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

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. Blocking methods partition the data sets into blocks or clusters of records which share a blocking attribute or are otherwise similar with respect to a defined criterion.

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

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

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. 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 computationally infeasible for large data sets. To reduce the 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.

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[7], Cohen and Richman[2], and others have proposed the use of high-dimensional similarity indexing to improve the efficiency of blocking methods. The similarity of blocking to clustering has been observed previously [2, 7].

We compare *Standard Blocking* [6], the *Sorted Neighbourhood* method [5], *Bigram Indexing* [1] and *Canopy Clustering with TFIDF* [7]. 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).

2. BLOCKING METHODS

The *Standard Blocking* (SB) method clusters records into blocks where they share the identical *blocking key* [6]. A

blocking key is defined to be composed from the record attributes in each data set. 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 $O(\frac{n^2}{b})$ [3]. 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.

The *Sorted Neighbourhood* (SN) method [5] 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 $O(wn)$ [3].

2.1 Bigram Indexing

The *Bigram Indexing* (BI) method as implemented in the *Febrl* [1] 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.

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 [2]. 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 $O(\frac{n^2}{b})$ [3]. However, as discussed above the number of blocks b will much larger in bigram indexing.

2.2 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 [2, 7].

The number of record pair comparisons resulting from canopy clustering is $O(\frac{fn^2}{c})$ [7] 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.

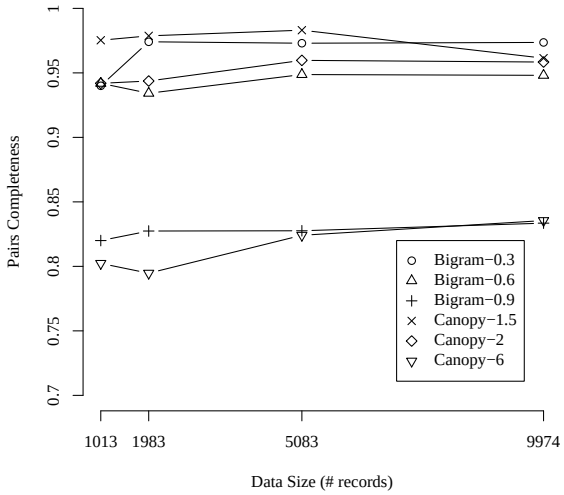

Figure 1: Pairs completeness for bigram indexing and canopy clustering methods

However, if f is too small, then the method will not be able to detect typographical errors.

3. EXPERIMENTAL RESULTS

Written in Python and published as open source software, *Febrl* is a useful platform for performing empirical comparisons for record linkage. *Febrl* (version 0.2) [1] implements standard blocking and bigram indexing. We implemented sorted neighbourhood and canopy clustering with TFIDF. We used *DBGen* [4] to artificially generate mailing list data containing surnames, given names and other attributes.

We evaluate three performance metrics for the blocking methods [3]. The three evaluation metrics used are *reduction ratio* (RR), *pairs completeness* (PC) and *F score*. Their definitions can be found in [3] and the longer version of this paper [1].

We calibrated our experimental methods by reproducing the standard blocking and sorted neighbourhood results produced by *TAILOR* [3] under a similar experimental framework. For bigram indexing and canopy clustering with TFIDF Figure 1 shows the pairs completeness results, Figure 2 the reduction ratio results and Figure 3 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 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 stage of record linkage.

Bigram indexing performs best with a threshold parameter of $t = 0.3$. This parameter results in blocking attributes made by concatenating 3 bigrams.

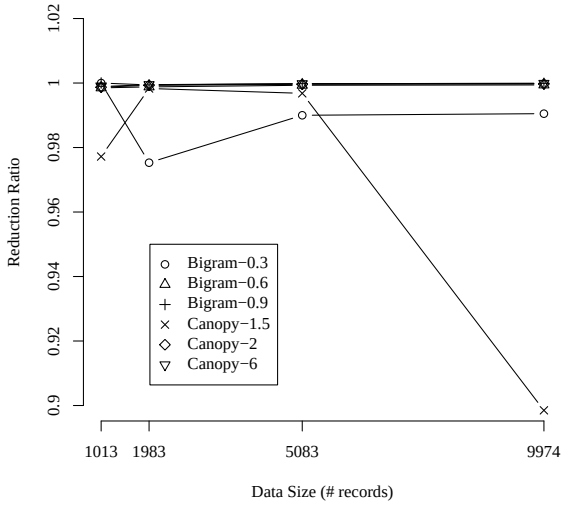

Figure 2: Reduction ratio for bigram indexing and canopy clustering methods

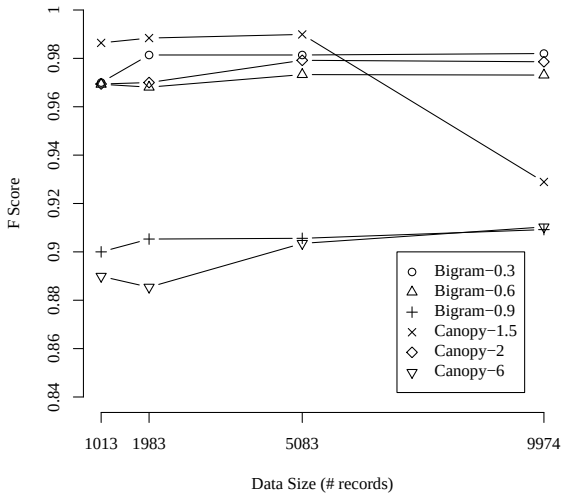

Figure 3: F score for bigram indexing and canopy clustering methods

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	45	571

Table 1: Canopy cluster average and maximum sizes for different data set size n and loose threshold = 1.5

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 to around 0.90 for the data set with $n = 9974$ records. A smaller reduction ratio for larger data sets causes very slow linkage performance. 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 1. For a fixed threshold, the cluster size is growing, leading to more record pairs being generated. The jump from cluster size average 3.9 for the data set with $n = 5083$ records to 45 for the data set with $n = 9974$ records is significant.

4. CONCLUSIONS AND FUTURE WORK

This paper describes work in progress. We wish to consider other promising fast indexing methods. The main contribution of this paper has been the direct evaluation of reduction ratio and pairs completeness for some diverse blocking methods on artificial data sets from a widely used database generator.

5. ACKNOWLEDGEMENTS

The authors would like to thank the anonymous reviewers for their valuable suggestions and comments.

6. REFERENCES

[1] P. Christen and T. Churches. *Febrl: Freely extensible biomedical record linkage Manual, V0.2, http://datamining.anu.edu.au/linkage.html*, April 2003.

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

[3] M. Elfeky, V. Verykios, and A. Elmagarmid. TAILOR: A Record Linkage Toolbox. In *ICDE*, 2002.

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

[5] M. Hernandez and S. Stolfo. Real-world data is dirty: data cleansing and the merge/purge problem. *J. DMKD*, 1(2), 1998.

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

[7] A. McCallum, K. Nigam, and L. Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *ACM SIGKDD*, pages 169–178, 2000.