High-Value Token-Blocking: Efficient Blocking Method for Record Linkage

KEVIN O'HARE and ANNA JUREK-LOUGHREY, School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast
CASSIO DE CAMPOS, Eindhoven University of Technology

Data integration is an important component of Big Data analytics. One of the key challenges in data integration is record linkage, that is, matching records that represent the same real-world entity. Because of computational costs, methods referred to as blocking are employed as a part of the record linkage pipeline in order to reduce the number of comparisons among records. In the past decade, a range of blocking techniques have been proposed. Real-world applications require approaches that can handle heterogeneous data sources and do not rely on labelled data. We propose high-value token-blocking (HVTB), a simple and efficient approach for blocking that is unsupervised and schema-agnostic, based on a crafted use of Term Frequency-Inverse Document Frequency. We compare HVTB with multiple methods and over a range of datasets, including a novel unstructured dataset composed of titles and abstracts of scientific papers. We thoroughly discuss results in terms of accuracy, use of computational resources, and different characteristics of datasets and records. The simplicity of HVTB yields fast computations and does not harm its accuracy when compared with existing approaches. It is shown to be significantly superior to other methods, suggesting that simpler methods for blocking should be considered before resorting to more sophisticated methods.

CCS Concepts: • **Information systems → Novelty in information retrieval;**

Additional Key Words and Phrases: Blocking, record linkage, entity resolution

ACM Reference format:
Kevin O'Hare, Anna Jurek-Loughrey, and Cassio de Campos. 2021. High-Value Token-Blocking: Efficient Blocking Method for Record Linkage. *ACM Trans. Knowl. Discov. Data.* 16, 2, Article 24 (July 2021), 17 pages.
https://doi.org/10.1145/3450527

1 INTRODUCTION

Society worldwide is generating more and more data giving rise to the data deluge problem. In the world of big data, data integration technology is crucial for maximising the capability of data-driven decision making. For example, analysis of the health data coming from sources such as electronic health records, drug and toxicology databases, and genomics and social media environments, is a key driver toward advancing precision medicine. As a part of the data integration

Authors' addresses: K. O'Hare and A. Jurek-Loughrey, School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, N.Ireland BT7 1NN, UK; emails: {kohare08, a.jurek}@qub.ac.uk; C. de Campos, Eindhoven University of Technology, 5600 MB, PO Box 513, Eindhoven, The Netherlands; email: c.decampos@tue.nl.

1556-4681/2021/07-ART24 $15.00
https://doi.org/10.1145/3450527

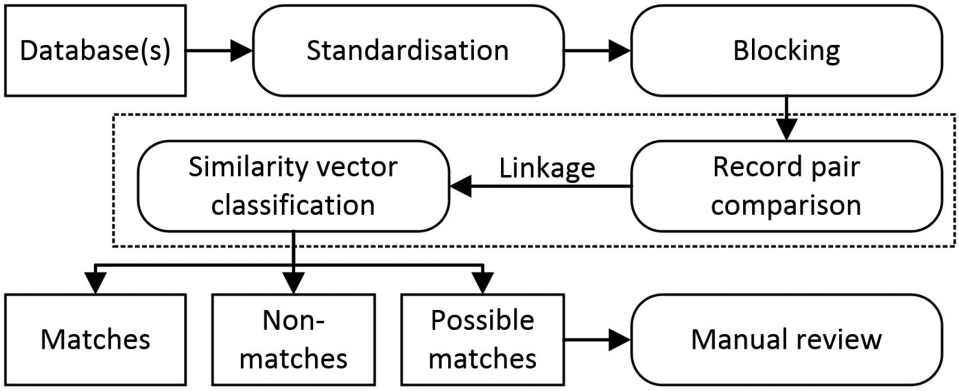

Fig. 1. General overview of the RL process.

process, records (from multiple data sources) that refer to the same real-world entity (e.g., a person) need to be linked. This process is referred to as **record linkage** (**RL**), data/record matching, or entity resolution.

An overview of a general RL process is depicted in Figure 1. Records are first standardised in order to remove inconsistencies between otherwise matching values. This may consist of removing non-alphanumeric characters or replacing multiple spelling variations of the same word with a single common spelling (e.g., "*John*", "*Johnny*", "*Jon*" → "*John*"). Record pair comparisons may then be performed with each record pair classified as either a matching, non-matching or possibly matching record pair requiring further manual review. However, naively performing RL would involve comparing every record of a data source with every other record (i.e., quadratic complexity). For even moderately sized datasets, this would result in a very large number of record pair comparisons, incurring significant cost to time and computational resources. Therefore, in order to improve efficiency, a blocking algorithm is typically applied before the linkage phase: Records are grouped in a smart manner such that matching pairs are placed together as much as possible. Subsequently, only records of a same group are compared against each other to find matching pairs.

Many existing blocking algorithms make different assumptions regarding data sources. Commonly they assume that the data are structured or that some amount of labelled data (i.e., with known matching status of each record pair) are available. Given the ever increasing amount of heterogeneous data sources, these assumptions are rather unrealistic. To address this issue some schema-agnostic blocking methods have been proposed. However, they tend to be inefficient with respect to run time and/or memory, and require tuning of parameters for good performance. For any blocking method that requires tuning of parameter values, one may need to perform potentially multiple evaluations upon a set of labelled data representative of the target dataset. Such labelled data are often not readily available.

We propose a simple unsupervised blocking approach that is very efficient in time and memory use, does not require fine tuning of parameters and it is easy to implement. In addition, such a simple approach has the further advantages of being unsupervised and being applicable to both structured and unstructured data. The contribution is not restricted to creating yet another blocking method, but to highlight that simple and thoughtful approaches shall be fully considered before creating more and more sophisticated methods. The evaluations show that this simple approach performs well against a range of competitors from the literature, including, schema/non-schema

agnostic and supervised/unsupervised methods. As a by-product, we have created a novel unstructured dataset, which we have made publicly available.

2 RELATED WORK

In a RL task we are given (one or more) datasets with (possibly unstructured) records where some records may refer to the same entity, that is, they are matching records. For simplicity, we will assume in the discussion that a single dataset is given (the case of multiple datasets can be tackled by an analogous approach). The goal of blocking is to cluster the records in small groups such that matching record pairs are placed in the same group. This is a hard task, as the blocking methods need to maximise the number of matching pairs whose records fall in the same group while minimising the sizes of the groups, thus allowing for fewer comparisons between records in the later stage when records of a same group are checked against each other. This brings the overall complexity of finding the matching record pairs from quadratic in the total number of records to quadratic only in the largest group size (while linear in the total number of records).

Existing blocking approaches are designed for either structured, semi-structured or unstructured data. Structured blocking scheme learning approaches use labelled data to evaluate potential individual blocking predicates and/or propositional combinations of them. Depending on the approach, they either assume that labelled data are available (supervised) [3, 26], or they apply an automatic method for data labelling (unsupervised) [20, 22]. The efficiency and efficacy of blocking scheme learning approaches are dependent upon the number and choice of predicates considered, and the quality and quantity of labelled data used. **Sorted Neighbourhood (SoNe)** [5, 11, 36, 44] is another unsupervised approach for structured datasets in which a window of fixed size is slid over records ordered by a key and only records that lie within the window form record pairs for comparison. Like the previous approaches SoNe also requires appropriate parameter selection for good performance, namely the choice of sorting key and size of sliding window.

Some blocking approaches have been proposed for semi-structured data. In [31], authors combine Token-Blocking with *Attribute Clustering* in order to improve its efficiency. Attribute clustering is a form of schema matching in which a similarity value is assigned to attribute column pairs. Highly similar attribute columns are clustered together and thought of as referring to the same value. Token-Blocking is then performed in a manner so that only records that share a common token by clustered attributes are grouped together. Token-Blocking is explained again later on.

Schema-agnostic blocking methods may be applied to any type of data regardless of structure. ***Canopy Clustering (CaCl)*** [2, 3, 19, 21, 25] groups records using a computationally inexpensive (potentially schema-agnostic) distance metric with records being placed in the group of chosen (potentially at random) centroids. The performance of CaCl is heavily dependent upon appropriate selection of parameters such as the distance metric and similarity threshold values for forming canopies. Overly low threshold values result in a small number of large clusters, failing to reduce the number of record pairs sufficiently. Whereas overly high similarity threshold values result in many small clusters that fail to form many of the matching record pairs of a dataset.

Another type of schema-agnostic blocking methods are **Locality Sensitive Hashing (LSH)** techniques [10, 12, 15, 23, 29, 40], including the prominent *MinHash-LSH* [6, 18, 24, 37, 42]. LSH blocking methods form k-shingles (i.e., sub-strings of length k) for each record, then form a boolean matrix using the universal set of all k-shingles to indicate which shingles are present in each record. Hash functions may then be used to identify records that share the same k-shingles and place them in the same block for comparison. Alternatively, MinHash-LSH forms MinHash signatures for each record by randomly permuting the rows of the boolean matrix, and identifying the index position of the first one value of each record. As only identical records would have identical MinHash signatures, the MinHash signatures are partitioned into bands, with each record placed into a

block for each respective band. MinHash-LSH removes the need for manual specification of any hash functions, but is still highly sensitive to parameter choices such as the choice of k, MinHash signature length and bandwidth for partitioning.

Another commonly used schema-agnostic blocking approach is *Token-Blocking*. Token-Blocking places records into blocks for each of their respective tokens with records that share common tokens being placed into the same blocks. Token-Blocking tends to capture most matching record pairs, but often fails to reduce the number of comparisons sufficiently due to the high commonality of some tokens e.g., *"the"* [14].

Some blocking methods generate block collections that contain many of the matching record pairs, but fail to reduce the number of record pair comparisons sufficiently. This can be addressed by using Block-Refinement techniques. *Block-Purging* [28] removes blocks above a specified maximum block size limit of a block collection. *Block-Filtering* [34] removes records from a proportion of their largest sized blocks. Both require appropriate parameters for good performance. Recently, it has been proposed to improve schema-agnostic token-blocking using *Meta-Blocking* [7, 31, 33, 35, 38, 39], which computes an inexpensive weight for each record pair relation and discards weak ones. Meta-Blocking frameworks model the record to record relations of a block collection as nodes (i.e., records), connected by weighted edges (i.e., relations) with respective edge weight values determined by an inexpensive weighting function. The weaker record pairs (indicated by a low edge weight value) are then pruned. In [32], five different weighting schemes are proposed. In [31, 34], weight based pruning tended to retain most matching record pairs but not reduce the comparisons sufficiently, with the opposite being true for cardinality based pruning. Although some Meta-Blocking frameworks are shown to work well for some datasets, there is little consensus as how to determine which framework should be used for any particular new dataset. In *BLAST* [37], Meta-Blocking is combined with Attribute Clustering [31] in order to avoid pairing records that share tokens by different attributes. In *BLOSS* [7], the non-redundant record pairs of a block collection are accurately labelled by banding them by their similarity according to a defined metric.

The overall goal of blocking is to reduce the search space of the RL process through reduction of the number of record comparison. The candidate record pairs provided as an output of the blocking process are further compared with a selected RL algorithm and determined as match or non-match based on their similarity. The existing RL approaches can be broadly divided into two categories. The first category of approaches rely on applying generic rules and different similarity measures to identify those pairs of records that are similar enough to be considered as matching [13, 41]. The second category of approaches relies on the application of Machine Learning algorithms [9]. With structured data, each pair of records is represented as a comparison vector representing a set of similarities, each calculated with a similarity measure on the corresponding pair of attributes values of the two vectors. Following the construction of the comparison vectors, a Machine Learning models can be trained to classify each vector (i.e., pair of records) as match or non-match. RL can also be performed in unsupervised or semi-supervised setting using techniques such as self-learning [16] or active learning [43]. Recently, Deep Learning based approaches applied with different text embedding models have been proposed for linking structured [8] and unstructured records [17].

3 HIGH-VALUE TOKEN-BLOCKING (HVTB)

The proposed **High-Value Token-Blocking** (**HVTB**) algorithm (shown in detail in Algorithm 1) is inspired by, and based on, the well-known **Term Frequency-Inverse Document Frequency** (**TF-IDF**) and its ability to identify *high-value* tokens. TF-IDF values indicate the significance of a token based on their commonality within a dataset, with lower values indicating highly frequent

insignificant tokens, and higher values indicating rarer significant tokens. In [20, 22], it is used to automatically generate labelled data to then rank individual blocking predicates with. The best are then used to form standard blocking schemes for structured and semi-structured datasets in an automatic and unsupervised manner. In [22], schema matching is additionally used so that RL may be performed between datasets with differing schemas. By contrast, the proposed HVTB algorithm uses TF-IDF to block the records of a dataset in a manner comparable to that of the highly efficient Token-Blocking but only using the most significant tokens of each record. As such, it is completely schema-agnostic in manner and can be applied efficiently to any dataset regardless of structure.

It consists of the following steps. First, tokens of each record are standardised and inverted indices are updated: one for the tokens of each record and another for the document frequencies of each token. Then, TF-IDF is calculated for each distinct token of each record together with the average TF-IDF values ($TF\text{-}IDF_{r_{avg}}$) of all non-unique tokens (i.e., those with a document-frequency greater than one) of each record.

In order to discriminate between the least and most discriminative tokens of each record, we use the implicitly defined $TF\text{-}IDF_{r_{avg}}$ value for each record as a threshold. Unique tokens tend to have relatively high TF-IDF values due to their rarity but inherently cannot form any record pairs making them redundant. As the inclusion of unique tokens would skew this threshold for each record, we therefore do not include them when calculating $TF\text{-}IDF_{r_{avg}}$. Finally, each record is assigned to a respective block for each of its tokens with a TF-IDF value greater than $TF\text{-}IDF_{r_{avg}}$, i.e., their high-value tokens.

A key property of the approach is to perform a similar but much more efficient alternative to Meta-Blocking in which we index each record pair of this already efficient block collection by their number of common blocks, to then prune those record pairs with a below average number of common blocks. This similarity to existing Meta-Blocking approaches lies in that all individual record pairs of a block collection are weighted by the number of common blocks of each record pair [7, 31, 33, 35, 38, 39]. However, instead of forming an inter-connected graph of all their relations, we only form a simple index that uses much less memory. The chosen weighting also allows us to implement what is referred to as the *Least Common Block* criterion [1, 30], which avoids redundant record pair comparisons from being performed at negligible additional cost. For each record pair of a block collection, the list of blocks for each record are first inspected. If the first (lexicographically speaking) common token of both lists corresponds to that of the currently inspected block of the block collection, then and only then is that record pair indexed. Therefore, even if the same record pair is placed in multiple different blocks of a block collection, it is only inspected once when the least common block criterion is met, and ignored in all other cases.

Indexing record pairs by their number of common tokens also allows for a much more efficient alternative to pruning than is used in related Meta-Blocking literature. Meta-Blocking approaches inspect the individual edge-weight values of a blocking graph, allowing the respective record pair of those that exceed a minimum threshold value to be compared. As our record pairs have already been indexed by their number of common tokens, we instead only consider for linkage those indexed record pairs with an above threshold number of common tokens. For our threshold, we select the average number of common blocks of all indexed record pairs. In addition to being implicitly defined, this value can also be calculated instantly using the index. In summary, HVTB is simple and combines the best of token-blocking, block-refinement techniques and Meta-Blocking but with much less computational demand and need for fine tuning of explicit parameter values. By first removing many of the less significant tokens of each record prior to token-blocking, an efficiently structured block collection is initially generated. This is in contrast to an inefficient block collection being first generated (as per normal token-blocking) and then improved via block-refinement techniques. Secondly, as the set of tokens (i.e., blocks) associated with each record has

ALGORITHM 1: High Value Token Blocking (HVTB)

Input: n records: $\mathbf{D} = r_1, \ldots, r_n$
Output: Set of record pairs, $\{r_i, r_j\}, \ldots, \{r_{i'}, r_{j'}\}$

```
// Tokenize records
Create index of tokens of each record, T = {}
Create index of document frequencies, F = {}
foreach record r ∈ D do
    Tokenise r as a set of standardised tokens, T_r
    Add T_r to T
    // Update document frequencies
    foreach distinct token t ∈ T_r do
        if document frequency of t (f_t) ∈ F then
            increment f_t by 1
        else
            assign f_t = 1 and add to F
// Calculate TF-IDF values
foreach record r ∈ D do
    foreach distinct token t ∈ T_r do
        calculate TF-IDF value of t
    Compute average non-unique token TF-IDF value, TF-IDF_ravg
// Block records by their high value tokens
Create empty block collection, B = {}
foreach record r ∈ D do
    foreach distinct token t ∈ T_r do
        if TF-IDF_{t∈T_r} > TF-IDF_ravg then
            if respective block of t (B_t) ∈ B then
                add r to B_t
            else
                assign B_t = {r} and add to B
// Improve block collection as an index
Create index of weights for record pairs, W = {}
foreach record pair {r_i, r_j} ∈ B do
    calculate w_{i,j} = |T_i ∩ T_j| and add to W
Calculate average weight value in W as w_Avg
foreach record pair {r_i, r_j} ∈ W do
    if w_{i,j} < w_Avg then
        remove {r_i, r_j} from W
Return record pairs of W, {r_i, r_j}, ..., {r_i', r_j'}
```

been reduced, finding the number of common blocks of each record pair of a block collection is made much more efficient than normal. Furthermore, as only the more significant tokens of each record are retained, for two records to share a common token is more indicative of a matching record pair than normal. Finally, by indexing the record pairs by their number of common blocks, rather than graphing their relations via a more computationally demanding weighting function, we can easily and instantly prune those with a below threshold number of common blocks. This is in contrast to Meta-Blocking approaches in which the edge-weight value of each record pair must be individually inspected for comparison.

Algorithm 1 has clearly a very low computational complexity. The loop of lines 3–10 spends linear time in the dataset size, as it runs over all tokens of all records updating a frequency table. The loop of lines 11–14 has the same linear complexity if the appropriate counts are kept in order to obtain TF-IDF. Lines 16–22 summarise the block information again by a single pass over the dataset. If an efficient dictionary data structure is used, this again can be done in (amortised) linear time in the dataset. Lines 24–25 and 27–29 clearly spend time that is asymptotically bounded by the output size (that is, number of returned record pairs). Therefore, the overall asymptotic complexity of HVTB is linear in the dataset size and in the number of pairs it proposes. This is an optimal complexity, as no algorithm can be faster than that (any algorithm needs to at least read its input, which already spends linear time in the input size, and return its output, which is linear time in the output size). Obviously, the algorithm can be slow if it yields too many record pairs, but empirically we will see that this is not the case.

4 EXPERIMENTAL SETUP

We evaluate blocking results using ***reduction ratio*** (RR) (1 minus the number of returned record pairs divided by the total number of pairs), ***pair completeness*** (PC) (overall percentage of returned matching pairs divided by the total number of matching pairs) and $F_{RR,PC}$ (the harmonic mean of them). We additionally present time and memory usage of each blocking approach to better understand the required resources. All algorithms were coded using Java Eclipse Mars.1 and run using a Dell Optiplex 9020 with 16 GB of RAM, an Intel(R) Core(TM) i7-4790 with 3.60 GHz and 64x Windows 7 Enterprise for fairness of comparisons.

We compare HVTB to a number of competitors. First are the unsupervised blocking approaches SoNe, CaCl, *MinHash-LSH* (MinH), the unsupervised blocking scheme learning approach named simply (C) of [20] as well as *BLAST* [37] and the Meta-Blocking frameworks (referred to as Meta) described in [32, 34]. We also include two supervised blocking scheme learning approaches (A and B) [3, 26] that assume labelled data are available (while these have limited applicability in unlabelled settings, it is worth to run a comparison to understand the differences in performance).

For all competing methods we use the same default parameter values of their seminal papers. For those that require training for parameter choices (i.e., *CaCl*, *MinH*, *Meta*, *A*, *B*, *C*) we use the optimal parameter values found for a respective training set consisting of 50% of the original dataset. Optimal in this context refers to values that are found to achieve the highest harmonic mean of precision and recall by TF-IDF linkage in the least amount of time [27]. For the SoNe approach, window size is set to 20 and records are sorted by each attribute lexicographically. For CaCl optimal parameter values are found for each dataset by varying the lower and upper threshold parameter values from $\{0.00 \rightarrow 1.00\}$ in increments of 0.01 for the respective training set. For *MinHash-LSH* optimal parameter values are found for each dataset by setting shingle length to 2, varying MinHash signature length from $\{20 \rightarrow 2{,}000\}$ in increments of 20, and setting band width to 5 for the respective training set. For the unsupervised blocking scheme learning approach of [20], we use the same default values for the automatic labelling algorithm as described in the same article to generate the necessary labelled data. For *BLAST* we use the same default values for Block-filtering and Block-purging as described in [37]; however, as all of the datasets used in the experiments share the same respective schemas, there is no need for Attribute-Clustering. For the Meta-Blocking frameworks, we use the same 5 weighting schemes, and 4 pruning schemes of [32, 34], forming 20 different Meta-Blocking baselines. We enhance these frameworks with Block-filtering and Graph-partitioning using the same default values as described in [34]. For the standard blocking scheme learning baselines (A, B, and C), we evaluate using ten-fold cross-validation and restrict conjunctions to a maximum length of 2, as higher values typically result in longer evaluation times with little or no gain to performance. Out of the

Table 1. Properties Table for Evaluated Datasets

Name	Number of Attributes	Number of Records	Number of Matches
Restaurant	5	864	112
Cora	4	1,295	17,184
Clean-Synth	10	10,000	2,000
Dirty-Synth	9	10,000	26,692
Discs25000	7	25,000	1,007
DBLP-ACM	4	2,616 + 2,294	2,224
Amazon-Google	4	1,362 + 3,225	1,300
DBLP-Scholar	4	2,616 + 64,263	5,347
Abt-Buy	4	1,081 + 1,092	1,097
Abstracts-Titles	1	6,704 + 6,704	6,704
Walmart-Amazon	10	2,554 + 22,074	1,154

eight baseline methods, SoNe, A, B, and C were designed for structured data (i.e., represented by pre-defined attributes), while CaCl, MinH, Meta, and Blast were presented as schema-agnostic methods in their original papers. It is clear that all those approaches have more intricate setups than HVTB.

Existing Datasets: In the experimental evaluation we used 10 existing datasets, including 8 real-world datasets, which are commonly applied by the RL community and two synthetically generated datasets. Five of the datasets (Restaurant, Cora, Discs25000, Clean-Synth, and Dirty-Synth) are used for de-duplication (i.e., matching pairs exist within a single dataset), and five (DBLP-ACM, Amazon-Google, DBLP-Scholar, Abt-Buy, and Walmart-Amazon) are across multiple datasets. To generate the synthetic datasets (Clean-Synth and Dirty-Synth), we use a modified version of the synthetic data generator defined by Christen [4]. In Table 1, we indicate different properties of each dataset such as their number of attributes, total number of records and total number of matching record pairs.

New Dataset: The majority of the publicly available RL datasets are structured. To provide better benchmark for schema-agnostic blocking techniques, as a part of this work we constructed an unstructured dataset which is composed of titles and abstracts of medical scientific articles. The linking task involves identifying pairs of titles and abstract that refer to the same publication. The dataset was obtained from the **PubMed Central** (**PMC**) archive using PMC ESearch API for collecting IDs of full text research articles and EFetch API for obtaining the full text content for each article ID. For each article we extracted the title and abstract. The dataset contains 6,704 matching record pairs and it has been made publicly available.[1]

5 RESULTS AND DISCUSSIONS

In this section, we present results obtained in the experimental evaluation of the proposed blocking methods and the baseline techniques. We evaluated all the methods with respect to the quality of the output block collections, performance of RL methods on the obtained blocks and the efficiency.

[1] https://github.com/DrKevinOHare/Titles-Abstracts-dataset.

Table 2. RR Value of Each Baseline for Each Dataset

Dataset	HVTB	SoNe	CaCl	MinH	Meta	BLAST	A	B	C
Restaurant	0.993	0.772	0.999	0.984	0.990	0.981	0.990	0.728	0.991
Cora	0.966	0.884	0.926	0.574	0.986	0.976	0.963	0.964	0.967
Clean-Synth	1.000	0.960	0.995	0.991	0.998	0.997	1.000	0.999	1.000
Dirty-Synth	0.999	0.964	0.993	0.966	0.993	0.996	0.999	0.999	0.999
Discs25000	1.000	0.990	0.929	0.999	1.000	0.997	0.999	1.000	0.999
DBLP-ACM	0.997	0.939	0.824	0.953	0.994	0.677	0.998	1.000	0.999
Amazon-Google	0.978	0.918	0.997	0.439	0.815	0.572	0.996	0.990	0.998
DBLP-Scholar	0.999	0.969	0.699	0.926	1.000	0.797	0.999	0.999	0.999
Abt-Buy	0.988	0.891	0.619	0.800	0.982	0.813	0.929	0.968	0.950
Abstracts-Titles	0.998	0.994	1.000	0.771	0.994	0.854	0.905	0.993	0.028
Walmart-Amazon	0.887	0.913	0.941	0.309	0.995	0.378	0.993	0.994	0.992

Table 3. PC value of each baseline for each dataset

Dataset	HVTB	SoNe	CaCl	MinH	Meta	BLAST	A	B	C
Restaurant	1.000	1.000	0.973	0.955	1.000	1.000	1.000	0.998	1.000
Cora	0.940	0.867	0.940	0.969	0.615	0.881	0.921	0.919	0.916
Clean-Synth	1.000	1.000	0.998	1.000	1.000	1.000	1.000	1.000	1.000
Dirty-Synth	1.000	1.000	0.998	0.999	1.000	1.000	1.000	1.000	1.000
Discs25000	0.946	0.997	0.775	0.886	0.977	0.998	0.827	0.774	0.950
DBLP-ACM	0.996	0.990	0.989	0.967	0.998	1.000	0.977	0.825	0.987
Amazon-Google	0.945	0.632	0.719	0.785	0.934	0.999	0.825	0.812	0.724
DBLP-Scholar	0.993	0.936	0.972	0.987	0.977	1.000	0.946	0.972	0.976
Abt-Buy	0.920	0.772	0.965	0.781	0.935	0.997	0.926	0.966	0.982
Abstracts-Titles	0.947	0.071	0.810	0.453	0.962	0.997	0.930	0.912	1.000
Walmart-Amazon	0.999	0.913	0.860	0.913	0.991	0.999	0.818	0.863	0.964

5.1 Blocking Quality

In Tables 2–4, we present blocking results in terms of RR, PC, and $F_{RR,PC}$. In these tables, *Meta* refers to the best individual results of the 20 different Meta-Blocking frameworks of [32, 34] (hence such meta-approach could have an unfair advantage because of that). All methods' details are presented in Section 4. We observe that HVTB achieves the overall best $F_{RR,PC}$ for most datasets. It was marginally outperformed by Meta, B and C on Abt-Buy, and by Meta on Abstracts-Titles. For Walmart-Amazon, HVTB was outrun by Meta and C and for Discs25000, it was beaten by SoNe, Meta, BLAST, and C. It is worth noting that even though HVTB did not always win, the difference in performance ($F_{RR,PC}$) was rather small (between 0.002 and 0.038) in those losing cases (apart from Meta on Walmart-Amazon). We can also see that HVTB performs consistently well across all the datasets while every other method obtains relatively low result for some datasets. For example, Meta obtained $F_{RR,PC}$ below 0.8 for Cora and below 0.9 for Amazon-Google. HVTB also performed consistently well in terms of *RR* and *PC*. It can also be observed from Table 4 that HVTB and Meta significantly outperformed any of the other techniques on the new Abstracts-Titles dataset, which is the only formally unstructured dataset under comparison. Methods such as SoNe, MinH, and C failed to provide a good performance.

Table 4. $F_{RR,PC}$ value of each baseline for each dataset

Dataset	HVTB	SoNe	CaCl	MinH	Meta	BLAST	A	B	C
Restaurant	0.996	0.871	0.986	0.970	0.995	0.990	0.995	0.835	0.996
Cora	0.953	0.876	0.933	0.721	0.757	0.926	0.941	0.941	0.941
Clean-Synth	1.000	0.980	0.997	0.995	0.999	0.999	1.000	1.000	1.000
Dirty-Synth	1.000	0.982	0.996	0.982	0.996	0.998	1.000	1.000	1.000
Discs25000	0.972	0.994	0.845	0.939	0.988	0.998	0.905	0.872	0.974
DBLP-ACM	0.997	0.964	0.899	0.960	0.996	0.807	0.987	0.903	0.993
Amazon-Google	0.961	0.749	0.836	0.563	0.870	0.727	0.903	0.890	0.839
DBLP-Scholar	0.996	0.952	0.813	0.955	0.988	0.887	0.972	0.985	0.988
Abt-Buy	0.953	0.827	0.754	0.790	0.958	0.895	0.926	0.967	0.966
Abstracts-Titles	0.972	0.132	0.895	0.571	0.978	0.920	0.914	0.951	0.054
Walmart-Amazon	0.940	0.913	0.899	0.462	0.993	0.548	0.896	0.918	0.978

Table 5. Hypothesis Testing on $F_{RR,PC}$ Results Between HVTB and Competitors Over 11 Datasets Using Wilcoxon Signed Rank Test

Method	p-value	Median	95% C.I. for Median
SoNe	0.005	0.074	$[0.023, 0.409]$
CaCl	0.001	0.075	$[0.022, 0.130]$
MinH	0.001	0.150	$[0.028, 0.282]$
Meta	0.688	0.001	$[-0.011, 0.090]$
BLAST	0.007	0.074	$[0.006, 0.197]$
A	0.009	0.034	$[0.012, 0.058]$
B	0.024	0.053	$[0.011, 0.097]$
C	0.441	0.008	$[-0.017, 0.463]$

Estimated median value (and 95% confidence interval C.I.) of $F_{RR,PC}$(HVTB) − $F_{RR,PC}$(competitor) (hence positive means HVTB is superior).

In order to reach conclusions, we have run a non-parametric hypothesis testing using Wilcoxon signed-rank test using the results in terms of $F_{RR,PC}$ for each method and dataset. Results in Table 5 show that HVTB is superior to each one of the competitors (the median of $F_{RR,PC}$(HVTB)−$F_{RR,PC}$(competitor) is always positive), while such difference is statistically significant for all but Meta and C (even in those two comparisons, the estimated 95% confidence interval of the median is mostly on the positive side). This suggests that most sophisticated approaches fail to improve on a simple idea such as HVTB, actually being significantly inferior to it.

5.2 Efficiency and Scalability

This section reports about run time and memory consumption for each of the blocking methods. First, we present run time values over the 11 already discussed datasets. Figure 2 shows that HVTB has one of the lowest blocking run time values among all approaches. The only consistent exception is SoNe, which has equal or marginally lower time than HVTB in all cases and CaCl in a single case.

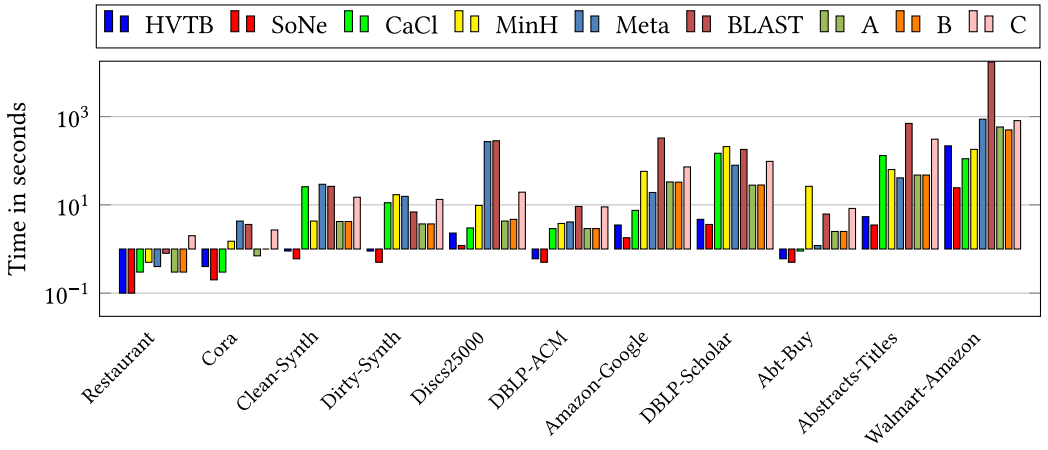

Fig. 2. Blocking run time of each blocking method for each dataset.

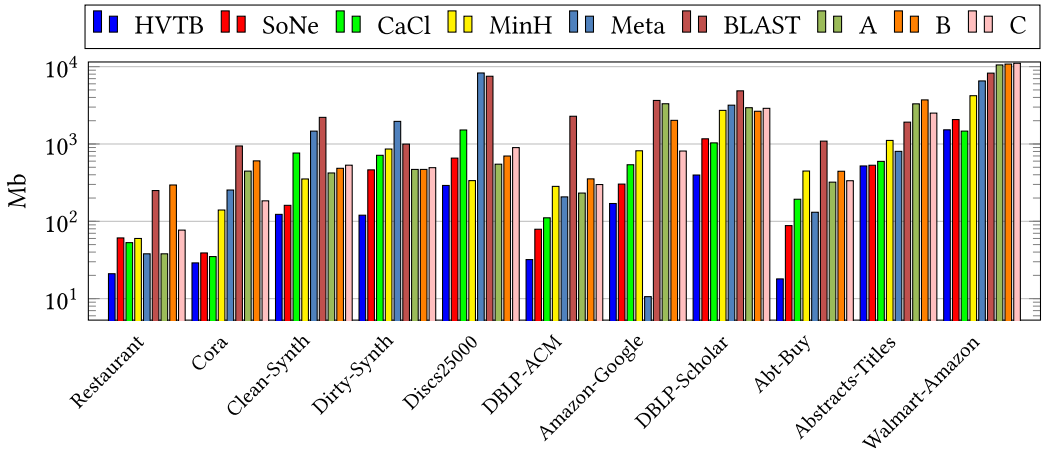

Fig. 3. Memory consumption (in Mb) of each baseline for each dataset.

Looking to the memory consumption of each blocking method for each dataset (Figure 3), we observe that HVTB is the least memory demanding of the evaluated approaches (often by a considerable margin). This low memory cost relates to the fact that HVTB reduces the sets of tokens for each record prior to the token-blocking phase, as well as not having to retain in memory the record to record relations of a block collection (as per Meta-Blocking). Some of the best competitors are observed to have quite significant memory footprints.

In Figures 4 and 5, we, respectively, present the time (in seconds) and memory consumption (in Mb) for the different blocking methods for varying dataset sizes. We generate datasets of different sizes up to 1 Million records using the same synthetic data generator used to generate Clean-Synth and Dirty-Synth. Note that a discontinued line indicates that the corresponding blocking method was incapable of completing for larger dataset sizes under our experimental setup limitations (in particular 16 Gb of RAM). For the largest evaluated dataset of 1 million records, HVTB was capable of running in under 30 seconds, while the next quickest approach (SoNe) required nearly 25 minutes to perform blocking. Most notably, Meta and BLAST are most similar in concept to HVTB

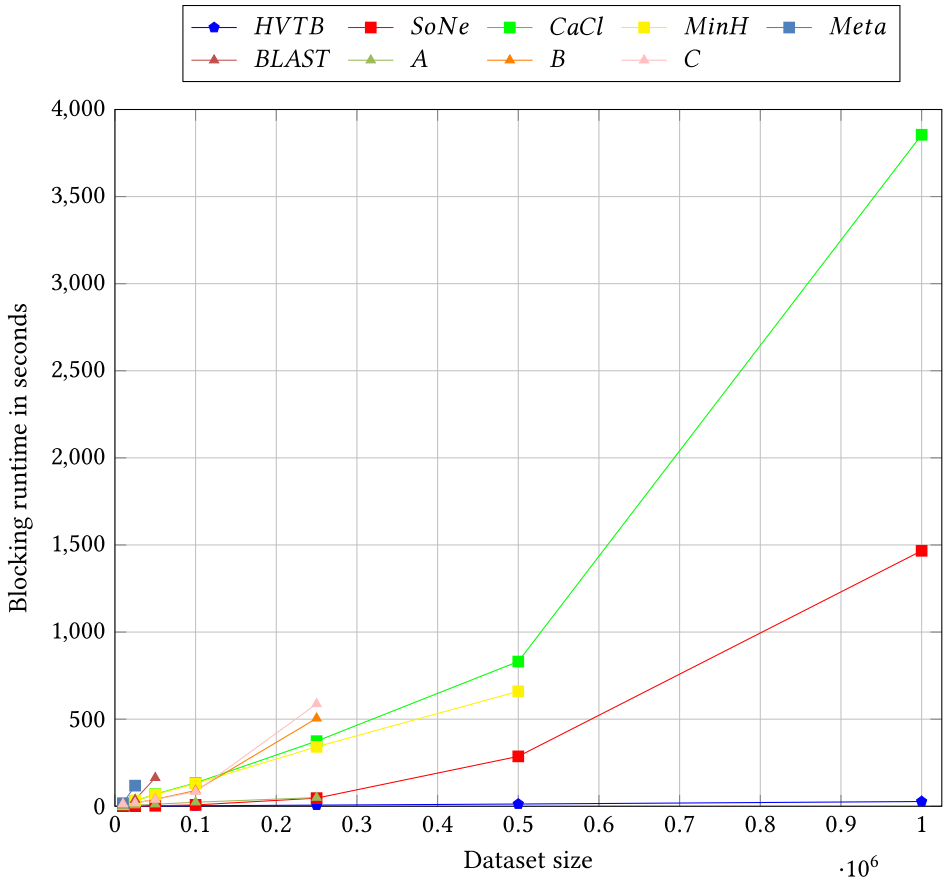

Fig. 4. Blocking run time (in seconds) of different approaches using a synthetic dataset of increasing size. Some curves stop early because the corresponding method was not able to run beyond such a dataset size.

but are much less efficient. A similar situation is observed in Figure 5 about the memory usage. We also see that HVTB has a linear increase, indicating that one may be able to accurately predict the necessary memory for an especially large dataset.

5.3 Blocking Diversity

We further analyse how HVTB differ from competitiors. First, we check how much the methods overlap in terms of the true matches captured during blocking. In other words, we wanted to see whether the methods tend to make the same mistakes or perhaps they outperform each other on distinct subgroups of records. Table 6 shows the number of true matches that were placed in the same blocks by HVTB but were missed by the other baseline methods and vice versa. In this analysis, we only included the unsupervised techniques (i.e., SoNe, CaCl, MinH, Meta, and BLAST) that do not require cross-validation. There is very little or no disagreement for Restaurant, Clean-Synth, Dirty-Synth, and DBLP-ACM datasets. This can be explained by the fact that majority of the methods performed well with those four datasets and a small number of mistakes have been made overall.

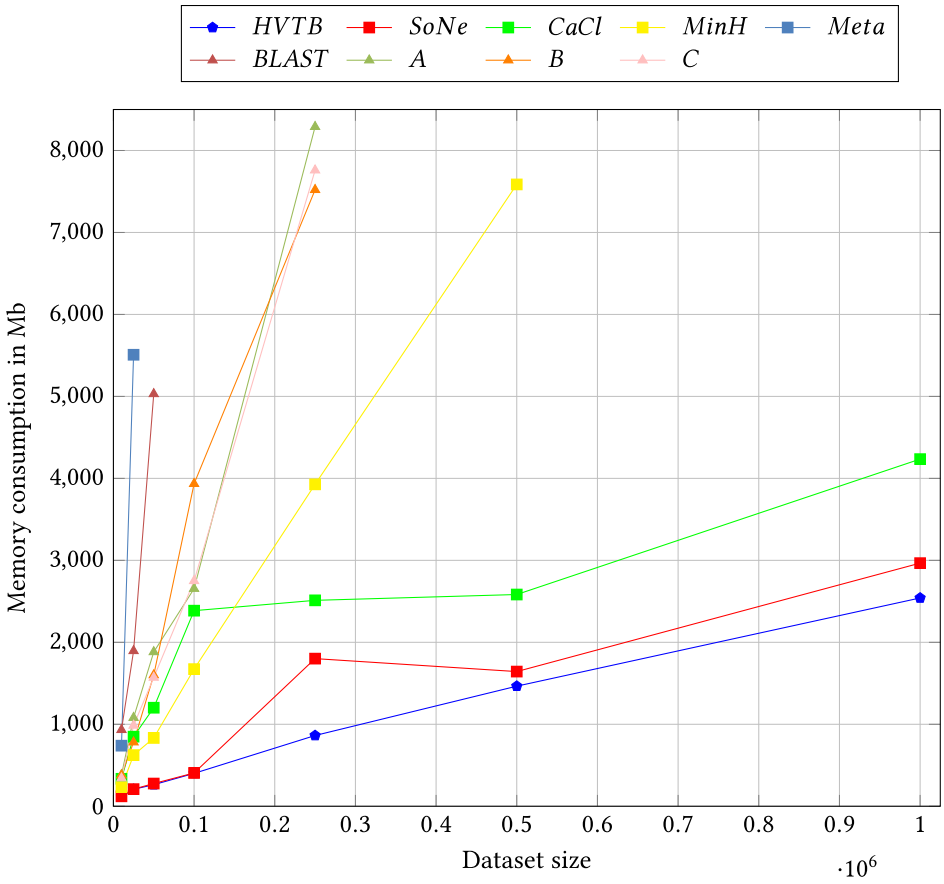

Fig. 5. Memory consumption (in Mb) of different approaches using a synthetic dataset of increasing size. Some curves stop early because the corresponding method was not able to run beyond such a dataset size.

The largest disagreements between HVTB and others can be observed for the Cora, Discs25000, Amazno-Google, Abt-Buy, and Abstract-Titles datasets. The large disagreement might simply reflect the fact that HVTB performed better than others. In some cases, however, the disagreement is high even though both methods performed equally well. This can be observed, for example, for HVTB versus Meta on Abt-Buy and Abstract-Titles datasets. This suggests that these methods make different mistakes, which may indicate that they are capable of detecting distinct subgroups of matching records pairs.

On Abstract-Titles, both HVTB and Meta performed best when record pairs had larger number of words (both with p-value $< 10^{-6}$) and larger number of unique characters (both with p-value $< 10^{-5}$). The two methods performed differently when there was more long words (p-value $< 10^{-6}$) and fewer short words (p-value $< 10^{-3}$). Over the records that they disagreed, HVTB performed best when there were more short words (p-value 0.041). On Abt-Buy, HVTB performed best when record pairs had more letters (p-value < 0.005), more long words (p-value $< 10^{-3}$), and fewer short words (p-value 0.001), while Meta did best when there were fewer numbers (p-value $< 10^{-4}$), fewer characters (p-value < 0.001), fewer long words (p-value $< 10^{-4}$), and fewer short words (p-value $< 10^{-4}$). They differed in prediction over record pairs with fewer short words (p-value < 0.01).

Table 6. Percentage Disagreement on True Matching Record Pairs Between HVTB and the Baseline Methods for Each of the Datasets

Dataset	BLAST	CaCl	Meta	MinH	SoNe
Restaurant	0	3	0	4	0
Cora	8	5	33	5	14
Clean-Synth	0	0	0	0	0
Dirty-Synth	0	0	0	0	0
Discs25000	5	26	5	14	5
DBLP-ACM	0	1	0	3	1
Amazon-Google	5	23	10	22	37
DBLP-Scholar	1	3	2	2	7
Abt-Buy	8	10	9	24	24
Abstracts-Titles	5	16	5	52	89
AmazonWalmart	9	14	1	9	9

Table 7. Association Between Methods and Record-Pair Characteristics that Are Repeatedly Significant (Mann–Whitney p-value < 0.05) in at Least 6 of the 11 Employed Datasets

Characteristic	HVTB	BLAST	CaCl	Meta	MinH	SoNe
Characters	–	–	–	–	fewer	–
Numbers	difference	–	difference	–	fewer	–
Non-numbers	difference	–	difference	difference	more	–
Short words (len.<4)	fewer	–	fewer	difference	fewer	fewer
Long words (len.>8)	more	–	more	–	more	difference

They show when a characteristic is prominently different between correctly and incorrectly blocked matching pairs (hence the test is run within each dataset and considers all record pairs that should have been matched). *Difference* means that the method indicated in the column performed better when there was a difference in that characteristic (described in the row) between compared records in the pair, while the adjectives *fewer* and *more* regard both records of the pair.

Over the records that they disagreed, HVTB performed best when there were more numbers (p-value $< 10^{-6}$), more characters (p-value $< 10^{-4}$), and more long words (p-value $< 10^{-4}$). All these hypothesis tests were performed using the non-parametric Wilcoxon (Mann–Whitney) rank sum test.

These analyses between HVTB and Meta on Abstract-Titles and Abt-Buy may help us to understand the differences in performance in these particular cases, but it is hard to generalise the results by looking to particular datasets. With that more general goal in mind, we have run the non-parametric Wilcoxon (Mann–Whitney) rank sum test to identify characteristics associated with each of the methods based on when the particular method has correctly blocked together matching record pairs. The hypothesis test is run for each blocking method within each dataset. In order to consolidate results, Table 7 reports only the characteristics that were recurrent in more than half of the datasets (with high statistical significance), so they likely represent characteristics of the methods that might generalise across datasets. In order to clarify the meaning of the table, let us consider the row *Numbers* and the column for method CaCl. Table 7 reports that, among all record pairs that should have been matched, CaCl performs best in blocking those pairs correctly when there is a *difference* in the amount of "numerical words" among the pairs to be matched. Let us consider the row *Short words* and column MinH. Here, the table shows that MinH performs better in blocking pairs correctly that have fewer short words (when considering all words of both

records in the pair). This also happens for HVTB, CaCl, and SoNe, as shown in the respective columns of Table 7. Results appear in the table only if they repeatedly happened in at least 6 of the 11 datasets.

6 CONCLUSIONS

This work focuses on the blocking task and proposes a simple but effective method named HVTB. Despite its simplicity, HVTB has been the fastest blocking method in the empirical study, has shown significantly less memory consumption and has always obtained good blocking results. It has been able to consistently achieve high reduction ratio of comparisons, high pair completeness, and high harmonic combination of those values. Statistical hypothesis testing shows that HVTB was superior to other methods. Based on the characteristics of the datasets, it is suggested that HVTB is more suitable for datasets that are unstructured or have attributes represented as lengthy unstructured text. While, some of the existing methods do perform well in some particular cases, these experiments suggest that a simple approach such as HVTB should be strongly considered before resorting to more sophisticated methods.

Many of the competitors are schema-aware/dependable, and have a hidden cost of time and effort for sourcing of labelled data and tuning of parameter values (typically requiring multiple evaluations). HVTB has performed consistently well despite being unsupervised, schema-agnostic, and only making use of implicitly defined parameter values, therefore avoiding the need for a domain expert.

In future work, we would like to explore whether the proposed blocking method can be even further improved in terms of automatic selection of linkage parameter values based on distribution of record pair similarities of block collections, as this would allow for an effective end-to-end RL solution that may be applied to any dataset and without the need for labelled data or domain expertise.

REFERENCES

[1] Tiago Brasileiro Araújo, Carlos Eduardo Santos Pires, and Thiago Pereira da Nóbrega. 2017. Spark-based streamlined metablocking. In *Proceedings of the 2017 IEEE Symposium on Computers and Communications*. IEEE, 844–850.

[2] B. Vijaya Babu and K. Jyotsna Santoshi. 2014. Unsupervised detection of duplicates in user query results using blocking. *International Journal of Computer Science and Information Technologies* 5, 3 (2014), 3514–3520.

[3] Mikhail Bilenko, Beena Kamath, and Raymond J. Mooney. 2006. Adaptive blocking: Learning to scale up record linkage. In *Proceedings of the 6th International Conference on Data Mining*. IEEE, 87–96.

[4] Peter Christen. 2008. 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, 1065–1068.

[5] Peter Christen. 2012. *Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*. Springer Science Business Media.

[6] Mingyuan Cui. 2014. *Towards a Scalable and Robust Entity Resolution-Approximate Blocking with Semantic Constraints*. Technical Report. Australian National University.

[7] Guilherme dal Bianco, Marcos André Gonçalves, and Denio Duarte. 2018. BLOSS: Effective meta-blocking with almost no effort. *Information Systems* 75 (2018), 75–89.

[8] Muhammad Ebraheem, Saravanan Thirumuruganathan, Shafiq Joty, Mourad Ouzzani, and Nan Tang. 2018. Distributed representations of tuples for entity resolution. In *Proceedings of the VLDB Endowment*. 11, 11 (2018), 1454–1467.

[9] Mohamed G. Elfeky, Vassilios S. Verykios, and Ahmed K. Elmagarmid. 2002. TAILOR: A record linkage toolbox. In *Proceedings of the18th International Conference on Data Engineering*. IEEE, 17–28.

[10] Aristides Gionis, Piotr Indyk, Rajeev Motwani, et al. 1999. Similarity search in high dimensions via hashing. In *Proceedings of the 25th International Conference on Very Large Data Bases*, Vol. 99. 518–529.

[11] Mauricio A. Hernández and Salvatore J. Stolfo. 1998. Real-world data is dirty: Data cleansing and the merge/purge problem. *Data Mining and Knowledge Discovery* 2, 1 (1998), 9–37.

[12] Ekaterini Ioannou, Odysseas Papapetrou, Dimitrios Skoutas, and Wolfgang Nejdl. 2010. Efficient semantic-aware detection of near duplicate resources. In *Proceedings of the 7th International Conference on Extended Semantic Web*. Springer, 136–150.

[13] Robert Isele and Christian Bizer. 2012. Learning expressive linkage rules using genetic programming. *Proceedings of the VLDB Endowment* 5, 11 (2012), 1638–1649.

[14] Delaram Javdani, Hossein Rahmani, Milad Allahgholi, and Fatemeh Karimkhani. 2019. Deep Block: A novel blocking approach for entity resolution using deep learning. In *Proceedings of the 5th International Conference on Web Research*.

[15] Liang Jin, Chen Li, and Sharad Mehrotra. 2003. Efficient record linkage in large data sets. In *Proceedings of the 8th International Conference on Database Systems for Advanced Applications*. IEEE, 137–146.

[16] Anna Jurek, Jun Hong, Yuan Chi, and Weiru Liu. 2017. A novel ensemble learning approach to unsupervised record linkage. *Information Systems* 71 (2017), 40–54.

[17] Anna Jurek-Loughrey. 2020. Siamese neural network for unstructured data linkage. In *Proceedings of the 22nd International Conference on Information Integration and Web-Based Applications & Services*. 417–425.

[18] Dimitrios Karapiperis and Vassilios S. Verykios. 2015. 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.

[19] Mayank Kejriwal and Daniel P. Miranker. 2014. Two-step blocking scheme learner for scalable link discovery. In *Proceedings of the 9th International Conference on Ontology Matching*.

[20] Mayank Kejriwal and Daniel P. Miranker. 2013. An unsupervised algorithm for learning blocking schemes. In *Proceedings of the 13th International Conference on Data Mining*. IEEE, 340–349.

[21] Mayank Kejriwal and Daniel P. Miranker. 2014. On linking heterogeneous dataset collections. In *Proceedings of the International Semantic Web Conference (Posters & Demos)*. Citeseer, 217–220.

[22] Mayank Kejriwal and Daniel P. Miranker. 2015. An unsupervised instance matcher for schema-free RDF data. *Web Semantics: Science, Services and Agents on the World Wide Web* 35, 2 (2015), 102–123.

[23] Hung-sik Kim and Dongwon Lee. 2010. HARRA: Fast iterative hashed record linkage for large-scale data collections. In *Proceedings of the 13th International Conference on Extending Database Technology*. ACM, 525–536.

[24] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. 2014. *Mining of massive datasets*. Cambridge university press.

[25] Andrew McCallum, Kamal Nigam, and Lyle H. Ungar. 2000. Efficient clustering of high-dimensional data sets with application to reference matching. In *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 169–178.

[26] Matthew Michelson and Craig A. Knoblock. 2006. Learning blocking schemes for record linkage. In *Proceedings of the 21st National Conference on Artificial Intelligence*. AAAI, 440–445.

[27] Kevin O. Hare, Anna Jurek, and Cassio de Campos. 2018. A new technique of selecting an optimal blocking method for better record linkage. *Information Systems Journal* 77 (2018), 151–166.

[28] George Papadakis, Ekaterini Ioannou, Claudia Niederée, and Peter Fankhauser. 2011. Efficient entity resolution for large heterogeneous information spaces. In *Proceedings of the 4th ACM International Conference on Web Search and Data Mining*. ACM, 535–544.

[29] George Papadakis, Ekaterini Ioannou, Claudia Niederée, Themis Palpanas, and Wolfgang Nejdl. 2011. Eliminating the redundancy in blocking-based entity resolution methods. In *Proceedings of the 11th Annual International ACM/IEEE Joint Conference on Digital Libraries*. ACM, 85–94.

[30] George Papadakis, Ekaterini Ioannou, Claudia Niederée, Themis Palpanas, and Wolfgang Nejdl. 2011. To compare or not to compare: making entity resolution more efficient. In *Proceedings of the International Workshop on Semantic Web Information Management*. ACM.

[31] George Papadakis, Ekaterini Ioannou, Themis Palpanas, Claudia Niederee, and Wolfgang Nejdl. 2013. A blocking framework for entity resolution in highly heterogeneous information spaces. *IEEE Transactions on Knowledge and Data Engineering* 25, 12 (2013), 2665–2682.

[32] George Papadakis, Georgia Koutrika, Themis Palpanas, and Wolfgang Nejdl. 2014. Meta-blocking: Taking entity resolution to the next level. *IEEE Transactions on Knowledge and Data Engineering* 26, 8 (2014), 1946–1960.

[33] George Papadakis and Themis Palpanas. 2016. Blocking for large-scale entity resolution: Challenges, algorithms, and practical examples. In *Proceedings of the 32nd International Conference on Data Engineering*. IEEE, 1436–1439.

[34] George Papadakis, George Papastefanatos, Themis Palpanas, and Manolis Koubarakis. 2016. Boosting the efficiency of large-scale entity resolution with enhanced meta-blocking. *Big Data Research* 6 (2016), 43–63.

[35] George Papadakis, George Papastefanatos, Themis Palpanas, and Manolis Koubarakis. 2016. Scaling entity resolution to large, heterogeneous data with enhanced meta-blocking. In *Proceedings of the 19th International Conference on Extending Database Technology*. 221–232.

[36] George Papadakis, Jonathan Svirsky, Avigdor Gal, and Themis Palpanas. 2016. Comparative analysis of approximate blocking techniques for entity resolution. In *Proceedings of the VLDB Endowment* 9, 9 (2016), 684–695.
[37] Giovanni Simonini, Sonia Bergamaschi, and H. V. Jagadish. 2016. BLAST: A loosely schema-aware meta-blocking approach for entity resolution. In *Proceedings of the VLDB Endowment* 9, 12 (2016), 1173–1184.
[38] Giovanni Simonini, George Papadakis, Themis Palpanas, and Sonia Bergamaschi. 2018. Schema-agnostic progressive entity resolution. In *IEEE 34th International Conference on Data Engineering*. 53–64.
[39] Giovanni Simonini, George Papadakis, Themis Palpanas, and Sonia Bergamaschi. 2019. Schema-agnostic progressive entity resolution (extended version). *IEEE Transactions on Knowledge and Data Engineering* 31, 6 (2019), 1208–1221.
[40] Rebecca C. Steorts, Samuel L. Ventura, Mauricio Sadinle, and Stephen E. Fienberg. 2014. A comparison of blocking methods for record linkage. In *Proceedings of the International Conference on Privacy in Statistical Databases*. Springer, 253–268.
[41] Jiannan Wang, Guoliang Li, Jeffrey Xu Yu, and Jianhua Feng. 2011. Entity matching: How similar is similar. In *Proceedings of the VLDB Endowment*. 4, 10 (2011), 622–633.
[42] Qing Wang, Mingyuan Cui, and Huizhi Liang. 2016. Semantic-aware blocking for entity resolution. *IEEE Transactions on Knowledge and Data Engineering* 28, 1 (2016), 166–180.
[43] Qing Wang, Dinusha Vatsalan, and Peter Christen. 2015. Efficient interactive training selection for large-scale entity resolution. In *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 562–573.
[44] William E. Winkler. 2006. *Overview of record linkage and current research directions*. Research Report Series (Statistics #2006-2). Statistical Research Division, U.S. Census Bureau, Washington, DC.

Received March 2020; revised November 2020; accepted February 2021