Li BH, Liu Y, Zhang AM *et al.* A survey on blocking technology of entity resolution. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 35(4): 769–793 July 2020. DOI 10.1007/s11390-020-0350-4

A Survey on Blocking Technology of Entity Resolution

Bo-Han Li[1,2,3], *Member*, *CCF*, *ACM*, Yi Liu[1], *Student Member*, *CCF*, An-Man Zhang[1], *Student Member*, *CCF* Wen-Huan Wang[1], *Student Member*, *CCF*, and Shuo Wan[1], *Student Member*, *CCF*

[1] *College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China*
[2] *Key Laboratory of Safety-Critical Software, Ministry of Industry and Information Technology, Nanjing 211106, China*
[3] *Collaborative Innovation Center of Novel Software Technology and Industrialization, Nanjing 210000, China*

E-mail: {bhli, yiliusx1916008, zhanganman, wangwenhuan, shuowan}@nuaa.edu.cn

Received January 30, 2020; revised June 8, 2020.

Abstract Entity resolution (ER) is a significant task in data integration, which aims to detect all entity profiles that correspond to the same real-world entity. Due to its inherently quadratic complexity, blocking was proposed to ameliorate ER, and it offers an approximate solution which clusters similar entity profiles into blocks so that it suffices to perform pairwise comparisons inside each block in order to reduce the computational cost of ER. This paper presents a comprehensive survey on existing blocking technologies. We summarize and analyze all classic blocking methods with emphasis on different blocking construction and optimization techniques. We find that traditional blocking ER methods which depend on the fixed schema may not work in the context of highly heterogeneous information spaces. How to use schema information flexibly is of great significance to efficiently process data with the new features of this era. Machine learning is an important tool for ER, but end-to-end and efficient machine learning methods still need to be explored. We also sum up and provide the most promising trend for future work from the directions of real-time blocking ER, incremental blocking ER, deep learning with ER, etc.

Keywords blocking construction, blocking optimization, data linkage, entity resolution

1 Introduction

Many enterprises, governments and research projects can collect data, which describes various entities from different fields. People get hundreds of millions of entities from different data sources [1]. However, the same object may have multiple entities and the information from multiple sources needs to be integrated and combined. Thus, an efficient and accurate process is needed to determine whether entities from the same source or different sources represent the same object in the real world. The process can improve data quality [2, 3] or enrich data to promote accurate data analysis [4] and guarantee the quality of linked data more efficiently and effectively.

Entity resolution (ER) is the process which can identify different entities that refer to the same object in the real world. It is a long-standing problem in machine learning, statistics, information retrieval, natural language processing, and database management. Different subdisciplines refer to it by a variety of names, such as record linkage [5, 6], record matching [7, 8], deduplication [9, 10], co-reference resolution, reference reconciliation, object consolidation, identity uncertainty and database hardening [11]. Accurate and fast ER has huge practical implications in a wide variety of commercial, scientific and security domains. Data records that need to be matched can correspond to entities that refer to people, such as customers, travelers, patients, employees, and students. As an example, the dramatic actor Baoqiang Wang can be referred to as "bao bao", "bao qiang" and "wang baoqiang" on different social networks while they all represent the same person. Other types of entities that sometimes

Survey
Special Section on Entity Resolution
This work was partially supported by the National Natural Science Foundation of China under Grant No. 61772268 and the Fundamental Research Funds for the Central Universities of China under Grant Nos. NS2018057 and NJ2018014.

need to be matched include records about businesses, consumer products, publications, web pages, genome sequences [12], etc.

Generally, ER suffers from a quadratic complexity because each entity must be compared with all the other entities, and string similarity measures are usually applied to pairwise comparison which always controls the overall cost of ER [13, 14]. Many techniques were proposed for solving the ER problem, which are divided into two main frameworks: blocking and filtering [15]. These two frameworks are mainly based on different environments and assumptions, and have independent research routes. The former attempts to identify which entity pairs may match in order to limit the comparison between them with the assumption that the generic entity profile is represented as attribute-value pairs and the matching function is unknown (i.e., the decision-making step). It usually allocates all entities to a group of small sized blocks, and attempts to quickly discard the absolutely mismatched pairs (i.e., entities that are not in the same block) to improve the efficiency of ER. Different from blocking, the latter represents entities as strings or sets and compares the similarity between different entities; thus it can quickly discard the pairs that are not matched indeed. Filtering is also an important technology to improve the computational efficiency of ER algorithms. However, due to the high efficiency and popularity of blocking, filtering is not our focus.

ER is an important task of data fusion and other applications, and has been widely studied for more than 70 years [16]. The blocking method is an important tool to improve the efficiency of ER and ensure the accuracy of the result. Since the Standard Blocking method was proposed by [13] in 1969, a large number of different blocking entity resolution algorithms have been proposed for different data structures. These algorithms have different construction methods for different application scenarios, and different optimization strategies were put forward for different construction methods. This paper focuses on blocking technology, which is the first to show and analyze the blocking algorithm, in an easy-to-understand way, from the perspective of data processing structure, blocking construction, and blocking optimization.

The rest of this paper is organized as follows. Section 2 provides the background knowledge and related definitions of ER and blocking technology, and Section 3 summarizes the blocking techniques under different construction methods and gives the analysis from the perspective of data processing structure. In Section 4, we analyze and summarize the techniques of blocking optimization. Section 5 discusses the future promising work and Section 6 summarizes the survey.

2 Preliminary and Overview

2.1 Definition

2.1.1 *Entity Resolution*

Records in different datasets may relate to the same entities in reality. The entity records in the dataset are generally represented by the entity profile, and an entity collection is a set of entity profile. Assuming infinite sets of attribute names N, attribute values V, and identifiers I, an entity profile can be defined as a tuple (i, A_i), with $i \in I$ being a unique identifier and A_i being a set of attribute-value pairs (n, v), where $n \in N$ and $v \in (V \cup I)$ [15, 17, 18]. For the two entity records p_i and p_j, if p_i and p_j refer to the same real-world entity, they match successfully, and we denote this as $p_i \equiv p_j$. The main task of entity resolution is to identify records associated with the same real-world entities within one entity collection or across more entity collections.

Assuming an entity collection denoted as ε and $p_i \in \varepsilon$, if there is no duplicate record in ε, it is said to be duplicate-free; otherwise it is said to be dirty. Given two input entity collections, ε_1 and ε_2, we can classify entity resolution methods into three categories: Clean-Clean ER, Dirty-Clean ER, and Dirty-Dirty ER. In Clean-Clean ER, both ε_1 and ε_2 are duplicate-free entity collections. In Dirty-Clean ER, ε_1 is a duplicate-free entity collection, and ε_2 is a dirty one. In Dirty-Dirty ER, both ε_1 and ε_2 are dirty.

2.1.2 *Blocking*

Since each entity needs to be compared with other entities, entity resolution is essentially a quadratic problem, which makes it difficult to handle large-scale data. To solve this problem, the blocking method puts similar entities in the dataset into the same block, and only compares entities in the same block, thereby improving the efficiency of entity resolution.

The blocking method consists of block building stage and block processing stage [15]. In the block building phase, the blocking method extracts from each entity profile the blocking key values (BKVs) that can include its distinguishing information to form a blocking scheme, hereby assigning one or more entities to a block set B. Each block corresponds to a specific instance of BKVs, and contains all entity records related to the instance value [19]. The goal of the block processing stage

is to improve the block set B obtained in the first stage. At this stage, either some blocks are discarded through block-refinement, or some comparisons are discarded in specific blocks through comparison-refinement.

2.2 Taxonomy

To understand a block construction algorithm from different perspectives, this subsection gives the following detailed classification perspectives [15, 20]. We show the taxonomy of the technique for blocking construction in Table 1 and Table 2. To be noticed, in Table 1 and Table 2, we use "aware" to denote that the algorithm is aware of the property, and "agnostic" means the algorithm does not have the property. In particular, we use the symbol "-" to indicate that the algorithm does not belong to this category.

Schema-Awareness. There are two ways to define blocking key values [20]. The schema-aware method needs to use a-priori schema information, and selects some specific attribute values or a combination of these attribute values as blocking key values. These selected attributes are considered suitable for matching because they are discriminative or contain less noisy data. Schema-agnostic methods are often used to process data without considering schema information. They extract blocking key values from all attribute values, which is beneficial to dealing with unstructured heterogeneous data.

Table 1. Taxonomy of Rule-Based Algorithms

Data Type	Algorithm	Schema-Awareness	Redundancy-Awareness	Constraint-Awareness	Matching-Awareness	Key Type
Structured data	Standard Blocking [21]	Aware	Free	Lazy	Static	Hash-based
	Suffix Arrays Blocking [22]	Aware	Positive	Proactive	Static	Hash-based
	Improved Suffix Arrays Blocking [23]	Aware	Positive	Proactive	Static	Hash-based
	Incremental Suffix Arrays Blocking [24]	Aware	Positive	Proactive	Static	Hash-based
	Q-gram-Based Blocking [25]	Aware	Positive	Lazy	Static	Hash-based
	Extended Q-Gram-Based Blocking [12, 26]	Aware	Positive	Lazy	Static	Hash-based
	MFIBlocking [27]	Aware	Positive	Proactive	Static	Hash-based
	Canopy Clustering Blocking [28]	Aware	Positive	Lazy	Dynamic	Sort-based
	Extended Canopy Clustering Blocking [12]	Aware	Positive	Proactive	Dynamic	Sort-based
	Sorted Neighborhood [29]	Aware	Neutral	Proactive	Static	Sort-based
	Adaptive Sorted Neighborhood [30, 31]	Aware	Neutral	Lazy	Static	Sort-based
	Duplicate Count Strategy (DCS) and DCS++ [32]	Aware	Neutral	Proactive	Dynamic	Sort-based
	Sorted Blocks [33]	Aware	Neutral	Lazy	Static	Hybrid
Non-structured data	Token Blocking (TB) [34]	Agnostic	Positive	Lazy	Static	Hash-based
	Attribute Clustering Blocking [18]	Agnostic	Positive	Lazy	Static	Hash-based
	Prefix-Infix (-Suffix) Blocking [35]	Agnostic	Positive	Lazy	Static	Hash-based
	RDFkeyLearning [36]	Agnostic	Positive	Lazy	Static	Hash-based
	Semantic Graph Blocking [37]	Agnostic	Neutral	Proactive	Static	-

Table 2. Taxonomy of Machine Learning (ML) Based Algorithms

ML and Data Type	Algorithm	Schema-Awareness	Redundancy-Awareness	Constraint-Awareness	Matching-Awareness	Key Type
Supervised learning with structured data	ApproxRBSetCover [38]	Aware	Positive	Lazy	Static	Hash-based
	ApproxDNF [38]	Aware	Positive	Lazy	Static	Hash-based
	Blocking Scheme Learner (BSL) [39]	Aware	Positive	Lazy	Static	Hash-based
	Blocking Based on Genetic Programming (BGP) [40]	Aware	Positive	Lazy	Static	Hash-based
	CBlock [41]	Aware	Positive	Proactive	Static	Hash-based
	DNF Learner [42]	Aware	Positive	Lazy	Dynamic	Hash-based
	Conjunction Learner [43]	Aware	Positive	Lazy	Static	Hash-based
Unsupervised learning with structured data	FisherDisjunctive [44]	Aware	Positive	Lazy	Static	Hash-based
	Non-Standard Key-Based Blocking [45]	Aware	Neutral	Lazy	Static	Sort-based
Unsupervised learning with non-structured data	TYPiMatch [46]	Agnostic	Positive	Lazy	Static	Hash-based
	Hetero [47]	Agnostic	Positive	Lazy	Static	Hash-based
	Extended DNF BSL [48]	Agnostic	Positive	Lazy	Static	Hash-based

Redundancy-Awareness. Considering the relationship between the blocks, various blocking algorithms can be divided into three categories [49, 50]. Redundancy-free methods assign each entity into a single disjoint block. With this kind of blocking methods, the larger the number of identical blocks to which two entities belong, the higher the similarity. Redundancy-positive methods place every entity into multiple blocks to yield overlapping blocks. The methods are generally used to improve the accuracy of the algorithm. In the middle of these two extremes lie redundancy-neutral methods, where most pairs of entities share the same number of blocks and the redundancy does not impart on the result too much.

Constraint-Awareness. According to the restrictions on blocks, the blocking algorithm can be divided into the lazy method and the proactive method. The lazy method has no restrictions on blocks, while the proactive method sets one or more restrictions on blocks, such as setting the maximum size of the blocks, or adding a restriction to reduce the number of comparisons in the block.

Matching-Awareness. The matching results of some blocking methods do not depend on the process of blocking. They are independent of the matching process and are called static methods. Other methods rely on the information obtained during the process to dynamically adjust the block, which are called dynamic methods. In the dynamic methods, the processes of blocking and matching are intertwined with each other.

Key Type. According to different measurement methods of key values, the blocking methods can be divided into hash- or equality-based methods and sort- or similarity-based methods. Hash- or equality-based methods put entity records into the same block when different entity records have the same blocking key value. The entities are placed in the same block in sort- or similarity-based methods when entities have similar key values.

2.3 Evaluation Measures

There are several ways to evaluate the entity resolution blocking algorithm [12, 15, 20]. Suppose the input entity set is recorded as ε. The block set is denoted as B, and $b_i \in B$ is a block. Then $\|B\| = \sum_{b_i \in B} \|b_i\|$, where $\|b_i\|$ means all comparison times in the block. D_ε means all duplicate record pairs in the input. D_B represents duplicate record pairs detected by the blocking algorithm.

Definition 1 (Pair Completeness (PC)). *PC is defined as $PC_B = D_B/D_\varepsilon$, and it corresponds to recall. The PC value is between 0 and 1, which is directly proportional to the effectiveness of the algorithm* [20].

Definition 2 (Pairs Quality (PQ)). *PQ is defined as $PQ_B = D_B/\|B\|$, and it corresponds to precision. The PQ value is between 0 and 1. The higher the value, the less the comparison between the unmatched entities, and the higher the efficiency of the algorithm* [20].

Definition 3 (Reduction Ratio (RR)). *RR is defined as $RR_B = 1 - \|B\|/\|\varepsilon\|$. It estimates the portion of comparisons that are saved by a blocking method with respect to the naive, brute-force approach. The value is between 0 and 1. The higher the value, the higher the efficiency of the algorithm* [20].

Definition 4 (Overhead Time (OT)). *OT represents the total time required by a blocking algorithm from obtaining the original input data ε to obtaining the block set B* [20].

Definition 5 (Resolution Time (RT)). *RT is the time required for performing all pair-wise record comparisons with a specific entity matching technique* [20].

3 Techniques for Blocking Construction

Blocking technology can effectively reduce the time complexity of the entity resolution. Its core is to produce efficient and accurate blocking standards, or blocking construction methods. In this paper, we divide construction methods into two categories, namely rule-based and machine learning based. While describing the features, advantages and disadvantages of the algorithms, we also analyze each algorithm from the perspective of data structure.

Typical blocking methods have been described in [15] and Fig.1 gives a rough description of their relationship. For each edge, if A points to B, it means that method B improves method A by modifying the definition of blocking key values or changing the way they are used to create blocks.

3.1 Rule-Based Techniques

A rule-based approach typically utilizes a rule from expert knowledge or data structure that is artificially specified by the application for blocking. We also generally classify algorithms that use heuristic methods into this category. After [21] creatively proposing the Standard Blocking (SB) algorithm, many algorithms based on SB use hash-based or sort-based methods to select blocking predicates to process structured

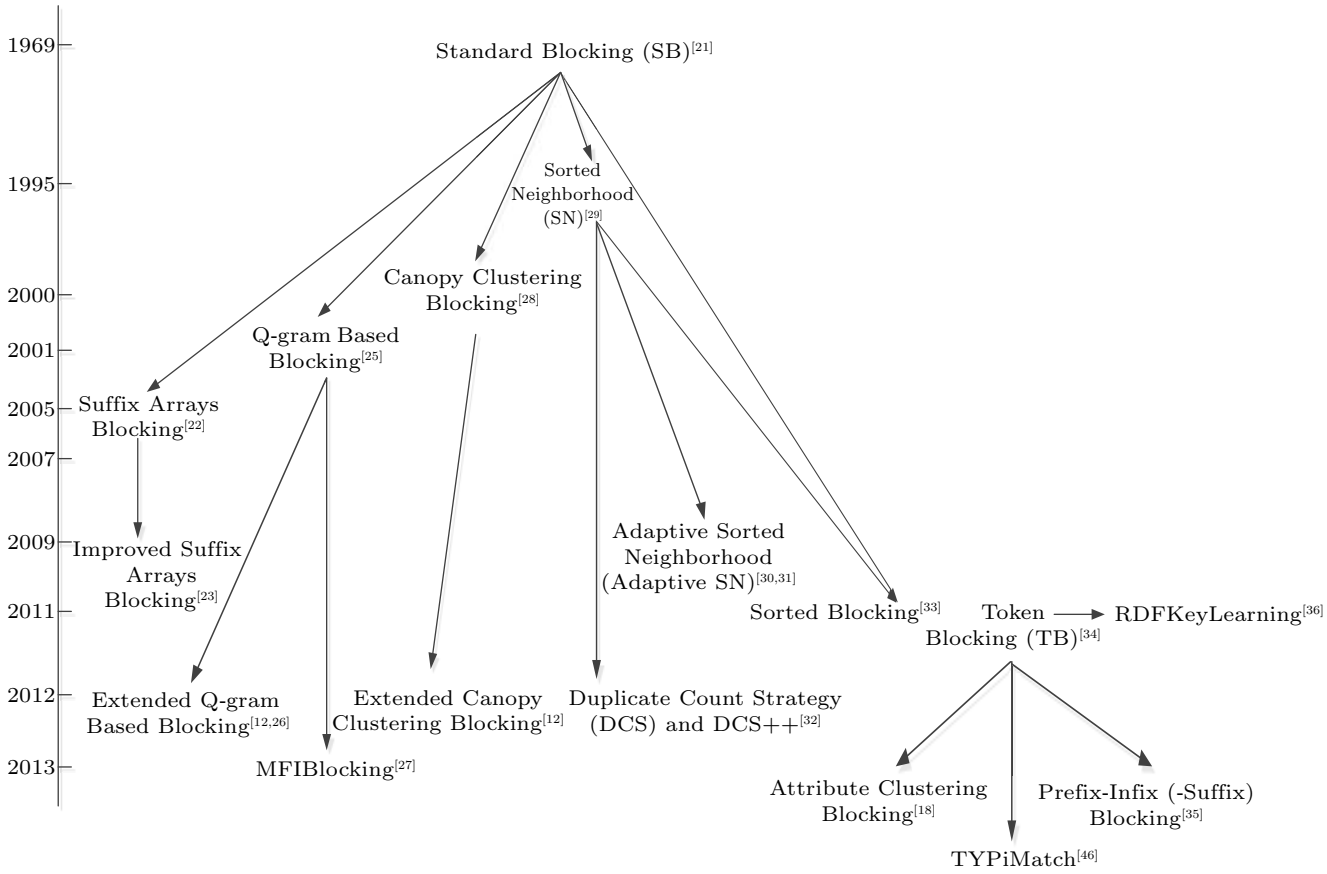

Fig.1. Genealogy trees of block building techniques in chronological order.

data. When Token Blocking (TB) was proposed based on the value of dataset instead of the scheme of data, many TB-based algorithms or algorithms just referring to some ideas of TB were proposed to deal with the non-structured data.

3.1.1 *Structured Data*

Standard Blocking. Standard Blocking [21] is the basis of the entity resolution blocking algorithm. The algorithm represents each entity with one or more key values. Each block corresponds to a specific key value and contains all entities represented by that key value. Records in the same block are compared, which can reduce the number of record comparisons in the original entity resolution. The definition of blocking key values by traditional blocking methods often affects the quality and the number of candidate record pairs for comparison [12]. Errors in records used to generate blocking key values result in blocks containing incorrect records. Meanwhile, the block size depends on the frequency distribution of the blocking key values. Therefore, it is difficult to predict how many pairs of records will be compared. Motivated by this, many blocking algorithms are improved and optimized based on Standard Blocking.

Suffix Arrays Blocking. [22] proposes a fast and efficient blocking scheme for the data linking problem of large-scale structured multiple data sources. The scheme uses a suffix array structure. The basic idea is to insert the blocking key value and its suffix into an inverted index based on the suffix array. Then the algorithm sorts this index in alphabetical order to form a new blocking scheme. For example, assume that the Surname attribute such as "Kristen" is used as the blocking key value, and $l_m = 4$. Then the suffixes that this key value can convert are "Kristen", "risten", "isten" and "sten". According to this method, the suffixes formed by records of the example in Table 3 are shown in Table 4.

As shown in Table 5, the records with the corresponding suffixes are inserted into their sorted suffix array. It should be noted that the method often introduces another parameter b_m to limit the block size. If the number of records in the block is greater than b_m, the block will be deleted. In the experiment of [22], four structured bibliographic databases are used, with more

than 10 million records. The attribute values were normalized and segmented into tokens depending on the attribute types during pre-processing. [22] selects the attributes such as the name of the first author, the title of the article and the title of the journal. The algorithm achieves good results but it is obviously suitable for structure data.

Table 3. Example Set of Homogeneous Entity Profiles

Identifier	Given Name	Surname	Suburb	Postcode
P1	Bill	Christen	Sydney	2020
P2	Peter	Smith	Canberra	2020
P3	Petes	Christen	Sydeny	2020
P4	Pete	Smith	Canberra Sth	2060
P5	Ciri	Kristen	Syd	2027

Table 4. Suffix-Array Based Blocking with Surnames Used as BKVs

Identifier	Blocking Key Value	Suffix
P1	christen	christen, hristen, risten, isten, sten
P2	smith	smith, mith
P3	christen	christen, hristen, risten, isten, sten
P4	smith	smith, mith
P5	kristen	kristen, risten, isten, sten

Table 5. Sorted Suffix Array Created by Suffix-Array Based Blocking

Suffix	Identifier
christen	P1, P3
hristen	P1, P3
isten	P1, P3, P5
kristen	P5
mith	P2, P4
risten	P1, P3, P5
smith	P2, P4
sten	P1, P3, P5

The main time consumption of Suffix Arrays Blocking is the sort operation which is used to generate the suffix array structure. Thus, it has low time complexity, $O(n)$, for text with n tokens. In addition, due to the small but relevant set of candidate matches and the high levels of redundancy, the algorithm is efficient and effective. However, the disadvantage of the algorithm is that it cannot deal with the noise at the end of BKVs. In order to handle this and improve the robustness of the algorithm, some modifications are made to the suffix generation step. For example, all substrings with a length greater than l_m are taken by sliding window to replace the suffixes generated by the previous method.

Improved Suffix Arrays Blocking. Because of some errors, there are slight differences in the same entities and the similarity is extremely high. The standard Suffix Arrays Blocking algorithm is difficult to process such data. [23] improves the technology based on the standard suffix index. The basic idea is to merge similar suffixes and their corresponding records. The algorithm can handle the noise at the end of BKVs by measures of similarity and the merging between different suffixes. At the same time, it has the same or higher PC and lower PQ, RR than the standard Suffix Arrays Blocking. The experimental data denotes that Improved Suffix Arrays Blocking improves accuracy by 20% with less than 5% efficiency loss, compared with the standard Suffix Arrays Blocking method.

Experiments are conducted using real datasets from insurance companies and synthetic datasets generated by Febrl tools [23]. In the experiment, the "identity" dataset from the real data consists of personally identifying information such as names and addresses, and the synthetic dataset was generated with the standard settings. Therefore, similar to the standard Suffix Arrays Blocking, the algorithm is also suitable for structured data without introducing any other technologies.

Incremental Suffix Arrays Blocking. In order to satisfy the requirement of incremental solutions in many modern applications such as warehouse applications, [24] adopts the idea of Suffix Arrays to propose the algorithm. [24] proposes a faster method to construct the suffix array by using two sliding windows of increasing size called the working windows w and the saved window w_s. The algorithm proposed saves the number of subsequent starting common words in an array called LCP. It sets $\text{cur}_{LCP} = LCP[i+1]$. As long as $min_{LCP} \leqslant cur_{LCP}, |w| \leqslant max_{APP}$ and $LCP[w^-] < cur_{LCP}$, the algorithm extends w while the constraints of min_{LCP} and max_{APP} are not violated. Finally, the output saves the set that contains all records corresponding to w_s and the algorithm repeats the above step until $i > |LCP|$. In order to support efficient incremental operators, [24] adopts a StrB-Tree [51] which can provide the efficient searching and incremental updates. The sliding window w is expanded to both directions using the bidirectional list of the StrB-tree SB connecting the leaf nodes.

The major advantage of the algorithm is that it is faster than the standard suffix arrays based blocking because it uses a dynamic sliding window which is very

cache-efficient and avoids iterating over all the prefixes of a certain suffix. At the same time, the algorithm also satisfies the requirement of the incremental solution. However, it still needs to be improved when dealing with noisy or heterogeneous data. From the perspective of structure of data, like the analysis of the standard suffix arrays blocking, the algorithm needs to select the value of the attribute to construct the suffix arrays such as the value of Name or Address. Thus, it is also suitable for structured data.

Q-gram Based Blocking. Originally, in order to manage the string data and deal with the problems of efficient string matching from various databases, [25] uses positional q-grams (i.e., sub-sequences of q characters with a position in the characters) for blocking based on Standard Blocking. Assuming $q = 2$ and attribute value is "Pet-Can", the positional q-grams formed by the algorithm are "(1, *P)", "(2, Pe)", "(3, et)", "(4, t-)", "(5, -C)", "(6, Ca)', "(7, an)". The algorithm is based on the assumption that the more likely the two entities match, the smaller the edit distance that they should have, that is, these matching entities share a large number of q-grams in common. To enable the blocking processing through the use of positional q-grams, [25] augments the database by creating auxiliary table $(A_i, Position, Q\_grams)$ with three attributes. The main idea of the algorithm is as follows: if the two entities are in the same block, the number of q-grams they share should be higher than threshold t.

As shown above, Q-gram-Based Blocking uses positional q-grams for blocking by constructing auxiliary tables, thus it is more suitable for structured data. Compared with Standard Blocking, the algorithm has higher resilience for noise data, having higher PC, but results in more and larger blocks and has lower PQ and RR.

Extended Q-gram Based Blocking. In order to improve the performance of the algorithm without reducing the accuracy of the algorithm, [12, 26] improve the Q-gram Based Blocking algorithm. Their efforts are to put those records that have similar rather than the same blocking key values in one block. The algorithm uses k to denote the size of this set and converts attribute values into a set S of q-grams. A parameter t ($t \leqslant 1$) is set artificially, and the L elements in the set S are combined to form a new blocking key value. Here $L = \max(1, \lfloor k \times t \rfloor)$. We give an illustrative example. When $BKV =$ "$Peter$", $q = 2$, and $t = 0.8$, the set $S = \{$"Pe", "et", "te", "er"$\}$. Thus, $k = 4$, $L = \max(1, \lfloor 4 \times 0.8 \rfloor) = 3$, and the set of bigrams is $\{[Pe, et, te, er], [et, te, er], [Pe, te, er], [Pe, et, er], [Pe, et, te]\}$. The final key values are "Peetteer", "etteer", "Peteer", "Peeter", "Peette".

Table 6 shows the execution results of P3 and P4 records. After that, according to the newly formed key values, the algorithm places the records in the corresponding blocks, as shown in Table 7. Different from the traditional blocking algorithm using inverted index, when parameter t is not equal to 1, the algorithm will put each record into a different block. But when $t = 1$, there is only one key value which is generated by the algorithm; thus each record will only be inserted into their corresponding block. The algorithm combines the element of the q-gram based set to increase the distinctness of blocking keys and decrease the number of blocks generated, thus raising PQ and RR at the limited cost of PC.

MFIBlocking. The MFIBlocking algorithm proposed in [27] is a more advanced q-gram based algorithm[15]. Different from the other algorithms, MFIBlocking gives up manually designing a blocking key to relieve the designer from the difficult task of constructing a blocking key. The algorithm uses a dynamic, automatic selection of a blocking key, and different blocks can be created based on different keys. [27] defines a block B^i as $\langle P^i, A^i, E^i, S^i \rangle$, where P^i is a subset of the dataset consisting of tuples, A^i is the subset of the attribute set, E^i is a set of values which are created by values of attributes in A^i, and S^i is a score which is assigned with B^i based on the similarity of the A^i values for the tuple in P^i. The algorithm adopts two principles of compact set (CS) and sparse neighborhood (SN). In order to satisfy the principles, for every two tuples $t_1, t_2, \sigma^i(t_1, t_2) > \sigma^i(t_1, t_3)$ where σ^i is the score function and $t_3 \notin P^i$. Then $|P^i| = 1$ or $\max_{t \in P^i} |N(t)| \leqslant p \times minsup$ where $N(t)$ is the neighborhood of a tuple t, $p > 1$ is a predefined

Table 6. Q-gram Based Blocking with Given Names Used as BKVs

Identifier	BKV	Bigram Set	New Key Value
P2	Peter	{[Pe, et, te, er], [et, te, er], [Pe, te, er], [Pe, et, er], [Pe, et, te]}	Peetteer, etteer, Peteer, Peeter, Peette
P3	Petes	{[Pe, et, te, es], [et, te, es], [Pe, te, es], [Pe, et, es], [Pe, et, te]}	Peettees, ettees, Petees, Peetes, Peette
P4	Pete	{[Pe, et, te], [et, te], [Pe, te], [Pe, et]}	Peette, ette, Pete, Peet

constant which can yield good results in the range of $[1.5, 4]$, and *minsup* is a parameter. At a high level, MFIBlocking operates a series of iterations, and in each iteration, the algorithm creates a new set of blocks. After each iteration, the blocks are scored and the final set of blocks consists of the largest possible set of high-scoring blocks such that the sparse neighborhood criterion is met.

Table 7. Inverted Index List Created by Q-gram Based Blocking

Key Value	Identifier
⋮	⋮
etteer	P2
Peette	P2, P3, P4
Peet	P4
⋮	⋮

From the description above, although the algorithm avoids manually designing blocking keys based on attribute values, it still needs to process various key attributes in all records to generate the blocks. In this preparatory phase of the experiment, each attribute value is separated into q-grams, and the q-grams, originating from the same attribute, receive the same ID. Thus, MFIBlocking is suitable to process structured data. In general, this algorithm reduces the number of blocks and record pairs used for comparison. However, it is likely to miss some records that should have been compared, which creates difficulty in processing noisy data.

Canopy Clustering Blocking. In [28], an entity resolution blocking algorithm based on clustering is proposed. It is meant to use a less computationally expensive clustering algorithm to form high-dimensional overlapping clusters, and generates blocks that contain pairs of records to be compared from these clusters. The algorithm puts all records into a candidate record pool, and uses two similarity thresholds t_l, t_{t} $(t_{\mathrm{t}} \geqslant t_l)$ to create overlapping clusters. The algorithm randomly selects record r_x from the candidate pool, and compares the similarity of this record with the other record r_y by using a similarity measure at low cost. If the similarity is greater than t_l, then r_y and r_x are put together to form a cluster. While if the similarity between them is greater than t_{t}, then r_y is deleted from the candidate pool.

In most cases, the user should leverage domain-specific features and design a cheap distance metric in order to make use of the canopies technique. In the experiment, [28] uses the bibliographic citations data to cluster these citations into the sets that each refers to the same article. They choose the string edit distance for references to build the canopies and design several transformation cost parameters specific to this domain. Therefore, we can see that the algorithm is suitable for the structured data. Compared with traditional clustering algorithms, this algorithm can reduce the calculation time by more than an order of magnitude, while also slightly improving the accuracy.

Extended Canopy Clustering Blocking. The Canopy Clustering Blocking (CCB) algorithm is sensitive to the value of weights. If t_l is high, then many entities have no blocks to put in. To overcome this shortcoming, the Extended Canopy Clustering Blocking (ECCB) algorithm is proposed in [12]. The algorithm replaces the original similarity thresholds t_l, t_{t} with two cardinality thresholds, i.e., n_1, n_2 $(n_1 \geqslant n_2 \geqslant 1)$ respectively. Experiments show that ECCB has a higher accuracy and robustness than CCB. However, the drawback of this approach is that if there are BKVs that are frequent, the generated canopies might not be big enough to include all the records with these BKVs and the true matches might be missed.

Sorted Neighborhood. In 1995, [29] proposed this algorithm when solving the problem of merging large datasets. The main idea is to sort the records in alphabetical order according to BKVs, and use a fixed-size sliding window to compare a certain number of records. That is, the candidate comparison pair only appears in the same sliding window.

As shown in Table 8, the BKVs of the two datasets are inserted into an array and sorted alphabetically. Assuming that the window size is 4, the window slides on the sorted array. Four records are taken out each time, and the two pairs are compared for comparison. The candidate comparison pairs generated in Table 8 are shown in Table 9.

The major advantage of the algorithm is the low time complexity: the key creating phase is $O(N)$, the sorting phase is $O(N \log N)$, and the merging phase is $O(wN)$, where N is the number of records in the database and w is the size of windows. However, when the size of the window is too small and too much data has the same BKVs, many record pairs that should be compared are missed. This problem can be alleviated by changing the construction of BKVs to reduce the number of records for the same BKV. Another disadvantage of this method is that, it is sensitive to the BKVs because it is sorted alphabetically by BKVs. For

example, if BKVs are generated using the surname in Table 3, BKVs of P1 and P5 are "'Christen" and "Kristen", respectively. These two similar records will be placed far away because of sorting. They cannot be put into a block. To some extent, this problem can be solved by using different attribute domains to construct different blocks. A structured tunable dataset generator was used to generate experimental data in [29], confirming that the algorithm works well in practice, but has a huge overhead. It is worth noting that [12, 52] mention the use of inverted index instead of sorted array. In the comparison phase, each different record pair is compared only once, thereby improving the performance of the algorithm.

Table 8. Sorted Array Created by Sorted Neighborhood Blocking

Array Position	BKV	Indentifier
1	Bill	P1
2	Ciri	P5
3	John	P6
4	Pete	P4
5	Peter	P2
6	Petes	P3

Table 9. Record Pairs in Each Window Based on SN Blocking

Window Range	Candidate Record Pair
1–4	(P1, P5), (P1, P6), (P1, P4), (P5, P6), (P5, P4) (P6, P4)
2–5	(P5, P6), (P5, P4), (P5, P3), (P6, P4), (P6, P3) (P4, P2)
3–6	(P6, P4), (P6, P2), (P6, P3), (P4, P2), (P4, P3) (P2, P3)

Sorted Neighborhood brings matching records close together by sorting the records over the most important discriminating key attribute of the data, and compares the records which are restricted to a neighborhood within the sorted list. In the experiment, all databases used were generated automatically by a database generator, and each record consists of the following fields: social security number, first name, initial, last name, address, apartment, city, state, and zip code. As we can see from the above description, the algorithm is suitable for structured data.

Adaptive Sorted Neighborhood. The sorted neighborhood method uses a fixed-size window to generate the blocks and get the record pairs to be compared, which often misses many true matches. [30, 31] try to solve this problem by adaptively changing parameters such as the window size. In [30], "window" is just a temporary tool used to generate the final blocks. All the blocks will have different (thus adaptive) sizes eventually.

[30] proposes two methods according to the entity similarity measurement method, namely Incrementally Adaptive SN and Accumulative Adaptive SN respectively. The basic idea of the former is to increase the size of the window until a boundary pair is obtained, as long as the similarity between the first record and the last record in the window is less than a predetermined threshold. Boundary pairs are actually pairs of records that are adjacent and have a similarity greater than the threshold in the array. Different from Incrementally Adaptive SN, in order to obtain the boundary pairs, the latter keeps increasing the window size by moving the window forward to form multiple windows of increasing size with a single overlapping record. Finally, these windows are merged to obtain blocks. [30] conducts experiments on both real datasets and synthetic datasets. The blocking key is chosen as a schema over the key attribute values, for example the first four characters of one field concatenated with the first four characters of the other. Thus, the data should be structured if there is not any other pre-processing of data in the algorithm. The results show that the adaptive scheme is robust to the size change of individual block, and has good resistance to the errors in the blocking fields.

Duplicate Count Strategy (DCS) and DCS++. If more duplicates or similar items are found in a window, then we can get more duplicates or similar items by increasing the window size. Therefore, the size of the window should be fluctuated based on the number of duplicates or similar items. Based on this assumption, [32] proposes Duplicate Count Strategy (DCS). The algorithm introduces a parameter assumption d/c, where d is the number of newly detected repetitions, and c is the number of comparisons performed. If d/c is greater than or equal to a preset threshold φ, then the window size increases. Here φ expresses the average number of duplicates per comparison.

[32] proposes DCS++ by adding for each detected duplicate the next $w-1$ adjacent records of that duplicate to the window instead of increasing the window one by one, even if the new ratio becomes lower than φ. The algorithm also saves some comparisons by calculating the transitive closure. For example, if pairs $(t_i, t_k), (t_i, t_l)$ are duplicate, (t_k, t_l) is also a duplicate pair, and window $W(k, k+w-1)$ will not be checked. [32] proves that with the right selection of the

threshold there will not be any missed matches. [32] shows that DCS++ with a threshold $\varphi \leqslant \frac{1}{w-1}$ is at least as efficient as Sorted Neighborhood and can save $w-2$ comparisons per duplicate compared with Sorted Neighborhood in the best case. The major advantage of DCS and DCS++ is that the algorithms can efficiently respond to different cluster sizes within a dataset and the algorithm is good alternative to Sorted Neighborhood. From the perspective of data structure, the algorithms are based on the Sorted Neighborhood and need to sort by the key which means the need to select the important attribute value as the key. In the experiment, [32] selects three datasets and uses the attribute new reference as sorting keys such as the concatenation of the first author's last name and the year of publication. Thus, the algorithms are suitable for structured data as Sorted Neighborhood.

Sorted Blocks. After combining the traditional blocking and Sorted Neighborhood technologies, [33] proposes a new entity resolution blocking technology, Sorted Blocks. The technology sorts all the blocking key values according to the lexicographic order. Then it divides the ordered entities into different blocks according to the prefix of the blocking key value, and compares the records in the block. In addition, the algorithm uses window technology to avoid missing any matches. Records from different blocks in the window must also participate in the comparison calculation. The Sorted Blocks algorithm does not limit the block size, which can lead to large blocks taking up much processing time. To solve this problem, [33] proposes two proactive variants that limit the maximum block size, namely Sorted Blocks New Partition and Sorted Blocks Sliding Window. Sorted Blocks New Partition creates a new partition if the maximum partition size is reached. Sorted Blocks Sliding Window chooses to avoid performing all comparisons in blocks whose sizes are greater than the upper limit by sliding a window equal to the maximum block over the entity of the current block.

The major advantage of Sorted Blocks is that the variable partition size allows more comparisons if several records have similar values, but requires fewer comparisons if only a few records are similar. This makes the algorithm more efficient. However the algorithm also has more parameters and becomes more complex because of it. Sorted Blocks sorts the records based on a sorting key firstly as Sorted Neighborhood. The sorting keys should be unique enough to obtain an unambiguous sorting order (e.g., zip_code and name); therefore more attributes can be included for sorting than for actually partitioning the data. [33] uses three datasets to evaluate the algorithms such as the CDs including artists, titles, and songs. The sorted keys are selected by a manual schema such as the first three letters of each artist concatenating with the CD title and the name of the first track. The algorithm is suitable for the structured data.

3.1.2 Non-Structured data

With the development of the network, a large amount of non-structured data which is highly dynamic, heterogeneous and loose schema-bound becomes accessible. For example, Google Base alone encompasses 100 000 distinct schemata that correspond to 10 000 entity types [53]. Existing blocking approaches rely on schema information such as the concatenating of attribute names or knowledge about the domains of the respective attributes in order to define effective blocking criteria. However these approaches did not work in the context of so high levels of heterogeneity. Many algorithms focus on the attribute-agnostic mechanism and give up the methods based on schema binding.

Token Blocking (*TB*). In order to develop an ER technique that can be efficiently applied in the cases where large volume of noisy and heterogeneous data are prevalent, [34] introduces the attribute-agnostic mechanism and proposes a new attribute-agnostic blocking method instead of relying on the schema binding. The main idea is based on the assumption that duplicate entities, which should be put into a block for subsequent comparison, have at least one common value (denoted as token) and the value is independent of the corresponding attribute name. [34] gives the relevant definitions and proposes a unifying data model with simple entity profiles that correspond to real-world entities. Given the attribute set AN, the corresponding value set V and infinite set of identifiers ID and local $id \in ID$ (id is an identifier of an entity profile in the considered dataset), $n_i \in AN$ is an attribute name and $v_i \in V \bigcup ID$ is an attribute name. The profile p of an entity can be defined as a tuple (id, A_p) where A_p is a set of the attribute a_i. $a_i \in A_p$ is represented by the tuple (n_i, v_i). The blocking standard consists of two functions, namely a transformation function f_{t} and a transitive, symmetric constraint function f^i_{cond}. Among them, f_{t} derives the appropriate representation for blocking from the complete entity profile (or parts of it). For example, f_{t} can be the representation of each entity with its value for the attribute "zip code". While f^i_{cond} encapsulates the condition for two entities to be

placed in a block. The formulas for f_{t} and f^i_{cond} are as follows:

$$f_{\mathrm{t}}(p) = \{t_i : \exists n_i, v_i : (n_i, v_i) \in A_p \wedge t_i \in tokenize(v_i)\},$$
$$f^i_{\mathrm{cond}}(p, q) = ((t_i \in f_{\mathrm{t}}(p)) \wedge (t_i \in f_{\mathrm{t}}(q))),$$

where $tokenize(v_i)$ is a function that returns the set of tokens comprising value v_i. The next task is how to find the blocking criterion which can be robust enough to produce a set of blocks of high quality. The attribute-agnostic approach employs a transformation function f_{t} to transform all values of entity profiles into sets of tokens firstly, and then constraint functions f^i_{cond} are defined individually on these value tokens. If the datasets are all duplicate-free individually, f^i_{cond} suffices to consider only the intersection of the token sets of these datasets. Finally, each block created by the combination of f_{t} with all f^i_{cond} consists of all the entities containing this token in their profile values. In order to keep the likelihood of missed matches low and reduce the number of unrequired pair-wise comparisons, [34] optimizes the algorithm by using the Block Scheduling and Block Processing processes methods.

The algorithm has two major advantages. First, it can be efficiently implemented using the well-established IR techniques. Second, it is very robust to noise and heterogeneity. However, the datasets should be duplicate-free individually during the phrase of block building, and it relies on redundancy to achieve high effectiveness, resulting in the cost of lower efficiency, since it produces overlapping blocks with a high number of unnecessary comparisons.

In order to deal with the voluminous, highly heterogeneous, and loosely structured data, the algorithm focuses on the similar or same values of data and waives the use of manual schema information. [34] uses two kinds of datasets in the experiment. D_{moive} is a collection of movies shared among IMDB and DBPedia. The number of attribute names and the average profile size from the two sources are both different. We can see that TB is an algorithm which was originally designed to handle the heterogeneous and non-structured data even though it can also be used to process the traditional structured data theoretically.

Attribute Clustering Blocking. In order to process the data in the context of highly heterogeneous information spaces (HHIS), [18] optimizes the TB algorithm, and proposes the Attribute Clustering Blocking algorithm. The idea of the algorithm is to divide attribute names into nonoverlapping clusters according to the similarity of attribute names. If the cluster set obtained by attribute value division is denoted as K, then given $k \in K$, for each token t_i in k, a block created with its values contains all entities having t_i assigned to an attribute name belonging to k. Compared with the TB algorithm, the block collection created by the algorithm is larger in size, but of lower aggregate cardinality. Therefore, attribute clustering is expected to achieve a PC-RR balance of a higher efficiency.

The major advantage of the algorithm is that it can achieve equally high effectiveness with the TB algorithm, but at a significantly lower redundancy and higher efficiency. However the algorithm only considers the clean-clean ER. The Attribute Clustering Blocking algorithm was originally designed in the context of HHIS and based on the TB algorithm. It derives attribute clusters that produce blocks with a comparison distribution instead of trying to partition the input into clusters by the semantically equivalent attributes. Thus, the algorithm is suitable for the non-structured data in HHIS.

Prefix-Infix (-Suffix) Blocking. Large-scale heterogeneous datasets have the characteristics of loose schema binding. Many identifiers contain semantic information, and there is relationship information between different entities. Based on the entity identifier and the relationship between entities, [35] introduces a novel blocking approach Prefix-Infix (-Suffix) Blocking [15]. It is also called URI Semantics Blocking [54].

[35] states that according to the research in [55], about 66% of the 182 million URIs of the BTC09 dataset follow a common pattern. To be specific, the prefix part contains the source of the URI. The Infix part has some local identifiers. The suffix part contains some detailed information [56]. [35] divides the algorithm by using the blocking method and the number of the three different parts. The blocking method that uses only one of the above parts is Atomic Blocking Schemes, and the one that uses multiple parts is Composite Blocking Schemes. For Atomic Blocking Schemes, [35] gives three blocking methods: Infix Blocking, Infix Profile Blocking and Literal Profile Blocking. Infix Blocking conveys a transformation function that extracts the infix from the ID of each entity profile, and every block is associated with an infix and consists of the entity with the same infixes. Infix Profile Blocking represents each entity profile by infix profile and divides the entities according to the relationships. Literal Profile Blocking uses the set of all

tokens of the literal values to denote the entity profile, and blocks are based on the equality of tokens. For Composite Blocking Schemes, [35] gives four blocking methods: Complete Infix Blocking, Infix-Literal Profile Blocking, Infix Profile-Literal Profile Blocking and Total Description Blocking.

The major advantage of the algorithm is that, different from the previous algorithm, Prefix-Infix (-Suffix) Blocking makes use of the semantics and the relationships of the entities. The composite of Atomic Blocking Schemes can both achieve high efficiency and robustness. [35] also uses the Block Purging method to improve the efficiency of the algorithm. From the perspective of data structure, the algorithm completely ignores the schema information and exclusively relies on the values of entity profiles because of the heterogeneity of the data, and this gives it the ability to process non-structured data.

RDFkeyLearning. [36] uses the TB algorithm on the values of some selected attributes independently, and proposes the RDFKeyLearning algorithm. The aim of the algorithm is to obtain a set of data type attributes as the candidate blocking key that both discriminates and covers the instances well in a domain-independent manner. The algorithm introduces the "discriminability score" to measure the value of a certain attribute. Discriminability score of an attribute is used to denote the diversity of its object values and having low scores means that many instances have the same object values on this attribute; therefore, when utilizing such object values to look up similar entities, we cannot get a suitable reduction ratio. For those attributes whose discriminability score is higher than the threshold, the algorithm will evaluate their coverage, that is, how many entities have this attribute. Next, the algorithm calculates discriminability and the mean of the coverage. It selects the attribute which is the largest and higher than another preset threshold to provide the blocking key value. If no attribute satisfies the condition, the algorithm will combine the attribute which get the highest discriminability score with other attributes for the next iteration.

The advantage of RDFkeyLearning is that we can get the optimal set of attributes for candidate selection. However, the process is exponential in the number of candidate attributes because of its two loop, and the algorithm targets datasets that are primarily composed of strings. RDFkeyLearning is originally designed to process the RDF data even though it can also commonly achieve the best or comparably good results on the structured data.

Sematic Graph Blocking. [37] proposes the Semantic Graph Blocking method based on the relationship between entities, as opposed to the use of the syntactic information of attributes, as in the classic methods. The assumption the algorithm depends on is that two occurrences are more likely to refer to a single individual if they are closely related. The basic idea of this method is to use the relationship between entities to build a collaborative graph without considering the value of the attribute. Nodes represent entities, and edges represent the relationship between entities.

The major advantage of the algorithm is that, different from the classic blocking approaches, it can avoid the problems caused by spelling errors and still works in the condition where the datasets do not have common categorical or regular attributes by only allowing comparisons between records that have a relationship among them. However, the effectiveness of the algorithm will reduce if the connectivity of the dataset is high. A large part of the blocking graph with a limited size will consist of all those entities directly connected to the explored entity, and this will reduce the probabilities of the method finding real matches.

As we can see the algorithm relies on the relationships between the entities instead of the attributes of datasets. Thus the field of the algorithm will not be constrained in structured data from the perspective of data structure. If we have a relational database as the data source, the context information can be obtained from the foreign keys of the database. However, if we have a plain text file or other unstructured files as a data source, we can also connect the entities based on the relationships of all the records that contain values with the same contextual meaning in one or more textual attributes.

3.2 Machine Learning (ML) Based Techniques

Rule-based blocking algorithms mostly need to construct index-based similar functions manually, or select some blocking predicates with parameters. While the problem that machine learning based blocking algorithms need to solve is how to automatically learn efficient and accurate blocking functions. Such methods can be divided into supervised and unsupervised learning. The former relies on labeled datasets, which contain the data from matching entities and non-matching entity record pairs, called positive and negative examples, respectively. The latter does not rely on the labeled data or generate the needed data automatically

and it is suitable for the case where the labeled data is lacking. We will mainly introduce the algorithms commonly used in these two categories.

3.2.1 Supervised Learning with Structured Data

ApproxRBSetCover. The ApproxRBSetCover algorithm [38] is one of the earliest proposed entity resolution blocking algorithms based on machine learning. The relevant definitions are given in [38], including the formal definition of the problem. The paper assumes that the training data $D_{\text{train}} = (X, Y)$, where $X = \{x_i\}_{i=1}^{n}$ represents a set of records, and $Y = \{y_i\}_{i=1}^{n}$ represents the real object (or entity) corresponding to the record X. A set of general blocking predicate set P is expressed as $\{p_i\}_{i=1}^{s}$. When the general block predicate is applied to different fields of the dataset, different special predicates will be generated. For example, suppose there are three predicates that can be applied to any text field, "Exact Match", "Same 1st Three Chars", and "Contains Same or Off-By-One Integer", which are applied to four different text fields, such as "Author", "title", "year" and "venue". Then $3 \times 4 = 12$ special block predicates are produced.

The algorithm uses the elements in the predicate set P to form an overall blocking function f_p. The multiple blocking predicate is composed of the blocking function f_p. The problem definition is given in [38] that given a potential blocking predicate set $P = \{p_i\}_{i=1}^{t}$, learning an optimal blocking function f_p^* requires selecting a predicate subset P_*, so that all or almost all coreferent pairs are covered by at least one predicate in P_*, and the minimum number of non-coreferent pairs is covered. The following formula [38] is used for formal description [29]:

$$\omega^* = \underset{\omega}{\operatorname{argmin}} \sum_{(x_i x_j) \in R} \left[\boldsymbol{\omega}^{\mathrm{T}} \boldsymbol{p}\left(x_i, x_j\right) > 0\right],$$

$$\text{s.t.} \quad |B| - \sum_{(x_i x_j) \in B} \left[\boldsymbol{\omega}^{\mathrm{T}} \boldsymbol{p}\left(x_i, x_j\right) > 0\right] < \varepsilon,$$

where $B = \{(x_i, x_j) : y_i = y_j\}$ represents a set of coreferent pairs, and $R = \{(x_i, x_j) : y_i \neq y_j\}$ represents a set of non-coreferent pairs. ε is an adjustable parameter, describing the number of coreferent pairs that have not been found. $\boldsymbol{\omega}$ is a binary vector of length t and it encodes which potential blocking criteria are selected. $\boldsymbol{p}(x_i, x_j)$ is a vector of binary values returned by the t predicates for pair (x_i, x_j). The problem is finally classified as a "Red-Blue Set Cover" problem to solve. Learning the optimal blocking function is equivalent to finding a subset of predicate vertices with their incident edges so that at least $\beta - \varepsilon$ blue vertices have at least one incident edge, while the cover cost, equal to the number of red vertices, is minimized.

ApproxDNF. In some areas, a disjunctive combination of blocking predicates cannot express the best blocking strategy in detail. Research [57] shows that in the United States of Census data, the use of conjunctions such as "Same Zip AND Same 1st Char in Surname" is effective. [38] proposes the ApproxDNF method based on the ApproxRBSetCover algorithm. Unlike the ApproxRBSetCover algorithm, the new method takes conjunctions of predicates into consideration. In each iteration, it selects k conjunctions that can maximize the ratio of coreferent pairs and non-coreferent pairs, thereby forming a candidate set of conjunctions $P^{(c)}$. Then $P^{(c)}$ is added to a single set of predicates P. ApproxDNF also classifies the problem as a "Red-Blue Set Cover" problem by adding a vertex to the middle row corresponding to the conjunction of blocking predicates, along with edges connecting it to the red and blue vertices from the intersection of covered vertex sets for the individual predicates in the conjunction.

The authors of ApproxRBSetCover and ApproxDNF both presented experiments on two datasets: Cora and Addresses. The Cora dataset contains different field citations to computer science papers, and Addresses is a dataset containing names and addresses of 10 000 individuals. The results show that the two algorithms significantly improve the efficiency of entity resolution and provide an attractive methodology for data mining tasks that rely on similarity computations between different records. However the algorithms still have some shortcomings. ApproxRBSetCover lacks considering the combination of the different blocking predicates, and that is compensated by ApproxDNF. But for ApproxDNF, the number of all possible conjunctions is exponential; therefore only conjunctions up to predetermined length k are considered. However, it is difficult to pre-set the parameter k, and that also limits the possibility of finding the better blocking key to some extent.

The aims of ApproxRBSetCover and ApproxDNF are both to learn an optimal blocking function which can find a combination of blocking predicates that captures all or nearly all true matches and a minimal number of non-coreferent pairs. In order to get the key values, the algorithms apply general blocking predicates on the different parts of the records. Thus it is diffi-

cult to get a good key if the data is heterogeneous or unstructured and the algorithms are suitable for the structured data.

Blocking Scheme Learner (*BSL*). Similar to [38], [39] proposes a blocking learning algorithm Blocking Scheme Learner (BSL) based on machine learning for constructing an efficient blocking scheme automatically. [39] indicates that an efficient blocking scheme should have the characteristics of increasing the number of correct comparison pairs while minimizing the number of candidate comparisons. [39] proposes that the conjunction is an intersection between {method, attribute} pairs and a blocking scheme is a disjunction of conjunctions. [39] uses the Sequential Covering Algorithm (SCA) to solve the construction problem of blocking schemes. The algorithm first learns the rules that can cover the positive examples which are the matches in the training data. The rule here refers to the conjunction of attributes. Then it deletes these covered positive examples and continues to learn new rules until it can no longer discover a rule with performance above a threshold. There are some differences from the traditional SCA, since each of the conjunctions is disjointed together, and any records covered by a more restrictive conjunction will be replaced by a less restrictive conjunction and if the rule contains any previously learned rules, the algorithm will remove these contained rules.

[39] defines the conjunction as the intersection between {method, attribute} pairs. An instance of the {method, attribute} pairs is like ({first-letter, first name}. In order to construct the pairs, the dataset had better be structured. The experimental data is the restaurant dataset, and the attributes for this data are {name, address, cuisine, city}. The major advantage of the algorithm is that, since the algorithm only considers the true pairs of the training data, it requires a small training set and its training speed is faster than those of ApproxRBSetCover and ApproxDNF. The disadvantage of the algorithm is to cover true matches and the ability is always limited by the number of true matches in the training data.

Blocking Based on Genetic Programming (*BGP*). Different from previous entity resolution blocking algorithms based on machine learning, [40] proposes a BGP algorithm based on the genetic programming method. The algorithm can use flexible rules to define blocking functions. It relies on tree structure to represent blocking scheme. Each leaf node represents a blocking predicate, and the other nodes are defined as Boolean operation "or" or "and". Each edge represents the relationship between different predicates and Boolean operators. [40] notices that blocking schemas with larger numbers of conjunctions of affinity rules can cover larger numbers of true matches, and proposes a new criterion on $f^*_{\text{Fit}} = f_{\text{Fit}} + \left[\frac{C}{100}\right]$, where $f_{\text{Fit}} = \frac{2}{1/PC+1/PF}$ and C represents the number of conjunctions found in the blocking schema. The algorithm is processed as the following sequence of steps: randomly generating an initial set of individual blocking schemas firstly and evaluating the schemas. Then the algorithm creates a new set of schemas by using genetic operators and samples which present true and false matches. Finally the best schema is generated after a predetermined number of generations.

BGP depends on genetic programming which allows for the use of more flexible rules and chooses the best values of parameters that can be used in each rule. By using genetic programming, BGP can achieve satisfactory results without analyzing a very large number of blocking combinations of predicates. However, the algorithm has too many parameters such as the number of generations and the number of individuals in each generation. That will also make the operation of tuning more difficult. When the amount of records is huge, many similar blocks will be generated and hence many pairs of duplicate records are repeated, and that will increase the computation time.

BGP was implemented in two versions: BGP-SR, which is based on standard rules, and BGP-PR, which was defined as rules based on parameters. The basic standard rules used in BGP-SR have no parameters such as perfect match and first three characters in common. The basic rules in BGP-PR are the rules with parameters such as N-grams in common or length of substrings in common. Like the other algorithms based on the structured data, BGP also depends on the attributes of the dataset. Thus the dataset should be structured in order to use these rules.

CBlock. The CBlock algorithm proposed in [41] also uses a tree structure to represent the blocking process. Different from previous studies that introduced the blocking algorithm, CBlock is executed in a map-reduce framework like Hadoop and uses "canopy" to represent the block. The canopies represented by the leaf nodes in the tree are the final blocking result. The blocking functions are constructed by hash functions which are based on the attributes of the records. The algorithm assigns a hash function to each node except the leaf node during execution. When the number of entities in a node (i.e., canopy) is greater than a pre-

defined maximum value, the algorithm greedily selects the best hash function by counting for all hash functions the number of duplicates that get eliminated when choosing the hash function. Then the hash function that minimizes the number of eliminated duplicates is chosen. In order to increase the overall recall, the algorithm combines multiple small canopies to maintain the size requirement. [41] also introduces the "drill-down" problem for a single attribute to generate hash function automatically. The aim is to optimally divide a single-attribute's domain into disjoint sets so as to cover as many duplicate pairs as possible and ensure that the cost is below a threshold. [41] designs an optimal polynomial-time algorithm to address the drill-down problem based on dynamic programming.

CBlock is a proactive method and can follow the guide of different applications by the blocking process based on architectural constraints, such as imposing disjointness of blocks in a grid environment or specifying a maximum size of each block based on memory requirements. It is noteworthy that the algorithm is the only ML-based method suitable for the MapReduce framework. CBlock was originally designed to process the noisy data whose attribute values may be polluted, for example, some attribute values have null values in some records. The hash functions (also called blocking predicates) used in the algorithm are still based on the restrictions of attributes. The datasets used in the experiment are movie data and restaurant data. The scheme of movie data from DBPedia and IMDb consisted of the attributes such as title, director, release year and runtime. The restaurant data have the attributes such as street, city, state, and zip. The hash functions are created using these attributes.

DNF Learner. To solve the problem of lack of labeled data for training, [42] proposes the DNF Learner algorithm. In addition to the blocking scheme, DNF Learner is used in the context of a matcher which it is going to be used with. The article proposes an algorithm that can automatically generate labels for learning the blocking scheme. It classifies the learning blocking scheme problem as a DNF learning [58] problem, and uses the Probably Approximately Correct (PAC) learning theory to guide the algorithm to learn the best blocking scheme. To make the algorithm feasible, the article gives two practical simplifications. The first is to optimize the rejection rate of blocking schemes whose recall value is higher than the preset acceptable threshold. The second is to limit the search for acceptable blocking schemes.

Conjunction Learner. The previous machine learning based entity resolution blocking technology uses labeled datasets. However, labeled data is not enough to characterize unlabeled data, which results in a poor blocking scheme. [43] proposes a semi-supervised algorithm Conjunction Learner that combines labeled data and unlabeled data to learn a blocking scheme. [43] uses the tunable parameter α to control the weight of unlabeled data in the algorithm. If $\alpha = 0$, the algorithm only considers the labeled data and degenerates to BSL. If $\alpha = 1$, the algorithm treats labeled data and unlabeled data equally.

[43] also uses SCA as the basic idea of the proposed algorithm. It is appropriately modified to adapt to new application scenarios. [43] assumes that the amount of unlabeled data is much larger than the amount of labeled data. Cost functions are defined according to the characteristics of labeled data and unlabeled data. [43] uses sampling to obtain several subsets containing unlabeled data, and the aggregation function to calculate the cost of them. The aggregation function uses the average or the maximum and the minimum values according to actual need.

3.2.2 *Unsupervised Learning with Structured data*

FisherDisjunctive. Methods based on supervised learning rely on labeled data. It is not easy to get even a small number of high-quality label datasets. [44] proposes an algorithm to generate a weakly labeled training set. The algorithm firstly comprises a simple Disjunctive Blocking scheme, then the scheme makes multi-pass over the datasets and tokenizes the corresponding field value of each tuple. The tuple is then placed in blocks. The algorithm presets several parameters such as the upper threshold ut, lower threshold lt and a window size c. After all the blocks have been generated, a sliding window passes over c tuples at a time and all tuple pairs within the window are generated. Each pair computes the similarity using TFIDF measure, and if the result falls between ut and lt, the pair is considered ambiguous and ignored. If the result is larger than ut, the tuple will be added to the list which contains the duplicate records. Otherwise, the tuple will be denoted as non-duplicate.

After getting the weakly labeled training datasets, FisherDisjunctive uses feature vectors to store the boolean results of each predicate, and the Fisher discrimination criterion is used to determine the best features. Whether the feature is eligible depends on the number of negative examples it covers. If the eligible

feature also covers positive examples that the current disjunction does not cover, it is added to the current disjunction.

[44] defines the blocking scheme as a function constructed in disjunctive normal form using a given set of special blocking predicates which are formed by combining general blocking predicates and the attributes of the datasets, such as (Contains Common Token, address) and (Exact Match, city). In order to learn the blocking scheme, the algorithm applies all the given specific blocking predicates on every duplicate or non-duplicate record and stores the results in a feature vector. The algorithm still depends on the scheme of the datasets to generate the BKVs, and gives priority to the structured data.

FisherDisjunctive adapts an unsupervised framework for learning good blocking schemes and achieves good results compared with a supervised algorithm. Although the algorithm is efficient, it has many parameters and the amount of parameters results in the problem of parameter tuning.

Non-Standard Key-Based Blocking. [45] proposes a new blocking algorithm that does not need any labelled data or manual fine-tuning of parameters. The algorithm has three phrases: blocking predicate reduction, blocking predicate weighting and record blocking using selected predicates. In order to waive the obviously weak blocking predicates without labeled data and manual fine-tuning of parameter values, the algorithm uses *Coverage* to indicates the proportion of records for which at least one blocking key is generated by a blocking predicate. The algorithm omits the useless blocking predicates with coverage less than the average value to reduce computation. In the second phrase, the algorithm uses a modified version of the automatic labelling algorithm to generate a set of labelled positive records. It is noteworthy that the algorithm uses the automatic labelling algorithm to obtain labelled records, but it only chose 5% of the record pairs which not only can keep computation low, but also are mostly matching record pairs, with a high chance. Then, the weights computed by the reduction ratio are assigned to each blocking predicate. Finally, the first record of a dataset is selected as the representative record of an initial block and each record is compared with the representative records by three best blocking predicates which are selected in the second phrase. The similar records will be put in the same block; otherwise, they will be used to create a new block.

The algorithm was among the fastest and most proficient in most of cases and was suitable in the situation where the domain expert and the labelled data are difficult to obtain. However, the small number of labelled data which is used to select the best blocking predicates and blocking predicates which is used to compare different records may also affected the robustness of the algorithm while bringing about speed improvement.

In every phrase of the algorithm, blocking predicates are applied on the records, and each indexing function for blocking predicates is combined with each attribute of a dataset. Nine datasets are used in the experiment and each of them has the fixed number of attributes which follow the unified scheme. Thus, the unsupervised algorithm is suitable for the structured data.

3.2.3 *Unsupervised Learning with Non-Structured Data*

TYPiMatch. [46] finds that the quality of the scheme-agnostic approach can be greatly improved [36, 55, 59] when using a special subtype from the data instance instead of the blocking key learned from the usual type. Based on this, [46] proposes an unsupervised algorithm TYPiMatch. TYPiMatch is an algorithm which also refers to the idea of TB. The algorithm relies entirely on the value of data for learning both the subtypes and the key values specific to these subtypes. First of all, [46] proposes that a subtype is a set of instances that share the same set of pseudo-schema features. The pseudo-schema features are the features in values which commonly co-occur in instance descriptions such as Title and Price in instances of type Product. Their solution is based on the observation that instances belonging to the same type share the same attributes in the schema. However these attributes, called schema features, are too few in heterogeneous data, and it is difficult to use them to distinguish different subtypes. Yet, instances of the same type also have some features in their values in common besides schema features, which means the identical parts of the entity description can be used to infer features. The example given by [46] is shown in Table 10. Instances of type "Camera" have a description of "Cyber-Shot" or a camera brand (such as "Sony"). These common descriptions are recorded as pseudo-schema features, and subtypes are deduced from them. To form the final blocking schema, the algorithm iterates through all features that appear in the key values and retrieve all the candidates for a given feature. Finally, blocking key values are obtained from these representative subtypes.

Table 10. Example Blocks for Different Subtypes [46]

Block	ID	Title	Subtype
Block 1	P1	Sony Cyber-Shot DSC-W650	Camera
	P2	Sony DSC-W620 Cyber-Shot	Camera
	P3	Sony Cyber-Shot DSC-W650	Camera
Block 2	P4	Sony Reader PRS-600SC	eReader
	P5	Sony Reader PRS-600SC	eReader
	P6	Sony PRS-T1 6" eBook Reader	eReader

[46] uses "DBLP-Scholar" data and "Abt-Buy" product data for experiments. The algorithm yields up to 32.62% improvement in terms of reduction ratio over existing solutions for blocking. However, the detection of entity types is time-consuming and too sensitive to its parameter configuration [60].

[46] considers general Web data including different types such as relational, XML and RDF data. The algorithm learns the subtype-specific blocking keys from the heterogeneous data instead of depending on the manual schema features. Thus, we can see that the algorithm is designed to process the non-structured data.

Hetero. In the current work, the dataset pairs between the to-be-linked entities are provided. In other words, each collection is a set of datasets, such as the government data which consists of batches of files, and datasets in one collection first need to be mapped to datasets in the second collection, after which a blocking scheme is learned on each mapped dataset pair. In order to address the problems, a link-discovery blocking scheme learner is proposed in [47], which includes the following steps. It first uses property tables to solve the problem of mixing RDF and tabular data in the input data. Then it relies on the similarity of the documents and represents each dataset in each collection as a term frequency vector. After obtaining the vector, the algorithm generates the matrices using the dot product of vectors of two datasets and maps pairs between collections according to a confidence function of the Max Hungarian algorithm [61]. The records selected from the dataset that complete the mapping are involved in the learning process of the blocking scheme. Hetero also adopts the feature selection technique like FisherDisjunctive to learn the best Blocking scheme. The number of training pairs is a given constant value and does not grow with the dataset. In order to handle a small number of fixed-size training sets, the algorithm also uses bagging [62] technique. In each bagging iteration, a small part of the overall training sample is chosen to apply the feature selection.

The algorithm uses mapping and technique of bagging to make up the loss quality of a learning algorithm. In this way, Hetero can work with small training sets. However, like FisherDisjunctive, the algorithm has many parameters and after adopting bagging technique, it is more difficult to tune the good parameters. The algorithm was originally designed in the context of heterogeneous data. [47] mainly considers RDF and tabular data, and uses property tables to combine the different type data. In the phrase of mapping, data is represented as a vector and the value is the frequency of the token in the datasets. Finally, the data of different structures can be unified into the same form for processing and this gives the algorithm the ability to process heterogeneous data.

Extended DNF BSL. Previous DNF Blocking scheme learners assume that the input of the algorithm should be tabular and structured. As the restriction limits the application of these algorithms, [48] proposes a general pipeline for learning DNF blocking schemes. Combined with an existing instance-based scheme matcher called Dumas [63], it gives an unsupervised method of this pipeline. The general pipeline is mainly divided into two modules, the Schema Matcher module and the Extended DNF BSL module respectively. The unsupervised method provided in this paper replaces the Schema Matcher module with the existing scheme matcher Dumas. The assignment problem in the algorithm is finally solved by applying the Hungarian Algorithm on this matrix [61]. After the Extended DNF BSL module getting the set of duplicate tuple pairs and the set of mapping generated by the Schema Matcher module, it gets the noisy duplicates as the negative records by permuting the duplicates set and outputs the DNF blocking scheme.

Although previous methods such as [43, 44] require less supervision, they require more parameters. While the method proposed in this paper only needs two parameters, in order to process the noisier datasets, a good n:m scheme matcher is still needed to be evaluated in the future work. Like Hetero, in order to process the heterogeneous datasets, Extended DNF BSL considers the RDF datasets and transforms the RDF data into a property table. Different from the previous algorithms, the pipeline also admits structurally heterogeneous tabular inputs, and this allows it to be applied to tabular datasets with different schemas. Thus, the algorithm is still feasible when dealing with the RDF-tabular heterogeneous data in the task of linking datasets between Linked Open Data and the relational Deep Web [48].

4 Technique for Blocking Optimization

The purpose of blocking optimization is to reduce the overall computational cost of the entity resolution algorithm, while ensuring that the accuracy of the algorithm is minimally affected. It improves the PQ and RR values with the least impact on the PC value. We divide the optimization method into block-refinement and comparison-refinement according to the level of operation.

4.1 Block Refinement

Among methods for improving the blocking technology, some rely on preset rules, which are called static methods. Another type of methods relies on the feedback of the algorithm results to improve the blocking, which is called dynamic method [15]. The followings are introduced from these two aspects.

4.1.1 Static Methods

It was found in [35] that large blocks require many records, resulting in high computational cost and little improvement to the PC. The authors [35] proposed the block-purging technology, which sets a maximum block size and discards blocks that exceed this maximum. This technology brings improvements to PQ and RR, and has a small impact on the PC. It was found in [64] that the Meta-Blocking method retains a large number of redundant comparisons, and has a high computational cost when processing large-scale datasets. In order to solve these problems, [64] proposes the Block-Filtering method. It is equivalent to a preprocessing operation, which can reduce the size of the block by shrinking the blocking graphs generated in the Meta-Blocking method.

Different applications have different block size requirements. For example, Real-Time ER requires subsecond response time, and the block size requires an upper limit. The purpose of privacy-preserving record linkage is to ensure anonymity. Each block contains at least k records, and the block size must have both an upper limit and a lower limit. [65] proposes a novel hierarchical clustering approach, namely Size-Based Block Clustering. This method divides the data into independent blocks by using the initial blocking key value. If some blocks are too small, they are merged according to a similar function. If blocks are too large, it uses other blocking key values for division. This method can form blocks in a specific size range. In addition, the paper proposes a cost function for achieving a balance between block quality and block size.

4.1.2 Dynamic Methods

Many previous algorithms deal with each block individually. However, the results of entity resolution in one block can be used in processing other blocks. The result of matching and merging two records in the same block will match with the records in other blocks. The processing of the record pair in one block can reduce the processing time of other blocks. Based on the above findings, [66] proposes the IteratIve Blocking method. When records r and s match successfully, they will be merged into a new record $<r, s>$. Next, the method replaces these two records in all blocks containing r, s with $<r, s>$, and $<r, s>$ will continue to be compared with other records. New record pairs are generated in the process. In this case, the PC value will be greatly improved. Since the algorithm avoids redundant comparison of the same pair of records in different blocks, the PQ and RR values will also be improved.

The same record pair may appear in different blocks, but these record pairs only need to be compared once. If the algorithm can find these duplicate record pairs early, it can reduce the number of duplicate comparisons. For this purpose, [34] proposes the Block Scheduling method. This method aims to sort all the block elements in the block set B, and finds a block processing order that can reduce the total number of comparisons. Each $b_i \in B$ is assigned a utility value. In the block b_i, the number of comparisons in other blocks is processed by the propagation operation. The proportion of all comparison times of b_i block can be expressed by the following formula [34]:

$$u_i = \frac{gain_i}{cost_i}.$$

Suppose the utility value is recorded as u_i, $gain_i$ represents the benefit brought by processing b_i blocks for processing other blocks, and $cost_i$ represents the cost of processing b_i blocks. [19] also gives the derivation of the approximate calculation of the formula. The algorithm sorts the processing order of all blocks according to the calculated utility value.

After proposing the Block Scheduling method, [34] finds that the block ranked at the end of the processing order contains few duplicates and is more computationally expensive. Duplicate propagation operation results in few duplicates. [34] concludes that the later the block in the processing order, the lower its processing value.

Then the block pruning method was proposed. This method sets a parameter duplicate overhead, which indicates the computational cost to find true matches. Once the parameter reaches a predetermined threshold, the calculation stops.

4.2 Comparison Refinement

4.2.1 Comparison Propagation

The accuracy of many entity resolution blocking algorithms is increased by adding redundant comparisons. To reduce the number of comparisons of the algorithm and reduce the computational cost, [19] proposes the comparison propagation method. Due to the fact that the same record pair may be calculated multiple times in different blocks, this method propagates the calculated record pair into the subsequent processed blocks. It temporarily stores the processed records in memory when processing datasets with a small amount of data, but it cannot handle datasets with a large amount of data. To solve this problem, [19] establishes an inverted index for each entity and its own block. Only when n is the smallest common block ID of the entities e_i and e_j, will e_i and e_j be compared in block b_n. As shown in Fig.2, the smallest common block ID of e_2 and e_4 is 2. This pair of entities will only be compared in b_2, not in b_4 or b_5. The method can improve the PQ and RR values without affecting the PC value, but it has a high spatial complexity.

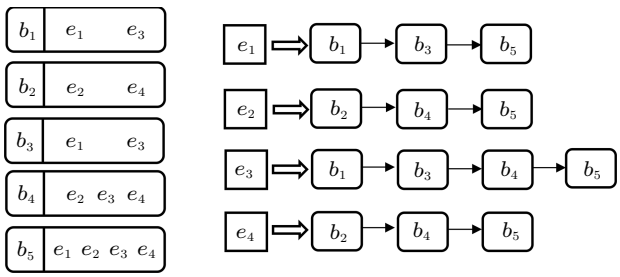

Fig.2. Block collection and corresponding entity index [15].

The comparison propagation method can avoid repeated comparisons of the same pair of entities, but it cannot avoid the comparison of two entities that are not successfully matched. Later, many methods adopted the comparison propagation technique proposed by [19], hoping to further reduce unnecessary comparisons.

4.2.2 Comparison Pruning

For the redundancy-positive blocking techniques, if the number of common blocks to which two entities belong is greater, the two entities are more likely to match successfully [50]. Based on this principle, [49] proposes the comparison pruning method. This method uses the Jaccard similarity metric to determine whether the pair of entities is likely to match before performing an exact comparison of the two entities. When the similarity of a pair of entities is lower than the threshold, no costly comparison is performed. Unlike block pruning, the comparison pruning method does not operate at the block level, but operates at the individual comparisons level.

4.2.3 Meta-Blocking

In terms of the redundancy-positive blocking techniques, [50] proposes the Meta-Blocking method. The core idea is to use the information extracted from the block-to-entity relationship to find the most similar entity pair. In essence, this method restructures a given block set into a new block set, where the new set contains as few comparisons as possible and its validity is unchanged. The method converts a block set into a graph structure representation. Nodes represent entity profiles, and undirected edges represent metrics between two entities. There are different specific schemes for measurement rules. In the process of creating the graph, all redundant comparisons can be discarded. Superfluous comparisons can be removed by the pruning strategy in Fig.3. Fig.3(a) is the profile of four entities, and Fig.3(b) is the block generated by the entity according to different BKVs. Assume that the edge weights are represented by the number of blocks to which two adjacent entities belong together, and Fig.3(c) is a graph based on the blocks.

On the basis of the blocking graph, [50] proposes four pruning strategies. The Weight Edge Pruning (WEP) strategy sets a global weight threshold. This method discards edges with weights less than the threshold, and finally outputs edges that meet the conditions. The Cardinality Edge Pruning (CEP) strategy sorts the edges by weight, and selects the K edges with the largest weight as the output. Weight Node Pruning (WNP) applies WEP to the neighborhood of each node (such as the subgraph of each node), and replaces the remaining edges with directed edges. Each neighborhood uses a different local weight threshold. The Cardinality Node Pruning (CNP) strategy selects K neighboring nodes with the highest weight for each node, and also replaces the remaining edges with directed edges.

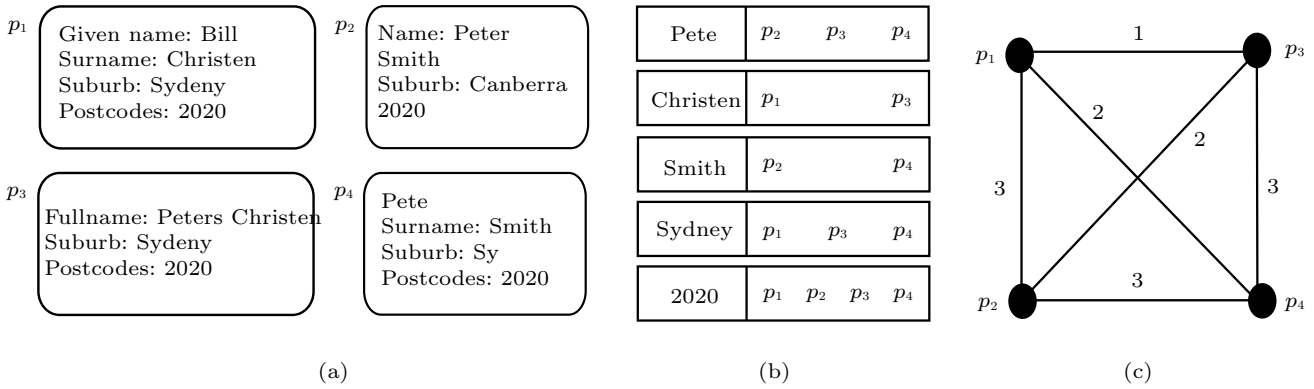

Fig.3. (a) Set of heterogeneous entity profiles. (b) Block collection B. (c) Blocking graph G_B [15].

4.2.4 *Canopy Clustering*

The Canopy Clustering algorithm proposed by [28] uses rough calculations at a single comparisons level to reduce the overall comparison time, which can also be seen as a pruning algorithm. First, the algorithm sets the similarity threshold, and puts all entities into the candidate pool. Then it randomly selects one entity to compare it with other entities at a low computational cost, thus eliminating superfluous comparisons. As the Canopy Clustering algorithm is sensitive to the threshold, [12] has extended it. Each iteration selects n_1 entities that are most similar to the specified entity, and removes n_2 ($< n_1$) entities from the candidate pool.

4.2.5 *Spectral Neighborhood (SPAN)*

To reduce the number of costly comparisons when processing large-scale datasets, [67] uses the clustering method and proposes the Spectral Neighborhood (SPAN) algorithm. The algorithm represents the block set as a matrix. The matrix is converted to a binary tree by using the spectral clustering algorithm [68]. The root node contains all entities, and the leaf nodes represent a subset of entities that are not included in each other. The algorithm introduces Newman-Girvan modularity as a stopping criterion for blocking. This method does not need to calculate a complex similarity matrix, but obtains blocks in the search process based on the tree structure.

5 Direction and Trend

There are many promising directions in blocking for future work, and we have mainly selected the blocking for real-time ER and incremental ER for introduction. Finally, due to the importance and wide application of entity resolution, we also give some other promising directions in the overall ER for future work.

5.1 Blocking for Real-Time ER

Traditionally, entity resolution has been applied on static databases; however more and more applications with large databases have a stream of query records that need to be matched in real time. Real-time entity resolution is the process of matching query records in (ideally) sub-second time with records in the available entity collections that represent the same real-world entity. To meet the requirements, [69] proposes an algorithm based on inverted index and [70] develops the technique and proposes the dynamic similarity-aware inverted indexing technique. [71, 72] convert the sorted list of blocking keys into a braided AVL tree [73] and propose a blocking method based on sorted neighbourhood indexing technique. [74] also uses the sorted neighborhood-based real-time indexing technique [71] and proposes an unsupervised learning technique that automatically selects optimal blocking keys. [75] combines MinHash LSH with Sorted Neighborhood and proposes a noise-tolerant approximate blocking approach that can be used in real-time ER.

We find that in order for the algorithm to solve the real-time ER, it is often necessary to incorporate some index or hash structure on the basis of the original algorithm. The existing algorithms are all aimed at structured data. However, we still need new algorithms to deal with the real-time ER in the context of HHIS such as linking datasets between Linked Open Data and Deep Web in real time. In the future, we can combine the techniques of hash or index with the schema-agnostic algorithms to solve the real-time ER problem of non-structure data.

5.2 Blocking for Incremental ER

In the big data era, the velocity of data updates is often high. Therefore, ER should not be a one-time process, but should be continuously improved with the update of linkage results. Specifically, to address incremental record linkage, we only need to compare the inserted record with the records of the dataset instead of comparing all the records in datasets. That means new algorithms that address the incremental ER should be devised.

R-Swoosh and F-Swoosh proposed by [76] are two of the first algorithms to address the incremental ER problems but both algorithms have quadratic cost and cannot scale to large datasets. Some existing incremental ER algorithms adopt distributed computational infrastructure to overcome the memory shortage caused by the incremental storage of existing blocks, such as PRIME proposed by [77] which is schema-agnostic and can deal with streaming and incremental data. Some use the index which supports efficient searching and incremental updates at lower time complexity such as incremental suffix arrays blocking proposed by [24] which uses suffix-based blocking method and StrB-tree. In the future, how to combine the above two technologies so that the algorithms can solve the incremental ER of large data volume efficiently is still worth exploring.

5.3 Work for the Overall ER

For the overall ER, there are many other promising research studies besides blocking. For example, in recent years, deep learning has provided impressive results in natural language processing and other fields. The novel ER system DEEPER, proposed by [78] is one of the attempts to combine deep learning with ER. [77] proposes an ER approach using word embedding technology and DNN models, and [79] analyzes the benefits and limitations of deep learning methods for solving ER problems. [80] proposes a methodology to produce explainable interpretations of the output of deep learning models for the ER task. Like the machine learning based ER, using deep learning to deal with the task of ER also faces the difficulty of tuning. At the same time, how to improve the interpretability of the model is also an urgent problem to be solved in the future. With the increase of the data, new progressive applications of ER have emerged and the goal is to provide the best possible solution within a limited budget of computational or time resources. [81,82] propose the scheme-aware methods based on the progressive sorted neighborhood. [83] proposes the a progressive solution for multi-source ER over different entity types. [84] adopts the MapReduce parallelization framework on the progressive ER method and the result achieves a higher efficiency.

In addition, all such as collective entity resolution [85–89], crowdsourced entity resolution [90–95], parameter configuration [96,97] and blocking methods that exploit entity evolution [98] are worthy of research in the future.

6 Conclusions

Entity resolution has been extensively studied for many years, and the blocking technology has always been an important tool to improve the efficiency of ER tasks. In this survey, we focused on blocking technology and used easy-to-understand methods to display and analyze the existing block technologies from multiple perspectives. We analyzed all typical blocking algorithms from the perspective of data structure, and displayed their respective characteristics. We classified the blocking construction techniques according to their essential characteristics and summarized them from different perspectives. In order to enable the algorithm to deal with more complex and larger datasets, many methods that optimize the blocking technology have been proposed, and we divided these methods into block-refinement methods and comparison-refinement methods. We found that in order to finish the task of ER, the traditional blocking ER methods which are designed to process the structured data, such as Standard Blocking and Suffix Arrays Blocking, construct the special structure arrays by sorting or creating auxiliary table based on the attribute values with some pre-defined schema to create the block. These methods may not work in the context of HHIS nowadays. In fact, combining and using the techniques flexibly such as the sliding window and the index of tree will make the methods still have the ability to meet the needs of new applications today. Discarding the schema like Token Blocking makes the algorithm not only able to handle non-structured data, but also improves the robustness. However, the algorithm also increases the unnecessary comparisons. In fact, we found that we do not have to completely discard the information in the attribute, which can improve the performance of the algorithm, such as the Attribute Clustering Blocking algorithm. At the same time, extracting the relationship information in data entities can make better use of new data

features in the context of HHIS with large amounts of data. We can also use the methods such as incorporating the external knowledge to further improve the algorithm. Machine learning based ER algorithms not only improve the efficiency of the ER, but also provide an attractive methodology for data mining tasks. Most of them face difficulties in parameter turning. At the same time, most of the unsupervised machine learning ER algorithms just use unsupervised methods to generate label data before learning the blocking functions. We still need the end-to-end unsupervised machine learning ER algorithm. Finally, we summarized the existing techniques and gave some promising directions for future work. For example, we still need efficient solutions to deal with the non-structured data in real-time ER. In the field of incremental ER, combining the index and distributed computing can further improve the efficiency of the algorithms. To use deep learning, we need to reduce the difficulty of tuning and improve the interpretability of the model.

References

[1] Bhattacharya I, Getoor L. A Latent Dirichlet model for unsupervised entity resolution. In *Proc. the 2006 SIAM International Conference on Data Mining*, Apr. 2006, pp.47-58.

[2] Liu X L, Wang H Z, Li J Z, Gao H. EntityManager: Managing dirty data based on entity resolution. *J. Comput. Sci. Technol.*, 2017, 32(3): 644-662.

[3] Winkler W E. Methods for evaluating and creating data quality. *Inf. Syst.*, 2004, 29(7): 531-550.

[4] Winkler W E. Overview of record linkage and current research directions. Technical Report, U.S. Bureau of the Census, 2006. http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=7D191FC85CD0D418884ACD2CECC2C190?doi=1-0.1.1.79.1519&rep=rep1&type=pdf, March 2020.

[5] Newcombe H B, Kennedy J M, Axford S J, James A P. Automatic linkage of vital records. *Science*, 1959, 130(3381): 954-959.

[6] Bhattacharya I, Getoor L. Iterative record linkage for cleaning and integration. In *Proc. the 9th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, Jun. 2004, pp.11-18.

[7] Pasula H, Marthi B, Milch B, Russell S J, Shpitser I. Identity uncertainty and citation matching. In *Proc. the 2002 Annual Conference on Neural Information Processing Systems*, Dec. 2002, pp.1401-1408.

[8] Fan W F, Jia X B, Li J Z, Ma S. Reasoning about record matching rules. *Proc. the VLDB Endowment*, 2009, 2(1): 407-418.

[9] Bilenko M, Mooney R J. Adaptive duplicate detection using learnable string similarity measures. In *Proc. the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Aug. 2003, pp.39-48.

[10] Bhattacharya I, Getoor L. Deduplication and group detection using links. In *Proc. the 2004 ACM SIGKDD Workshop on Link Analysis and Group Detection*, August 2004.

[11] Getoor L, Machanavajjhala A. Entity resolution: Theory, practice & open challenges. *Proc. the VLDB Endowment*, 2012, 5(12): 2018-2019.

[12] Christen P. A survey of indexing techniques for scalable record linkage and deduplication. *IEEE Transactions on Knowledge and Data Engineering*, 2011, 24(9): 1537-1555.

[13] Christen P. Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection. Springer, 2012.

[14] Dong X L, Srivastava D. Big data integration. In *Proc. the 29th International Conference on Data Engineering*, Apr. 2013, pp.1245-1248.

[15] Papadakis G, Skoutas D, Thanos E, Palpanas T. A survey of blocking and filtering techniques for entity resolution. arXiv:1905.06167, 2019. https://arxiv.org/abs/1905.06167, March 2020.

[16] Dunn H L. Record linkage. *American Journal of Public Health and the Nation's Health*, 1946, 36(12): 1412-1416.

[17] Christophides V, Efthymiou V, Stefanidis K. Entity resolution in the web of data. *Synthesis Lectures on the Semantic Web*, 2015, 5(3): 1-122.

[18] Papadakis G, Ioannou E, Palpanas T, Niederée C, Nejdl W. A blocking framework for entity resolution in highly heterogeneous information spaces. *IEEE Transactions on Knowledge and Data Engineering*, 2012, 25(12): 2665-2682.

[19] Papadakis G, Ioannou E, Niederée C, Palpanas T, Nejdl W. Eliminating the redundancy in blocking-based entity resolution methods. In *Proc. the 11th Annual International ACM/IEEE Joint Conference on Digital Libraries*, Jun. 2011, pp.85-94.

[20] Papadakis G, Alexiou G, Papastefanatos G, Koutrika G. Schema-agnostic vs schema-based configurations for blocking methods on homogeneous data. *Proc. the VLDB Endowment*, 2015, 9(4): 312-323.

[21] Fellegi I P, Sunter A B. A theory for record linkage. *Journal of the American Statistical Association*, 1969, 64(328): 1183-1210.

[22] Aizawa A, Oyama K. A fast linkage detection scheme for multi-source information integration. In *Proc. the 2005 International Workshop on Challenges in Web Information Retrieval and Integration*, Apr. 2005, pp.30-39.

[23] de Vries T, Ke H, Chawla S, Christen P. Robust record linkage blocking using suffix arrays. In *Proc. the 18th ACM Conference on Information and Knowledge Management*, Nov. 2009, pp.305-314.

[24] Allam A, Skiadopoulos S, Kalnis P. Improved suffix blocking for record linkage and entity resolution. *Data & Knowledge Engineering*, 2018, 117: 98-113.

[25] Gravano L, Ipeirotis P G, Jagadish H V, Koudas N, Muthukrishnan S, Srivastava D. Approximate string joins in a database (almost) for free. In *Proc. the 27th International Conference on Very Large Data Bases*, Sept. 2001, pp.491-500.

[26] Baxter R, Christen P, Churches T. A comparison of fast blocking methods for record linkage. In *Proc. the ACM SIGKDD 2003 Workshop on Data Cleaning, Record Linkage and Object Consolidation*, Aug. 2003.

[27] Kenig B, Gal A. MFIBlocks: An effective blocking algorithm for entity resolution. *Information Systems*, 2013, 38(6): 908-926.

[28] McCallum A, Nigam K, Ungar L H. Efficient clustering of high-dimensional data sets with application to reference matching. In *Proc. the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Aug. 2000, pp.169-178.

[29] Hernández M A, Stolfo S J. The merge/purge problem for large databases. *ACM SIGMOD Record*, 1995, 24(2): 127-138.

[30] Yan S, Lee D, Kan M Y, Giles L C. Adaptive sorted neighborhood methods for efficient record linkage. In *Proc. the 7th ACM/IEEE Joint Conference on Digital Libraries*, Jun. 2007, pp.185-194.

[31] Draisbach U, Naumann F. A comparison and generalization of blocking and windowing algorithms for duplicate detection. In *Proc. the International Workshop on Quality in Databases*, Aug. 2009, pp.51-56.

[32] Draisbach U, Naumann F, Szott S, Wonneberg O. Adaptive windows for duplicate detection. In *Proc. the 28th IEEE International Conference on Data Engineering*, Apr. 2012, pp.1073-1083.

[33] Draisbach U, Naumann F. A generalization of blocking and windowing algorithms for duplicate detection. In *Proc. the 2011 International Conference on Data and Knowledge Engineering*, Sept. 2011, pp.18-24.

[34] Papadakis G, Ioannou E, Niederée C, Fankhauser P. Efficient entity resolution for large heterogeneous information spaces. In *Proc. the 4th International Conference on Web Search and Web Data Mining*, Feb. 2011, pp.535-544.

[35] Papadakis G, Ioannou E, Niederée C, Palpanas T, Nejdl W. Beyond 100 million entities: Large-scale blocking-based resolution for heterogeneous data. In *Proc. the 5th International Conference on Web Search and Web Data Mining*, Feb. 2012, pp.53-62.

[36] Song D, Heflin J. Automatically generating data linkages using a domain-independent candidate selection approach. In *Proc. the 10th International Semantic Web Conference*, Oct. 2011, pp.649-664.

[37] Nin J, Muntes-Mulero V, Martínez-Bazan N, Larriba-Pey J. On the use of semantic blocking techniques for data cleansing and integration. In *Proc. the 11th International Database Engineering and Applications Symp.*, Sept. 2007, pp.190-198.

[38] Bilenko M, Kamath B, Mooney R J. Adaptive blocking: Learning to scale up record linkage. In *Proc. the 6th IEEE International Conference on Data Mining*, Dec. 2006, pp.87-96.

[39] Michelson M, Knoblock C A. Learning blocking schemes for record linkage. In *Proc. the 21st National Conference on Artificial Intelligence and the 18th Innovative Applications of Artificial Intelligence Conference*, Jul. 2006, pp.440-445.

[40] Evangelista L O, Cortez E, da Silva A S, Meira W. Adaptive and flexible blocking for record linkage tasks. *Journal of Information and Data Management*, 2010, 1(2): 167-167.

[41] Sarma A D, Jain A, Machanavajjhala A, Bohannon P. An automatic blocking mechanism for large-scale deduplication tasks. In *Proc. the 21st ACM International Conference on Information and Knowledge Management*, Oct. 2012, pp.1055-1064.

[42] Giang P H. A machine learning approach to create blocking criteria for record linkage. *Health Care Management Science*, 2015, 18(1): 93-105.

[43] Cao Y, Chen Z, Zhu J, Yue P, Lin C, Yu Y. Leveraging unlabeled data to scale blocking for record linkage. In *Proc. the 22nd International Joint Conference on Artificial Intelligence*, Jul. 2011, pp.2211-2217.

[44] Mayank K, Daniel P M. An unsupervised algorithm for learning blocking schemes. In *Proc. the 13th International Conference on Data Mining*, Dec. 2019, pp.340-349.

[45] O'Hare K, Jurek-Loughrey A, de Campos C. An unsupervised blocking technique for more efficient record linkage. *Data & Knowledge Engineering*, 2019, 122: 181-195.

[46] Ma Y T, Tran T. TYPiMatch: Type-specific unsupervised learning of keys and key values for heterogeneous web data integration. In *Proc. the 6th ACM International Conference on Web Search and Data Mining*, Feb. 2013, pp.325-334.

[47] Kejriwal M, Miranker D P. A two-step blocking scheme learner for scalable link discovery. In *Proc. the 9th International Workshop on Ontology Matching Collocated with the 13th International Semantic Web Conference*, Oct. 2014, pp.49-60.

[48] Kejriwal M, Miranker D P. A DNF blocking scheme learner for heterogeneous datasets. arXiv:1501.01694. https://arxiv.org/abs/1501.01694, Jan. 2020.

[49] Papadakis G, Ioannou E, Niederée C, Palpanas T, Nejdl W. To compare or not to compare: Making entity resolution more efficient. In *Proc. the International Workshop on Semantic Web Information Management*, Jun. 2011, Article No. 3.

[50] Papadakis G, Koutrika G, Palpanas T, Nejdl W. Meta-blocking: Taking entity resolution to the next level. *IEEE Transactions on Knowledge and Data Engineering*, 2013, 26(8): 1946-1960.

[51] Ferragina P, Grossi R. The string B-tree: A new data structure for string search in external memory and its applications. *Journal of the ACM*, 1999, 46(2): 236-280.

[52] Christen P. Towards parameter-free blocking for scalable record linkage. Technical Report, Faculty of Engineering and Information Technology, 2007. http://users.cecs.anu.edu.au/~Peter.Christen/publications/tr-cs-07-03.pdf, March 2020.

[53] Madhavan J, Jeffery S R, Cohen S, Dong X L, Ko D, Yu C, Halevy A. Web-scale data integration: You can only afford to pay as you go. In *Proc. the 3rd Biennial Conference on Innovative Data Systems Research*, Jan. 2007, pp.342-350.

[54] Papadakis G, Palpanas T. Blocking for large-scale entity resolution: Challenges, algorithms, and practical examples. In *Proc. the 32nd International Conference on Data Engineering*, May 2016, pp.1436-1439.

[55] Chaudhuri S, Chen B, Ganti V, Kaushik R. Example-driven design of efficient record matching queries. In *Proc. the 33rd International Conference on Very Large Data Bases*, Sept. 2007, pp.327-338.

[56] Papadakis G, Demartini G, Fankhauser P, Kärger P. The missing links: Discovering hidden same-as links among a billion of triples. In *Proc. the 12th International Conference on Information Integration and Web-Based Applications and Services*, Nov. 2010, pp.453-460.

[57] Winkler W E. Approximate string comparator search strategies for very large administrative lists. Technical Report, U.S. Census Bureau, 2005. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.79.402&rep=rep1&type=pdf, March 2020.

[58] Valiant L G. A theory of the learnable. *Communications of the ACM*, 1984, 27(11): 1134-1142.

[59] Suchanek F M, Abiteboul S, Senellart P. PARIS: Probabilistic alignment of relations, instances, and schema. *Proceedings the VLDB Endowment*, 2011, 5(3): 157-168.

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

[61] Jonker R, Volgenant T. Improving the Hungarian assignment algorithm. *Operations Research Letters*, 1986, 5(4): 171-175.

[62] Verikas A, Gelzinis A, Bacauskiene M. Mining data with random forests: A survey and results of new tests. *Pattern Recognition*, 2011, 44(2): 330-349.

[63] Bilke A, Naumann F. Schema matching using duplicates. In *Proc. the 21st International Conference on Data Engineering*, Apr. 2005, pp.69-80.

[64] Papadakis G, Papastefanatos G, Palpanas T, Koubarakis M. Scaling entity resolution to large, heterogeneous data with enhanced meta-blocking. In *Proc. the 19th International Conference on Extending Database Technology*, Mar. 2016, pp.221-232.

[65] Fisher J, Christen P, Wang Q, Rahm E. A clustering-based framework to control block sizes for entity resolution. In *Proc. the 21st International Conference on Knowledge Discovery and Data Mining*, Aug. 2015, pp.279-288.

[66] Whang S E, Menestrina D, Koutrika G, Theobald M, Garcia-Molina H. Entity resolution with iterative blocking. In *Proc. the International Conference on Management of Data*, Jun. 2009, pp.219-232.

[67] Shu L, Chen A, Xiong M, Meng W. Efficient SPectrAl Neighborhood blocking for entity resolution. In *Proc. the 27th International Conference on Data Engineering*, Apr. 2011, pp.1067-1078.

[68] Shi J, Malik J. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2000, 22(8): 888-905.

[69] Christen P, Gayler R, Hawking D. Similarity-aware indexing for real-time entity resolution. In *Proc. the 18th ACM Conference on Information and Knowledge Management*, Nov. 2009, pp.1565-1568.

[70] Ramadan B, Christen P, Liang H, Gayler R W, Hawking D. Dynamic similarity-aware inverted indexing for real-time entity resolution. In *Proc. the 2013 PAKDD Workshop on Data Mining Applications in Industry and Government*, Apr. 2013, pp.47-58.

[71] Ramadan B, Christen P. Forest-based dynamic sorted neighborhood indexing for real-time entity resolution. In *Proc. the 23rd International Conference on Information and Knowledge Management*, Nov. 2014, pp.1787-1790.

[72] Ramadan B, Christen P, Liang H, Gayler R W. Dynamic sorted neighborhood indexing for real-time entity resolution. *Journal of Data and Information Quality*, 2015, 6(4): Article No. 15.

[73] Rice S V. Braided AVL trees for efficient event sets and ranked sets in the SIMSCRIPT III simulation programming language. In *Proc. the 2007 Western Multiconference on Computer Simulation*, Jan. 2007, pp.150-155.

[74] Ramadan B, Christen P. Unsupervised blocking key selection for real-time entity resolution. In *Proc. the 19th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, May 2015, pp.574-585.

[75] Liang H, Wang Y, Christen P, Gayler R. Noise-tolerant approximate blocking for dynamic real-time entity resolution. In *Proc. the 18th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, May 2014, pp.449-460.

[76] Benjelloun O, Garcia-Molina H, Menestrina D, Su Q, Whang S E, Widom J. Swoosh: A generic approach to entity resolution. *The VLDB Journal*, 2009, 18(1): 255-276.

[77] Araújo B T, Stefanidis K, Pires C E S, Nummenmaa J, da Nóbrega P T. Incremental blocking for entity resolution over web streaming data. In *Proc. the 2019 IEEE/WIC/ACM International Conference on Web Intelligence*, Oct. 2019, pp.332-336.

[78] Ebraheem M, Thirumuruganathan S, Joty S, Ouzzani M, Tang N. Distributed representations of tuples for entity resolution. *Proc. the VLDB Endowment*, 2018, 11(11): 1454-1467.

[79] Mudgal S, Li H, Rekatsinas T, Doan A, Park Y, Krishnan G, Deep R, Arcaute E, Raghavendra V. Deep learning for entity matching: A design space exploration. In *Proc. the 2018 International Conference on Management of Data*, Jun. 2018, pp.19-34.

[80] di Cicco V, Firmani D, Koudas N, Merialdo P, Srivastava D. Interpreting deep learning models for entity resolution: An experience report using LIME. In *Proc. the 2nd International Workshop on Exploiting Artificial Intelligence Techniques for Data Management*, Jul. 2019, Article No. 8.

[81] Papenbrock T, Heise A, Naumann F. Progressive duplicate detection. *IEEE Transactions on Knowledge and Data Engineering*, 2014, 27(5): 1316-1329.

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

[83] Altowim Y, Kalashnikov D V, Mehrotra S. Progressive approach to relational entity resolution. *Proc. the VLDB Endowment*, 2014, 7(11): 999-1010.

[84] Altowim Y, Mehrotra S. Parallel progressive approach to entity resolution using MapReduce. In *Proc. the 33rd IEEE International Conference on Data Engineering*, Apr. 2017, pp.909-920.

[85] Bhattacharya I, Getoor L. Collective entity resolution in relational data. *ACM Transactions on Knowledge Discovery from Data*, 2007, 1(1): Article No. 5.

[86] Globerson A, Lazic N, Chakrabarti S, Subramanya A, Ringaard M, Pereira F. Collective entity resolution with multi-focal attention. In *Proc. the 54th Annual Meeting of the Association for Computational Linguistics*, Aug. 2016.

[87] Kouki P, Pujara J, Marcum C, Koehly L, Getoor L. Collective entity resolution in familial networks. In *Proc. the 2017 IEEE International Conference on Data Mining*, Nov. 2017, pp.227-236.

[88] de Assis Costa G, de Oliveira J M P. A relational learning approach for collective entity resolution in the web of data. In *Proc. the 5th International Workshop on Consuming Linked Data Co-Located with the 13th International Semantic Web Conference*, Oct. 2014.

[89] Kouki P, Pujara J, Marcum C, Koehly L, Getoor L. Collective entity resolution in multi-relational familial networks. *Knowledge and Information Systems*, 2019, 61(3): 1547-1581.

[90] Wang J, Kraska T, Franklin M J, Feng J. CrowdER: Crowdsourcing entity resolution. arXiv:1208.1927, 2012. http://arxiv.org/abs/1208.1927, Aug. 2018.

[91] Vesdapunt N, Bellare K, Dalvi N. Crowdsourcing algorithms for entity resolution. *Proc. the VLDB Endowment*, 2014, 7(12): 1071-1082.

[92] Gong S S, Hu W, Ge W Y, Qu Y Z. Modeling topic-based human expertise for crowd entity resolution. *Journal of Computer Science and Technology*, 2018, 33(6): 1204-1218.

[93] Zhang A Z, Li J Z, Gao H, Chen Y B, Ma H Z, Bah M J. CrowdOLA: Online aggregation on duplicate data powered by crowdsourcing. *Journal of Computer Science and Technology*, 2018, 33(2): 366-379.

[94] Mazumdar A, Saha B. A theoretical analysis of first heuristics of crowdsourced entity resolution. In *Proc. the 31st AAAI Conference on Artificial Intelligence*, Feb. 2017, pp.970-976.

[95] Chai C, Li G, Li J, Deng D, Feng J. A partial-order-based framework for cost-effective crowdsourced entity resolution. *The VLDB Journal*, 2018, 27(6): 745-770.

[96] Maskat R, Paton N W, Embury S M. Pay-as-you-go configuration of entity resolution. *Transactions on Large-Scale Data-and Knowledge-Centered Systems*, 2016, 29: 40-65.

[97] Li H, Konda P, G.C. P S, Doan A, Snyder B, Park, Y, Krishnan G, Deep R, Raghavendra V. MatchCatcher: A debugger for blocking in entity matching. In *Proc. the 21st International Conference on Extending Database Technology*, Mar. 2018, pp.193-204.

[98] Papadakis G, Giannakopoulos G, Niederée C, Palpanas T, Nejdl W. Detecting and exploiting stability in evolving heterogeneous information spaces. In *Proc. the 11th Annual International ACM/IEEE Joint Conference on Digital Libraries*, Jun. 2011, pp.95-104.

Bo-Han Li received his Ph.D. degree in computer application from Harbin University of Science and Technology, Harbin, in 2009. He is currently an associate professor at the College of Computer Science and Technology of Nanjing University of Aeronautics and Astronautics (NUAA), Nanjing. His current research interests include spatiotemporal database, recommendation system, sentiment analysis, etc. He is a member of CCF and ACM.

Yi Liu received his B.E. degree in computer science and technology from Nanjing Normal University, Nanjing, in 2019. Now he is a Master student of Nanjing University of Aeronautics and Astronautics, Nanjing. His research is about artificial intelligence, knowledge graph and spatio-temporal data mining. He is a student member of CCF.

An-Man Zhang received her B.E. degree in information management and information system from Jiangsu University of Technology, Changzhou, in 2017. Now she is a Master student of Nanjing University of Aeronautics and Astronautics, Nanjing. Her research is about social network, text sentiment analysis and crowdsourcing. She is a student member of CCF.

Wen-Huan Wang received her B.E. degree in information management and information system from Jiangsu University of Technology, Changzhou, in 2018. Now she is a Master student of Nanjing University of Aeronautics and Astronautics, Nanjing. Her research is about social network, text sentiment analysis and knowledge graph. She is a student member of CCF.

Shuo Wan received his B.E. degree in computer science and technology from East China Jiaotong University, Nanchang, in 2017. Now he is a Master student of Nanjing University of Aeronautics and Astronautics, Nanjing. His research interests are sentiment analysis and spatio-temporal data mining. He is a student member of CCF.