

Adaptive Filtering for Efficient Record Linkage

Lifang Gu
CSIRO ICT Centre
GPO Box 664
Canberra ACT 2601, Australia
Lifang.Gu@csiro.au

Rohan Baxter
CSIRO ICT Centre
GPO Box 664
Canberra ACT 2601, Australia
Rohan.Baxter@csiro.au

Abstract

The process of identifying record pairs that represent the same real-world entity in multiple databases, commonly known as record linkage, is one of the important initial steps in many data mining applications. Record linkage of millions of records is a computationally expensive task. Various blocking methods have been used in record linkage systems to reduce the number of record pairs for comparison.

A good blocking key is critical to the success of a blocking method and will ideally result in lot of small blocks. However, in practice, there are almost always large blocks no matter how good the blocking key is. For example, when blocking on *surname* for an Anglo-Celtic population, ‘Smith’ and ‘Taylor’ are populous and result in very large block sizes. The efficiency of a blocking method is hindered by these large blocks since the resulting number of record pairs is dominated by the sizes of these large blocks. In this paper, we present an adaptive filtering algorithm to post-process large blocks to enhance the blocking efficiency.

Experimental results show that our filtering algorithm can reduce the number of record pairs produced by the standard blocking method by 88% on a small real-world data set. The algorithm also reduces the number of record pairs generated by a 3-pass standard blocking method by 50% on several synthetic test data sets, with minimal loss of accuracy.

Keywords: record linkage, filtering, data integration, data reduction, indexing, pre-processing.

1 Introduction

Record linkage is the task of identifying records corresponding to the same entity from one or more data sources. Entities of interest include individuals, companies, geographic regions, families, or households. Record linkage has applications in customer systems for marketing, relationship management, fraud detection,

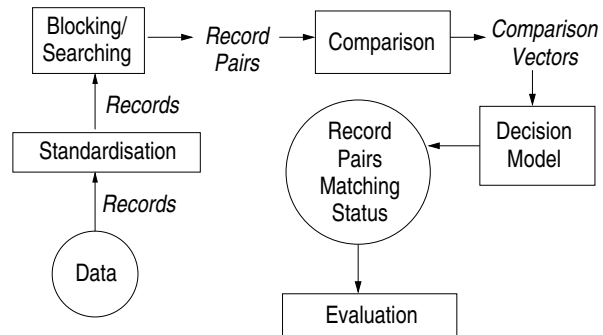


Figure 1: Information flow diagram of a record linkage system.

law enforcement and government administration.

In many data mining projects it is necessary to collate information about an entity from more than one data source. If a unique identifier or key of the entity of interest is available in all of the data sources to be linked, conventional ‘join’ operations can be used for record linkage, which assumes error-free identifying fields and links records that exactly match on these identifying fields. However, real-world data is ‘dirty’ and sources of variation in identifying fields include lack of a uniform representation or format, misspellings, abbreviations, and typographical errors.

Record linkage can be considered as part of the *data cleaning* process, which is a crucial first step in the knowledge discovery process [4]. Fellegi and Sunter [5] were the first to introduce a formal mathematical foundation for record linkage, following a number of experimental papers that were published in the medical domain since 1959 [11]. Winkler [12] extended and enhanced the original model.

No matter what technique is used, a number of issues need to be addressed when linking data. Figure 1 shows the information flow diagram of a typical record linkage system as implemented in *TAILOR* [3] and *Febrl* [2]. Often, data is recorded or captured in

various formats, and data fields may be missing or contain errors. Standardisation is an essential first step in every linkage process to clean and standardise the data. Since potentially every record in one dataset has to be compared with every record in a second data set (i.e. the number of record pairs to be compared grows quadratically with the number of records to be matched), *blocking* or *searching* techniques are often used to reduce the number of comparisons. The data sets are partitioned into smaller blocks (clusters) using blocking variables. Only records within the same blocks are then compared in details using the defined comparison variables. The comparison vectors generated by such detailed comparison functions are then passed to the decision model to determine the final status (match, non-match or potential match) of the record pairs. The results of the record linkage can be assessed by the evaluation model.

In this paper, we focus on the blocking/searching component in the information flow diagram. The performance bottleneck in a record linkage system is usually the detailed comparison of record pairs. A good blocking method can greatly reduce the number of record pair comparisons and achieve significant performance speed-ups. The main contribution of this paper is the development of a fast adaptive filtering algorithm, which can be combined with any blocking method as a post-processing step to further reduce the number of record pair comparisons with minimal accuracy loss. A key innovation is that the filtering is applied only to blocks with a significant number of records.

The rest of the paper is organised as follows. Section 2 reviews several blocking methods and identifies their limitations. Section 3 then presents our adaptive filtering algorithm for addressing these limitations. Experimental results are described in Section 4 and conclusions are made in Section 5.

2 Blocking Methods

In order to reduce the search space (i.e. the number of record pairs to be compared), many blocking methods have been recently proposed in the literature and some of them are reviewed in this section. The main idea in blocking is to group similar records together, called blocks or clusters, using available information from the records.

2.1 Standard Blocking The *Standard Blocking* (SB) method partitions records into mutually exclusive blocks where records in each block share an identical *blocking key* value [8]. A blocking key is a variable composed from the record attributes in a data set. An example of a blocking key is the first four characters of the *surname* attribute. A blocking key can also be com-

posed of more than one attribute, for example, a *post-code* attribute combined with a *given name* attribute.

There is a cost-benefit trade-off to be considered in choosing the blocking keys [9]. A further consideration in the selection 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 as the blocking key. Multiple blocking keys are also used as a way to mitigate the effects of errors in blocking keys. Another strategy of reducing the effects of spelling and transcription errors [3] in record attributes used as blocking keys (typically name and address attributes) is to use phonetic encodings, such as Soundex or NYSIIS, of these attributes.

The number of generated record pairs depends on the number of blocks and their sizes. Assuming two data sets with n records each are to be linked, the blocking key results in b blocks, and each block contains n/b records, the resulting number of record pairs is $O(\frac{n^2}{b})$ [3]. This is of course the ideal case, hardly ever achievable with real data. In general, the number of record pairs is $O(\sum_{i=1}^b n_i^2)$ where n_i is the number of records in block i . Thus, the number of record pairs to be compared can be dominated by large blocks.

2.2 Sorted Neighbourhood The *Sorted Neighbourhood* (SN) method [7] 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 number of record pair comparisons (assuming two data sets with n records each) of the SN method is $O(wn)$ [3].

One problem with the SN method arises if the number of records in a block is larger than the window size. For example, having a sorting key *surname*, hundreds of records can have a value of 'smith', and if the window size is small not all records with a surname value 'smith' will be compared. However, this problem can be avoided by using the transitivity property.

2.3 Bigram Indexing The *Bigram Indexing* (BI) method [2] allows for *fuzzy blocking*. The basic idea is to convert the blocking key values into a list of bigrams (sub-strings containing two characters) and build sub-lists of all possible permutations using a threshold. 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.

Like standard blocking, the number of record pair comparisons with two data sets with n records each and

b blocks all containing the same number of records is $O(\frac{n^2}{b})$ [3]. However, the number of blocks b will be much larger in bigram indexing.

2.4 Canopy Clustering *Canopy Clustering* is a two-step efficient clustering method for high-dimensional data sets [10]. The key idea involves using a cheap, approximate distance measure to efficiently divide the data into overlapping subsets (called canopies). Clustering is then performed by measuring exact distances only between points that occur in a common canopy. In the case of blocking, only the first step is performed to form canopies using *TF/IDF* (Term Frequency/Inverse Document Frequency) as the approximate distance measure. Only records in the same canopy will be compared in detail. The algorithm details can be found in [10].

The number of record pair comparisons resulting from canopy clustering is $O(\frac{fn^2}{c})$ [10], where c is the number of canopies and f is the average number of canopies a record belongs to. The parameters should be set so that f is small and c is large, in order to reduce the amount of computation.

3 Adaptive Filtering

The previous section shows that the number of record pairs generated by any blocking method depends on the number of blocks it generates (linearly) and their sizes (quadratically). Very large blocks have therefore dominant effects on the efficiency of blocking methods. It is generally difficult to avoid large blocks no matter what blocking methods/keys are chosen. For example, block 'blac' will be much larger than block 'szep' if the first 4 characters of the *surname* attribute are used as the blocking key, since frequent surnames such as 'black', 'blackburn' and 'blackman' will be included in block 'blac'.

To improve the blocking efficiency, we propose an adaptive filtering algorithm as a post-processing step of the blocking process. The filtering is adaptive in the sense that the number of blocks to be filtered is dependent on the results from a blocking method. Filtering is only applied to larger blocks.

Specifically, it is observed that not all record pairs within the two similar blocks are potential matches, as the blocking key used might be incomplete or contain errors, or the blocking key does not have enough discriminating power. As a result, complete variables of a blocking key or other information (mainly name and address) in records can be used to perform fast approximate comparisons to filter out unlinkable record pairs before detailed comparisons are performed. The key objective of filtering is therefore to efficiently eliminate

those unlinkable record pairs using the information contained in a chosen *filtering variable*. A filtering variable should be chosen to be different from the blocking key in the following sense. It should contain independent information differentiated from that of the blocking key for the filtering process to be able to quickly remove unlinkable record pairs. For example, a filtering variable can be an entire surname instead of the first 4 characters of the surname, or it can be a string composed of the complete given name and surname.

Next, we need an efficient method for deciding whether a record pair is unlinkable. We use the fast approximate comparison method proposed by Gravano et al. [6] for our adaptive filtering. The method was initially devised for performing efficient approximate string joins by exploiting q -gram properties. One of the measures used for string comparison is the *edit distance*. Gravano et al. [6] relate simple q -gram properties to lower bounds on string edit distances. Given a filtering variable (a string), we convert it into a list of bigrams. In the following two subsections, we describe some key properties of bigrams and show how they can be used to perform fast filtering without calculating the edit distance.

3.1 Length Filtering It is observed that string length provides useful information to quickly eliminate those very dissimilar strings. Dissimilar strings are defined to be those that are not within the desired edit distance. Specifically, it can be proved [6] that if two strings s_1 and s_2 are within an edit distance k , their lengths cannot differ by more than k .

For our particular case, if the string length difference of the filtering variable values in two records is larger than a predefined value k , this pair of records is declared as unlinkable and eliminated from the record pair list.

3.2 Count Filtering The above length filtering is not very effective for strings with a uniform length distribution. There are other features that can be used to perform the filtering. One such feature is based on the observation that similar strings share a large number of common bigrams. The basic idea of *count filtering* is therefore to make use of the information conveyed in the sets B_{s_1} and B_{s_2} of bigrams of the strings s_1 and s_2 , ignoring the positional information of the bigrams, in determining whether s_1 and s_2 are within the edit distance k .

It can be shown [6] that the following proposition holds: If s_1 and s_2 are within an edit distance k , then the cardinality of $B_{s_1} \cap B_{s_2}$, ignoring positional information, must be at least $C_{min} = \max(|s_1|, |s_2|) - 2k + 1$.

Name	number of original records	number of duplicates	maximal duplicates per record	distribution
dataset1	500	500	1	uniform
dataset2	4,000	1,000	5	poisson
dataset3	2,000	3,000	5	zipf
dataset4	5,000	5,000	1	uniform

Table 1: Properties of the four synthetic test data sets produced by the Febrl database generator.

Specifically, we count the number of common bigrams in two string values of the filtering variable and compare it with the threshold C_{min} for a particular k . If the number of common bigrams in two records is smaller than the threshold, this pair of records is considered unlinkable and removed from the record pair list.

4 Experimental Results

We have implemented the above filtering algorithm within the *Febrl* system [2], where several blocking methods have been implemented.

4.1 Data A data generator has been implemented in *Febrl* and delivered as part of its distribution [2]. Four data sets are generated using this 'dbgen' and they all have attributes such as given name, surname, date of birth, address, and postcode. The properties of these four data sets are listed in Table 1.

A small real-world data set (bibliographic reference data) is also obtained from the public domain (the Cora web site [10]) and tested by our filtering algorithm. The Cora data set has 1,295 records with 10 attributes such as label, author, title, year, venue, pages, etc. The *label* attribute is composed of the first 4 characters of the surname of the first author and the year in which the paper is published.

4.2 Experimental Protocol We have applied a 3-pass standard blocking on the synthetic data sets using three composite blocking keys. The complete surname is chosen as our filtering variable.

For the Cora data set, we also apply the standard blocking method but use the first four characters of NYSIIS codes of the attribute *label* as the blocking key. The *title* attribute is chosen as the filtering variable.

As the final number of record pair comparisons is dominated by the sizes of large blocks, filtering is applied only to blocks with a size larger than a threshold, T_b . As this threshold increases, less blocks will be filtered. In the extreme case, filtering is turned off when the threshold exceeds the size of the largest block in a data set. In the experiments, we vary this threshold from 3 to an upper bound, which turns off

T_B	#filtered blocks	SF	SF_M	FRR	RR	PC
40	0	25,820	4,634	0.000	0.999	0.927
25	3	24,904	4,633	0.036	0.999	0.927
20	11	23,365	4,626	0.095	0.999	0.925
15	29	21,251	4,617	0.177	0.999	0.923
10	84	18,193	4,582	0.295	0.999	0.916
5	404	12,813	4,452	0.504	0.999	0.890
3	1,110	9,453	4,236	0.634	0.999	0.847

Table 2: Adaptive filtering results for data set 4.

the filtering step.

Another parameter is the desired edit distance k . The smaller the parameter k , the more effective the filtering. At the same time, more true matches could be missed. However, the variation of the effectiveness is not very large when k changes from 1 to 3 [6]. Since we aim for no false dismissals, we set the parameter k to 3 in our experiments.

4.3 Performance Metrics To evaluate the performance of our filtering algorithm, metrics for evaluating the performance of blocking methods [1] are used with a minor modification. Specifically, we replace the number of record pairs generated by a blocking method, S , with the total number of record pairs generated by a blocking method and the filtering process, SF . There are three such metrics, namely *reduction ratio* (RR), *pairs completeness* (PC) and the *Fscore*. Two of them are listed here: $RR = 1 - \frac{SF}{N}$ where N is the number of all possible record pairs in the entire data sets, and $PC = \frac{SF_M}{N_M}$ where subscript M refers to matched pairs.

The number, N , is usually very large for real-world data sets and hence the metric RR becomes inappropriate as it will approach one no matter how the value SF changes. As a result, we propose to use another metric, the *filtering reduction ratio* (FRR), to measure the relative reduction in the number of record pairs to be compared purely due to filtering. FRR is defined as $FRR = 1 - \frac{SF}{S}$ where S is the number of record pairs produced by the blocking method only. If there is no filtering, FRR will be zero.

4.4 Results and Discussion Table 2 shows the results of our adaptive filtering applied to data set 4. The 3-pass blocking is very effective and produces very small blocks. The reduction ratio (RR) values all reach up to 0.999. Despite of this, our filtering algorithm still manages to reduce the number of record pairs by up to 63% (from 25,820 to 9,453). This efficiency improvement has been achieved with very minimal accuracy loss. The PC values have only decreased by 5%. Results on other 3 data sets are consistent with those shown and available upon request.

T_B	#filtered blocks	SF_{M1}	SF_{M2}	PC_1	PC_2
40	0	4,634	4,634	0.927	0.927
25	3	4,633	4,634	0.927	0.927
20	11	4,626	4,629	0.925	0.926
15	29	4,617	4,627	0.923	0.925
10	84	4,582	4,610	0.916	0.922
5	404	4,452	4,547	0.890	0.909
3	1,110	4,236	4,470	0.847	0.894

Table 3: Comparison of the PC values under two different filtering variables for data set 4.

T_B	#filtered blocks	SF	FRR
400	0	107,341	0.000
300	1	48,207	0.551
100	2	21,068	0.804
50	5	14,396	0.866
20	9	13,208	0.877
10	10	13,139	0.878
5	11	13,114	0.878
3	13	13,106	0.878

Table 4: Adaptive filtering results for the Cora data set.

It has been found that the slight accuracy loss is mostly caused by the field swapping problem, e.g., given name and surname swapping, and missing values in the filtering variable. Table 3 compares the values of the *pairs completeness* metric under two different filtering variables for data set 4 (subscript 1 refers to surname only and subscript 2 refers to the concatenation of the given name and surname). It can be seen that the *pairs completeness* metric value is higher (0.894 versus 0.847) when the concatenated names are used as the filtering variable.

Table 4 shows the results of our filtering algorithm applied to the Cora data set. Filtering on the two big blocks (each containing several hundred records) brings the number of record pairs down by more than 80% (from 107,341 to 21,068). This shows the power of filtering for improving the efficiency of blocking in record linkage, when the assumptions made on blocking keys are not upheld. Since true matches are not known for the Cora data set, accuracy cannot be evaluated.

It is also observed that filtering effectiveness is dependent on data characteristics (such as string length distribution and error distribution). It is therefore expected that the degree of efficiency improvement of the filtering algorithm will vary, depending on the data to be linked and the blocking method/key chosen.

5 Conclusions

In this paper, we have proposed the adaptive filtering algorithm as a novel post-processing step of blocking methods to achieve more efficient record linkage. Experimental results have shown that our adaptive filtering algorithm can reduce the number of record pairs

generated by a standard blocking method by 88% for the Cora data set. For the four synthetic data sets with up to 10,000 records, a 50% reduction in the number of record pairs has been achieved even when the block sizes are not very large. Since real-world data is usually large and dirty, the blocking performance can be expected to be greatly enhanced by our filtering algorithm.

Further experiments should be performed on larger real-world data sets and also under different parameter settings to confirm the robustness of our filtering algorithm. We observed a significant reduction in overall run-time in initial tests (run-time performance results were not presented here due to limited space), but further investigations are required.

References

- [1] R. Baxter, P. Christen, and T. Churches. A Comparison of fast blocking methods for record linkage. In *Proc. of ACM SIGKDD'03 Workshop on Data Cleaning, Record Linkage, and Object Consolidation*, pages 25–27, Washington, DC, USA, August 2003.
- [2] P. Christen and T. Churches. *Febrl: Freely extensible biomedical record linkage Manual*, release 0.2 edition, April 2003. <https://sourceforge.net/projects/febrl/>.
- [3] M. Elfeky, V. Verykios, and A. Elmagarmid. TAILOR: A Record Linkage Toolbox. In *Proc. of the 18th Int. Conf. on Data Engineering*. IEEE, 2002.
- [4] U. Fayyad, G. Piatesky-Shapiro, and P. Smith. From data mining to knowledge discovery in databases (a survey). *AI Magazine*, 17(3):37–54, 1996.
- [5] L. Fellegi and A. Sunter. A theory for record linkage. *J. of the Amer. Stat. Soc.*, 64:1183–1210, 1969.
- [6] L. Gravano, P. Ipeirotis, H. Jagadish, et al. Approximate string joins in a database (almost) for free. In *Proc. 27th Int. Conf. on VLDB*, pages 491–500, 2001.
- [7] 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.
- [8] M. Jaro. Advances in record linkage methodology as applied to matching the 1985 Census of Tampa. *J. of the Amer. Stat. Soc.*, 84(406):414–420, 1989.
- [9] R. Kelley. Advances in record linkage methodology: a method for determining the best blocking strategy. In *Proc. of the Workshop on Exact Matching Methodologies in Arlington - Record Linkage Techniques*, 1985.
- [10] 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.
- [11] H. Newcombe, J. Kennedy, et al. Automatic linkage of vital records. *Science*, 130:954–959, 1959.
- [12] W. Winkler. The state of record linkage and current research problems. Technical Report RR/1999/04, Statistical Research Report Series, US Bureau of the Census, Washington DC, 1999.