Neighbourhood Blocking for Record Linkage

Daniel Elias[1] and Josiah Poon[2]

[1] Commonwealth Bank, Sydney, Australia daniel.elias@cba.com.au
[2] University of Sydney, Australia josiah.poon@sydney.edu.au

Abstract. This paper describes Neighbourhood Blocking – a novel method for the indexing step in the record linkage process. Record Linkage is the task of identifying database records referring to the same entity without the aid of definitive key fields. It has applications in data integration, fraud detection and other areas. If all pairs of records are compared indiscriminately, the size of this task is quadratic in dataset size. Various indexing methods are typically used to reduce the number of record pairs subjected to detailed comparison. Neighbourhood Blocking generalizes two existing indexing methods – Standard Blocking and Sorted Neighbourhood Indexing. It also allows wildcard matching of missing values and a limited number of blocking field mismatches. Numerical experiments and tests on benchmark datasets are reported in which Neighbourhood Blocking is compared to the algorithms that it generalizes. In our experiments, Neighbourhood Blocking frequently produces superior index quality at the expense of modestly increased runtime.

Keywords: record linkage, sorted neighbourhood indexing, standard blocking

1 Introduction

1.1 Record linkage

Applications such as data integration, deduplication and fraud detection require the identification of distinct records referring to the same entity without the aid of unambiguous identifying fields. For example, many census datasets lack a field suitable for unambiguously identifying individual people. In such cases, the task of resolving object identity must rely on a combination of other fields (eg: name, date of birth, address), each serving as a partial indicator of object identity. For example, several people might live at the same address, and the same person may have different addresses at different times.

Since [2] investigated record linkage in 1940s, the subject has been pursued separately in several disciplines. Consequently, many names are now used to refer to it. These include: conflation, data linkage, deduplication, disambiguation, entity resolution, record linkage and several others. In this paper, it will be referred to generally as "record linkage" and as "deduplicaion" when the records to be matched are in the same table.

[1] describes the record linkage process in terms of distinct steps:

1. *Preprocessing* – Feature extraction from individual records
2. *Indexing* – Identification of candidate record pairs
3. *Comparison* – Feature extraction from record pairs
4. *Classification* – Final identification of record pairs representing matches
5. *Evaluation* – Assessment and checking of the result

This paper focuses on the *Indexing* step. The simplest form of indexing is a Full Index – all possible record pairs. However, since the number of pairs is quadratic in the number of records, there is often value in using other methods which are more selective.

Indexing involves competing objectives: It must be sufficiently scalable to handle the dataset. Given this constraint, there is a balance between size reduction and recall.

1.2 Terms and Abbreviations

Key A field used for grouping records into Blocks (blocking key) or for sorting them (sort key)
BKV Blocking Key Value - one of the values contained in a blocking key
Block A group of all records with a particular combination of BKVs
NB Neighbourhood Blocking
SB Standard Blocking
SNI Sorted Neighbourhood Indexing

1.3 Motivation

Standard Blocking (SB) and Sorted Neighbourhood Indexing (SNI) are two simple but commonly used indexing techniques. To illustrate their features, consider the set of points illustrated in Figure 1.

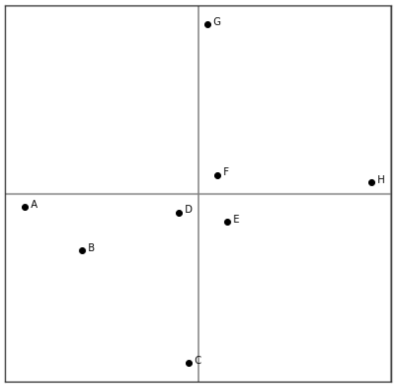

Fig. 1. Bivariate blocks

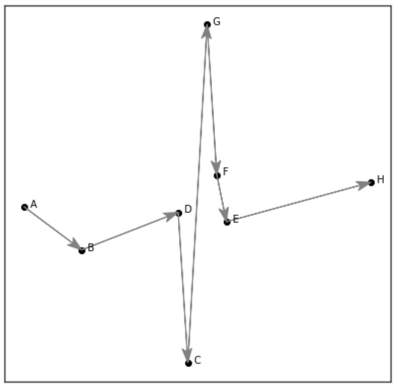

Fig. 2. Sorting by horizontal variable

Row	Given name	Surname	Address	Birth Date	SSID
1	Catherine	Bourke	42 Black Stump Cres	15-Mar-1958	3984257
2	Cathy	Smythe	42 Black Stump Cres	15-Mar-1958	398425
3	Timothy	Bourke	42 Black Stump Cres	06-Dec-1959	3939872
4	Timothy	Bourk	110 Beachfront Drive	06-Dec-1995	3939872

Table 1. Example Records for Deduplication

SB is like a standard database join based on *equality* matches in one or more fields. Discretization of the points in Figure 1 effectively divides the space into regions, each representing a "block" of points that are grouped together. A shortcoming of this approach is that some pairs of nearby points (eg: D-E) are excluded from the index because they straddle block boundaries.

This problem can be addressed by introducing matching by *proximity* rather than equality. SNI does this in a restricted way by allowing proximity matching based on a *single ordering* of the records. This is equivalent to a one-dimensional route that visits all the points such as the one illustrated in Figure 2. A problem with using a one-dimensional route to traverse a two (or many) dimensional space is that there will typically be distant pairs of points (like C-G in Figure 2) that are closer in this route than nearby pairs like D-F.

Another problem with both SB and SNI is that neither provide meaningful treatment of missing values. Wildcard matching would be desirable.

A third problem is illustrated in Table 1 where the only two individuals referred to are Catherine and Timothy Bourke. In situations like this, a record pair selection rule like "any three of these five fields approximately agree" would be desirable.

As outlined in Section 3 Neighbourhood Blocking addresses these issues through the features illustrated in Figure 6 without requiring comparison of all possible pairs of records.

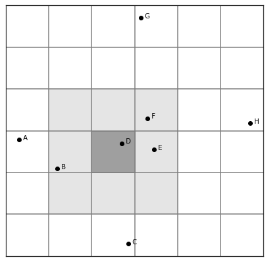

Fig. 3. Simultaneous proximity matching

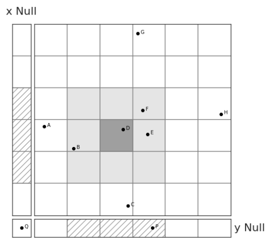

Fig. 4. Wildcard matching of nulls

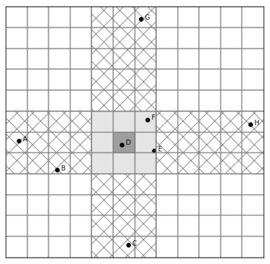

Fig. 5. Allowances for non-matches

Fig. 6. Features of Neighbourhood Blocking

1.4 This paper's contributions

This paper's contributions are:

– Proposal and description of Neighbourhood Blocking
– Theorems relating to its properties
– Tests comparing Neighbourhood Blocking with SB and SNI

2 Related Work

The purpose of indexing in record linkage is to produce pairs of records for further consideration in the Comparison and Classification steps. The need for this was identified in [3] where the indexing technique now known as Standard Blocking is also described. It is still a field of active research.

The simplest way of selecting record pairs for further consideration is simply to select all possible pairs. This can be manageable in the case of smaller datasets. For example, [8] uses it in a study that focuses on the Comparison phase of the record linkage process.

Standard Blocking (SB), described by [3], produces all pairs of records which have exact matches in all fields designated as "blocking keys".

Sorted Neighbourhood Indexing (SNI), described in [4] selects all pairs of records that are within a fixed "rank distance" of one another in a (single) sorted list. SB can be thought of as a special case of SNI where the maximum rank distance is zero. [9] describes an efficient method for implementing an online version of SNI using a tree-based approach to maintain the sort order as new records are added.

There are also several methods which map individual blocking key values to one or more alternative versions.

Many-to-one mappings (such as Soundex which maps strings to sound codes) are a way of coarsening blocking (thereby increasing the number of record pairs included).

Many-to-many mappings (such as string modifications using q-grams or suffix arrays) result in blocking where each record is effectively a member of multiple blocks, thereby reducing the effects of block boundaries. Some such methods are described in chapter four of [1]. Many-to-many mappings can often result in very large indexes. [5] describes an approach to address this by "pruning" the size of the mapping.

Ther are also other methods which don't fit easily into these categories. Canopy Clustering which makes some limited use of non-ordinal record comparison. Progressive Blocking ([7]) integrates *indexing* with *comparison.*

3 Neighbourhood Blocking

3.1 Intuition

Neighbourhood Blocking (NB) is a generalization of SB and SNI which adds the following features:

– matching by *simultaneous* proximity in multiple record orderings
– wildcard matching of missing values, and
– allowance for complete mismatches in a limited number of blocking keys

Each of these preclude the use of a sorting-based algorithm. Recursion will therefore be used instead.

3.2 Algorithm Description

The steps in the recursive implementation are:

Neighbourhood Blocking can be implemented using the following steps:

Normalize BKVs Replace all non-null BKVs with integers representing their rank.

Atomic blocking Produce a (single) master table of block BKV combinations by taking all distinct combinations of BKV ranks in the table(s) being indexed. Assign a distinct block ID to each row in this table.

Produce a linkage index of candidate block pairs NB is used (recursively) to achieve this as follows:

– If the blocking is maximally coarse (ie: each blocking key has only one non-null value), produce a Full Index.
– Otherwise: produce a NB index using the same parameters for blocking keys, wildcard limit and mismatch limit on a *coarsened version of the block table* where each non-null BKV x is replaced with $\left\lfloor \frac{x}{a} \right\rfloor$ where $a > 1$. Each rank distance limit r is replaced by $\left\lceil \frac{r}{a} \right\rceil$.

The number of recursive steps is logarithmic in the maximum number of distinct values in any blocking key.

Identify pairs of matching blocks Compare the blocks in each of the block pairs identified in the previous step and determine which ones satisfy the record matching conditions. Put their block IDs into a link table (ie: each row is a pair of block IDs).

Translate block pairs to record pairs Use database-style joins (via the link table found in the previous step) to determine the pairs of row IDs corresponding to matching row pairs.

For deduplication, filter the record pairs In the case of deduplication of a single table (as opposed to linkage of two tables), filter the list of record id pairs to only include unique pairs (regardless of order).

3.3 Comparison to other methods

SB and SNI are both special cases of NB where no wildcard matching or match condition violations are allowed.

Unlike Progressive Blocking (outlined in [7]), NB is separable from the Comparison and Classification steps. It uses multiple sorting orders to determine block proximity and allows for wildcard matching and field mismatches.

A comparison of key features of NB and some of its counterparts is summarized in Table 2.

Feature	SB	SNI	Progressive	NB
multiple block keys	✓	✓	✓	✓
multiple orderings	N/A			✓
block combination / overlap		✓	✓	✓
separable from Comparison	✓	✓		✓
nulls as wildcards				✓
limited non-matches				✓

Table 2. Comparison of Index Algorithm Features

3.4 Properties

Proximity matching allows the inclusion of record pairs which straddle block boundaries. By Theorem 1, where there is a notion of position within blocks (making the notion of "close pairs straddling block boundaries" meaningful), NB includes all record pairs closer than a specific "inclusion distance", regardless of the specific locations of block boundaries.

Theorem 1 (Inclusion distance). *A NB index using blocking keys that are discretized versions of continuous variables will include all pairs of records whose non-discretized Euclidean distance is less than the product of (a) the length of the unit of discretization, and (b) the lowest of its rank distance limits. This is true regardless of the locations of block boundaries.*

Clearly, a NB Index is a superset of a SB index which uses the same keys. However, by Theorem 2, it is also a superset of a SB Index where the granularity of any or all of the blocking keys is coarsened by combining groups of $1 + r_j$ adjacent values where r_j is the j$^{\text{th}}$ blocking key's rank distance limit.

Theorem 2 (Superset of SB). *If:*

1. *X_N is a NB index with keys $k_1, k_2 \cdots k_n$ and corresponding rank distance limits of $r_1, r_2 \cdots r_n$*
2. *Each k'_j ($j \in \{1 \cdots n\}$) is a $(1 + r_j)$:1 mapping of k_j such that each distinct value of k'_j corresponds to $(1 + r_j)$ consecutive sorted values of k_j*

3. X_S is a SB Index with keys $k'_1, k'_2 \cdots k'_n$

Then: $X_N \supseteq X_S$

Theorem 3 relates to an idealized database where:

– all keys have sufficiently many distinct values for edge effects to be negligible,
– records are uniformly distributed throughout the key space, and
– no keys have any null values

Under these idealized conditions, it is shown that the size of a NB index is larger than a SB Index using the same keys by a factor of $\prod_j(1+2r_j)$, where r_j is the rank distance limit for the j^th^ blocking key. By Corollary 1, for the same idealized database and where all blocking keys are also sorting keys, a NB Index has the same reduction ratio as a SB Index with each key coarsened by a factor of $1+2r_j$.

Theorem 3 (Index size relative to SB). *In datasets where each block contains the same number of records and each sorting key has the same number of distinct values, the index sizes for SB and NB are related by:*

$$\lim_{d,v\to\infty} \frac{\mid X_N \mid}{\mid X_S \mid} = \prod_j (1+2r_j) \tag{1}$$

where:

X_S is the set of record pairs from SB
X_N is the set of record pairs from NB where no field mismatches or wildcards are allowed.
r_j is the rank distance limit used in NB for the j^{th} blocking key
d is the number of records per block,
v is the number of distinct values of each blocking key

Corollary 1. *In the limiting case described in Theorem 3, NB produces the same index size as SB with each blocking key coarsened by a factor of $(1+2r_j)$ where r_j is the j^{th} blocking key's rank distance limit.*

4 Application to Benchmark Datasets

Comparisons of index quality (defined in Section 4) between NB, SB and SNI were made on the following benchmark datasets:

– FEBRL: 4 datasets about people
– DBG-Leipzig: Amazon-Google Products, ABT-Buy (products); DBLP-ACM, DBLP-Scholar (publications)

Some calculated fields were added to the datasets and a number of indexes were calculated for each dataset using each of the indexing methods. For each index produced, a point representing its recall and reduction ratio was computed. These were grouped by indexing method and the "frontier points" among them were identified as those that:

- are on the convex hull surrounding all points for the indexing method, and
- do not have *both* lower recall and lower reduction ratio than any other point

Selected index quality frontiers are shown in Figure 10.

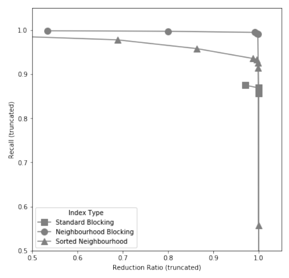

Fig. 7. FEBRL3

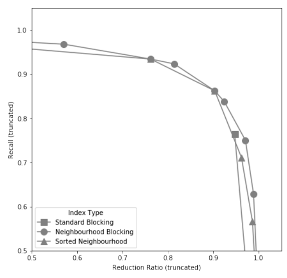

Fig. 8. Abt-Buy

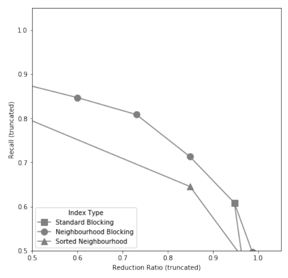

Fig. 9. Amazon-GoogleProducts

Fig. 10. Index quality frontiers by dataset

The results for all the FEBRL datasets are broadly similar, so only Febrl3 is shown (Figure 7). For all these datasets, index quality is moderate to high and the ranking of index quality frontiers is the same. That ranking is:

1. Neighbourhood Blocking with a near-perfect result
2. Sorted Neighbourhood Indexing
3. Standard Blocking

In the DBG Leipzig datasets(eg: Figure 8), all three indexing methods produce far lower index quality than those achieved in the FEBRL datasets. [6] reports using string similarity measures in the indexing step to achieve higher index quality in these datasets. In the case of the Amazon-GoogleProducts dataset (Figure 9), NB did produce a noticeable improvement over the other methods.

5 Scalability comparison

Numerical experiments were conducted to compare the scalability of NB with that of SB and SNI on simulated datasets with up to 1 million rows. It was found that runtime for NB depends most strongly on two factors:

- How sparsely populated the blocks are (ie: the ratio of the number of records in the database to the number of distinct combinations of BKVs)
- The size of the final index produced

When database sparsity is low:

Method	Filtering	Intercept	Slope	R^2
NB - No wildcards or adjacency		4.92	3.18	0.33
NB - no wildcards		8.68	3.12	0.08
NB - 1 wildcard		9.23	3.02	0.09
NB - 2 wildcards		10.27	3.15	0.07
Full		1.19	1.08	0.99
SNI		1.22	10.34	0.96
SB		1.55	3.23	0.82
NB	Non-sparse	0.26	3.20	0.99
SB	Non-sparse	0.23	3.13	0.99

Table 3. Indexing times - lines of best fit by method (seconds per million row pairs)

- runtime for all three indexing methods is approximately linear in the size of the index produced, and
- the rates of index production for Standard and NB are similar

Table 3 shows relationships between index production times (in seconds) and index size (in millions of record pairs) for the methods tested. For non-sparse datasets (ie: those with many records per block), index production rates for Standard Blocking and Neighbourhood Blocking are similar.

6 Discussion

Since SB and SNI are both special cases of Neighbourhood Blocking:

- NB can produce any index that SB or SNI can (as well as others which might have higher quality), and
- SB and SNI can be implemented using any algorithm suitable for NB (as well as others which might be less resource intensive)

Therefore, whether or not NB is preferable to SB and/or SNI depends on whether it produces an increment in index quality that justifies any increment in resource consumption.

The benchmark datasets examined in Section 4 include several cases where the improvement in index quality is material (as well as some where all three methods behave poorly).

The timings in Section 5 indicate that the difference in resource consumption is larger for small, sparse datasets than for large, dense ones.

7 Conclusion

Neighbourhood Blocking always produces indexes of at least as high quality as Standard Blocking and Sorted Neighbourhood Indexing (both being special cases of NB), and requires at least as many resources.

Compared to the other two methods, NB has several advantages which often result in higher index quality. Namely:

– simultaneous proximity matching using multiple record orderings
– wildcard matching of missing values
– tolerance for complete mismatches in a limited number of keys

Neighbourhood Blocking can be efficiently implemented through the use of recursion.

Scalability tests indicate similar index production speeds for NB and SB in sufficiently large datasets.

8 Further Work

This work could be extended by making a progressive version of Neighbourhood Blocking. This would be similar to Progressive Blocking as described by [7], except that additional match types would be allowed and block proximity would be determined by multiple sorting orders.

References

1. Christen, P.: Data Matching. Springer (2012)
2. Dunn, H.L.: Record linkage. American Journal of Public Health and the Nation's Health **39**, 1412–1416 (1946)
3. Fellegi, I.P., Sunter, A.B.: A theory for record linkage. Journal of the American Statistical Association **64**, 1183–1210 (1969)
4. Hernandez, M.A., Stolfo, S.J.: The merge/purge problem for large databases. In: Proceedings of the 1995 ACM SIGMOD international conference on Management of data. ACM, San Jose, California, USA (1995)
5. Khairul, N.B., M.N.Shahrul, A.: Efficient identity matching using static pruning q-gram indexing approach. Decision Support Systems **73**, 97–108 (2015)
6. Kopcke, H., Thor, A., Rahm, E.: Learning-based approaches for matching web data entities. IEEE Internet Computing **14**(4), 23–31 (2010). https://doi.org/10.1109/MIC.2010.58
7. Papenbrock, T., Heise, A., Naumann, F.: Progressive duplicate detection. IEEE Transactions on Knowledge and Data Engineering **27**(5), 1316–1329 (2015). https://doi.org/10.1109/TKDE.2014.2359666
8. Poon, S.K., Poon, J., Lamb, M.K., Yin, Q., Sze, D.M.Y., Wu, J.C., Mok, V.C., Ching, J.Y., Chan, K.L., Cheung, W.H., Lau, A.Y.: An ensemble approach for record matching in data linkage. Studies in health technology and informatics **227**, 113–119 (2016)
9. Ramadan, B., Peter Christen, H.L., Gayler, R.W.: Dynamic sorted neighborhood indexing for real-time entity resolution. Journal of Data and Information Quality **6** (2015)