value *'baxter'* will result in a bigram list *('ba','ax','xt','te','er')*. With a threshold of 0.8 the following sub-lists of length 4 (calculated as the length of the bigram list times the threshold: 5×0.8) will be inserted into the inverted index:

('ax','xt','te','er')
('ba','xt','te','er')
('ba','ax','te','er')
('ba','ax','xt','er')
('ba','ax','xt','te')

All record numbers which contain the blocking key value *'baxter'* will be inserted into five blocks, thus increasing the number of record pair comparisons compared to standard blocking.

The number of sub-lists created for a blocking key value both depends on the length of the value and the threshold. The lower the threshold the shorter the sub-lists, but also the more sub-lists there will be per blocking key value, resulting in more (smaller blocks) in the inverted index.

In the information retrieval field, bigram indexing has been found to be robust to small typographical errors in documents [3].

Like standard blocking, the number of record pair comparisons with two data sets with n records each, b blocks all containing the same number of records is $O(\frac{n^2}{b})$ [4]. However, as discussed above the number of blocks b will much larger in bigram indexing. This is demonstrated for a particular data set in Table 1.

2.4 Canopy Clustering with TFIDF

Canopy Clustering with TFIDF (Term Frequency/Inverse Document Frequency) forms blocks of records based on those records placed in the same canopy cluster. A canopy cluster is formed by choosing a record at random from a candidate set of records (initially, all records) and then putting in its cluster all the records within a certain *loose threshold* distance of it. The record chosen at random and any records within a certain *tight threshold* distance of it are then removed from the candidate set of records. We use the TFIDF distance metric, where bigrams are used as tokens. The algorithm and details are found in [3, 11].

The number of record pair comparisons resulting from canopy clustering is $O(\frac{fn^2}{c})$ [11] where n is the number of records in each of the two data sets, c is the number of canopies and f is the average number of canopies a record belongs to. The threshold parameter should be set so that f is small and c is large, in order to reduce the amount of computation. However, if f is too small, then the method will not be able to detect typographical errors.

3. DATA SETS AND EVALUATION METRICS

3.1 Linkage Software

We use *Febrl* [2] as a flexible, modular testbed [1]. *Febrl* (version 0.2) implements standard blocking and bigram indexing. We implemented sorted neighbourhood and canopy clustering with TFIDF. Written in the object-oriented scripting language Python and published as open source software, *Febrl* is an ideal platform to implement new algorithms and techniques for all aspects of data cleaning, standardisation and record linkage.

3.2 Data Sets and Blocking Fields

We used *DBGen* [5] to artificially generate mailing list data containing surnames, given names and other attributes. The results of the experiments are dependent on the error characteristics of this data source, which are documented elsewhere [6]. The *DBGen* default parameters were used, which specify uniformly sized clusters. We choose the number of clusters to be half the number of records, which results in most records having a single duplicate. However some records will have more than one other record with a true match and some will have no duplicates.

The blocking key in *Febrl* was specified to be the surname attribute truncated to 4 characters concatenated with the given name attribute truncated to 4 characters. Note that not all names in the data sets are 4 characters long and so the actual length of the blocking key is variable with a maximum length of 8 characters. We parameterise the blocking key by its length, so that the values can be truncated to 1, 3 or 6 characters. As an example a surname value *'baxter'* and given name *'rohan'* will result in a blocking key value *'baxtroha'*.

Febrl is configured so that no standardisation or cleaning is performed on the input data sets, rather the values generated by *DBGen* are directly used to form the blocking indexes.

3.3 Evaluation Metrics

We added an evaluation module to *Febrl* to calculate three performance metrics for blocking methods [4]. These methods require that the true identity information for record pairs is available in the test data sets. The first metric is the *reduction ratio* (RR), which is defined as: $RR = 1 - \frac{s}{N}$, where s is the number of record pairs produced by a blocking method for comparison and N is the number of possible record pairs in the entire data sets (assuming we link two data sets with n records each $N = n \times n$). RR is the relative reduction in the number of record pairs to be compared.

RR does not measure the time taken for a particular implementation of a blocking algorithm. The time taken for two methods with the same RR could vary considerably. Some methods may require a sort, which for large data sets is a time consuming operation [13].

The second metric is the *pairs completeness* (PC) metric which is defined as $PC = \frac{s_M}{N_M}$, where s_M is the number of true match record pairs in the set of record pairs produced for comparison by the blocking method and N_M is the total number of true match record pairs in the entire data.

The third metric is the F *score*, which combines RR and PC via a harmonic mean, $Fscore = \frac{2 \times PC \times RR}{PC + RR}$. It captures the trade-off between the pairs completeness and reduction ratio metrics.

We focus on RR and PC, rather than precision and recall of the overall linkage results, because that allows direct evaluation of the indexing methods without possible confounding of results with the comparison and decision methods.

4. EXPERIMENTAL RESULTS

We considered standard blocking with the blocking key values truncated to 1, 3 and 6 characters, denoted *B-1*, *B-3* and *B-6* in the graphs. For the sorted neighbourhood method, we used windows with $w = 5, 10$ and 20, denoted by *W-5*, *W-10* and *W-20* in the graphs. For the standard blocking and the sorted neighbourhood methods Figure 2 shows the pairs completeness results, Figure 3 the reduction ratio results and Figure 4 the F score results.

These results show trends similar to those produced by *TAILOR* [4] under a similar experimental framework. Standard blocking trades off pairs completeness with reduction ratio performance as the number of blocks b increases. More smaller blocks results in less comparisons but more true match pairs are missed.

The sorted neighbourhood method avoids the extremes in performance of standard blocking and its behaviour changes predictably as the window size w in increased. With larger windows, pairs completeness results improve, but the reduction ratio decreases.

For bigram indexing and canopy clustering with TFIDF Figure 5 shows the pairs completeness results, Figure 6 the reduction ratio results and Figure 7 the F score results.

Both bigram indexing and canopy clustering outperform the two earlier blocking methods with the right parameter settings. The increased performance is very significant. For

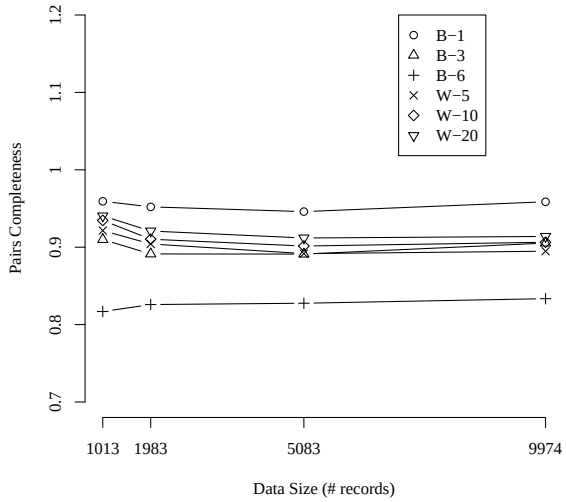

Figure 2: Pairs completeness for standard blocking and sorted neighbourhood methods

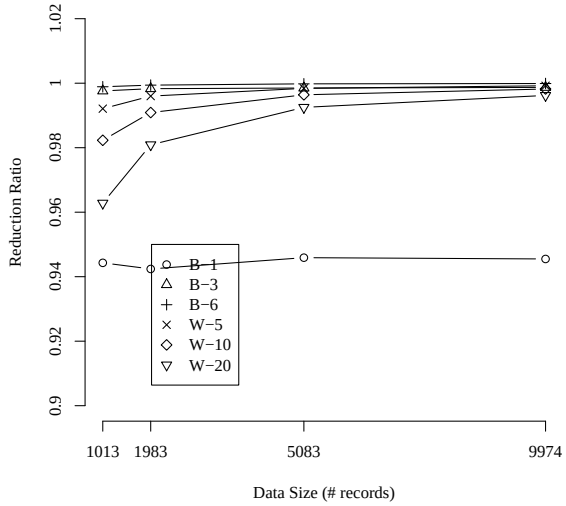

Figure 3: Reduction ratio for standard blocking and sorted neighbourhood methods

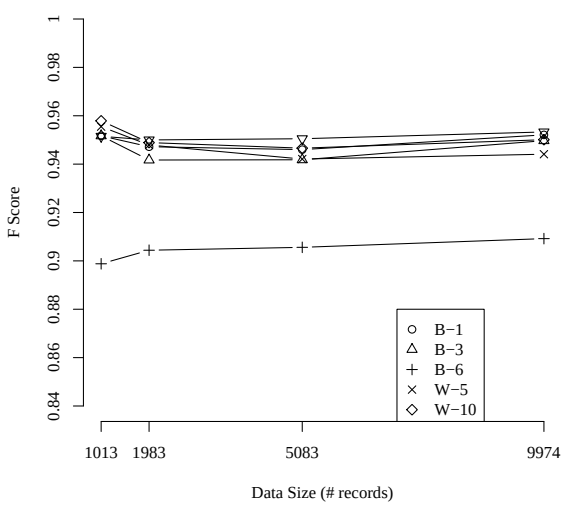

Figure 4: F score for standard blocking and sorted neighbourhood methods

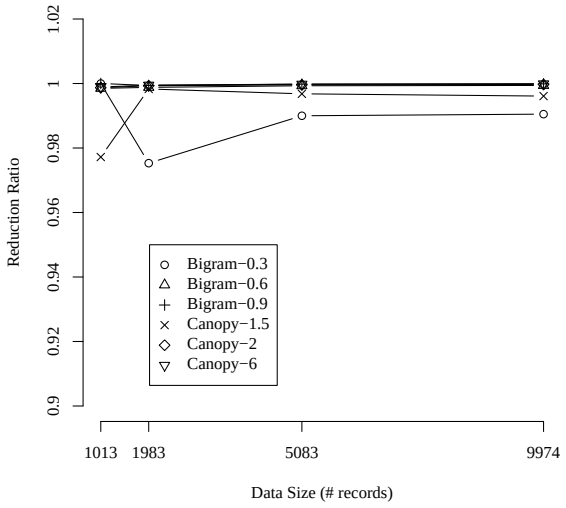

Figure 6: Reduction ratio for bigram indexing and canopy clustering methods

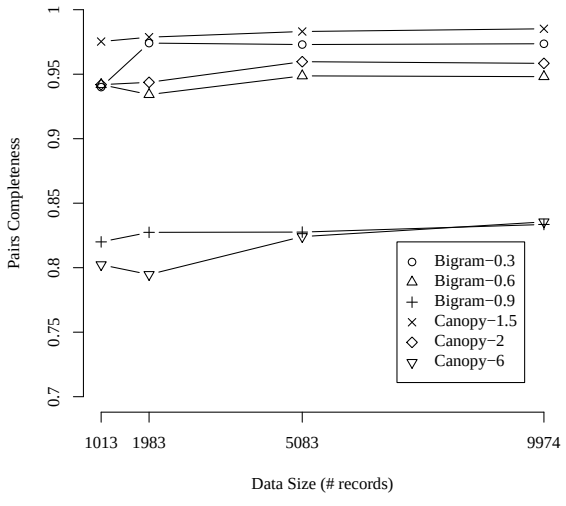

Figure 5: Pairs completeness for bigram indexing and canopy clustering methods

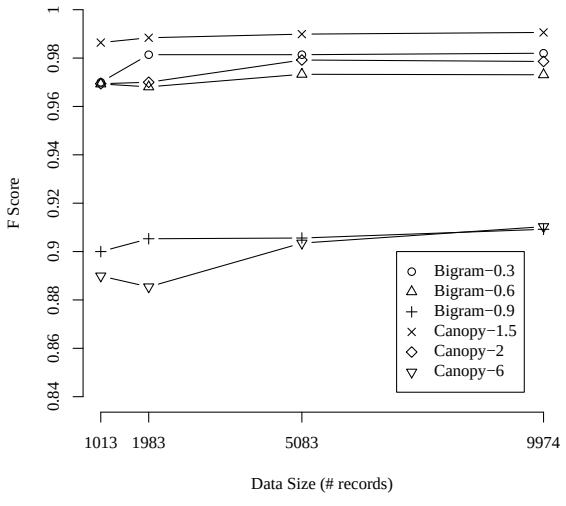

Figure 7: F score for bigram indexing and canopy clustering methods

Data set size n	Canopy cluster average size	Canopy cluster maximum size
1013	1.9	9
1983	2.3	21
5084	3.9	65
9974	5.9	157

Table 2: Canopy cluster average and maximum sizes for different data set size n and loose threshold $= 1.5$

example, pairs completeness with the two earlier methods had a maximum of about 0.96, whereas the maximum for canopy clustering with TFIDF (with optimal parameter settings) is 0.98. For the data set with $n = 9974$ records, that amounts to missing $(0.98 - 0.96) \times 9974 = 200$ (2%) true matches at the end of the blocking component of record linkage.

Bigram indexing performs best with a threshold parameter of $t = 0.3$. This results in blocks made of 3 bigrams for these data sets.

Canopy clustering performs best with the loose threshold set to 1.5. A loose threshold of 1.0 leads to a very poor reduction ratio and huge run times. Canopy clustering reduction ratio drops slightly for the data set with $n = 9974$ records. We looked more closely at what causes this degradation by tracking the size of the canopy clusters produced for loose threshold of 1.5. This is shown in Table 2. For a fixed threshold, the cluster size is growing, leading to more record pairs being generated.

Canopy clustering is the only stochastic blocking method discussed here and so its results will vary from run to run. We ran each experiment ten times and produce the average. The standard deviation of the ten results for any of the three metrics considered was never larger than 0.002. This shows the variation between canopy clustering runs does not affect the ranking of the results.

5. CONCLUSIONS AND FUTURE WORK

This paper describes work in progress. We wish to consider other promising fast indexing methods. One class of candidates is vector subspace mapping [9]. Recent work by Jin, Li and Mehrotra [9] have applied one of these methods, Fastmap, to record linkage. An empirical and theoretical comparison of Fastmap and other vector subspace mapping methods with the methods discussed in this paper is of interest.

The experimental comparisons should be extended to non-*DBGen* data sets to investigate dependencies on data sources and their error characteristics. The indexing methods should also then be extended to non-name and address attributes, such as dates and postcodes. Implementation issues for indexing methods and effects on the run time will also be considered in further work.

The main contribution of this paper has been the direct evaluation of reduction ratio and pairs completeness for some diverse indexing methods on artificial data sets from a widely used database generator.

6. ACKNOWLEDGMENTS

The development of the *Febrl* record linkage system was funded by the Australian National University (ANU) and the NSW Department of Health under an AICS (ANU-Industry Collaboration Scheme) AICS #1-2001. Additional funding was provided by the Australian Partnership for Advanced Computing (APAC).

7. REFERENCES

[1] P. Christen and T. Churches. *Febrl: Freely extensible biomedical record linkage Manual*, release 0.2 edition, April 2003.

[2] P. Christen, T. Churches, et al. http://datamining.anu.edu.au/linkage.html.

[3] W. Cohen and J. Richman. Learning to Match and Cluster Large High-Dimensional Data Sets for Data Integration. In *SIGKDD'02*, 2002.

[4] M. Elfeky, V. Verykios, and A. Elmagarmid. TAILOR: A Record Linkage Toolbox. In *Proc. of the 18th Int. Conf. on Data Engineering*. IEEE, 2002.

[5] M. Hernandez and S. Stolfo. The Merge/Purge Problem for Large Databases. In *Proc. of 1995 ACT SIGMOD Conf.*, pages 127–138, 1995.

[6] M. Hernandez and S. Stolfo. Real-world data is dirty: data cleansing and the merge/purge problem. *Journal of Data Mining and Knowledge Discovery*, 1(2), 1998.

[7] M. Jaro. Software Demonstrations. In *Proc. of an International Workshop and Exposition - Record Linkage Techniques*, Arlington, VA, USA, 1997.

[8] M. A. Jaro. Advances in Record Linkage Methodology as Applied to Matching the 1985 Census of Tampa, Florida. *Journal of the American Statistical Society*, 84(406):414–420, 1989.

[9] L. Jin, C. Li, and S. Mehrotra. Efficient Record Linkage in Large Data Sets. In *Int. Conf. on Database Systems for Advanced Applications (DASFAA)*, Tokyo, Japan, March 2003.

[10] B. Kilss and W. Alvey, editors. *Record Linkage Techniques - 1985. Proceedings of the Workshop on Exact Matching Methodologies in Arlington, Virginia May 9-10*. Internal Revenue Service Publication, Washington, DC, 1985.

[11] A. McCallum, K. Nigam, and L. Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *Proc. of the sixth ACM SIGKDD Int. Conf. on KDD*, pages 169–178, 2000.

[12] V. Verykios, G. Moustakides, and M. Elfeky. A Bayesian decision model for cost optimal record matching. *The VLDB Journal*, 2002.

[13] W. Yancey. Big Match: A program for extracting probably matches from a large file for record linkage. Rrc2002/01, U.S. Census Bureau, 2002.