Hindawi
Journal of Sensors
Volume 2018, Article ID 2094696, 12 pages
https://doi.org/10.1155/2018/2094696

Research Article

A Type-Based Blocking Technique for Efficient Entity Resolution over Large-Scale Data

Hui-Juan Zhu,[1] **Zheng-Wei Zhu**,[1] **Tong-Hai Jiang**,[2,3] **Li Cheng**,[2,3] **Wei-Lei Shi**,[2] **Xi Zhou**,[2,3] **Fan Zhao**,[2,3] **and Bo Ma**[2,3]

[1]*School of Information Science & Engineering, Changzhou University, Changzhou 213164, China*
[2]*The Xinjiang Technical Institute of Physics and Chemistry, Chinese Academy of Sciences, Urumqi 830011, China*
[3]*Xinjiang Laboratory of Minority Speech and Language Information Processing, Urumqi 830011, China*

Correspondence should be addressed to Zheng-Wei Zhu; zhuzw@cczu.edu.cn and Tong-Hai Jiang; jth@ms.xjb.ac.cn

Received 21 April 2017; Accepted 11 September 2017; Published 28 January 2018

Academic Editor: Qing Tan

In data integration, entity resolution is an important technique to improve data quality. Existing researches typically assume that the target dataset only contain string-type data and use single similarity metric. For larger high-dimensional dataset, redundant information needs to be verified using traditional blocking or windowing techniques. In this work, we propose a novel ER-resolving method using a hybrid approach, including type-based multiblocks, varying window size, and more flexible similarity metrics. In our new ER workflow, we reduce the searching space for entity pairs by the constraint of redundant attributes and matching likelihood. We develop a reference implementation of our proposed approach and validate its performance using real-life dataset from one Internet of Things project. We evaluate the data processing system using five standard metrics including effectiveness, efficiency, accuracy, recall, and precision. Experimental results indicate that the proposed approach could be a promising alternative for entity resolution and could be feasibly applied in real-world data cleaning for large datasets.

1. Introduction

Entity resolution (ER), also known as record linkage, entity reconciliation, or merge/purge, is the procedure of identifying a group of entities (records) representing the same real-world entity [1–3]. Generally speaking, ER has become the first step of data processing and widely used in many application domain, such as digital libraries, smart city, financial transactions, and social networks. Especially with the rapid development of Internet of Things technology, it is common that data contains a huge amount of inaccurate information and different types of ambiguities [4]. Accordingly, developing proper ER techniques to clear and integrate data collected from multiple sources has received much attention [1, 5, 6]. The ultimate goal of ER technologies is to improve data quality or to enrich data to facilitate more detailed data analysis.

Researchers have proposed many automatic methods and techniques to resolve ER problem across multiple resources, to detect if they refer to the same entity and therefore can be merged. Wang and Madnick [7] proposed a rule-based method by using rules and unique key attribute developed by experts. However, this kind of method has some additional restrictions, such as the result of the rules must always be correct. Bilenko and Mooney [8] proposed an adaptive duplicate detection method, but it mainly focus on string similarity measures. Recently, based on different similarity metrics, some researchers proposed machine learning methods to classify the entity pairs to "match," "non-match," or "possible-match", such as the [9] proposed automatic record linkage tool by using of support vector machine classification of Christen. However, this kind of methods requires a large amount of manually labelled data, and if the entity pairs are classified to "possible-match," a manual review process is required to assess and classify them into "match" or "non-match." Thus, this is usually a time-consuming, cumbersome, and error-prone process [10].

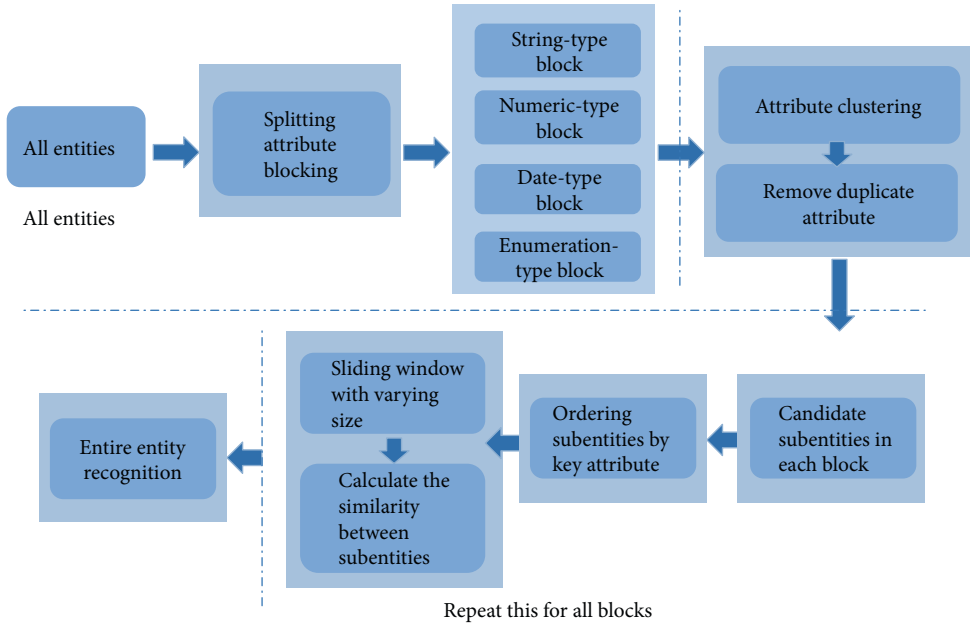

Figure 1: The Overall framework of the proposed entity-resolution technique.

The mentioned work in ER mainly focused on the development of automatic algorithms [11]. A general procedure for solving the ER problem includes calculating the similarity of all entity pairs by means of similarity metric methods [12], such as Jaccard similarity coefficient and Levenshtein distance. Entities whose similarity values are higher than the specified threshold are considered to be the same entity. However, in the face of large datasets, the performance deteriorates drastically as a comparison of dense attributes [2]. To solve this problem, blocking and windowing mechanisms [2, 10, 13, 14] have been introduced. The goal of the blocking scheme is to group entities into block-based clustering technique to reduce comparison searching space into one block. Windowing methods, such as sorted neighborhood method (SNM) [13, 15] or multipass sorted neighborhood method (MPN) [16, 17], sort entities according to the keywords, and then slide a fixed size window over them to compare the entities within the window.

Algorithmic approaches have been improving in quality, but there are still some problems that have not been fully studied [1]. First, existing research typically assumes that the target dataset only contain string-type data and use single similarity metric [1, 11, 18, 19]. Second, for large high-dimensional target dataset, redundant information is verified, which not only increases the computational complexity but also may deteriorate the quality of ER results [20]. Third, the common drawback of most blocking techniques is that they put the complete entities in one block (or multiple blocks) according to the block keywords, which leads to the lack of the necessary flexibility for composing ER workflows of higher performance in combination with specialized, complementary methods [5, 21]. Fourth, the performance of typical windowing methods (SNM or MPN) depends strongly on the size of sliding window [21], but they often employ a fixed window size. The larger the fixed window size, the more comparisons are executed, and the lower the overall efficiency gets, however, small size may lead to a high number of missed matches (e.g., the closest entities are not placed in the same window) and to low effectiveness [2, 21–23].

Based on above problems, this paper introduces a novel ER resolving method. First, we introduce a novel blocking scheme based on attribute value types. In these different type of blocks, we adopt diverse similarity metrics to achieve maximum efficiency gains, because even though a metric method showed robust and high performance for one type of data [24, 25], it may perform poorly on others [26–28]. Second, we provide an attribute clustering method, a novel and effective blocking approach, which is a preprocessing scheme to resolve ER in a significantly lower redundancy and higher efficiency. Its core is to divide attribute names into nonoverlapping clusters according to the similarity or correlation of the attribute values. Third, we introduce a comparison mechanism, which combines a dynamically adjustable window size to specify the processing order of all individual comparisons in each block. It aims at identifying the closest pairs of entities involving significantly fewer comparisons while maintaining the original level of effectiveness. Fourth, during the ER completion step, we adopt a weighted undirected graph to gather the output of each block and used the improved weighting and pruning scheme to enhance its efficiency. Finally, we assess the performance of our methods with using a real-life dataset, and the experimental results validate the exceptional performance of our methods.

2. The Overall Framework

The overall framework of the proposed ER resolving method is shown in Figure 1. It consists of three parts: the first part

divides the complete entities into different blocks according to the attribute value types; the second part introduces the method of attribute clustering, which is used to remove redundant attributes; the third part performs comparisons solely between the subentities within the scalable window in each block and gathers the outputs from individual blocks to finish the ER problem.

3. Type-Based Blocking Approach

At the core of our approach lies in the notion of type-based blocking (defined as Definition 1), varying window size and blocking graph. Given an entity set R which is defined as definition 2, the common attribute value types include string, numerical, enumeration, and date. In order to avoid the defects of fixed window size, the proposed method uses different dynamic scalable window strategies (refer to Section 3.3.2) in different type of blocks, also known as varying window size. In addition, given a Map *closestMap*, which is used to store the closest entities, the corresponding blocking graph G_e (also known as edge-weight graph) is constructed according to the value of *closestMap*: each entity identifier contained in *closestMap* is mapped to a node in the blocking graph, and each pair of cooccurring entities (e.g., entities that are marked as matched at least in one block) is connected with an undirected edge.

Definition 1. (Type-Based Blocking) In this paper, according to attribute value types, the dataset is divided into the following blocks: numeric-type block, string-type block, date-type block, and enumeration-type block, and these blocks are nonoverlapping.

Definition 2. (Entity Set) Entity set is defined as $R = \{r\_1, r\_2, \ldots, r\_n\}$, and each entity can be expressed as $r\_i = \{x\_i1, x\_i2, \ldots, x\_im\}$ $(1 \le i \le n)$. The attribute vector is defined as $A = \{A\_1, A\_2, \ldots, A\_m\}$, and A_l $(1 \le l \le m)$ is used to represent the lth attribute. Accordingly, the value of A_l in entity r_i is represented as x_il $(1 \le l \le m,\ 1 \le i \le n)$.

Definition 3. (Blocking Graph) Given an entity collection *closestMap*, the undirected blocking graph $G\_e = \{V\_e, E\_e, \mathrm{WS}\}$ is generated by it, where V_e is the set of its nodes, E_e is the set of its undirected edges, and WS is the weighting scheme that determines the weight of every edge.

3.1. Splitting Attributes to Different Blocks. The comparison between attribute values is an important task for resolving the ER problem. A variety of methods has been developed for that [29–32], which typically rely on string comparison techniques. However, the dataset might contain other types of data, such as numerical, enumeration, and date, and there is still much work to be done about these types of data [33–35].

The purpose of splitting attributes into different blocks is to facilitate the introduction of a variety of flexible similarity metrics to identify redundant attributes or closest entities. For instance, entities (Jimi, F) and (Jimi, M), in which "F" and "M," respectively, represent the "female" and "male" for the gender attribute. The similarity between these two entities achieved by traditional methods, such as Levenshtein distance [36], is very high. However, these two entities represent two individuals with same name but with different gender. The attribute gender is an enumeration type, and the similarity of the enumeration type should be calculated using equal or unequal, rather than the methods often used for string-type data. The functionality of splitting attributes into different blocks is outlined in Algorithm 1.

3.2. Attribute Clustering Method. The increase of data dimensionality brings new challenges to the efficiency and effectiveness of many existing ER methods. In this article, we introduce an attribute clustering method to remove redundant attributes. As a result, we obtain a more compact and easily interpretable representation of the target concept. The core of the proposed attribute clustering method is to divide attributes into disjoint clusters according to the similarity or correlation of the attribute values within the same type block, which is based on the concept of redundant. This scheme offers a better balance between the computational cost and the precision for resolving ER [37]. In order to facilitate the discussion, the following definitions are introduced.

3.2.1. String-Type Block. The attribute clustering method for string type depends on two parts: (1) the model that congruously represents the values of attribute; (2) the similarity metric that catches the common pattern between the two sets of attribute values. In this paper, the weight of each term is obtained by TF-IDF (term frequency-inverse document frequency) algorithm [38]. Accordingly, the two sets of attribute values can be represented as $v\_i = \{wi\_1, wi\_2, \ldots, wi\_n\}$ and $v\_j = \{wj\_1, wj\_2, \ldots, wj\_n\}$, and θ is the angle between v_i and v_j. Thus, the similarity between two sets of attribute values with cosine similarity can be defined as

$$\mathrm{sim}(v\_i, v\_j) = \cos\theta = \frac{\sum_{t=1}^{n} (wi\_t . wj\_t)}{\sqrt{\sum_{t=1}^{n} wi\_t^2 . \sum_{t=1}^{n} wj\_t^2}}, \quad (1)$$

where $\mathrm{sim}(v\_i, v\_j)$ takes values in the interval [0, 1], with higher values indicating higher similarity between the given attributes. If it is higher than an established threshold, that means the given attributes are redundant.

3.2.2. Numeric-Type Block. In this paper, the numerical attributes are divided into three categories [34]: (1) nominal attribute, it refers to only a set of numerical symbols, such as ID card number, which does not have a meaningful order, and is not quantitative; (2) ordinal attribute, its possible values have the meaningful orders or ranking, but the difference between successive is unknown; (3) general numerical attribute, its possible values are measurable, such as age and height. Assume that there are two numerical attributes A and B, and the values of A is expressed as $\{a\_1, a\_2, \ldots, a\_c\}$ and the values of B is expressed as $\{b\_1, b\_2, \ldots, b\_r\}$. The correlation between

```
Input: sd: dataset;
       δ: the threshold for the number of possible values of an enumeration attribute;
       λ: the number of values which are randomly selected from sd;
Output: Map < BT(block type), list of attribute names > BT∈{NUME, STRING, DATE, ENUM}
(a)  Map < attribute, List < v1, v2,…, vλ> > mediateData ← sd;// Using Map to store attribute and its
     values.
(b)  blockMap← new HashMap < String, List>; // blockMap is used to store the return value;
(c)  For each attribute in mediateData Do
(d)       valuesNoRep← Remove duplicate elements from List < v1,v2,…,vλ>;
(e)       n← valuesNoRep.size();
(f)       If ((double)n/λ < δ) then {//the type of this attribute is enumeration
(g)            List enumAttributes ← blockMap.get("ENUM");
(h)            If (enumAttributes == null) then {enumAttributes ← new ArrayList;
(i)                 blockMap.put("ENUM", enumAttributes);}
(j)            enumAttributes← enumAttributes.add(attribute name);
(k)       }Else if (the elements of valuesNoRep conform to the date type rules) then {
(l)            List dateAttributes ← blockMap.get("DATE");
(m)            If (dateAttributes == null) then {dateAttributes ← new ArrayList;
(n)                blockMap.put("DATE", dateAttributes);}
(o)            dateAttributes ← dateAttributes.add(attribute name);
(p)       }Else if (The elements of listWithoutDu conform to the numerical type rules) then {
(q)            List numericAttributes ← blockMap.get("NUME");
(r)            If (numericAttributes == null) then {numericAttributes ← new ArrayList;
(s))                blockMap.put("NUME", numericAttributes);}
(t)            numericAttributes ← numericAttributes. add(attribute name);
(u)       } Else {
(v)            //other attributes will be treated as string type
(w)            List stringAttributes ← blockMap.get("STRING");
(x)            If (stringAttributes == null) then {stringAttributes ← new ArrayList;
(y)                 blockMap.put("STRING", stringAttributes);}
(z)            stringAttributes ← stringAttributes.add(attribute name);
(aa)      }
(bb) End For each attribute in mediateData
(cc) return blockMap;
(Note: the δ and λ should be adjusted according to the size of dataset)
```

Algorithm 1: Splitting attributes into different blocks.

two nominal numerical attributes is discovered by χ^2 test [39], which is shown in

$$\chi^2 = \sum_{i=1}^{c}\sum_{j=1}^{r}\frac{(o\_ij - e\_ij)^2}{e\_ij}, \tag{2}$$

$$e\_ij = \frac{\text{count}(A = a\_i) \times \text{count}(B = b\_j)}{n}, \tag{3}$$

where o_ij is the observation frequency and e_ij is the expected frequency. The calculation method of e_ij can be referred to (3), where n is the number of data tuple and $\text{count}(A = a\_i)$ is the count of value a_i which appeared in the A attribute.

For the general numerical attribute, their correlation is measured by Pearson correlation coefficient [40], which is shown in

$$r\_A, B = \frac{\sum_{i=1}^{n}(a\_i - \bar{A})(b_i - \bar{B})}{n \cdot \sigma\_A \cdot \sigma\_B} = \frac{\sum_{i=1}^{n}(a\_i \cdot b\_i) - n\overline{A}\cdot\overline{B}}{n \cdot \sigma\_A \cdot \sigma\_B}, \tag{4}$$

where σ_A and σ_B are the standard deviation of attributes A and B. $\bar{A}$ and $\bar{B}$ are the mean values. r_A, B is their correlation coefficient, whose value space is $[-1,1]$. If r_A, B is bigger than the established threshold, that means one of the given attributes can be removed as redundant [34].

For the ordinal numerical attribute, it is divided three steps: (1) we sort and divide the values of attribute A into set $\{M\_1, M\_2, \ldots, M\_100\}$, also known as percentile [41, 42]. The processing of the attribute B is the same as that of A; (2) we count the number of attribute values that fall into each interval and use this count value ca_i to replace M_i $(1 \le i \le 100)$. Thus, the replacement set $\{ca\_1, ca\_1, \ldots, ca\_100\}$ and $\{cb\_1, cb\_1, \ldots, cb\_100\}$ can be obtained; (3) $\{ca\_1, ca\_1, \ldots, ca\_100\}$ and $\{cb\_1, cb\_1, \ldots, cb\_100\}$ can be regarded as the general numerical attribute to calculate their correlation.

3.2.3. Date-Type Block. The correlation between date-type attributes, which is the handling process, is same with that of the ordinal numerical attribute.

```
Input: Map < attribute name, List ⟨v_1, v_2, … , v_n⟩ >//this is for same type block
       δ: the threshold for the attributes similarity
Output: Set of attribute names clusters: C
(a) connects ←{};
(b) noOfAttributes ←the size of Map < attribute name, List ⟨v_1, v_2, … , v_n⟩ >;
(c) For (int i = 0; i < noOfAttributes; i++) do
(d)       For (int j = i + 1; j < noOfAttributes; j++) do
(e)             Sim_i, j ←attribute[i].getSimilarAttribute(attribute[j]);//refer to the formula (1)~(4)
(f)             If (Sim_i, j > δ) then connects ← connects.add(new Connect(i, j));
(g)       End For(j)
(h) End For (i)
(i) cons ←computerTransitiveClosure(connects);
(j) C←getConnectedComponents(cons);
(k) For each c_i ∈ C do
(l)       If (c_i.size() == 1) then C.remove(c_i);
(m) End For (c_i);
(n)  Return C;
```

Algorithm 2. Attribute clustering algorithm.

3.2.4. Enumeration-Type Block. For the numeration-type attributes, their correlation measurement is same with that of the nominal numerical attribute.

The detailed functionality of the proposed attribute clustering method is described in Algorithm 2, which is based on the method proposed by Papadakis et al. in research [43].

In principle, the attribute clustering method proposed in this paper works as follows: each attribute name in the input *map* is associated with the most similar or the strongest correlation attribute name (lines c–h). The *Connect* between two attribute names is stored in a data structure on the condition that the similarity or correlation of their values is more than the threshold δ (line f). The transitive closure of the stored *connects* is then obtained to build the basis for partitioning attribute names into cluster (line i). As a result, each cluster is taken from each connected component of the transitive closure (line j), and the singleton clusters are removed from C (lines k–m).

3.3. The Closest Entity Detection. SNM was first proposed in the mid-1990s [44]. Its basic idea is to sort the dataset (entity set) according to the sort keywords and to sequentially move a fixed size window over the sorted entities. Candidate entity pairs are then generated only from entities within the same window. This technique reduces the complexity from $O(n \times n)$ to $O(n \times w)$, where n is the number of input entities and w is the window size. The linear complexity makes SNM more robust against load balancing problems [16, 17]. However, this method still has a room for improvement, such as the following:

(a) When some attribute values are missing or the length difference between the attribute values is larger, the weighting summation method for calculating the similarity between entities will no longer appropriate.

(b) The interference caused by redundant attributes will increase the comparison's cost and affect and the quality of ER results [20].

(c) The comparison strategy in SNM is that when each new entity enters the current window, it needs to be compared with the previous w-1 (w is the window size) entities to find "matching" entities. However, it is difficult to determine the sliding window size [2].

(d) The real-life dataset often contains a variety of data types, such as enumeration, numerical, date, and string, but the traditional SNM usually assumes that the dataset only contains string-type data and just employ single similarity metric based on string type [11].

To solve the above problems, this paper proposes multiblocking sorted neighborhood (MBN) algorithm. In order to facilitate the discussion, the definition of valid weight is introduced.

Definition 4. (Valid Weight) When comparing the attribute values r_il and r_jl, if one of them is missing or their length ratio is lower than the specified threshold, the valid weight $\vartheta\_ij^{(l)}$ will be set to 0, the length ratio are shown in (5), and the length ratio is only adopted in string- and enumeration-type blocks:

$$\text{lenRatio} = \frac{\text{len\_big}}{\text{len\_small}}, \tag{5}$$

$$\text{len\_big} = \text{Max}(\text{length}(r\_il), \text{length}(r\_jl)), \tag{6}$$

$$\text{len\_small} = \text{Min}(\text{length}(r\_il), \text{length}(r\_jl)). \tag{7}$$

3.3.1. The Similarity Metrics. As mentioned above, the data types in this article include string, numeric, date, and enumeration. Attribute value similarity metrics are discussed based on these four types.

(1) String-Type Attribute. Cosine similarity [45–47] is used to calculate the similarity of attribute value x_il and x_jl, which can be defined as $\text{sim\_att\_string}(x\_il, x\_jl)$. Accordingly, the similarity of entity r_i and r_j is

calculated by (8), and t is the number of attributes in this string-type block.

$$\text{sim\_ent\_str}(r\_i, r\_j) = \frac{\sum_{l=1}^{t} \vartheta\_ij^{(l)} \text{sim\_att\_str}(x\_il, x\_jl)}{\sum_{l=1}^{t} \vartheta\_ij^{(l)}}. \quad (8)$$

(2) Numeric-Type Attribute. The numeric type is divided into nominal, ordinal, and general numerical as mentioned above. The diversity is adopted in numeric type, and the diversity of attribute values x_il and x_jl from attribute A_l ($1 \le i, j \le n$, $1 \le l \le m$) is defined as div_ij^l. The calculation methods are defined as the following:

(a) A_l is one of the general numerical attributes, $\text{div}\_ij^l = |x\_il - x\_jl|/(\max\_A\_l - \min\_A\_l)$, where $\max\_A\_l$ is the maximum value of A_l and $\min\_A\_l$ is the minimum one.

(b) A_l is one of the ordinal numerical attributes, and A_l has the ordered state M which can be expressed as $\{1, \ldots, M\_k\}$. Using the corresponding ranking number s_il (when the $x\_il \in M\_r$ ($1 \le r \le k$), the ranking number of x_il is r) to replace x_il and calculating $z\_il = (s\_il - 1)/(M\_r - 1)$ accordingly, next, s_il is replaced by z_il. Finally, the processing of z_il can refer to step (a).

(c) A_l is one of the nominal attributes: if $x\_il = x\_jl$, then $\text{div}\_ij^l = 0$; else $\text{div}\_ij^l = 1$.

Consequently, the diversity of entity r_i and r_j is defined in (9), and the similarity between them is shown in (10), where r is the total number of attributes in this numeric block.

$$\text{div\_ent}(r\_i, r\_j) = \frac{\sum_{l=1}^{r} \vartheta\_ij^{(l)} \ \text{div}\_ij^{(l)}}{\sum_{l=1}^{r} \vartheta\_ij^{(l)}}, \quad (9)$$

$$\text{sim\_ent\_num}(r\_i, r\_j) = 1 - \text{div\_ent}(r\_i, r\_j). \quad (10)$$

(3) Enumeration-Type Attribute. Because the possible values for enumeration attribute are a set of predefined specific symbols, there is usually no implicit semantic relationship between them. This article uses equality or inequality to measure the similarity between two enumerated attribute values, which is shown as follows:

$$\text{sim\_att\_enum}(x\_il, x\_jl) = \begin{cases} 0 & (x\_il \ne x\_jl), \\ 1 & (x\_il = x\_jl). \end{cases} \quad (11)$$

Accordingly, the similarity of entity r_i and r_j is obtained by (12), where s is the amount of attributes in enumeration-type block.

$$\text{sim\_ent\_enum}(r\_i, r\_j) = \frac{\sum_{l=1}^{s} \vartheta\_ij^{(l)} \text{sim\_att\_enum}(x\_il, x\_jl)}{\sum_{l=1}^{s} \vartheta\_ij^{(l)}}. \quad (12)$$

(4) Date-Type Attribute. The processing of date-type attributes, firstly, is to divide the attribute values into a set of ordered states $\{1, \ldots, M\_k\}$. Correspondingly, the similarity between x_il and x_jl from the date attribute A_l is defined as (13), and the similarity of entity r_i and r_j is calculated by (14).

$$\text{sim\_att\_date}(x\_il, x\_jl) = \begin{cases} 1 & (x\_il \in M\_i, x\_jl \in M\_i), \\ 0 & (x\_il \in M\_i, x\_jl \in M\_j, i \ne j), \end{cases} \quad (13)$$

$$\text{sim\_ent\_date}(r\_i, r\_j) = \frac{\sum_{l=1}^{s} \vartheta\_ij^{(l)} \text{sim\_att\_date}(x\_il, x\_jl)}{\sum_{l=1}^{s} \vartheta\_ij^{(l)}}. \quad (14)$$

3.3.2. The Varying Window Sizing Strategies. The window size ws in traditional SNM is fixed, and the challenge here is how to select a reasonable value for ws. If the window size is too large that will increase the comparison cost, in contrast, too small may lead to missing matches. In this paper, depending on the type-based blocks, we use varied strategies to dynamically adjust the window size in these blocks.

For string-type block, we adopt the method proposed by [15] to dynamically adjust the window size based on the amount of closest entities in the current window, which can be defined as follows:

$$\text{ws\_}i = \text{Int}\left(\text{ws\_min} + \frac{n\_\text{actclo}}{\text{ws\_}(i\text{-}1) - 1}(\text{ws\_max} - \text{ws\_min})\right), \quad (15)$$

where ws_i is the window size of the ith window, ws_min is the minimum of window size and $W_{\max}$ is the maximum one, and n_actclo is the number of the closest entities. But the calculation method of the minimum and maximum window size is not presented in research [15]. In this paper, we obtained the minimum and maximum with the following steps: (1) selecting the key attribute; (2) clustering the values of the key attribute based on *Cosine* similarity metric; (3) ws_min is equal to the number of elements of the minimum cluster and ws_max is equal to the number of elements in the biggest cluster.

For numerical-type and date-type block, the conditions for expanding window size are $(1\text{-div}\_iw_i^{(l)}) > \delta$ and $\text{sim\_att\_date}(x\_il, x\_w_il) > \delta$, and the conditions for narrowing window size are $(1\text{-div}\_iw_p^{(l)}) < \delta$ and $\text{sim\_att\_date}(x\_il, x\_w_pl) < \delta$, where x_il is from the entity of the latest sliding into the current window, x_w_il is from the last one in the current window, and x_w_pl is from the pth entity in the current window. In addition, the expansion step ws_step (ws_step = 1) and the minimal window size ws_min are also set in advance. Accordingly, the expanded window size ws_i is shown in (16), and the narrowed window size is described by (17).

$$\text{ws\_}i = \text{ws\_}i + \text{ws\_step}, \quad (16)$$

Input: *db*: dataset in each block
*ws*_min: the minimal window size(the string type block need also provide *ws*_max)
*A*_*l*: the sort key attribute in each block
δ: the similarity threshold in each block
Output: Map < sub record identifier, Set < the identifiers of the closest entities>>
(a) *winSize*←*ws*_min;
(b) *type_att* ← the type of *A*_*l*;
(c) if (*type_att*.equals("STRING")) then *winSize* ←*ws*_max;
(d) else if (*type_att*.equals("ENUM")) then *winSize* ←calculate the window size according to (18);
(e) Sorting entities in each block based on *A*_*l*;
(f) Sliding the window from the first entity in each block;
(g) *closestMap*←new HashMap < Integer, Set < Integer>>;
(h) While (it is not the last entity) Do
(i) *i*←0; //initialize the comparison times
(j) *latestEnt*←the latest entity that sliding into the current window;
(k) While (*i* < *winSize*) Do
//according to the (8), (10), (12), and (14) to calculate the similarity between entities
(l) *similarity*←the similarity between entity *latestEnt* and the i^{th} entity in current window;
(m) If (*similarity* > δ) then Do{
(n) *closestSet*← *closestMap*.get(the identifier of entity *latestEnt*);
(o) If (*closestSet* == null) then *closestSet* ←new HashSet;
(p) *closestMap*.put(the identifier of entity *latestEnt*, *closestSet*);
(q) *closestSet* ← *closestSet*.add(the identifier of the i^{th} entity);
(r) } else if (*type_att*.equals("NUME") or *type_att*.equals("DATE")) Then {*winSize* ← narrow the window size according to (17);}
(s) *i* ←*i*++;
(t) If (*type_att*.equals("STRING") and *i* > *ws*_min) then recalculate the window size with (15);
(u) If (*i* > *winSize* and *similarity* > δ) then {*winSize* ←*winSize*++; Exit the current loop;//only for date or numerical}
(v) Sliding into the next window;
(w) return *closestMap*;

Algorithm 3: Calculate the similarity between entities in each block.

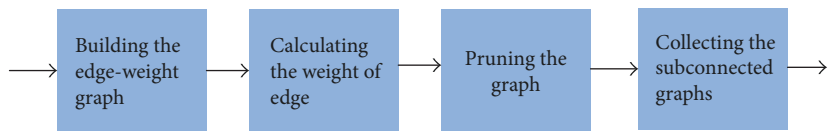

Figure 2: The internal functionality of our gathering method.

$$\text{ws\_}i = \begin{cases} \text{ws\_min} & (p \le \text{ws\_min}), \\ p & (\text{ws\_min} < p < \text{ws\_}i). \end{cases} \quad (17)$$

For the enumeration-type block, the window size is defined as follows:

$$\text{ws\_}i = \text{Max}(\text{frequency}(v\_i)), \quad (18)$$

where v_i represents the possible values of enumeration attribute.

3.3.3. MBN Algorithm Implementation. Based on the definitions of similarity metric in Section 3.3.1 and the variable window strategies in Section 3.3.2, we implement the closest entity detection in each block through Algorithm 3.

In order to obtain the final solution of the complete entities, we gather the output of each block obtained by Algorithm 3. Generally speaking, the gathering process is divided into the following four steps, which is intuitively represented as Figure 2.

(a) Building the edge-weight graph *G*_*e*, which is an undirected graph, is used to express *closestMap*. The nodes in this graph are the entity identifiers contained in the *closestMap*.

(b) If the likelihood of two entities matched is proved at least in one block, they will be connected by an edge. The purpose of this step is to determine the weight of each edge.

(c) Pruning edge-weight graphing aims at selecting the globally best pairs by iterating over the edges of an edge-weight graph in order to filter out edges that do not satisfy the pruning criterion, such as the edges with low weight.

(d) Collecting the connected subgraphs from the pruned edge-weight graph constitutes the final output of the complete entities.

In general, the strong indication of the similarity of two entities is provided by the number of blocks they have in common; the more blocks they share, the more likely they are to match. Therefore, the weight of an edge connecting entities r_i and r_j is set to the number of blocks which marked that these two entities are matched. However, the contribution of different blocks to the computation of the similarity between the complete entities is also different. For example, the output from a string-type block has a positive effect because their values are sufficient dispersion. As a result, we improved the common block scheme (CBS) method proposed in research [14, 35] and named it as improved common block scheme (ICBS), which is shown in (19). For the sake of discussion, we use e_i, j to describe the edge between entities r_i and r_j; correspondingly, e_i, j. weight is used to express its weight.

$$e\_i, j.\text{weight} = \sum_{k=1}^{d} e\_i, j^k.\text{weight} \times B\_i, j^k, \tag{19}$$

where e_i, j^k.weight is the weight of kth block, and if the entities r_i and r_j are marked as matched in the kth block, then $B\_i, j^k = 1$; else $B\_i, j^k = 0$. We adopt weight edge pruning (WEP) method [14] to prune the edges with lower weight.

4. Experiment Evaluation

This section aims to experimentally evaluate the techniques presented in this paper, respectively. We begin our analysis with the dataset collection, as shown in Section 4.1. In addition, the following experiments are conducted: the splitting attributes into different blocks and the attribute clustering methods are examined in Section 4.2. In order to evaluate the improved common block scheme (ICBS), we compare it with CBS presented in [14] in Section 4.3. At last, we compare our MBN method with the classical SNM in Section 4.4 to evaluate the effectiveness of our similarity metric schemes and dynamic adjustment window size strategies in different blocks. Among them, all approaches and experiments were fully implemented using Java 1.8 version, and development tool is IntelliJ IDEA. For the implementation of attribute clustering and pruning of edge-weight graph, we referred to and improved the source codes publicly released at http://SorceForge.net by Papadakis et al. [21]. We also employed some open source libraries in our implementation, such as jgrapht and commons-math.

4.1. Dataset Collection. To thoroughly evaluate our techniques, we employ a large-scale, real-world dataset, which is generated and obtained from one of real-world Internet of Things project that is developed and maintained by our research group. The time span of the dataset is from June 2016 to November 2016. Because if the terrorist can easily get flammable items (such as gasoline and natural gas), it will cause huge potential danger to the society. Therefore, the purpose of this project is to monitor these combustible materials in order to predict and monitor potential violent attacks in Xinjiang (one of China's provinces). In addition to refueling records, this project also collected information about drivers, vehicles, and so forth and stores them in the cloud platform. In combining machine learning algorithms, such as random forests or rotation forests, we will use these clean data output by our proposed method in this paper to propose some smart models in our future research, for example, to predict illegal vehicles.

4.2. Evaluation of Splitting Attributes to Different Blocks and Attribute Clustering Methods. For assessing the performance of our splitting attributes and attribute clustering methods, *Accuracy* is used as evaluation criteria in this part and its definition is shown as follows:

$$\text{Acc}_{\text{splitting}} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} = \frac{\#\text{CorrectSplAttNum}}{\#\text{TotalAttNum}}, \tag{20}$$

$$\text{Acc}_{\text{clustering}} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} = \frac{\#\text{CorrectCluAttNum}}{\#\text{TotalAttNum}}, \tag{21}$$

where true positive (TP) is the amount of positive testing samples correctly predicted as positive, false positive (FP) is the amount of negative testing samples incorrectly predicted as positive, true negative (TN) is the amount of negative testing samples correctly predicted as negative, and false negative (FN) is the amount of positive testing samples wrongly predicted as negative.

In Table 1, in this experiment, we selected four tables to verify our methods. For the purpose of verifying the attribute clustering method, we added some redundant attributes and noise values in these four tables. The experimental results demonstrated that the average accuracy achieved by our splitting attributes to different blocks was 98.18% and the average accuracies obtained by our attribute clustering method from four tables were, respectively, 89.28%, 86.83%, 87.64%, and 92.08%. It can be proved that our methods work well. The main reason is that we adopt different methods to handle attributes based on the attribute value types and value distributions. For instance, in enumeration-type block, the confidence level was set as $\alpha = 0.001$, the similarity threshold was set to 0.80 in numerical- and date-type blocks, and it was set to 0.76 in string-type block.

4.3. Evaluation of ICBS Method. Combined with WEP [14] scheme, Figure 3 presents the accuracy comparison between CBS [14] and our ICBS. As shown in Figure 3, the experimental results exhibited that our ICBS had high efficiency in these four tables mentioned in Section 4.2 (i.e., accuracy > 84%). For ICBS, the weight in different type block

Table 1: The accuracy of splitting attributes to different blocking and attributes clustering.

Name	Attribute number	$Acc_{splitting}$	$Acc_{clustering}$			
			Enumeration	Date	String	Numerical
Cylinder_out	68	97.06%	86.67%	85.71%	92.6%	99.67%
Cylinder_check	69	98.8%	91.3%	89.47%	84.62%	85.71%
Vehicle_info	61	98.36%	87.5%	84.62%	90%	83.33%
Gas_cylinder	66	98.48%	91.67%	87.5%	83.33%	99.56%
Average	**66**	**98.18%**	**89.28%**	**86.83%**	**87.64%**	**92.08%**

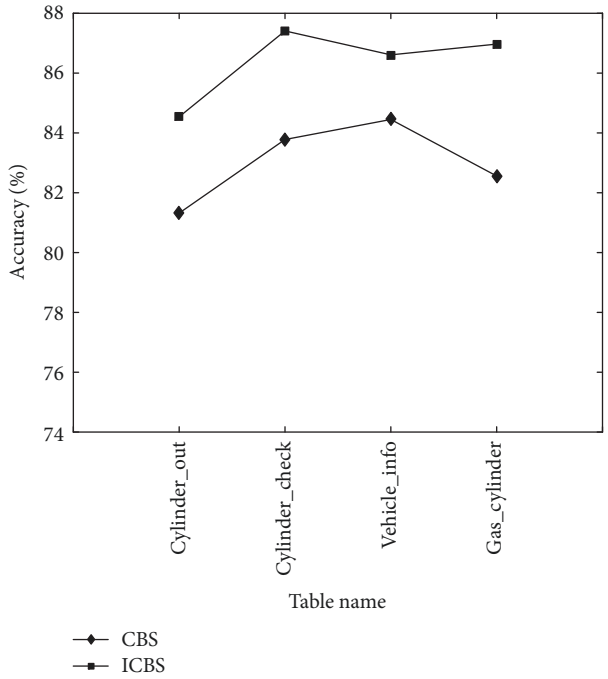

Figure 3: Accuracy comparison between ICBS and CBS.

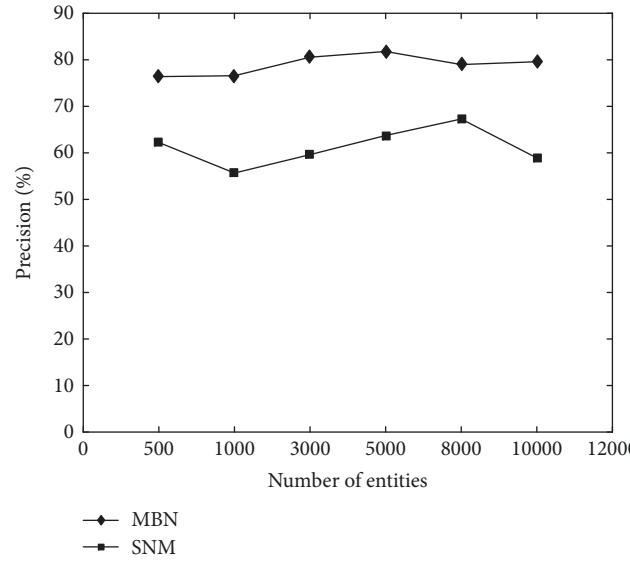

Figure 4: Precision comparison between MBN and SNM.

was also varied; generally speaking, the weight in string and numerical type was enlarged in these four tables, and the weight in enumeration and date was reduced. But for the CBS, the weight in different types of block was all set to 1. The experimental results demonstrated that these four types of block have different contributions to the calculation of the similarity of the complete entity and our ICBS is an appropriate and important part in our ER workflow.

4.4. Evaluation of MBN Method. In this section, we compared our MBN method with existing entity resolution technique SNM. The metrics are the following: (1) Precision (Pre.); (2) Recall (Rec.); (3) *F*-score, which are shown as follows:

$$\text{Pre.} = \frac{\text{TP}}{\text{TP} + \text{FP}} \times 100\%, \tag{22}$$

$$\text{Rec.} = \frac{\text{TP}}{\text{TP} + \text{FN}} \times 100\%, \tag{23}$$

$$F\text{-score} = \frac{2 \times \text{Pre.} \times \text{Rec.}}{\text{Pre.} + \text{Rec.}}, \tag{24}$$

where the TP and FP are the number of true-matched and true-nonmatched candidate entity pairs generated by the ER method. The FN is the number of nonmatched entity pairs.

We extracted the *Cylinder_check* table from the dataset to verify the effectiveness of our MBN method. This table has a total of 534,017 entities. We carried out this experiment based on five hundred, 1 thousand, 3 thousand, 5 thousand, 8 thousand, and 10 thousand entities, respectively. The entity similarity threshold was set to 0.79 both in MBN and SNM. The window size was set to 16 in SNM. In MBN, the initial window size for date- and numerical-type blocks was set to 8, the minimum window size was set to 5, the maximum window size was set to 30 for string type, and the window size of enumeration type was calculated in real-time. The input data of SNM and MBN were all from *Cylinder_check* table, in which the redundant attributes have been removed. The comparisons based on *precision*, *recall*, and *F-score* achieved by MBN and SNM are shown in Figures 4 and 5.

Figures 4 and 5 demonstrated that, compared with SNM, our MBN method improved the precision and recall for the closest entity detection. That means MBN is a more objective

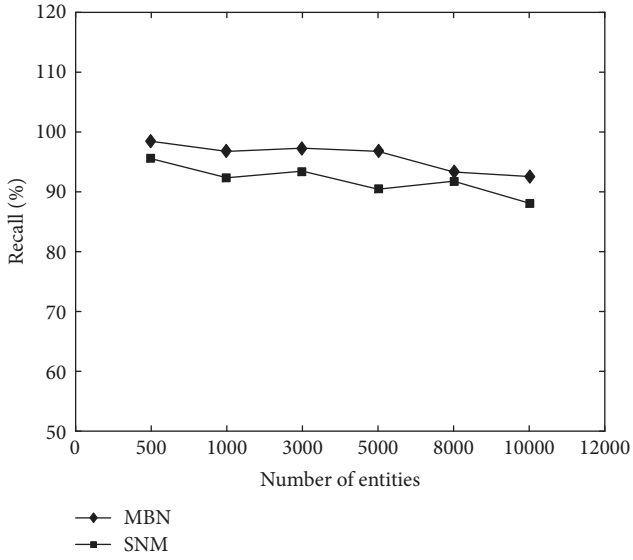

Figure 5: Recall comparison between MBN and SNM.

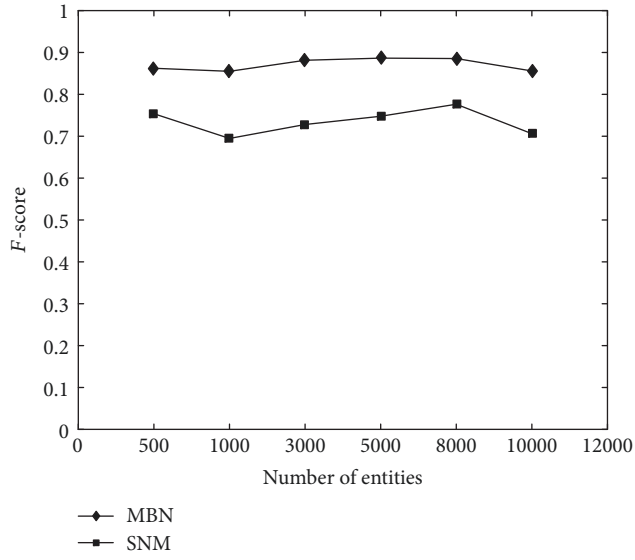

Figure 6: *F*-score comparison between MBN and SNM.

method to evaluate the similarity between entities because it split entities into different blocks based on the attribute value types and adopted different similarity metrics in these blocks. In addition, compared with the fixed window size in SNM, the dynamic adjustable window schemes introduced in our ER resolving method improves the performance while reducing the missing matched. Moreover, in SNM, the performance of detecting the closest entities mainly depends on its single sort key attribute, which means that if the key attribute are not selected properly or the key attribute contains noise data, its performance will be greatly reduced. Thus, it is necessary to select multiple key attributes and flexibly integrate several ER workflows to achieve better performance in MBN method. Generally speaking, these two methods all had acceptable recall under the same dataset. In Figure 6, the *F*-score comparison also indicated that MBN performed better than the original SNM method.

5. Conclusion and Future Work

In this paper, we introduce a novel hybrid approach, which is as an effective method for entity resolution in the context of voluminous and highly dimension data. In contrast to existing entity resolution methods, our approach commits to reducing the searching space for entity pairs by the constraint of redundant attributes and matching likelihood and to meeting multiple similarity comparison requirements based on attribute value types. Our thorough experimental evaluation verified the effectiveness, as well as the efficiency of our method on real-world dataset. We believe our approach is a promising alternative to entity resolution problem that can be effectively adopted to many other applications, such as the National Census area (e.g., it can be used to match data from different census collections to detect and correct conflicting or missing information) and Business Mailing Lists area (e.g., it can be used to identify redundant entities about the same customer to avoid money being wasted on mailing several copies of an advertisement flyer to one person). Of course, many interesting problems remain to be addressed in the future, including a general guidance for selecting key attribute in each block, and we will attempt to introduce parallelization technique into our approach to further improve its efficiency. Another interesting direction of our research is how to apply and develop our method to improve the efficiency of approaches that rely on entity resolution in machine learning and probabilistic inference.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is supported in part by the Xinjiang Key Laboratory Fund, under Grant no. 2016D03019, in part by the Xinjiang High-Tech R&D Program, under Grant no. 201512103, and in part by the Thousand Talents Plan, under Grant no. Y32H251201. The authors would like to thank all anonymous reviewers for their constructive advices.

References

[1] J. Wang, T. Kraska, M. J. Franklin, and J. Feng, "CrowdER: crowdsourcing entity resolution," *Proceedings of the Vldb Endowment*, vol. 5, no. 11, pp. 1483–1494, 2012.

[2] U. Draisbach and F. Naumann, "A generalization of blocking and windowing algorithms for duplicate detection," in *2011 International Conference on Data and Knowledge Engineering (ICDKE)*, Milan, Italy, 2011.

[3] G. Papadakis, E. Ioannou, and P. Fankhauser, "Efficient entity resolution for large heterogeneous information spaces," in *Proceedings of the fourth ACM international conference on Web search and data mining*, Hong Kong, China, 2011.

[4] J. Efremova, B. Ranjbar-Sahraei, H. Rahmani et al., "Multi-source entity resolution for genealogical data," in *Population*

Reconstruction, pp. 129–154, Springer International Publishing, Cham, 2015.

[5] S. E. Whang, D. Menestrina, G. Koutrika, M. Theobald, and H. Garcia-Molina, "Entity resolution with iterative blocking," in *SIGMOD '09 Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, Providence, Rhode Island, USA, 2009.

[6] R. Mahmoud, N. El-Bendary, H. M. Mokhtar, and A. E. Hassanien, "Similarity measures based recommender system for rehabilitation of people with disabilities," in *Advances in intelligent systems and computing*, Springer, Cham, 2016.

[7] Y. R. Wang and S. E. Madnick, "The inter-database instance identification problem in integrating autonomous systems," in *[1989] Proceedings. Fifth International Conference on Data Engineering*, Los Angeles, CA, USA, 1989.

[8] M. Bilenko and R. J. Mooney, "Adaptive duplicate detection using learnable string similarity measures," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '03*, pp. 39–48, Washington, D.C, 2003.

[9] P. Christen, "Automatic record linkage using seeded nearest neighbour and support vector machine classification," in *KDD '08 Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 151–159, Las Vegas, Nevada, USA, 2008.

[10] P. Christen, "A survey of indexing techniques for scalable record linkage and deduplication," *IEEE Transactions on Knowledge & Data Engineering*, vol. 24, no. 9, pp. 1537–1555, 2012.

[11] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios, "Duplicate record detection: a survey," *IEEE Educational Activities Department*, vol. 19, pp. 1–16, 2007.

[12] C. Xiao, W. Wang, X. Lin, J. X. Yu, and G. Wang, "Efficient similarity joins for near-duplicate detection," *ACM Transactions on Database Systems*, vol. 36, no. 3, p. 15, 2011.

[13] U. Draisbach and F. Naumann, "A comparison and generalization of blocking and windowing algorithms for duplicate detection," in *Proceedings of International Workshop on Quality in Databases (QDB)*, pp. 51–56, 2011.

[14] G. Papadakis, G. Koutrika, T. Palpanas, and W. Nejdl, "Meta-blocking: taking entity resolutionto the next level," *IEEE Transactions on Knowledge & Data Engineering*, vol. 26, no. 8, pp. 1946–1960, 2014.

[15] J. Z. Zhang, Z. Fang, Y. J. Xiong, and X. Y. Yuan, "Optimization algorithm for cleaning data based on SNM," *Journal of Central South University*, vol. 41, no. 6, pp. 2240–2245, 2010.

[16] L. Kolb, A. Thor, and E. Rahm, "Multi-pass sorted neighborhood blocking with MapReduce," *Computer Science - Research and Development*, vol. 27, no. 1, pp. 45–63, 2012.

[17] L. He, Z. Zhang, Y. Tan, and M. Liao, "An efficient data cleaning algorithm based on attributes selection," in *2011 6th International Conference on Computer Sciences and Convergence Information Technology (ICCIT)*, Seogwipo, South Korea, 2012.

[18] G. Chesi and K. Hashimoto, "Automatic image annotation approach based on optimization of classes scores," *Computing*, vol. 96, no. 5, pp. 381–402, 2014.

[19] Q. Tan, X. Zhang, and R. M. G. Kinshuk, "The 5R Adaptation Framework for Location-Based Mobile Learning Systems," *Modern Distance Education Research*, 2012.

[20] D. Srivastava, D. Srivastava, and D. Srivastava, "Less is more: selecting sources wisely for integration," in *International Conference on Very Large Data Bases*, 2012.

[21] G. A. Papadakis, *Blocking Techniques for Efficient Entity Resolution over Large, Highly Heterogeneous Information Spaces*, 2013.

[22] J. Sedano, S. González, C. Chira, Á. Herrero, E. Corchado, and J. R. Villar, "Key features for the characterization of android malware families," *Logic Journal of IGPL*, vol. 25, no. 1, pp. 54–66, 2016.

[23] J. R. Villar, P. Vergara, M. Menéndez, E. de la Cal, V. M. González, and J. Sedano, "Generalized models for the classification of abnormal movements in daily life and its applicability to epilepsy convulsion recognition," *International Journal of Neural Systems*, vol. 26, no. 06, article 1650037, 2016.

[24] J. Sedano, C. Chira, S. González, Á. Herrero, E. Corchado, and J. R. Villar, "Characterization of android malware families by a reduced set of static features," in *Advances in Intelligent Systems and Computing*, Springer International Publishing, Cham, 2016.

[25] Z. Q. Hao, Z. J. Zhang, and H. C. Chao, "A cluster-based fuzzy fusion algorithm for event detection in heterogeneous wireless sensor networks," *Journal of Sensors*, vol. 2015, Article ID 641235, 11 pages, 2015.

[26] M. Bilenko, R. Mooney, W. Cohen, P. Ravikumar, and S. Fienberg, "Adaptive name matching in information integration," *Intelligent Systems IEEE*, vol. 18, no. 5, pp. 16–23, 2003.

[27] A. Pomeroy and Q. Tan, "Effective SQL injection attack reconstruction using network recording," in *IEEE International Conference on Computer and Information Technology*, Pafos, Cyprus, 2011.

[28] S. Tejada, C. A. Knoblock, and S. Minton, "Learning domain-independent string transformation weights for high accuracy object identification," in *KDD '02 Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, Edmonton, Alberta, Canada, 2002.

[29] W. W. Cohen, P. Ravikumar, and S. E. Fienberg, "A comparison of string metrics for matching names and records," *Kdd Workshop on Data Cleaning & Object Consolidation*, 2003.

[30] P. Christen, "A comparison of personal name matching: techniques and practical issues," in *Sixth IEEE International Conference on Data Mining - Workshops (ICDMW'06)*, Hong Kong, China, 2006.

[31] A. Thor and E. Rahm, "Evaluation of entity resolution approaches on real-world match problems," *Proceedings of the VLDB Endowment*, vol. 3, pp. 484–493, 2010.

[32] Q. Tan, P. H. Wu, Y. L. Jeng, and Y. M. Huang, "Mobile computing architecture for multi-platform adaptation," in *Iadis Wireless Applications and Computing*, 2009.

[33] M. Ghantous and M. Bayoumi, "MIRF: a multimodal image registration and fusion module based on DT-CWT," *Journal of Signal Processing Systems*, vol. 71, no. 1, pp. 41–55, 2013.

[34] J. Han and M. Kamber, "Data mining: concepts and techniques," *Data Mining Concepts Models Methods & Algorithms Second Edition*, vol. 5, no. 4, pp. 1–18, 2006.

[35] Q. Tan and F. Pivot, "Big data privacy: changing perception of privacy," in *2015 IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity)*, Chengdu, China, 2015.

[36] A. Backurs and P. Indyk, "Edit distance cannot be computed in strongly subquadratic time (unless SETH is false)," in *STOC*

'15 Proceedings of the forty-seventh annual ACM symposium on Theory of computing, Portland, Oregon, USA, 2015.

[37] B. Karim, Q. Tan, I. El Emary, B. A. Alyoubi, and R. S. Costa, "A proposed novel enterprise cloud development application model," *Memetic Computing*, vol. 8, pp. 287–306, 2016.

[38] T. Joachims, "A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization," in *Proceedings 14th International Conference on Machine Learning (ICML'97)*, pp. 148–155, 1997.

[39] S. Loisel and Y. Takane, "Partitions of Pearson's chi-square statistic for frequency tables: a comprehensive account," *Computational Statistics*, vol. 31, no. 4, pp. 1429–1452, 2016.

[40] M. T. Puth, M. Neuhäuser, and G. D. Ruxton, "Effective use of Pearson's product–moment correlation coefficient," *Animal Behaviour*, vol. 93, pp. 183–189, 2014.

[41] D. Bogdanov, L. Kamm, S. Laur, P. Pruulmann-Vengerfeldt, R. Talviste, and J. Willemson, "Privacy-preserving statistical data analysis on federated databases," in *Lecture Notes in Computer Science*, pp. 30–55, Springer International Publishing, 2014.

[42] M. Denojean-Mairet, Q. Tan, F. Pivot, and M. Ally, "A ubiquitous computing platform - affordable telepresence robot design and applications," in *2014 IEEE 17th International Conference on Computational Science and Engineering*, Chengdu, China, 2014.

[43] G. Papadakis, E. Ioannou, T. Palpanas, C. Niederee, and W. Nejdl, "A blocking framework for entity resolution in highly heterogeneous information spaces," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 12, pp. 2665–2682, 2013.

[44] M. A. Hernández and S. J. Stolfo, "The merge/purge problem for large databases," in *IGMOD '95 Proceedings of the 1995 ACM SIGMOD international conference on Management of data*, vol. 24, pp. 127–138, 1995.

[45] J. Ye, "Cosine similarity measures for intuitionistic fuzzy sets and their applications," *Mathematical and Computer Modelling*, vol. 53, no. 1-2, pp. 91–97, 2011.

[46] A. Abdelgawad and M. Bayoumi, "Proposed centralized data fusion algorithms," in *Lecture Notes in Electrical Engineering*, pp. 37–57, Springer US, Boston, MA, 2012.

[47] L. Muflikhah and B. Baharudin, "Document clustering using concept space and cosine similarity measurement," in *2009 International Conference on Computer Technology and Development*, pp. 58–62, Kota Kinabalu, Malaysia, 2009.