An unsupervised blocking technique for more efficient record linkage

An Unsupervised Blocking Technique For More Efficient Record Linkage

Kevin O'Hare[a], Anna Jurek-Loughrey[a], Cassio de Campos[b]

[a]*School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, Computer Science Building, 18 Malone Road, BT9 5BN Belfast, United Kingdom*
[b]*Buys Ballotgebouw, Utrecht University, 3584 CC Utrecht, The Netherlands*

Abstract

Record linkage, referred to also as entity resolution, is the process of identifying pairs of records representing the same real-world entity (for example, a person) within a dataset or across multiple datasets. This allows for the integration of multi-source data which allows for better knowledge discovery. In order to reduce the number of record comparisons, record linkage frameworks initially perform a process commonly referred to as blocking, which involves separating records into blocks using a partition (or blocking) scheme. This restricts comparisons among records that belong to the same block during the linkage process. Existing blocking techniques often require some form of manual fine-tuning of parameter values for optimal performance. Optimal parameter values may be selected manually by a domain expert, or automatically learned using labelled data. However, in many real world situations no such labelled dataset may be available. In this paper we propose a novel unsupervised blocking technique for structured datasets that does not require labelled data or manual fine-tuning of parameters. Experimental evaluations, across a large number of datasets, demonstrate that this novel approach often achieves superior levels of proficiency to both supervised and unsupervised baseline techniques, often in less time.

Keywords: Unsupervised Blocking, Record Linkage, Entity Resolution

Email addresses: kohare08@qub.ac.uk (Kevin O'Hare), a.jurek@qub.ac.uk (Anna Jurek-Loughrey), c.decampos@uu.nl (Cassio de Campos)

1. Introduction

Record Linkage (RL) is a process of identifying and linking pairs of records representing the same real-world entity. An overview of a general RL process is depicted in Figure 1. As the number of record pairs that require comparison during linkage grows exponentially with dataset sizes, linkage often incurs great computational expense even for moderately sized datasets. For this reason, a blocking phase is implemented prior to linkage to reduce the high computational cost of exhaustively comparing all record pairs.

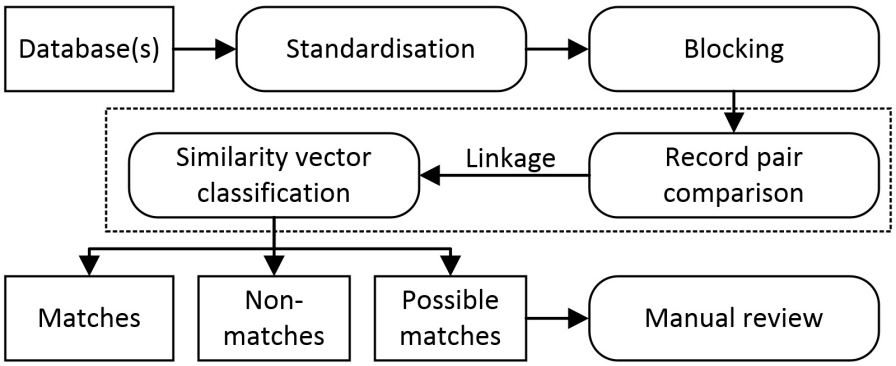

Figure 1: General overview of record linkage process.

Blocking is a process of dividing records into groups (blocks) in such a way that records within each group hold a high chance of being linked in the subsequent linkage process [1]. A good blocking approach places many matching record pairs, and few non-matching record pairs, into the generated blocks thus allowing for an efficient subsequent linkage phase. Blocking methods are commonly evaluated using labelled data (with known matching status of each record pair). Following the blocking process, linkage is performed exclusively upon the record pairs within each of the generated blocks. A large number of different linkage methods exist which classify each record pair within each block as either match or non-match based on the similarity between them [2, 3, 4, 5, 6].

Appropriate blocking rules can be determined manually [3, 7], however this requires time and domain expertise. They may also be learned automatically [8, 9, 10], however this requires the existence of a sufficient amount of labelled data which is often unavailable in real-world applications. Heuristic blocking approaches [11, 12] also exist but, like the automatic approaches, they typically require some manual fine-tuning of parameters for optimal performance. A set of parameter values that tends to work well for one dataset is not guaranteed to work well for another. In this paper we therefore make the following contributions:

(1) We propose a robust unsupervised blocking technique for structured datasets that does not require provision of labelled data.

(2) The proposed method does not require any manual effort since all of the parameters used in the method are selected automatically for each dataset.

(3) We demonstrate the proficiency of the proposed method by comparing it against state-of-the-art supervised and unsupervised blocking baselines.

The paper is structured as follows. In the proceeding section we provide some preliminary details so as to make the general overview of entity resolution and blocking more understandable. In Section 3 we provide an extensive overview of different existing supervised and unsupervised blocking techniques, detailing their advantages and disadvantages. In Section 4 we detail the proposed approach and how it overcomes the disadvantages of other existing approaches. In Section 5 we evaluate the proposed blocking approach in comparison to a number of relevant baselines. We make a number of observations regarding the results which outline the benefits of the proposed approach in comparison to the baselines. We conclude by summarising our findings and outlining where the proposed approach could ideally be further improved.

2. Preliminaries

Consider a dataset of records $R = r_1, ..., r_n$, where each record comprises values it takes for attributes from a scheme $A = a_1, ..., a_m$. Accordingly, we can

represent a record r_i as $[r_{i,1}, ..., r_{i,m}]$, where $r_{i,j}$ is the value that the i^{th} record takes for the j^{th} attribute.

2.1. Standard Blocking

During a standard blocking process a set of blocking predicates is used to determine which records should be placed in the same block. A blocking predicate is defined as follows.

Definition 2.1.1: (Blocking Predicate) A blocking predicate is an $\langle a_j, h \rangle$ combination where $a_j \in A$ is an attribute and h is an indexing function. For each $r_i \in R$, h takes $r_{i,j}$ as an input and provides a set of values, referred to as blocking keys, as an output.

For example, the blocking predicate $\langle Name, Common\ tokens \rangle$ applied to a record containing "A Brief History of Time" in the $Name$ attribute field would generate the following blocking keys: {"A", "Brief", "History", "of", "Time"}. Blocking keys determine into which block(s) records are placed, with each unique blocking key referring to a specific block. Our example record would therefore be placed in five different blocks, each associated with one of the five aforementioned blocking keys.

Due to the complexity of datasets (for example, missing values, typographical errors, acronyms, initialisations) a single blocking predicate is rarely likely to capture all matching record pairs efficiently, therefore multiple blocking predicates may be needed in the form of a standard blocking scheme.

Definition 2.1.2: (Blocking Scheme) Given a set of individual blocking predicates, $P = p_1, ..., p_{p'}$, a blocking scheme is a combination of blocking predicates, which can be disjunctive ($\langle p_i \rangle \cup \ldots \cup \langle p_j \rangle$), conjunctive ($\langle p_i \rangle \cap \ldots \cap \langle p_j \rangle$) or of disjunctive normal form ($\langle \langle p_i \rangle \cap \ldots \cap \langle p_j \rangle \rangle \cup \ldots \cup \langle \langle p_{i'} \rangle \cap \ldots \cap \langle p_{j'} \rangle \rangle$).

2.2. Blocking Evaluation Metrics

Blocking schemes are commonly evaluated with labelled data (with known matching status of each record pair) using evaluation metrics such as *reduction ratio (RR)*, *pairs quality (PQ)*, *pairs completeness (PC)* and/or a harmonic mean $F_{\mathrm{RR,PC}}$ of RR and PC [13].

Definition 2.2.1: (Reduction Ratio) For two datasets, A and B, reduction ratio is defined as:

$$RR = 1 - \frac{N}{|A| \cdot |B|}, \tag{1}$$

where $|A| \cdot |B|$ is the total possible number of record pairs between A and B, and $N \leq (|A| \cdot |B|)$ is the number of record pairs formed by a blocking method.

RR indicates how much a comparison space is reduced after a blocking phase. Blocking schemes with high RR are desirable as they form few record pairs for comparison.

Definition 2.2.2: (Pairs Quality) Pairs quality is defined as:

$$PQ = \frac{N_m}{N} \tag{2}$$

with $N_m \leq |N|$ being the number of matching record pairs contained within the reduced comparison space after blocking.

PQ indicates the proportion of blocked record pairs that are matching record pairs. Blocking schemes with high PQ are desirable as they form mostly matching record pairs and few non-matching record pairs.

Definition 2.2.3: (Pairs Completeness) Pairs completeness is defined as:

$$PC = \frac{N_m}{|M|}, \tag{3}$$

with $|M|$ being the number of matches within the entire dataset.

PC is the ratio of blocked matching record pairs out of all possible matching record pairs. Blocking schemes with high PC are desirable as they retain many matching record pairs.

One can notice that there is a trade-off between RR and PC. Placing all the records in the same block minimises RR but maximises PC, whereas placing each record in an individual block maximises RR and minimises PC. Ideally one looks for a blocking scheme that optimises both RR and PC. A commonly applied evaluation metric, which balances the trade-off between RR and PC, is the harmonic mean of RR and PC.

Definition 2.2.4: (Harmonic mean of RR and PC) For a given RR and PC, the harmonic mean is defined as:

$$F_{RR,PC} = \frac{2 \cdot RR \cdot PC}{RR + PC}. \tag{4}$$

3. Related Work

Blocking approaches can typically be categorised as either supervised (requires labelled data) or unsupervised (does not require labelled data).

Supervised blocking scheme learning approaches [1, 8, 10] commonly evaluate an initial set of blocking predicates using a set of labelled data. The best predicates, according to a predetermined selection criterion, may then continue to form longer conjunctions of predicates that also require evaluation. Blocking schemes are then formed by applying the top predicates, or conjunctions, until a pre-determined proportion of the labelled positives are detected. The computational demand of these approaches relies on the number of predicates (or conjunctions) requiring evaluation, and the amount of labelled data used to evaluate them. Using strict selection criterion reduces the number of predi-

cates and conjunctions but increases the likelihood of omitting combinations of predicates that would work well in a conjunction.

A number of unsupervised blocking techniques were proposed over the past decade. Kejriwal and Miranker [9] automatically generate labelled data from a target dataset as part of their unsupervised blocking scheme learning approach. Records are first grouped by their shared common tokens, a window of predetermined size is then slid over the contained records of each group. Record pairs within the window at any given time are then compared using the (log) Term Frequency-Inverse Document Frequency (TF-IDF) [14] measure. A pre-determined number of the most similar pairs, above an upper similarity threshold value, are labelled as positives, as are a pre-determined number of the most similar pairs, below a lower similarity threshold value, labelled as negatives. Following this, blocking predicates are ranked by their Fisher Score [15] of the labelled data. Each top predicate may be extended iteratively by others as a conjunction so that the resultant Fisher Score is higher than the average of all predicates or conjunctions in the previous iteration. Although the unsupervised approach overcomes the need for labelled data, it requires a number of parameters to be tuned, for example, window size, number of pairs to label and similarity threshold values. Additionally, the computational demand of the automatic labelling process is directly proportional to dataset size. For large and complex datasets one can therefore expect the automatic labelling process to be computationally expensive.

A different unsupervised blocking technique, which is commonly applied in the literature, is referred to as *Canopy Clustering*. Clustering is an unsupervised process by which records are divided into groups so that each record is similar to the records within the same cluster (similar according to a given distance function) and dissimilar to records from different clusters. *Canopy Clustering* [8, 16, 17, 18, 19] is a variation of clustering that can be used for unsupervised blocking, as it forms diverse overlapping clusters containing similar records. Given a set of records, a record is selected (potentially at random) as a centroid (that is, a representative record) of an initial cluster and removed from the original

set. All remaining records within a loose distance threshold of this centroid are assigned to it forming a cluster. Additionally, all records within a tight distance threshold are excluded from forming any further clusters. Following this, a new record is selected from the set and the clustering process repeats until no records are left. The efficiency and effectiveness of *Canopy Clustering* is very much reliant upon appropriate selection of parameters, such as the distance function and the loose and tight distance threshold values.

Sorted Neighbourhood [2, 20, 21, 22] is another unsupervised blocking technique. With this approach records are first sorted in lexicographical order according to their respective values from a sorting key (that is, a concatenation of sub-string values from an attribute or number of attributes). A window of fixed size is then passed over the ordered sorting key values. Only records that fit within the window at any given time are considered in the linkage phase. Domain expertise is needed to appropriately select the sorting key and window size for optimal performance. As a single sorting key is unlikely to place all matching record pairs adjacently as part of an ordered list, multiple sorting keys may be required.

Locality Sensitive Hashing (LSH) [23] is an unsupervised blocking technique that has inspired a number of other approaches, including FastMap [24], SparseMap [25], MetricMap [26] and StringMap [27]. *MinHash LSH* [28, 29, 30, 31, 32] is a more recent variation that extends upon LSH. With this approach records are first divided into sets of sub-strings of length k referred to as k-shingles. A sparse Boolean matrix is then formed in which 1's and 0's indicate if the respective record of a column contains the respective shingle of a row. A $MinHash$ value is the first non-zero row value of a column. If the rows of a boolean matrix are randomly permuted, the probability that two columns share the same $MinHash$ value is approximately equal to their Jaccard Similarity [33]. Over multiple permutations the $MinHash$ values for each column are concatenated to form what are referred to as $MinHash$ signatures. Similar records share similar, but not necessarily identical, $MinHash$ signatures. Therefore, $MinHash$ signatures are partitioned into smaller bands and the records of each

band are blocked together. *MinHash LSH* is easy to implement but relies upon appropriately selected parameter values for speed and effectiveness, for example, k, number of random permutations, the size and number of bands by which to partition $MinHash$ signatures.

Token blocking is a simple unsupervised blocking technique that treats each record as a bag of tokens, placing the record in a respective block for each of its tokens. Token blocking may be applied to any dataset(s) regardless of their structure (for example, structured, semi-structured, unstructured). Token blocking typically forms overly large sized blocks which hinder efficiency. *Block-refinement* methods may be applied that improve the efficiency of a blocking method such as token-blocking. For example, a list of highly frequent tokens (referred to as stop-words) to overlook may be manually provided in order to improve efficiency of token-blocking. Alternatively, a maximum block size limit value can be set with all blocks of greater size purged from the resultant block collection (Block-purging [34, 35]). Also, the generated blocks may be ranked by a utility function and linkage performed upon their respective record pairs until a cost/gain ratio determines further linkage to be overly expensive (Block-Pruning [34]). A common limitation of these block-refinement techniques is that optimal parameter values must be provided, which is non-trivial and requires domain expertise.

In [36] the authors combine token blocking with *Attribute Clustering* in order to improve token blocking for semi-structured datasets. Attribute clustering is a form of schema matching in which a similarity value is assigned to attribute column pairs between datasets. Highly similar attribute columns are clustered together and thought of as representing the same value. Token blocking is then performed in a way that only records that share a common token by clustered attributes are grouped together. In the experimental evaluations of [36], attribute clustering combined with token blocking was seen to obtain near equal PC to that of standard token blocking, but in substantially less comparisons in almost every case.

Meta-Blocking [31, 36, 37, 38, 39, 40, 41] describes a series of unsupervised

approaches in which the block collection of a blocking method is improved in terms of efficiency. This is typically achieved by forming a graph of nodes (records) connected by edges (their relations). The edges are then assigned with weights according to a weighting function, so that the weaker edges can be pruned (omitted) by a pruning scheme. Although Meta-Blocking may be applied analogously to any blocking method, most Meta-Blocking research aims at improving the blocking efficiency of token blocking, as it is unsupervised and schema-agnostic. In [35] five different edge-weight schemes, and four different pruning schemes, are proposed for Meta-Blocking, effectively forming 20 different Meta-Blocking frameworks. As these frameworks are later used as baselines in our experimental evaluations we define the respective edge-weight and pruning schemes accordingly. In these definitions $e_{i,j}$ indicates the edge-weight value of records i and j, and $B_{i,j}$ indicates the number of their common blocks. v_i indicates the number of edges connected to record i, and E_B indicates the total number of distinct edges of a graph formed by all records of a block collection (B). The five edge-weight schemes are defined as follows:

(i) *Common Blocks Scheme (CBS)* is simply the number of common blocks of a record pair,

$$e_{i,j}.weight = |B_{i,j}|$$

(ii) *Enhanced Common Blocks Scheme (ECBS)* considers both the number of common blocks of a record pair and the number of blocks of each record,

$$e_{i,j}.weight = |B_{i,j}|.log\frac{|B|}{|B_i|}.log\frac{|B|}{|B_j|}$$

(iii) *Aggregate Reciprocal Comparisons Scheme (ARCS)* is the sum of the reciprocals of the sizes of the common blocks of a record pair,

$$e_{i,j}.weight = \sum_{b_k \in B_{i,j}} \frac{1}{||b_k||}$$

(iv) *Jaccard Scheme (JS)* is the Jaccard Similarity of the blocks associated with each record,

$$e_{i,j}.weight = \frac{|B_{i,j}|}{|B_i|+|B_j|-|B_{i,j}|}$$

(v) *Enhanced Jaccard Scheme (EJS)* considers both the Jaccard Similarity of the blocks a record pair and the number of edges of each record,

$$e_{i,j}.weight = \frac{|B_{i,j}|}{|B_i|+|B_j|-|B_{i,j}|}.log\frac{|E_B|}{|v_i|}.log\frac{|E_B|}{|v_j|}$$

The four pruning schemes are defined as follows:

(i) *Weight Edge Pruning (WEP)*, all edges of the graph below a weight threshold value are pruned.
(ii) *Cardinality Edge Pruning (CEP)*, all but the top-K edges of the graph are pruned.
(iii) *Weight Node Pruning (WNP)*, all edges of a node below a weight threshold value are pruned.
(iv) *Cardinality Node Pruning (CNP)*, all but the top-K edges of each node are pruned.

For any of the pruning schemes the most important factor is determining the amount of edges (record pairs) to prune. By removing too many you miss many matching record pairs, by removing too few you fail to reduce the number of comparisons sufficiently. For the two weight-based pruning schemes the authors opt for the average edge-weight value of either the entire graph globally, or each node locally, as the pruning weight threshold value. For the other two edge-based pruning schemes the authors use a blocking-cardinality function to determine the top-K number of edges to retain for either the entire graph, or each individual node. In [36, 42] the weight-based pruning schemes tend to achieve comparatively high PC but poor RR, whereas the cardinality based pruning schemes tended to achieve comparatively high RR but poor PC.

In [42] the authors propose a number of improvements for the Meta-Blocking frameworks of [35]. For Entity Resolution (ER) cases in which a record from one dataset may match at most with one record of another dataset (that is, Clean-Clean ER) they suggest *Graph partitioning* in which, only nodes (records) from the smaller dataset form edges with those of the larger dataset, thus avoiding redundant (repeat) pairs with little to no negative effect on the quality of re-

sults. As such, the existing pruning techniques (WEP, CEP, WNP and CNP) of [35] are modified slightly to accommodate this new framework. They also introduce *Block filtering* in which they remove each entity (record) from the largest of its associated blocks. In their experimental evaluations they opt to retain each entity in the smallest 80% of their respective blocks. In their experimental evaluations *Graph partitioning* and *Block filtering* were both found to improve RR with little to no negative impact upon PC. In both [36] and [42] the authors recommend different Meta-Blocking framework combinations for different circumstances but no overall winner for all cases is decided.

As we will see later, our experiments show that for each of the aforementioned blocking approaches, different values of parameters may be optimal for different datasets. Therefore domain expertise is required for each new dataset. Standard blocking scheme learning approaches must evaluate individual blocking predicates or blocking schemes. Evaluation of large numbers of blocking predicates is time consuming. It can be improved by using smaller amounts of labelled data or stricter selection criteria, however this may come at a cost to evaluation quality. For the Meta-Blocking approaches different frameworks are recommended for different datasets based on their characteristics. As such, there is domain expertise needed when selecting an appropriate Meta-Blocking framework for user needs.

4. Proposed Approach

In order to address the limitations of existing blocking techniques, we propose a new blocking technique that does not require any labelled data or manual fine-tuning of parameters. We will empirically show that it achieves higher proficiency, in less time, than most baseline techniques for most datasets. There are three key steps in the proposed approach; (1) blocking predicate reduction, (2) blocking predicate weighting and (3) record blocking using selected predicates. In the proceeding subsections we detail each step in greater detail.

4.1. Blocking Predicate Reduction

Standard blocking scheme learning approaches typically omit obviously weak blocking predicates from consideration in order to reduce computation, for example, those that have insufficient PC or cover too many labelled negatives. However, this often requires a set of labelled data, manual fine-tuning of parameter values and incurs computational cost as each predicate is evaluated individually. The proposed technique efficiently omits blocking predicates with poor RR, and PC, without labelled data or manual fine-tuning of parameters.

To achieve this we first form an index of the generated values (that is, blocking keys) of each blocking predicate for each record. *Coverage* [43, 44] indicates the proportion of records in a dataset for which at least one blocking key is generated by a respective blocking predicate. The higher the Coverage of a blocking predicate the more useful it is for blocking, as fewer records will be overlooked during the blocking process. However, only non-unique blocking keys result in a record pair being formed. We therefore remove all unique blocking keys of a blocking predicate before calculating its respective Coverage value. An average Coverage value of all blocking predicates is then calculated. All blocking predicates with Coverage less than this average value are then omitted as they are indicated as not forming a significant enough number of record pairs (that is, high RR but poor PC). We then calculate an average RR value of the remaining blocking predicates and remove those blocking predicates with an RR value less than the average. By first removing those blocking predicates with high RR but poor PC, we ensure the average RR threshold value is not unfairly skewed. As we will see later on, using this approach consistently resulted in the efficient removal of many non-useful blocking predicates.

4.2. Blocking Predicate Weighting

Standard blocking scheme learning approaches typically rank blocking predicates by a supervised evaluation function so that only the best predicates are used as part of a blocking scheme. Labelled record pairs must first be sourced or

generated. Each blocking predicate is then evaluated individually using the labelled record pairs incurring further additional computational cost. We instead propose an efficient unsupervised blocking predicate weighting process (Algorithm 1) that can assign weights quickly for even the largest of our evaluation datasets.

Algorithm 1: Blocking predicate weighting process

Input: n records: $\mathbf{R} = r_1, ..., r_n$

Set of blocking predicates, $\mathbf{P} = p_1, ..., p_{p'}$

Output: Set of blocking predicate weights, $\mathbf{W} = w_1, ..., w_{p'}$

1 **foreach** *record* $r \in \boldsymbol{R}$ **do**
2 Tokenise r
3 Block r based on tokens
4 Compute average block size
5 **forall** *pairs* (r_i, r_j) *from blocks of below average size* **do**
6 Determine the blocking predicates by which (r_i, r_j) agree
7 Label the top 5% of all blocked record pairs, that agree by the highest number of blocking predicates, as positives, Pos
8 **foreach** $p \in \boldsymbol{P}$ **do**
9 assign $w_p = F_{RR_p, \frac{Pos_p}{|Pos|}}$ as the weight of blocking predicate p
10 add w_p to $\mathbf{W}$
11 Return $\mathbf{W} = w_1, ..., w_{p'}$

In order to obtain a set of labelled positives (that is, pairs of matching records) we use a modified version of the automatic labelling algorithm defined by Kejriwal and Miranker [9] (see Section 3). Records are first grouped by their tokens (Lines 1 to 3) and only blocks of below average size are considered (Lines 4 to 5, Block-purging [34]). For each record pair within these blocks we determine the blocking predicates that the record pair *agrees* by (Line 6, that is, both records share at least one common blocking key by a predicate). The top

5% of record pairs (that is, those that agree by the most blocking predicates) are then labelled as positives (Line 7). Weights may then be instantly assigned to each blocking predicate, based on their reduction ratio and proportion of labelled positives they agree for, as follows (Lines 8 to 11):

Definition 4.2.1: (Blocking Predicate Weight) For a blocking predicate (p), the assigned weight is defined as:

$$F_{RR_p, \frac{Pos_p}{|Pos|}} = \frac{2 \cdot RR_p \cdot \frac{Pos_p}{|Pos|}}{RR_p + \frac{Pos_p}{|Pos|}} \tag{5}$$

where RR_p is the Reduction Ratio of blocking predicate p, Pos_p is the number of labelled positives blocking predicate p agrees for, and $|Pos|$ is the total number of labelled positives.

It is important to label a sufficient number of record pairs as positives, but without having too many non-matching record pairs. It must also be a small enough proportion so that computation is kept low. We consider it reasonable to expect that in most datasets the top 5% of token blocked record pairs are, with high chance, mostly matching record pairs. As we will see later in our experimental evaluations, this resulted in fast and accurate labelling for every dataset. An ideal blocking predicate would both agree for every positive record pair ($\frac{Pos_p}{|Pos|} = 1.0$), and have a high reduction ratio ($RR_p \sim 1.0$). This is why we consider the best blocking predicates to be those with the highest $F_{RR_p, \frac{Pos_p}{|Pos|}}$ value, that is, the harmonic mean of RR_p and $\frac{Pos_p}{|Pos|}$.

4.3. Blocking

With the proposed blocking technique (Algorithm 2) we select the first record of a dataset (Line 3) as the *representative* record of an initial block (that is, a single record used to represent each block). Every subsequent record is then checked to see if it can be assigned to this initial block (Lines 5-8), or is considered distinct enough to become the representative record of a new block (Lines 9-10). Previously assigned records may also be assigned to any newly created

block (Lines 11-13). Each record is processed in this manner resulting in a set of dissimilar blocks of similar records (Line 15).

Algorithm 2: Non-Standard Key-Based Blocking

Input: n records: $\mathbf{R} = r_1, ..., r_n$

Set of blocking predicates, $\mathbf{P} = p_1, ..., p_{p'}$

Output: Blocks containing record pairs for linkage, $\mathbf{B} = B_1, ..., B_{b'}$

```
1  Create an empty set of blocks, B = {}
2  Select the top 3 blocking predicates by F_{RR_p, Pos_p/|Pos|} from P, P' = top 3
     blocking predicates
3  Assign r_1 as representative record b_1 of block B_1
4  Add block B_1 to B
5  foreach record r ∈ R do
6      foreach representative record b_i ∈ B do
7          if r agrees with b_i by any p ∈ P' then
8              assign r to respective block B_i
9      if r is not assigned to any block then
10         create new block B_r with r as its representative record b_r
11         foreach previously assigned record r' ∈ R do
12             if r' agrees with b_r by any p ∈ P' then
13                 assign r' to B_r
14         add B_r to B
15 Return B = B_1, ..., B_{b'}
```

A blocking criterion must be defined for the proposed approach which stipulates how records are either assigned to an existing block, or become the representative record of a new block. We assign a record to an existing block if it agrees by at least one of the top 3 blocking predicates with its representative record. We consider this to be reasonable as it uses a low enough number of blocking predicates that computation will be kept low, but large enough to allow for diverse groupings of similar records.

The proposed blocking algorithm shares some similarities with some of the approaches presented in Section 3, but is distinct in a number of ways. Standard blocking scheme learning approaches create a new block for every blocking key a blocking predicate generates. In the proposed approach a new block is only created if a record being assigned does not share any blocking keys (by any of the selected blocking predicates) with any of the current representative records. Clustering techniques typically perform many unnecessary comparisons between dissimilar records using computationally expensive distance functions. In the proposed approach all records that share blocking key(s) with another are instead instantly *retrieved* using the mapped relations. We tested the robustness of the proposed approach by running the blocking algorithm multiple times for each dataset, with the records of the dataset shuffled between each attempt. For every dataset there is little or no variation in the resultant proficiency or run time values.

5. Experimental Evaluation

All algorithms were coded using Java Eclipse Mars.1. Evaluations were ran using a Dell Optiplex 9020 with 16G of RAM, an Intel(R) Core(TM) i7-4790 with 3.60GHz and 64x Windows 7 Enterprise.

Indexing Functions for Blocking Predicates: For the proposed blocking method we used 15 of the 25 indexing functions defined by Bilenko et. al. [8]. We omitted some of the original indexing functions as experimental evaluations indicated them to have comparatively high computational cost. Each indexing function is combined with each attribute of a dataset resulting in $15 \times a$ individual blocking predicates (see Definition 2.1) for a dataset containing a attributes.

Evaluation Metrics: To compare the proposed blocking algorithm with the baseline approaches, we present tables of results for both the blocking and subsequent linkage phases of each approach for each dataset. It was demonstrated

in our previous paper [45] that blocking evaluation metrics often fail to indicate the best blocking method as part of a record linkage framework, that is, including a subsequent linkage phase. For the blocking phase we use the evaluation metrics RR, PC, $F_{RR,PC}$ and PQ (detailed in Section 2), as well as a respective time value and the number of blocked pairs. The respective time value in this case also includes the time incurred for any pre-processing steps such as standardisation and gathering of TF-IDF statistics. For the linkage results we use the evaluation metrics Precision ($Prec$), Recall (Rec) and $F_{Prec,Rec}$. Precision is the proportion of correctly classified matches out of all classified matches by a linkage technique, Recall is the proportion of correctly classified matches out of all known matches in a dataset and $F_{Prec,Rec}$ is the harmonic mean of both Precision and Recall. Time values are also presented for the linkage results which represent the combined blocking and linkage times (as well as any pre-processing) of each approach by each linkage technique. In [45] we stated that a blocking method that achieves a higher $F_{Prec,Rec}$ value in less time than all others for a dataset is inarguably a better blocking method than its peers. In some cases there may not be such a clear winner as some blocking methods may be observed to achieve higher results than others but take more time. In each table we therefore indicate in bold the blocking method that arguably achieves the best overall performance for each dataset, that is, the best balance of maximising $F_{Prec,Rec}$ and minimising Time in comparison to all other methods.

Linkage Techniques: Two different linkage techniques have been combined with the blocking methods to form distinct RL frameworks. First was a hypothetical instantaneous *Perfect* linkage technique where each record pair is perfectly classified and no time is incurred for linkage. For this linkage technique Precision is always equal to one, and Recall is always equal to the Pairs Completeness value of the corresponding blocking method. Similarly, as *Perfect* linkage is performed instantly the overall Time value is equal to that of the corresponding blocking method as well. As such only $F_{\mathrm{Prec,Rec}}$ values are provided for *Perfect* linkage as all other values (that is, Precision, Recall, Time) can be

inferred. For the second linkage technique we employ an approach [4] which uses Log TF-IDF (Term Frequency Inverse Document Frequency) for measuring similarity between records. We chose this technique as it does not require any labelled data. The Log TF-IDF measure [9] is formally defined as:

$$\text{sim}(r_1, r_2) = \sum_{q \in r_1 \cap r_2} w(r_1, q) \cdot w(r_2, q), \tag{6}$$

where

$$w(r, q) = \frac{w'(r, q)}{\sqrt{\sum_{q \in r} w'(r, q)^2}}, \tag{7}$$

and

$$w'(r, q) = \log(tf_{r,q} + 1) \cdot \log(\frac{|R|}{df_q} + 1) \tag{8}$$

where (r_1, r_2) represents a record pair, $w(r, q)$ is the normalised TF-IDF weight of a term q in a record r, $tf_{r,q}$ represents the term frequency of q in r, $|R|$ is the total number of records in the dataset R, df_q is the document frequency of the term q in the cohort, that is, how many records in the dataset contain q. In order to classify record pairs by this linkage technique a similarity threshold value is required. For each dataset an optimal TF-IDF linkage threshold value is selected for each baseline, these are also presented in our results under the heading $THold$.

Baselines: We compare our blocking algorithm to a number of baseline approaches described in Section 3 including: unsupervised blocking approaches, *Sorted Neighbourhood*, *Canopy Clustering*, *MinHash-LSH*, the unsupervised blocking scheme learning approach defined by Kejriwal and Miranker [9] and the Meta-Blocking frameworks described in [35, 42]. For the *Sorted Neighbourhood* approach, window size is set to 20 and rather than define a sorting key for each dataset, we opt for sorting records by each attribute column lexicographically. For *Canopy Clustering* and *MinHash-LSH* we evaluate across a large range of parameter values and present the best $F_{Prec,Rec}$ value achieved in the least time. For *Canopy Clustering* the lower and upper threshold parameter values

are varied from $\{0.00 \rightarrow 1.00\}$ in increments of 0.01. For *MinHash-LSH* shingle length is set to 2, MinHash signature length is varied from $\{20 \rightarrow 2,000\}$ in increments of 20, and band width is set to 5. Note that in reality a domain expert or labelled data would be needed to identify such optimal parameter values. For the unsupervised blocking scheme learning approach defined by Kejriwal and Miranker [9], the automatic labelling algorithm outlined in the same paper is implemented to generate the necessary labelled data. For the Meta-Blocking frameworks we use the same 5 weighting schemes, and 4 pruning schemes of [35], forming 20 different Meta-Blocking baselines. We enhance these frameworks with Block-filtering (r=0.80) and Graph-partitioning as described in [42]. We opt for block-filtering rather than block-purging as the latter resulted in poor results for many of our evaluation datasets. We additionally implement as a baseline the default configuration of a particular Meta-Blocking framework as detailed in [21]. Namely token blocking with Block filtering (r=0.55), Common blocks scheme weighting and weighted edge pruning.

We also include 2 supervised blocking techniques, namely those of Bilenko et. al. [8] and Michelson and Knoblock [10]. As both are supervised no run time cost is incurred for the generation of their respective labelled data. For the standard blocking scheme learning baselines, conjunctions are restricted to a maximum length of 2 and are evaluated using ten-fold cross-validation. For the unsupervised blocking scheme learning baseline, parameter values are needed for the maximum proportion of labelled negatives a blocking predicate (or conjunction) may cover, and the maximum proportion of labelled positives allowed to remain uncovered by a learned scheme. In the original paper these parameter values were varied across different ranges with average and best results for the different parameter combinations being reported in the paper. In our experimental evaluations we extend this to all of the standard blocking scheme learning baselines by presenting the best results across all parameter combinations.

For any baseline evaluated across a range of parameter values we present the best results (that is, highest $F_{Prec,Rec}$ value in the least time) as well as the optimal parameter values at which this result was obtained.

Datasets: There are 9 different datasets used in our experimental evaluations with varying characteristics (Table 1). Five of the datasets (Restaurant, Cora, Clean-Synth, Dirty-Synth and CDDB10000) are used for deduplication (that is, within a single dataset), and four (DBLP-ACM, Amazon-Google, DBLP-Scholar and Abt-Buy) are for record linkage (that is, across multiple datasets).

Table 1: Table of characteristics for each of the evaluated datasets. DD or RL indicates if a dataset is for Deduplication (DD) or Record Linkage (RL). $|\bar{p}|$ indicates the average number of attribute-value pairs per entity of a dataset.

Name	DD or RL	Number of Attributes	$\|\bar{p}\|$	Number of Records	Number of Matches
Restaurant	DD	5	4.999	864	112
Cora	DD	4	3.776	1,295	17,184
Clean-Synth	DD	10	10.000	10,000	2,000
Dirty-Synth	DD	9	8.972	10,000	26,692
CDDB10000	DD	7	6.181	10,000	251
DBLP-ACM	RL	4	3.997	2,616+2,294	2,224
Amazon-Google	RL	4	3.281	1,362+3,225	1,300
DBLP-Scholar	RL	4	3.252	2,616+64,263	5,347
Abt-Buy	RL	4	2.261	1,081+1,092	1,097

CDDB10000 is a 10,000 sized sample of a much larger CD DataBase (CDDB, 750,000 records) dataset (http://www.freedb.org). Clean-Synth and Dirty-Synth are synthetically generated datasets using a modified version of the synthetic data generator defined by Christen [46]. Each record of Clean-Synth may match with at most one other record, whereas those of Dirty-Synth may potentially match with up to 20 other records. Up to 2 minor typographical errors may apply to the attribute values of the synthetic datasets and a small proportion of similar, but not matching, record pairs were deliberately generated in each.

6. Results and Discussion

In Tables 2 to 10 we present blocking and linkage results for the proposed blocking technique alongside those of the baselines. In these tables BL refers to the proposed blocking technique, $SoNe$ refers to *Sorted Neighbourhood*, $CaCl$ refers to *Canopy-Clustering* and $MinH$ refers to *MinHash-LSH*. $Meta$ refers to the Meta-Blocking frameworks of [35, 42], and CSB refers to the default configuration of a state-of-the-art blocking workflow from [21]. For clarity we present the average results of the different $Meta$ frameworks for each dataset. A refers to the supervised blocking scheme learning approach defined by Bilenko et. al. [8], B refers to the supervised blocking scheme learning approach defined by Michelson and Knoblock [10] and C refers to the unsupervised blocking scheme learning approach defined by Kejriwal and Miranker [9]. The adjoining 1 and 2 values of A, B and C indicate variations of each blocking scheme learning approach in which maximum conjunction lengths of 1 (Disjunctive blocking scheme) and 2 (Disjunctive Normal Form blocking schemes) are used respectively (see Definition 1.2). By analysing these tables of results we can make a number of interesting observations.

Looking to the $Perfect$ linkage results we observe that BL performs consistently well, achieving the highest or near highest Perfect $F_{Prec,Rec}$ value in all but one case. Interestingly, in the worst case (Table 6) BL achieved a Perfect $F_{Prec,Rec}$ value much lower than all other baselines, yet it achieves the highest TFIDF $F_{Prec,Rec}$ value for the same dataset. We believe this to be because although BL has a much lower PC (and therefore $Perfect$) value for this dataset than the other baselines, the blocked record pairs are more easily classified. This is indicated by its considerably higher Precision than the other baselines in this case.

Looking to the TF-IDF linkage results we observe again that BL achieves the highest, or near highest, $F_{Prec,Rec}$ value in every case. BL is also observed to achieve equal or better proficiency than $SoNe$ in every single case. In the worst case (Table 3) $CaCl$, $MinH$, A1, A2, B1, B2, C1 and C2 achieve a considerably

Table 2: Detailed numerical results upon *Restaurant* with time measured in seconds. lt and ut indicate the lower and upper threshold values used for Canopy Clustering respectively. l indicates the Min-Hash signature length used for MinH. η and ϵ indicate the maximum proportion of labelled negatives a blocking predicate may cover, and the maximum proportion of labelled positives allowed to remain uncovered by a learned blocking scheme respectively.

		Blocking						Perfect	TF-IDF				
Method	Param	RR	PC	$F_{\text{RR,PC}}$	PQ	No.Pairs	Time	$F_{\text{Prec,Rec}}$	THold	Prec	Rec	$F_{\text{Prec,Rec}}$	Time
BL		0.999	0.991	0.995	0.466	238	0.3	0.996	0.633	0.931	0.964	0.947	0.3
SoNe		0.772	1.000	0.871	0.001	85117	0.0	1.000	0.633	0.931	0.964	0.947	0.2
CaCl	**lt:0.20 ut:0.36**	**0.998**	**1.000**	**0.999**	**0.176**	**637**	**0.2**	**0.982**	**0.633**	**0.964**	**0.964**	**0.964**	**0.2**
MinH	l:240	0.981	1.000	0.990	0.016	7120	0.5	1.000	0.633	0.939	0.964	0.952	0.5
Meta		0.936	0.998	0.965	0.015	23935	0.4	0.981	0.634	0.931	0.963	0.946	0.4
CSB		0.990	1.000	0.995	0.032	3555	0.3	0.982	0.633	0.931	0.964	0.947	0.4
A1	η:0.0 ϵ:0.0	0.990	1.000	0.995	0.034	3649	0.3	1.000	0.633	0.931	0.964	0.947	0.3
A2	η:0.0 ϵ:0.3	0.991	1.000	0.996	0.038	3303	0.4	1.000	0.633	0.931	0.964	0.947	0.4
B1	η:0.5 ϵ:0.0	0.695	0.999	0.815	0.001	113659	0.3	1.000	0.633	0.931	0.964	0.947	0.4
B2	η:0.5 ϵ:0.0	0.728	0.998	0.835	0.001	101580	0.3	0.999	0.634	0.931	0.962	0.946	0.5
C1	η:0.0 ϵ:0.5	0.991	1.000	0.995	0.037	3492	1.9	1.000	0.633	0.931	0.964	0.947	1.9
C2	η:0.0 ϵ:0.5	0.991	1.000	0.995	0.037	3450	1.9	1.000	0.633	0.931	0.964	0.947	1.9

higher $F_{Prec,Rec}$ value than BL. However, most of these are observed to perform poorly in other cases whereas BL is always one of the best performing. One must also remember that results presented for $CaCl$, $MinH$, A1, A2, B1, B2, C1 and C2 are the best possible across multiple parameter combinations. Despite this, BL still achieves a higher $F_{\text{Prec,Rec}}$ value than them, including $CaCl$, in some other cases. In other words, even when $CaCl$, $MinH$, A1, A2, B1, B2, C1 and C2 are operating optimally BL may still perform better, or at least comparable.

For CDDB10000, Amazon-Google and Abt-Buy very low $F_{Prec,Rec}$ values by TF-IDF linkage are achieved by all approaches despite having quite strong $Perfect$ linkage results. This may be explained by the fact that these datasets contain some substantially longer attribute values (for example, product descriptions) than is found in the other datasets. Longer attribute values may allow two dissimilar records to share a token with low document frequency, which may result in a disproportionally high similarity score by TF-IDF linkage (See Eqns. 6 - 8). In other papers that these datasets were evaluated in, similarly low

Table 3: Detailed numerical results upon *Cora* with time measured in seconds. *lt* and *ut* indicate the lower and upper threshold values used for Canopy Clustering respectively. l indicates the Min-Hash signature length used for MinH. η and ϵ indicate the maximum proportion of labelled negatives a blocking predicate may cover, and the maximum proportion of labelled positives allowed to remain uncovered by a learned blocking scheme respectively.

		Blocking						Perfect	TF-IDF				
Method	Param	RR	PC	$F_{\text{RR,PC}}$	PQ	No.Pairs	Time	$F_{\text{Prec,Rec}}$	THold	Prec	Rec	$F_{\text{Prec,Rec}}$	Time
BL		0.954	0.857	0.903	0.383	38432	0.5	0.923	0.391	0.767	0.827	0.796	0.5
SoNe		0.884	0.867	0.876	0.140	96872	0.0	0.929	0.435	0.768	0.790	0.779	0.2
CaCl	**lt:0.26 ut:0.41**	**0.952**	**0.946**	**0.949**	**0.401**	**40543**	**0.1**	**0.928**	**0.437**	**0.791**	**0.866**	**0.827**	**0.1**
MinH	l:40	0.935	0.878	0.905	0.276	54653	0.3	0.935	0.428	0.804	0.832	0.817	0.5
Meta		0.857	0.780	0.780	0.435	119318	2.2	0.817	0.403	0.824	0.703	0.746	2.3
CSB		0.942	0.961	0.952	0.341	48428	0.8	0.909	0.473	0.776	0.833	0.803	0.9
A1	η:0.0 ϵ:0.6	0.973	0.914	0.943	0.689	22801	0.8	0.955	0.402	0.768	0.877	0.819	0.9
A2	η:0.0 ϵ:0.7	0.973	0.914	0.943	0.689	22801	0.9	0.955	0.402	0.768	0.877	0.819	0.9
B1	η:0.0 ϵ:0.0	0.952	0.926	0.939	0.425	39861	0.8	0.962	0.457	0.777	0.847	0.810	0.9
B2	η:0.0 ϵ:0.0	0.968	0.919	0.943	0.598	26815	0.8	0.958	0.437	0.772	0.858	0.813	1.0
C1	η:0.5 ϵ:0.0	0.963	0.922	0.942	0.586	30966	2.7	0.959	0.412	0.768	0.875	0.818	2.7
C2	η:0.1 ϵ:0.8	0.971	0.920	0.945	0.652	24523	2.8	0.958	0.404	0.766	0.880	0.819	2.9

Table 4: Detailed numerical results upon *Clean-Synth* with time measured in seconds. *lt* and *ut* indicate the lower and upper threshold values used for Canopy Clustering respectively. l indicates the Min-Hash signature length used for MinH. η and ϵ indicate the maximum proportion of labelled negatives a blocking predicate may cover, and the maximum proportion of labelled positives allowed to remain uncovered by a learned blocking scheme respectively.

		Blocking						Perfect	TF-IDF				
Method	Param	RR	PC	$F_{\text{RR,PC}}$	PQ	No.Pairs	Time	$F_{\text{Prec,Rec}}$	THold	Prec	Rec	$F_{\text{Prec,Rec}}$	Time
BL		0.997	1.000	0.998	0.012	166797	2.6	1.000	0.576	1.000	1.000	1.000	2.8
SoNe		0.960	1.000	0.980	0.001	1997819	0.2	1.000	0.576	1.000	1.000	1.000	6.7
CaCl	lt:0.08 ut:0.16	0.985	1.000	0.992	0.003	746158	25.8	1.000	0.576	1.000	1.000	1.000	26.6
MinH	l:100	0.991	1.000	0.996	0.005	432362	3.7	1.000	0.576	1.000	1.000	1.000	4.2
Meta		0.987	1.000	0.993	0.016	642204	33.5	1.000	0.576	1.000	1.000	1.000	34.2
CSB		**1.000**	**1.000**	**1.000**	**0.607**	**3295**	**1.4**	**1.000**	**0.576**	**1.000**	**1.000**	**1.000**	**1.4**
A1	η:0.0 ϵ:0.6	1.000	1.000	1.000	0.618	4100	4.0	1.000	0.576	1.000	1.000	1.000	4.0
A2	η:0.5 ϵ:0.7	1.000	1.000	1.000	0.364	5501	4.2	1.000	0.576	1.000	1.000	1.000	4.2
B1	η:0.0 ϵ:0.0	0.999	1.000	1.000	0.228	36334	4.0	1.000	0.576	1.000	1.000	1.000	4.1
B2	η:0.0 ϵ:0.0	1.000	1.000	1.000	0.675	3618	4.2	1.000	0.576	1.000	1.000	1.000	6.7
C1	η:0.5 ϵ:0.2	1.000	1.000	1.000	0.618	4100	15.0	1.000	0.576	1.000	1.000	1.000	15.0
C2	η:0.1 ϵ:0.1	1.000	1.000	1.000	0.979	2044	15.2	1.000	0.576	1.000	1.000	1.000	16.2

Table 5: Detailed numerical results upon *Dirty-Synth* with time measured in seconds. *lt* and *ut* indicate the lower and upper threshold values used for Canopy Clustering respectively. *l* indicates the Min-Hash signature length used for MinH. η and ϵ indicate the maximum proportion of labelled negatives a blocking predicate may cover, and the maximum proportion of labelled positives allowed to remain uncovered by a learned blocking scheme respectively.

		Blocking						Perfect	TF-IDF				
Method	Param	RR	PC	$F_{\text{RR,PC}}$	PQ	No.Pairs	Time	$F_{\text{Prec,Rec}}$	THold	Prec	Rec	$F_{\text{Prec,Rec}}$	Time
BL		0.997	1.000	0.998	0.173	154320	1.7	1.000	0.370	1.000	1.000	1.000	1.9
SoNe		0.964	1.000	0.982	0.015	1792356	0.2	1.000	0.370	1.000	1.000	1.000	6.4
CaCl	lt:0.14 ut:0.16	0.999	1.000	1.000	0.969	27549	10.5	1.000	0.370	1.000	1.000	1.000	10.5
MinH	l:360	0.967	0.999	0.983	0.016	1658740	13.1	1.000	0.370	1.000	0.999	1.000	14.9
Meta		0.993	0.926	0.953	0.239	376227	17.9	0.957	0.372	1.000	0.926	0.957	18.3
CSB		**0.999**	**1.000**	**1.000**	**0.973**	**27428**	**1.1**	**1.000**	**0.370**	**1.000**	**1.000**	**1.000**	**1.1**
A1	η:0.5 ϵ:0.2	0.999	1.000	1.000	1.000	26692	3.9	1.000	0.370	1.000	1.000	1.000	3.9
A2	η:0.5 ϵ:0.8	0.999	1.000	1.000	1.000	26692	4.0	1.000	0.370	1.000	1.000	1.000	4.3
B1	η:0.0 ϵ:0.1	0.999	1.000	1.000	0.826	40190	3.9	1.000	0.370	1.000	1.000	1.000	3.9
B2	η:0.0 ϵ:0.1	0.999	1.000	1.000	0.931	32632	4.0	1.000	0.370	1.000	1.000	1.000	4.3
C1	η:0.5 ϵ:0.2	0.999	1.000	1.000	1.000	26692	14.0	1.000	0.370	1.000	1.000	1.000	14.0
C2	η:0.1 ϵ:1.0	0.999	1.000	1.000	1.000	26703	14.3	1.000	0.370	1.000	1.000	1.000	14.5

Table 6: Detailed numerical results upon *CDDB10000* with time measured in seconds. *lt* and *ut* indicate the lower and upper threshold values used for Canopy Clustering respectively. *l* indicates the Min-Hash signature length used for MinH. η and ϵ indicate the maximum proportion of labelled negatives a blocking predicate may cover, and the maximum proportion of labelled positives allowed to remain uncovered by a learned blocking scheme respectively.

		Blocking						Perfect	TF-IDF				
Method	Param	RR	PC	$F_{\text{RR,PC}}$	PQ	No.Pairs	Time	$F_{\text{Prec,Rec}}$	THold	Prec	Rec	$F_{\text{Prec,Rec}}$	Time
BL		**0.963**	**0.502**	**0.660**	**0.000**	**1840147**	**2.6**	**0.668**	**0.656**	**0.539**	**0.382**	**0.448**	**3.7**
SoNe		0.975	1.000	0.988	0.000	1234611	0.2	1.000	0.670	0.291	0.578	0.387	4.2
CaCl	lt:0.04 ut:0.05	0.955	0.809	0.876	0.000	2228203	0.3	0.894	0.670	0.385	0.494	0.433	1.3
MinH	l:40	1.000	0.789	0.882	0.012	16988	1.7	0.882	0.596	0.329	0.685	0.444	3.3
Meta		0.978	0.981	0.979	0.003	1077863	20.8	0.750	0.662	0.306	0.602	0.404	21.6
CSB		0.999	0.984	0.991	0.005	54328	2.9	0.729	0.670	0.304	0.574	0.398	2.9
A1	η:0.0 ϵ:0.0	0.975	1.000	0.987	0.000	1240613	3.0	1.000	0.669	0.284	0.578	0.380	3.9
A2	η:0.0 ϵ:0.0	0.978	0.997	0.987	0.000	1101283	3.4	0.999	0.669	0.284	0.578	0.380	21.7
B1	η:0.0 ϵ:0.0	0.995	0.997	0.996	0.001	242592	3.0	0.998	0.669	0.282	0.576	0.379	3.3
B2	η:0.0 ϵ:0.0	0.998	0.993	0.995	0.003	118208	3.3	0.997	0.669	0.284	0.578	0.381	10.4
C1	η:0.0 ϵ:0.0	0.974	0.997	0.985	0.000	1306772	12.0	0.999	0.669	0.283	0.578	0.380	13.0
C2	η:0.0 ϵ:0.0	0.991	0.995	0.993	0.001	425428	12.2	0.997	0.669	0.283	0.578	0.380	13.5

Table 7: Detailed numerical results upon *DBLP-ACM* with time measured in seconds. lt and ut indicate the lower and upper threshold values used for Canopy Clustering respectively. l indicates the Min-Hash signature length used for MinH. η and ϵ indicate the maximum proportion of labelled negatives a blocking predicate may cover, and the maximum proportion of labelled positives allowed to remain uncovered by a learned blocking scheme respectively.

		Blocking						Perfect	TF-IDF				
Method	Param	RR	PC	$F_{\text{RR,PC}}$	PQ	No.Pairs	Time	$F_{\text{Prec,Rec}}$	THold	Prec	Rec	$F_{\text{Prec,Rec}}$	Time
BL		0.990	0.987	0.988	0.036	61564	1.1	0.993	0.555	0.889	0.958	0.922	1.1
SoNe		0.939	0.990	0.964	0.005	366154	0.1	0.995	0.652	0.943	0.897	0.920	1.1
CaCl	**lt:0.09 ut:0.27**	**0.897**	**0.979**	**0.937**	**0.004**	**615485**	**1.9**	**0.990**	**0.571**	**0.917**	**0.953**	**0.935**	**2.8**
MinH	l:100	0.958	0.929	0.943	0.008	254911	1.8	0.963	0.555	0.923	0.916	0.919	2.2
Meta		0.960	0.999	0.979	0.034	238663	4.2	0.951	0.647	0.933	0.905	0.919	4.4
CSB		0.995	0.999	0.997	0.081	27345	1.0	0.949	0.652	0.935	0.902	0.919	1.0
A1	η:0.5 ϵ:0.0	0.928	0.995	0.960	0.051	432653	3.0	0.998	0.568	0.891	0.956	0.922	3.4
A2	η:0.1 ϵ:0.4	0.999	0.972	0.985	0.347	8838	3.1	0.986	0.532	0.897	0.953	0.923	4.1
B1	η:0.0 ϵ:0.0	0.999	0.975	0.987	0.467	5100	3.0	0.987	0.617	0.924	0.907	0.915	3.0
B2	η:0.1 ϵ:0.0	0.999	0.975	0.987	0.644	3801	3.1	0.987	0.574	0.910	0.929	0.918	13.8
C1	η:0.0 ϵ:0.4	0.999	0.987	0.993	0.599	3720	9.7	0.993	0.570	0.904	0.943	0.922	9.7
C2	η:0.0 ϵ:1.0	0.999	0.987	0.993	0.599	3720	9.7	0.993	0.570	0.904	0.943	0.922	9.7

Table 8: Detailed numerical results upon *Amazon-Google* with time measured in seconds. lt and ut indicate the lower and upper threshold values used for Canopy Clustering respectively. l indicates the Min-Hash signature length used for MinH. η and ϵ indicate the maximum proportion of labelled negatives a blocking predicate may cover, and the maximum proportion of labelled positives allowed to remain uncovered by a learned blocking scheme respectively.

		Blocking						Perfect	TF-IDF				
Method	Param	RR	PC	$F_{\text{RR,PC}}$	PQ	No.Pairs	Time	$F_{\text{Prec,Rec}}$	THold	Prec	Rec	$F_{\text{Prec,Rec}}$	Time
BL		0.779	0.954	0.858	0.001	971767	4.5	0.976	0.432	0.361	0.438	0.396	7.0
SoNe		0.933	0.627	0.750	0.002	296593	0.5	0.771	0.371	0.308	0.369	0.336	2.3
CaCl	lt:0.11 ut:0.23	0.956	0.820	0.883	0.006	193409	3.1	0.901	0.432	0.366	0.442	0.401	3.8
MinH	l:1640	0.418	0.802	0.549	0.000	2560547	75.5	0.890	0.430	0.375	0.402	0.388	99.7
Meta		0.907	0.913	0.904	0.009	410830	28.9	0.625	0.425	0.365	0.455	0.404	30.3
CSB		**0.971**	**0.907**	**0.938**	**0.009**	**128160**	**5.9**	**0.627**	**0.427**	**0.361**	**0.457**	**0.403**	**6.8**
A1	η:0.1 ϵ:0.2	0.997	0.747	0.851	0.075	14647	34.0	0.852	0.417	0.367	0.448	0.401	34.2
A2	η:0.0 ϵ:0.1	0.997	0.713	0.825	0.086	12326	34.2	0.826	0.411	0.367	0.439	0.398	36.0
B1	η:0.1 ϵ:0.0	0.987	0.906	0.945	0.022	55163	33.2	0.950	0.429	0.354	0.458	0.400	33.8
B2	η:0.1 ϵ:0.0	0.989	0.862	0.919	0.040	49033	33.3	0.924	0.429	0.356	0.453	0.399	66.5
C1	η:0.0 ϵ:0.6	0.997	0.740	0.849	0.086	12070	74.7	0.850	0.403	0.360	0.469	0.405	74.9
C2	η:0.0 ϵ:0.5	0.997	0.740	0.849	0.087	12024	74.7	0.850	0.403	0.360	0.469	0.405	74.9

Table 9: Detailed numerical results upon *DBLP-Scholar* with time measured in seconds. *lt* and *ut* indicate the lower and upper threshold values used for Canopy Clustering respectively. l indicates the Min-Hash signature length used for MinH. η and ϵ indicate the maximum proportion of labelled negatives a blocking predicate may cover, and the maximum proportion of labelled positives allowed to remain uncovered by a learned blocking scheme respectively.

		Blocking						Perfect	TF-IDF				
Method	Param	RR	PC	$F_{\text{RR,PC}}$	PQ	No.Pairs	Time	$F_{\text{Prec,Rec}}$	THold	Prec	Rec	$F_{\text{Prec,Rec}}$	Time
BL		0.959	0.994	0.976	0.001	6900488	12.1	0.997	0.583	0.825	0.848	0.836	17.6
SoNe		0.975	0.935	0.954	0.001	4243241	0.7	0.966	0.546	0.826	0.837	0.832	179.9
CaCl	lt:0.05 ut:0.13	0.803	0.967	0.877	0.000	33069370	80.8	0.983	0.583	0.835	0.826	0.830	173.0
MinH	l:240	0.962	0.962	0.962	0.001	6433871	86.2	0.980	0.582	0.839	0.838	0.839	117.0
Meta		0.974	0.990	0.982	0.048	4303859	89.1	0.917	0.583	0.821	0.847	0.833	97.4
CSB		0.999	0.997	0.998	0.021	252150	12.6	0.918	0.583	0.819	0.849	0.833	13.0
A1	**η:0.1 ϵ:0.1**	**0.999**	**0.953**	**0.975**	**0.109**	**113361**	**33.1**	**0.976**	**0.514**	**0.835**	**0.876**	**0.855**	**33.2**
A2	η:0.0 ϵ:0.2	1.000	0.880	0.936	0.357	24432	31.9	0.936	0.495	0.862	0.832	0.846	32.0
B1	η:0.5 ϵ:0.0	0.998	0.982	0.990	0.047	362265	30.7	0.991	0.584	0.826	0.843	0.834	31.5
B2	η:0.0 ϵ:0.0	1.000	0.921	0.958	0.269	39284	30.7	0.958	0.533	0.838	0.830	0.833	180.7
C1	η:0.0 ϵ:0.6	0.999	0.977	0.988	0.048	128947	85.8	0.989	0.583	0.827	0.842	0.835	86.0
C2	η:0.0 ϵ:0.0	0.999	0.977	0.988	0.049	127576	85.8	0.989	0.583	0.827	0.842	0.835	87.6

Table 10: Detailed numerical results upon *Abt-Buy* with time measured in seconds. *lt* and *ut* indicate the lower and upper threshold values used for Canopy Clustering respectively. l indicates the Min-Hash signature length used for MinH. η and ϵ indicate the maximum proportion of labelled negatives a blocking predicate may cover, and the maximum proportion of labelled positives allowed to remain uncovered by a learned blocking scheme respectively.

		Blocking						Perfect	TF-IDF				
Method	Param	RR	PC	$F_{\text{RR,PC}}$	PQ	No.Pairs	Time	$F_{\text{Prec,Rec}}$	THold	Prec	Rec	$F_{\text{Prec,Rec}}$	Time
BL		0.924	0.982	0.952	0.012	89866	1.0	0.991	0.375	0.602	0.587	0.594	1.1
SoNe		0.918	0.769	0.837	0.006	96709	0.1	0.869	0.362	0.619	0.508	0.558	0.5
CaCl	lt:0.10 ut:0.34	0.938	0.968	0.953	0.015	72816	1.2	0.736	0.375	0.584	0.583	0.584	1.3
MinH	l:1900	0.883	0.688	0.773	0.005	137917	27.0	0.815	0.363	0.567	0.464	0.510	27.4
Meta		0.970	0.961	0.966	0.037	35163	0.9	0.730	0.379	0.589	0.576	0.582	1.0
CSB		**0.993**	**0.842**	**0.912**	**0.116**	**7960**	**0.6**	**0.755**	**0.358**	**0.586**	**0.607**	**0.596**	**0.6**
A1	η:1.0 ϵ:0.2	0.941	0.891	0.915	0.015	69923	2.3	0.942	0.375	0.639	0.541	0.586	2.4
A2	η:1.0 ϵ:0.1	0.986	0.881	0.930	0.077	16516	2.5	0.936	0.375	0.625	0.565	0.593	3.0
B1	η:0.1 ϵ:0.0	0.966	0.972	0.969	0.028	39855	2.4	0.986	0.375	0.580	0.585	0.582	2.5
B2	η:0.5 ϵ:0.0	0.983	0.948	0.965	0.055	20459	2.5	0.973	0.379	0.596	0.572	0.583	6.0
C1	η:0.1 ϵ:0.9	0.942	0.987	0.964	0.018	68122	7.9	0.993	0.380	0.597	0.574	0.585	8.0
C2	η:0.1 ϵ:0.1	0.980	0.954	0.967	0.047	23762	8.0	0.976	0.376	0.594	0.579	0.586	9.5

results were also achieved [47, 48, 49, 50, 51, 52], suggesting that these datasets are simply difficult for the task of record linkage.

Looking at the runtime values for TF-IDF linkage we observe that BL is the quickest or near-quickest method in every case. Although $SoNe$ and $CaCl$ are marginally faster than BL in some cases, they are considerably slower in others, in particular for the largest dataset (Table 9) where $SoNe$ and $CaCl$ each take nearly 3 minutes to complete, whereas BL requires less than 18 seconds. For the specific Meta-Blocking framework we observe that CSB is often, but not always, faster than BL. In the worse case (Table 9) BL is still remarkably fast compared to all other baselines even if not as fast as CSB. BL is faster than $MinH$, $Meta$, A1, A2, B1, B2, C1 and C2 in almost every case, often by a considerable margin. The only exceptions are A1 in Table 2, B1 and $MinH$ in Table 6 and $Meta$ in Table 10 which are less than $0.5s$ faster than BL in each respective case. This is especially significant considering A1, A2, B1 and B2 assume labelled data to be available at no computational cost.

In summary, BL is always amongst the quickest and most proficient blocking methods for every case. Although other blocking methods may be quicker or more proficient than BL for some cases, they are also often among the slowest or least proficient in others. For example, CSB performs quickly and proficiently in most cases, but has considerably lower proficiency than BL in Table 6. Moreover, we are assuming the best parametrisation for those competing methods, which in practice might not be so easy to achieve.

6.1. Scalability of Blocking Methods

In Figure 2 we present the time (in seconds) for the blocking phase of the different blocking methods for datasets of various size. For the datasets we gather samples of different sizes from the 750,000 record CDDB dataset from which CDDB10000 was originally sourced. For baselines that require parameters we present runtime values using the same parameter values for which they each performed optimally for CDDB10000 in Table 6.

We observe that the proposed approach performs best in terms of scalability. We note that although SoNe and MinH also scale relatively well they often perform much slower than the proposed blocking method when linkage is considered (Tables 2- 10). We believe this to be because although these approaches often block quickly, they often block poorly, most notably in Table 9 when the proposed blocking method completes both its blocking and subsequent linkage in under 18 seconds, whereas SoNe and MinH each require 2-3 minutes. We also remind the reader that MinH is operating using optimal parameters. This would not typically be the case unless a domain expert went to great effort or by using labelled data to identify such optimal parameters. If using non-optimal parameter values (for example, longer MinHash signatures) one would expect higher blocking times.

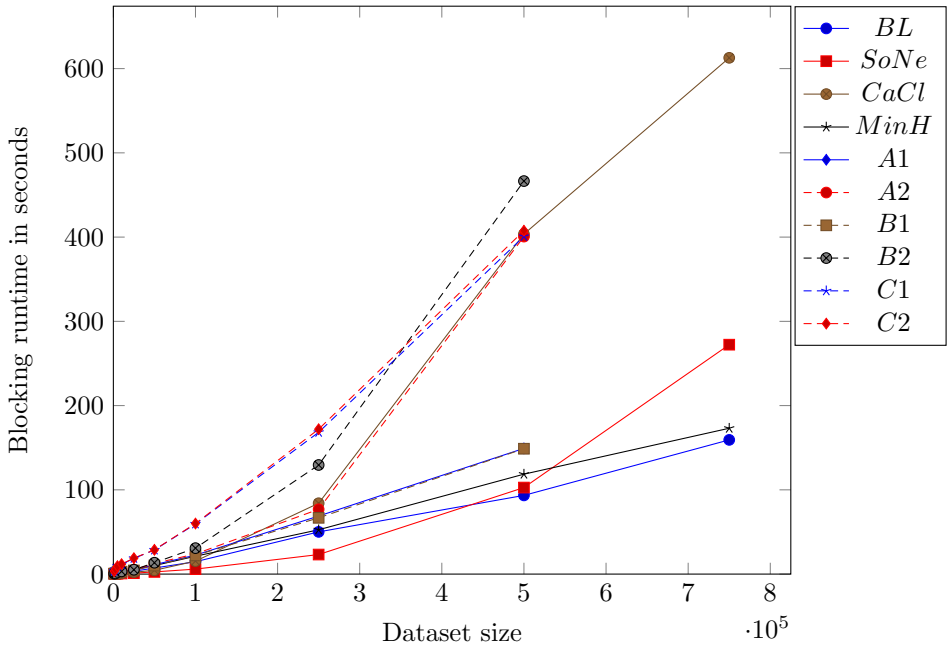

Figure 2: Blocking runtime (in seconds) of the proposed approach in comparison to the baselines for datasets of various size

The Meta blocking approaches (Meta and CSB) are excluded from this graph as they were observed to scale poorly, failing to complete for many of the larger datasets (often due to memory limitations). We believe this is the case because the Meta-Blocking approaches must first fully form a graph of edge-weight values for every record pair of a block collection, before reducing this number via a pruning technique. For large datasets this can be computationally demanding. This is further validated in Tables 2- 10 as Meta and CSB often have higher blocking times than the proposed approach.

Results for blocking scheme learning approaches (A1, A2, B1, B2, C1, C2) are only presented as far as the 500,000 record dataset, as they were not able to fully complete for the largest dataset. We again note that for the supervised blocking scheme learning approaches (A1, A2, B1 and B2) the time needed for sourcing or generating labelled data is not taken into consideration, and that in reality their times would actually be much greater. Despite this the proposed approach is still observed to scale better than these supervised approaches even when they assume labelled data. As the unsupervised approaches of C1 and C2 must first automatically label data, to then learn optimal blocking schemes with, it is unsurprising that these approaches scale poorly. It is worth noting however that the time values for C1 and C2 are dominated by this labelling process.

7. Future Work and Conclusion

In this paper a novel unsupervised blocking algorithm for structured datasets was introduced that overcomes many of the issues associated with other blocking techniques, namely the significant amount of manual fine-tuning of parameters, need for labelled data and overly long runtime.

The proposed approach was among the fastest and most proficient in every case, including the best results of some baselines allowed to run multiple times using different parameter values. This indicates a significant advantage of the proposed approach over others, as there may not always be a domain expert,

time or labelled data available.

The proposed blocking algorithm follows a similar format to that of the unsupervised blocking scheme learning baseline, in that labelled data is first automatically generated and then each blocking predicate is evaluated individually using supervised evaluation metrics. However, in the proposed approach this process is made much more efficient. By exploiting the relations between individual records and their blocking keys, record pairs that agree by many blocking predicates can be efficiently retrieved and labelled as positives. This is in contrast to the original approach in which computationally expensive distance functions are used to search for highly similar record pairs. As such, the proposed blocking approach was observed to operate consistently fast in comparison to the baselines. This is most evident for the largest dataset in which the proposed blocking approach is significantly faster than all other approaches except one, albeit assuming optimal parameter values.

In the experimental evaluations the proposed technique was shown to perform consistently well for every dataset, achieving either the best, or near best, balance of $F_{Prec,Rec}$ and time in every case. Although some baselines may have been faster or more proficient than the proposed algorithm in some cases, they were all shown to perform poorly in others.

In future work we would like to adapt the proposed approach for semi-structured and unstructured datasets. For semi-structured datasets we believe this can be achieved by combining the proposed blocking approach with *Attribute Clustering* [31]. For unstructured datasets we would like to further explore combining the proposed blocking approach with Meta-Blocking and token-blocking block-refinement techniques.

References

[1] K. O Hare, A. Jurek-Loughrey, C. de Campos, A review of unsupervised and semi-supervised blocking methods for record linkage, in: Linking and Mining Heterogeneous and Multi-view Data, Springer, 2019, pp. 79–105.

[2] P. Christen, Data matching: concepts and techniques for record linkage, entity resolution, and duplicate detection, Springer Science Business Media, 2012.

[3] M. G. Elfeky, V. S. Verykios, A. K. Elmagarmid, Tailor: A record linkage toolbox, in: Data Engineering, 2002. Proceedings. 18th International Conference on, IEEE, 2002, pp. 17–28.

[4] A. K. Elmagarmid, P. G. Ipeirotis, V. S. Verykios, Duplicate record detection: A survey, IEEE Transactions on knowledge and data engineering 19 (1).

[5] A. Jurek-Loughrey, P. Deepak, Semi-supervised and unsupervised approaches to record pairs classification in multi-source data linkage, in: Linking and Mining Heterogeneous and Multi-view Data, Springer, 2019, pp. 55–78.

[6] H. Köpcke, E. Rahm, Frameworks for entity matching: A comparison, Data & Knowledge Engineering 69 (2) (2010) 197–210.

[7] M. Y. Bilenko, Learnable similarity functions and their application to record linkage and clustering, Ph.D. thesis (2006).

[8] M. Bilenko, B. Kamath, R. J. Mooney, Adaptive blocking: Learning to scale up record linkage, in: Data Mining, 2006. ICDM'06. Sixth International Conference on, IEEE, 2006, pp. 87–96.

[9] M. Kejriwal, D. P. Miranker, An unsupervised algorithm for learning blocking schemes, in: Data Mining (ICDM), 2013 IEEE 13th International Conference on, IEEE, 2013, pp. 340–349.

[10] M. Michelson, C. A. Knoblock, Learning blocking schemes for record linkage, in: AAAI, 2006, pp. 440–445.

[11] S. Lata, A heuristic approach to record deduplication, Vol. 3, National Conference on Real Time Systems, International Journal of Engineering Research and Technology, IJERT, 2015.

[12] S. E. Whang, D. Marmaros, H. Garcia-Molina, Pay-as-you-go entity resolution, IEEE Transactions on Knowledge and Data Engineering 25 (5) (2012) 1111–1124.

[13] F. Guillet, H. J. Hamilton, Quality measures in data mining, Vol. 43, Springer, 2007.

[14] A. Bilke, F. Naumann, Schema matching using duplicates, in: Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference on, IEEE, 2005, pp. 69–80.

[15] R. O. Duda, P. E. Hart, D. G. Stork, et al., Pattern classification. 2nd, Edition. New York (2001) 55.

[16] D. B. V. Babu, K. J. Santoshi, Unsupervised detection of duplicates in user query results using blocking, International Journal of Computer Science and Information Technologies, IJCSIT 5 (3) (2014) 3514–3520.

[17] M. Kejriwal, D. P. Miranker, A two-step blocking scheme learner for scalable link discovery, in: OM'14 Proceedings of the 9th International Conference on Ontology, 2014, pp. 49–60.

[18] M. Kejriwal, D. P. Miranker, On linking heterogeneous dataset collections., in: International Semantic Web Conference (Posters & Demos), Citeseer, 2014, pp. 217–220.

[19] A. McCallum, K. Nigam, L. H. Ungar, Efficient clustering of high-dimensional data sets with application to reference matching, in: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2000, pp. 169–178.

[20] M. A. Hernández, S. J. Stolfo, Real-world data is dirty: Data cleansing and the merge/purge problem, Data mining and knowledge discovery 2 (1) (1998) 9–37.

[21] G. Papadakis, J. Svirsky, A. Gal, T. Palpanas, Comparative analysis of approximate blocking techniques for entity resolution, Proceedings of the VLDB Endowment 9 (9) (2016) 684–695.

[22] W. E. Winkler, Overview of record linkage and current research directions, in: Bureau of the Census, Citeseer, 2006.

[23] A. Gionis, P. Indyk, R. Motwani, et al., Similarity search in high dimensions via hashing, in: Vldb, Vol. 99, 1999, pp. 518–529.

[24] C. Faloutsos, K.-I. Lin, FastMap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets, Vol. 24, ACM, 1995.

[25] G. Hristescu, M. Farach-Colton, Cluster-preserving embedding of proteins, Tech. rep., Technical Report 99-50, Computer Science Department, Rutgers University (1999).

[26] J. T.-L. Wang, X. Wang, K.-I. Lin, D. Shasha, B. A. Shapiro, K. Zhang, Evaluating a class of distance-mapping algorithms for data mining and clustering, in: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 1999, pp. 307–311.

[27] L. Jin, C. Li, S. Mehrotra, Efficient record linkage in large data sets, in: Database Systems for Advanced Applications, 2003.(DASFAA 2003). Proceedings. Eighth International Conference on, IEEE, 2003, pp. 137–146.

[28] J. Leskovec, A. Rajaraman, J. D. Ullman, Mining of massive datasets, Cambridge university press, 2014.

[29] M. Cui, Towards a scalable and robust entity resolution-approximate blocking with semantic constraints, Tech. rep., Australian National University (2014).

[30] D. Karapiperis, V. S. Verykios, An lsh-based blocking approach with a homomorphic matching technique for privacy-preserving record linkage, IEEE Transactions on Knowledge and Data Engineering 27 (4) (2015) 909–921.

[31] G. Simonini, S. Bergamaschi, H. Jagadish, Blast: a loosely schema-aware meta-blocking approach for entity resolution, Proceedings of the VLDB Endowment 9 (12) (2016) 1173–1184.

[32] Q. Wang, M. Cui, H. Liang, Semantic-aware blocking for entity resolution, IEEE Transactions on Knowledge and Data Engineering 28 (1) (2016) 166–180.

[33] A. K. Jain, R. C. Dubes, Algorithms for clustering data, Prentice-Hall, Inc., 1988.

[34] G. Papadakis, E. Ioannou, C. Niederée, P. Fankhauser, Efficient entity resolution for large heterogeneous information spaces, in: Proceedings of the fourth ACM international conference on Web search and data mining, ACM, 2011, pp. 535–544.

[35] G. Papadakis, G. Koutrika, T. Palpanas, W. Nejdl, Meta-blocking: Taking entity resolutionto the next level, IEEE Transactions on Knowledge and Data Engineering 26 (8) (2014) 1946–1960.

[36] G. Papadakis, E. Ioannou, T. Palpanas, C. Niederee, W. Nejdl, A blocking framework for entity resolution in highly heterogeneous information spaces, IEEE Transactions on Knowledge and Data Engineering 25 (12) (2013) 2665–2682.

[37] G. dal Bianco, M. A. Gonçalves, D. Duarte, Bloss: Effective meta-blocking with almost no effort, Information Systems 75 (2018) 75–89.

[38] A. Karakasidis, G. Koloniari, V. S. Verykios, Scalable blocking for privacy preserving record linkage, in: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2015, pp. 527–536.

[39] G. Papadakis, T. Palpanas, Blocking for large-scale entity resolution: Challenges, algorithms, and practical examples, in: Data Engineering (ICDE), 2016 IEEE 32nd International Conference on, IEEE, 2016, pp. 1436–1439.

[40] G. Papadakis, G. Papastefanatos, G. Koutrika, Supervised meta-blocking, Proceedings of the VLDB Endowment 7 (14) (2014) 1929–1940.

[41] G. Papadakis, G. Papastefanatos, T. Palpanas, M. Koubarakis, Scaling entity resolution to large, heterogeneous data with enhanced meta-blocking., in: EDBT, 2016, pp. 221–232.

[42] G. Papadakis, G. Papastefanatos, T. Palpanas, M. Koubarakis, Boosting the efficiency of large-scale entity resolution with enhanced meta-blocking, Big Data Research 6 (2016) 43–63.

[43] D. Song, J. Heflin, Scaling data linkage generation with domain-independent candidate selection, In Proceedings of the 10th International Semantic Web Conference, ISWC, 2011.

[44] D. Song, J. Heflin, Automatically generating data linkages using a domain-independent candidate selection approach, in: International Semantic Web Conference, Springer, 2011, pp. 649–664.

[45] K. O Hare, A. Jurek-Loughrey, C. de Campos, A new technique of selecting an optimal blocking method for better record linkage, Vol. 77, Information Systems Journal, ISJ, 2018, pp. 151–166.

[46] P. Christen, Febrl-: an open source data cleaning, deduplication and record linkage system with a graphical user interface, in: Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2008, pp. 1065–1068.

[47] M. Kejriwal, D. P. Miranker, Semi-supervised instance matching using boosted classifiers, in: European Semantic Web Conference, Springer, 2015, pp. 388–402.

[48] P. Christen, A survey of indexing techniques for scalable record linkage and deduplication, IEEE transactions on knowledge and data engineering 24 (9) (2012) 1537–1555.

[49] C. Dou, D. Sun, R. K. Wong, Unsupervised blocking of imbalanced datasets for record matching, in: International Conference on Web Information Systems Engineering, Springer, 2016, pp. 172–186.

[50] H. Köpcke, A. Thor, E. Rahm, Evaluation of entity resolution approaches on real-world match problems, Proceedings of the VLDB Endowment 3 (1-2) (2010) 484–493.

[51] H. Kopcke, A. Thor, E. Rahm, Learning-based approaches for matching web data entities, IEEE Internet Computing 14 (4) (2010) 23–31.

[52] T. Papenbrock, A. Heise, F. Naumann, Progressive duplicate detection, IEEE Transactions on knowledge and data engineering 27 (5) (2015) 1316–1329.