Unsupervised Blocking of Imbalanced Datasets for Record Matching

Chenxiao Dou[1,2], Daniel Sun[2,1], and Raymond K. Wong[1,2]

[1] School of Computer Science & Engineering, University of New South Wales, Sydney, Australia
[2] Data61, Commonwealth Scientific and Industrial Research Organisation(CSIRO), ACT, Australia

Abstract. Record matching in data engineering refers to searching for data records originating from same entities across different data sources. The solutions for record matching usually employ learning algorithms to train a classifier that labels record pairs as either matches or non-matches. In practice, the amount of non-matches typically far exceeds the amount of matches. This problem is so-called imbalance problem, which notoriously increases the difficulty of acquiring a representative dataset for classifier training. Various blocking techniques have been proposed to alleviate this problem, but most of them rely heavily on the effort of human experts. In this paper, we propose an unsupervised blocking method, which aims at automatic blocking. To demonstrate the effectiveness, we evaluated our method using real-world datasets. The results show that our method significantly outperforms other competitors.

Keywords: Record matching, Blocking, Imbalance, Heuristics

1 Introduction

A crucial step in integrating data from multiple sources is detecting and eliminating duplicate records [9]. This process is called Entity Matching, Record linkage, or Record Matching, which is a well known problem that arises in many applications such as address matching and citation de-duplication [3, 6, 5, 16]. The goal of record matching is to identify records that represent the same real-world entities from a variety of data sources.

Machine learning has been playing an increasingly important role in the Record Matching problem. Some classical algorithms such as Support Vector Machine (SVM) and Decision Tree are adopted to predict whether a record pair is matched or not, based on the similarities between the entries [5, 16]. However, selecting a training dataset is one of the most significant steps before learning process. To achieve a good classifier in terms of accuracy and recall, people have to label as many samples as possible, and consequently hiring human experts to hand-label the instances is too expensive in general.

In a real-world record matching task on a dataset with n records, it is obvious that the number of truly matched pairs is not larger than n. However, the total

number of candidate pairs including matched pairs and non-matched pairs can reach $\theta(n^2)$. To reduce the number of candidate pairs, blocking techniques have been proposed to filter out the pairs that are unlikely matched [15, 8].

The general idea is to divide the records into several blocks and only the records in the same blocks need to be compared. For the records in different blocks, it assumes that the pairs are unlikely to match. Take a dataset of academic publications for example. When detecting the duplicates, we may partition the dataset into small groups according to conference names. Papers published in different conferences cannot refer to the same real-world entities. To increase comparison opportunities for truly matched records, users may employ several blocking criteria to ensure that every duplicated pair is assigned into at least one block [18]. However, most of blocking criteria are designed manually and decided based on human experience.

In this paper, we propose a new blocking method that can automatically filter record pairs. For many imbalanced datasets, most candidate record pairs have low similarity. As a result, in a similarity space, areas resided by non-matched pairs have higher density than those with matched pairs. Therefore, unlike most blocking methods that take similarity thresholds as input parameters, our method uses density as the measure and requires two thresholds (to be discussed in detail later) of density provided by users. We target at matched and non-matched pairs that are entangled in a similarity space. Furthermore, to improve the efficiency of our method, we admit the monotonicity assumption on detecting the target region. Through our empirical studies, it is explicit that the property of monotonicity generally holds in real-world datasets. Our results demonstrate that the proposed method can significantly block non-matched pairs. In summary, the contributions of this paper are summarized as follows:

- Compared to manually blocking methods, our method is able to determine complex blocking criteria automatically.
- The performance of the proposed density based approach will not be affected as much by the selection of similarity functions.
- The monotonicity of datasets is examined and used to improve the efficiency of the proposed method.

The remainder of this paper is structured as follows. We discuss related work in Section 2. In Section 3, we present the background of this paper including the problem, the assumptions, and the definitions. Our proposed method is then presented in Section 4. After that, the proposed method is evaluated in Section 5. Finally, we conclude this paper in Section 6.

2 Related Work

Entity matching is a well-studied problem of determining whether two records refer to the same entity or not. To measure the similarity between records, a variety of similarity functions have been proposed. EditDistance, Jaccard similarity, Cosine similarity are three widely used measures for this problem [7].

These different similarity measures are the key criteria to identify truly matched records.

Many researchers have explored machine learning techniques [5, 16] to solve the entity matching problem. Each similarity is regarded as a data feature, whose importance weight is automatically decided by a learning algorithm. But, there exists an imbalance issue that greatly affects the learning, that is, no learning algorithm can achieve an outstanding performance with an extremely imbalanced training set.

In order to reduce non-matched pairs, various blocking techniques are used to filter out the pairs that are unlikely matched. The traditional blocking criteria are manually designed according to the attributes of datasets [14]. Some learning algorithms are adopted to produce the blocking criteria automatically in [4, 13]. Canopy Clustering [12] fast groups record pairs into overlapping sets with a loose threshold and then filters the non-matches with a tight threshold. An inappropriate blocking criterion may separate the truly matched records into different clusters. Most of the existing work use several blocking criteria to increase the chance of assigning the matched records to the same clusters. Whang et al. [18] proposed an iterative blocking framework, which enables record matching across different clusters. After that, they discussed how to update the existing blocking rules when new records are added in [17].

Most of the previous methods [12, 4, 13, 3, 6] use textual similarity as the main measure to filter pairs. In our work, we employ the density of pairs on a similarity space as the measure to find the non-matches. We also adopt the monotonicity assumption to improve the efficiency of our proposed density based method. The monotonicity assumption has also been introduced in [3, 6] but they focused on the precision during learning, while in our approach, we focus on the property of monotonicity on the density during blocking.

3 Background

3.1 Problem Definition

In Table 1, the upper table contains three citation records from DBLP dataset, and the other has two records from ACM dataset. As shown in the tables, only the last records in the two tables are matched and the other three are non-matched. Thus, out of all 6 possible pairs between the two tables, there is only one matched pair, and this results in the imbalance problem. Given the entire datasets, the matched and the non-matched will be extremely unbalanced, and consequently from such datasets, it is difficult to sample a balanced training set, whose quality is important for the following learning process.

We denote the set of all record pairs as $\mathbb{C}$, the set of all matched pairs as $\mathbb{C}_{match}$, and the set of all non-matched pairs as $\mathbb{C}_{non}$. Our goal is to design a blocking method that could prune pairs in $\mathbb{C}_{non}$ and preserve pairs in $\mathbb{C}_{match}$ as many as possible in order to achieve a balanced data pool (the ideal imbalance ratio is $1:1$). The set of all blocked pairs is denoted as $\mathbb{B}$. Since it is difficult

Table 1. Citation Dataset

Title	Authors	Venue	Year
Safe query languages for constraint databases	Peter Z. Revesz	TODS	1998
Efficient Filtering of XML Documents for Selective Dissemination of Information	Mehmet Altinel, Michael J. Franklin	Very Large Data Bases	2000
Standards for databases on the grid	Susan Malaika, Andrew Eisenberg, Jim Melton	ACM SIGMOD Record	2003

Title	Authors	Conf	Year
Database techniques for the World-Wide Web: a survey	D. Florescu, A. Levy, A. Mendelzon	SIGMOD	1998
Standards for databases on the grid	S. Malaika, A. Eisenberg, J.	SIGMOD	2003

to prune the instances perfectly, the aim of our work is to make the output closely approximate the ideal result. It is measured based on three metrics: *Recall*, *Reduction Ratio* and *Imbalance Ratio*. We do not use *Precision* as a metric, because the aim of blocking is to balance the dataset other than classify the instances, and the high imbalance ratio will prevent any algorithm from directly finding the true matches.

$$Recall = 1 - \frac{\sum_{x \in \mathbb{C}_{match}} 1[x \in \mathbb{B}]}{|\mathbb{C}_{match}|} \tag{1}$$

$$Reduction\ Ratio = \frac{\sum_{x \in \mathbb{C}_{non}} 1[x \in \mathbb{B}]}{|\mathbb{C}_{non}|} \tag{2}$$

$$Imbalance\ Ratio = \frac{\sum_{x \in \mathbb{C}_{non}} 1[x \notin \mathbb{B}]}{\sum_{x \in \mathbb{C}_{match}} 1[x \notin \mathbb{B}]} \tag{3}$$

3.2 Similarity Space

Textual similarity is one of the main measures to decide whether two records are matched or not. Given a record pair $(r, s) \in R \times S$, we can use a variety of similarity functions to measure the similarity between attributes of R and attributes of S. Without loss of generality, we assume the returned scores from all the similarity functions are in $[0, 1]$. Given d different similarity measures, we can map a pair $(r, s) \in R \times S$ into a point $(s_1, ..., s_d) \in [0, 1]^d$ where s_i is the similarity score returned by the i-th similarity measure. We call $[0, 1]^d$ *similarity space*. Each dimension represents one similarity measure to evaluate a similarity between r and s.

In this paper, for the convenience of discussion, a similarity space is partitioned into regular cells. A *region* refers to a sub-space consisting of several consecutive cells and an *area* refers to an arbitrary sub-space in the similarity space.

3.3 Monotonicity

Monotonicity has been assumed for entity matching in [3, 6]. For example, Arvind et al.[3] have studied the monotonicity of precision on improving the efficiency of their algorithm. In practice, the monotonicity assumption generally holds in many datasets. Table 1 is example records in two citation tables. Jaccard similarity on *Name* and EditDistance on *Title* are two frequently used similarity in Citation datasets. When we map the record pairs onto a 2D similarity panel shown in Fig. 1, we can observe that the dark area of lower similarity has much more points than the area of high similarity. Generally, in the similarity space $[0, 1]^d$, the area of lower similarities may contain more points. We call this phenomenon as *Monotonicity of Density*. Intuitively, a pair of records with a lower textual similarity would have less possibilities to be matched. Thus, in this paper, we focus on how to use the monotonicity assumption to block record pairs.

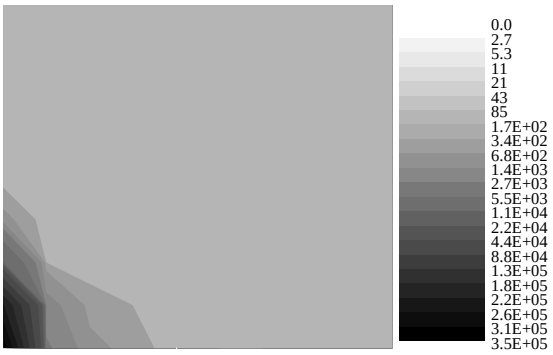

Fig. 1. Contour of citation pairs. The data are the publication records from DBLP and Google Scholar [1]. The panel on the left is the similarity space of two similarities. The right are the contour levels, which represent the numbers of points per unit area.

Prior to formally introducing *Monotonicity of Density*, we first give a formal definition of density. Due to the imbalance issue, the number of matched points is much smaller than that of non-matched ones after record pairs are mapped into $[0, 1]^d$. So the density of non-matches around a given point is roughly defined as follows:

Definition 1. *Given a threshold r and a point f, the density of point f, denoted as $\rho(f)$, is defined as the number of points within a certain distance r to f.*

And then, we define the density of region as follow:

Definition 2. *In a similarity space $[0, 1]^d$, the density of a region is represented by the density of point f that has the lowest $\rho(f)$ among all points in the same region.*

We also define a partial ordering $\preceq$ on the points in a similarity space.

Definition 3. *Given two points* $f = (f_1, ..., f_d)$ *and* $f' = (f'_1, ..., f'_d)$*, if* $f_i \leq f'_i$ *for all* $1 \leq i \leq d$*, we say that* f' dominates f*, denoted as* $f \preceq f'$*; if* $f \preceq f'$ *and* $f_i \neq f'_i$ *for some* $1 \leq i \leq d$*, it is denoted as* $f \prec f'$*.*

Then, we formalize *Monotonicity of Density* as follows:

Definition 4. *In a similarity space, for any two points* f *and* f' *such that* $f \preceq f'$*, if* $\rho(f) \geq \rho(f')$*, we say that the density is monotonic w.r.t.* $\preceq$*.*

Table 2. Notation Table

Notation	Description
$\mathbb{C}$	the set of all pairs
$\mathbb{C}_{match}$	the set of matched pairs
$\mathbb{C}_{non}$	the set of non-matched pairs
$\mathbb{B}$	the set of blocked pairs
$\preceq, \prec$	the partial ordering (*dominate*) defined in Section 3
$\rho(f)$	the density of point f
d	the number of dimensions of similarity space
r	the distance threshold for computing density
T_1	the tight threshold
T_2	the loose threshold
k	the granularity parameter
p, f	the points
R, S	the datasets
$\mathbb{V}$	the set of granulated points
$\mathbb{M}$	the set of *MaxBound* points
$\mathbb{U}$	the set of *MinUnknown* points
$\mathbb{Z}$	the set of enumerated points with density no less than T_2

4 Proposed Method

4.1 Overview of Method

Motivated by the observation that non-matched pairs crowd greatly in the area of low similarity, we propose a heuristic method that can automatically block the non-matches. Our method consists of three main steps:

– **Preprocessing**. After mapping entity pairs into the similarity space, the number of points is rather huge. Due to the huge number, it is too expensive to compute the density of every points for finding the region of high density. To improve the efficiency, we split the similarity space into finite cells and only consider the corners of each cell as the candidate boundary points of high density region.

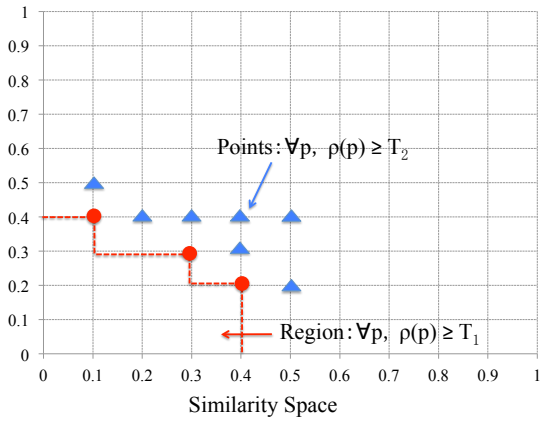

Fig. 2. Overview of the proposed method. The region with the density no less than T_1 will be blocked in the second step. The points with the density no less than than T_2 will be blocked in the third step.

- **Search for the region with density no less than** T_1. The property of monotonicity notably holds in the region of rather low similarity. So in the second step, we use a *Binary Search* algorithm to find all sub-regions with density at least equal to T_1. The union of every sub-region is the target region where the density is at least equal to T_1 . An example is shown in Fig. 2.
- **Search for the points with density no less than** T_2. In some region, the monotonicity may not hold strictly as well as in the region of density T_1, but there still exist many non-matches. Thus, in the rest of space, we run a recursive algorithm to enumerate the points of density no less than T_2, as shown in Fig. 2.

4.2 Preprocessing

In our method, the most expensive operation is to compute the defined density $\rho(f)$ of time complexity $\mathcal{O}(|\mathbb{C}|)$. As $\rho(f)$ is called frequently in the algorithms, we use an *R-tree* index[3] to mark the points and lower the complexity to $\mathcal{O}(\log(|\mathbb{C}|))$.

When searching the boundary points of high density region, it is infeasible to compute the density of every points in the similarity space. To improve the efficiency, we use an approximation technique that split the similarity space into finite cells and only check the points at the corners of each cell.

[3] *R-tree* is a kind of tree data structures for indexing multi-dimensional information [10].

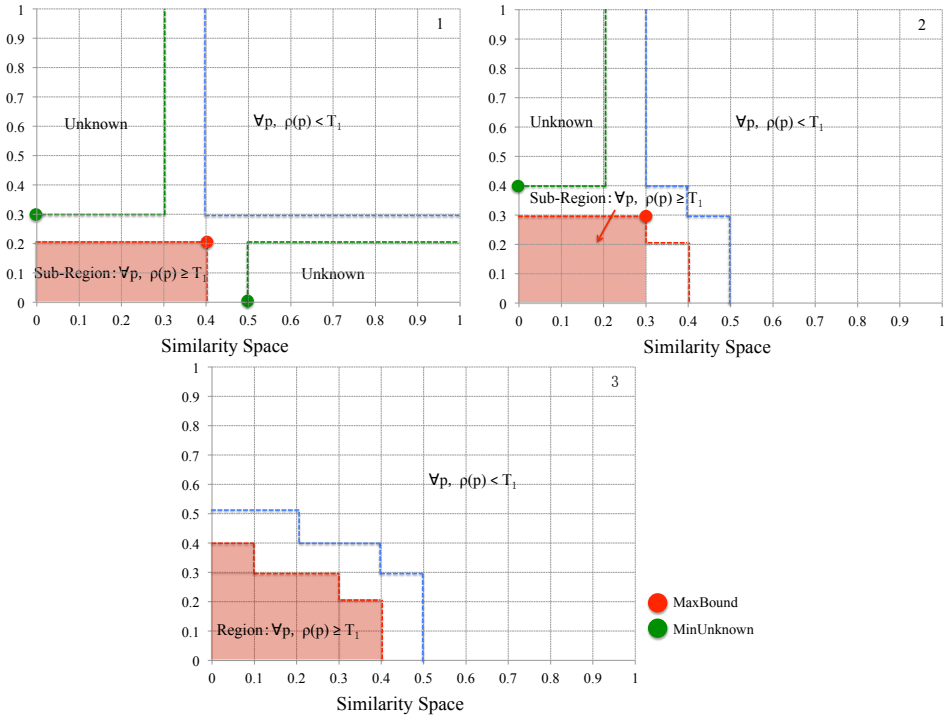

Table 3. Search the region with the density no less than T_1.

In order to split the space, we fix an integer value k, called *granularity parameter*. Then we define a set $\mathbb{V}$ of points, any of which is in the form $(v_1, ..., v_d)$ where $v_i = j/k, j \in 0, 1, ..., k$. The set $\mathbb{V}$ partitions the similarity space into $(1+k)^d$ *cells*. And a *region* is an area formed by several consective cells. When considering the density of region, we only need to check the $(1+k)^d$ points in $\mathbb{V}$. The points are at the grid line crosses in Fig. 2.

4.3 Search for the region with density no less than T_1

Now we show our solution of exploiting the monotonicity of density. Given two points $p \preceq p'$, if the monotonicity holds, $\rho(p) \geq \rho(p')$. From the practical experience in Fig. 1, this monotonicity generally applies in the region when the similarities are rather low. Considering the monotonicity of density, if we can find a point f with $\rho(f) \geq T_1$ and $\forall f' \succ f, \rho(f') < T_1$, this implies that the points f'' with $f'' \preceq f$ are all in the regions with the density no less than T_1. Formally, we define such point f as following:

Definition 5. *Given a threshold* T_1 *and* Monotonicity of Density*, we say a point* $p \in [0,1]^d$ *is maximally dense if* $\rho(p) \geq T_1$ *and* $\forall p' \succ p, \rho(p') < T_1$*. Such a point is called* MaxBound *point. And the set of such points is denoted as* $\mathbb{M}$*.*

According to the monotonicity, the point p that have the property $\exists p' \in \mathbb{M}, p \preceq p'$ should be in the region of density at least equal to T_1. In order to find the redundant non-matched points correctly, we set a relatively high threshold T_1 for enumerating the *MaxBound* points in $\mathbb{M}$. The detected record pairs, which have higher density than T_1, will be blocked out.

Next we propose our searching algorithm. The general idea is that when we locate one *MaxBound* point, it could always help divide the similarity space into three parts: unknown density, density at least equal to T_1 and density lower than T_1. In the unknown region, every point p has the property $\forall p' \in \mathbb{M}, p \npreceq p'$ and $p \nsucc p'$. For the point $f \preceq \forall f'$ in one unknown region, we call such a point f as *MinUnknown* point, which has minimum density in this unknown region. And the set of all *MinUnknown* points is denoted as $\mathbb{U}$. To detect the density of unknown regions, our algorithm repeats the process of finding the *MaxBound* points on the remaining unknown regions.

Our algorithm starts at the point $(1/k, ..., 1/k)$ because it should be the first *MinUnknown* point in $\mathbb{U}$. Then we use *Binary Search* algorithm on every dimension to locate the *MaxBound* point. When finding a *MaxBound* point f, we know that the region resided by the points $p \preceq f$ has a density at least equal to T_1 and the region resided by the points $p \succ f$ has a density less than T_1. But the density of the remaining region is unknown. We put the point with minimum density of unknown region into $\mathbb{U}$ and start a new round of *Binary Search* on next *MinUnknown* point in $\mathbb{U}$. Repeat the same process until no region is unknown in $[0, 1]^d$. We give a simple example on the 2-D similarity space in Fig. 3. In the first round, our algorithm starts at $(0.1, 0.1)$. After the process of *Binary Searching*, the first found *MaxBound* point, $(0.4, 0.2)$, divides the space into four regions. The one resided by the *MaxBound* point is one sub-region of the target region of density T_1. The two unknown ones are the regions which need to be detected in next iteration. And the left one is the region with density less than T_1. In the second round, we start the searching algorithm from the two *MinUnknown* points founded in the last round, $(0, 0.3)$ and $(0.5, 0)$. After this round, the area of unknown regions is reduced and a new sub-region of density T_1 is formed as shown in the second step of Fig. 3. Then, we repeatedly detect the remaining unknown regions until the similarity space has been entirely explored. The union region of all founded sub-regions is the target region with density no less than T_1.

As *MaxBound* points in $\mathbb{M}$ are on the boundary of region with density at least equal to T_1, we then enumerate each point $f \in \mathbb{M}$ and block all the points $p \preceq f$. According to *Monotonicity of Density*, the blocked points must have a density no less than T_1. For the second step, since we use *Binary Search* to locate the *MaxBound* points, the total number of points enumerated is $\mathcal{O}(\log k \cdot d \cdot |\mathbb{M}|)$.

```
1  Procedure EnumerateAllMaxBound(T_1)
2      𝕌 = {(1/k, ..., 1/k)}
3      foreach point p ∈ 𝕌 do
4          if ρ(p) ≥ T_1 then
5              p_max ← FindMaxBound( p, T_1 )
6              𝕄 ← 𝕄 ∪ p_max          /* p_max is a MaxBound Point        */
7              𝕌 ← UpdateMinUnknown( 𝕌, p_max )
8          end
9      end
10 Procedure UpdateMinUnknown(𝕌, p_max)
11     𝕌_new ← ∅
12     foreach point p ∈ 𝕌 do
13         if p ≺ p_max then
14             for i ← 1, d do
15                 p' ← p              /* p' is a MinUnknown point          */
16                 p'[i] ← p_max[i] + 1/k /* p'[i] is the i-th dimension of p' */
17                 𝕌_new ← 𝕌_new ∪ p'
18             end
19         end
20     end
21 Procedure FindMaxBound( p, T_1)
22     for i ← 1, d do
23         hi = 1
24         lo = p_i while hi − lo ≥ 2 do
25             p[i] ← (hi + lo)/2
26             if ρ(p) ≥ T_1 then
27                 lo = p[i]
28             else
29                 hi = p[i]
30             end
31         end
32     end
33     return p
```

Algorithm 1: Enumerate all *MaxBound* points

4.4 Search for the points with density no less than T_2

As mentioned above, the monotonicity property can coarsely distinguish points, but in some region distinguish-ability is not so strong. For a finer grain solution, the points in non-distinguishable region but with a high density should also be blocked out. In this section, we present a recursive algorithm to find the points.

In this step, we set another threshold $T_2 < T_1$ and search for the points of density no less than T_2. Our algorithm starts at some point p in the $\mathbb{M}$. If the neighbour point p' of p has $T_2 \leq \rho(p') < T_1$, we put it into an enumeration set $\mathbb{Z}$, which is equal to $\mathbb{M}$ at the very beginning. Then, we repeat this enumeration process at points in $\mathbb{Z}$, until no more eligible points are found and added into $\mathbb{Z}$.

```
1  ℤ ← 𝕄
2  foreach point p ∈ ℤ do
3  |  for i ← 1, d do
4  |  |  for j ∈ {−1, 1} do
5  |  |  |  p′ ← p            /* p′ is a neighbour point of p          */
6  |  |  |  p′[i] ← p′[i] + j/k    /* p′[i] is the i-th dimension of p′   */
7  |  |  |  if p′ ∉ ℤ and ρ(p′) ≥ T2 then
8  |  |  |  |  ℤ ← ℤ ∪ {p′}
9  |  |  |  end
10 |  |  end
11 |  end
12 end
```

Algorithm 2: Enumerate points with density no less than T_2

With assuming that the points close to a point $f \in \mathbb{Z}$ would also have a density at least equal to T_2, our blocking strategy is to remove all points within r to any one point in $\mathbb{Z}$.

The time consumption of this step is determined by T_2. The case that takes the longest time is when the density of each point in $\mathbb{V}$ on the rest space is just equal to T_2. When computing the density of point $p \in \mathbb{V}$, we need to count the number of points that are within a distance r to p. With some value of r, the density of p may count some points that are also counted by other $2d$ neighbour points of p in $\mathbb{V}$. To make the density identical, the number of shared points counted by two adjacent points in $\mathbb{V}$ should be same and equal to $T_2/2d$. After the last blocking step, the number of remaining points in $\mathbb{C}$, denoted as N_C, and the number of remaining points in $\mathbb{V}$, denoted as N_V, are known to users. There should exist $\frac{N_C}{T_2/2d}$ groups of points counted by two adjacent points in $\mathbb{V}$. And for each point, its density would be counted on $2d$ groups of shared points. Thus, if the maximal number of iterations does not exceed N_V, it can be represented as $\frac{N_C}{T_2/2d} \times \frac{2}{2d}$ equal to $\frac{2N_C}{T_2}$. The result shows the number of iterations does not matter with d. Even though the time complexity is $\mathcal{O}(\frac{N_C}{T_2})$, it should be noticed that N_C is much smaller than $|\mathbb{C}|$. The reason is that most of the non-matches are located in the region of high density and have been blocked in the last step.

5 Experiments

Our experiment were conducted on three datasets: *Restaurant*, *Goods* and *Citation*. *Restaurant* [1] is a collection of addresses of real-world restaurants. *Citation* [2] dataset contains a variety of publication records from DBLP and Google Scholar. *Goods* [2] is a dataset recording the product information on the websites of Google and Amazon. All of the datasets have two tables for record matching. The statistics of records in the datasets are shown in Table 4.

Table 4. Statistics of Records

Datasets	Table A	Table B	Matched Pairs
Restaurant	533	331	112
Citation	2294	2616	2224
Goods	1363	3266	1300

5.1 Method Comparison

In this section, we compare our method with *Standard Blocking*(SB) method [11] and *Canopies*(CANOPY) [12]. The two methods exploit different metrics for blocking. *Standard Blocking* clusters the records that share the same blocking keys into one block. In this experiment, we take the attributes of a dataset as the blocking keys. After having tried a variety of attribute combinations, we pick the one with good performance in terms of *Reduction Ratio* and *Recall* for comparison.

Canopies is an unsupervised clustering agorithm, which requires one loose similarity threshold θ_1 and one tight similarity threshold θ_2. It clusters the records that are textually similar to one block. From the similarity thresholds in the range $[0.3, 0.9]$, we choose the setting that achieves higher scores of *Reduction Ratio* and *Recall* for comparison.

Our method is a density-based clustering algorithm, which requires a loose threshold and a tight threshold of density. It blocks the records that are in high-density areas in the similarity space. We have four parameters all related to density computation. To simplify the process of tuning, we fixed $k = 20$ $r = 0.15$ and vary T_1, T_2 in $[100, 10000]$. In the following, the results are shown in Table 5, 6, 7.

Table 5. Restaurant

	SB	CANOPY	OURS
Reduction Ratio	0.847	0.924	0.992
Recall	1.000	1.000	0.982
Imbalance Ratio	376	253	24

Table 6. Citation

	SB	CANOPY	OURS
Reduction Ratio	0.824	0.944	0.986
Recall	1.000	0.999	0.994
Imbalance Ratio	393	151	38

Table 7. Goods

	SB	CANOPY	OURS
Reduction Ratio	0.831	0.997	0.996
Recall	0.274	0.377	0.885
Imbalance Ratio	180	25	15

From the results, we can observe that the three methods all work well on *Citation* and *Restaurant*. The reason why there is no significant difference is that the two datasets have been cleaned, that is, they do not contain much noisy information. In such a scenario, similarity functions are highly credible, grading non-matched pairs a low score and fully duplicated pairs a high score. But for *Goods*, a dataset with noise, the two competitors do not work well. As the noise generally exists in the attributes of the dataset, the similarity scores of truly matched data pairs may decrease to the level of non-matches'. Thus, a method targeting at the pairs with high similarity scores may not work anymore. As shown in Table 7, *Canopy* only captures 37.7% matched pairs that reside in the area of high similarity.

Our method achieves an outstanding result when noise exists. As the algorithms are to search for the area of non-matches, the noise does not make an influence. For *Goods* dataset, the noise makes matched points enter into the low-similarity area, but also makes non-matched points move to the area with further low similarity. It is still possible to locate the area of non-matches, with using the property of density. Though some matches of low similarity cannot be found, our density-based method still achieves a high recall 0.885 on *Goods* dataset.

5.2 Evaluation on Density

As the density is defined as the number of instances in the area centerred with r, the value of r plays an important role in the algorithms. A small r would make density more local and a big r is closer to the global density. In this section, we focus on studying the value of r.

Table 8. Restaurant

r	0.05	0.1	0.125	0.15	0.2
Reduction Ratio	0.074	0.074	0.731	0.971	0.992
Recall	1.000	1.000	1.000	1.000	0.982
Imbalance Ratio	3081	3081	895	96	24

Table 9. Citation

r	0.05	0.1	0.125	0.15	0.2
Reduction Ratio	0.388	0.388	0.902	0.934	0.986
Recall	1.000	1.000	1.000	0.996	0.994
Imbalance Ratio	1666	1666	266	178	38

Table 10. Goods

r	0.05	0.1	0.125	0.15	0.2
Reduction Ratio	0.429	0.996	0.996	0.996	0.997
Recall	0.987	0.889	0.881	0.881	0.877
Imbalance Ratio	1895	15	15	15	11

To study the effect of r, we fixed $k = 20, T_1 = 1000, T_2 = 200$ and vary the value of r in $\{0.05, 0.1, 0.125, 0.15, 0.20\}$. The experiment is conducted on three datasets as shown in Table 8, 9, 10.

From the results, we can make two conclusions: First, with the increase of value of r, *Reduction Ratio* tends to be greater, while *Recall* and *Imbalance Ratio* tend to be smaller; Second, with a smaller r, our method does not have a good performance.

For the first conclusion, when r becomes greater, according to the definition of density, the thresholds T_1 and T_2 become relatively looser. For a point p with $\rho(p) \leq T_2$, after the increase of r, the area centered at p would become larger and include more points, making the new value returned by $\rho(p)$ possibly larger than T_2. Thus, with a loose threshold, there should be more pairs being blocked by the algorithm, and this leads to a higher *Reduction Ratio* and a lower *Recall*. The second conclusion is due to the noise. When enumerating the points with density no less than T_2, if the algorithm finds a point p with $\rho(p) < T_2$, the neighbour point $p' \succeq p, \rho(p') \geq T_2$ will have no chance to be enumerated. p is the noise point that may make our algorithm early stop, and is the reason why the results of $r = 0.05$ in all tables are bad. With a smaller r, the density has more chance to be affected by some noise points. But with a greater r, the effect of the noise would be balanced off, as shown in the experiment that the results of $r = 0.15, 0.2$ are all good.

6 Conclusion

In this paper, we studied the problem of blocking records for entity matching. Unlike the methods using similarity to filter entities, we used density as main measure for blocking. Our proposed method also exploits the monotonicity of density on improving the algorithm efficiency. Finally, we compared our density based approach with one similarity based approach and one index based approach. We evaluated our method on real datasets and demonstrate the superiority of our approach in terms of *Reduction Ratio* and *Recall*.

References

1. http://archive.ics.uci.edu/ml/.
2. http://dbs.uni-leipzig.de/en/research/projects/object_matching.
3. A. Arasu, M. Götz, and R. Kaushik. On active learning of record matching packages. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 783–794. ACM, 2010.
4. M. Bilenko, B. Kamath, and R. J. Mooney. Adaptive blocking: Learning to scale up record linkage. In *Data Mining, 2006. ICDM'06. Sixth International Conference on*, pages 87–96. IEEE, 2006.
5. M. Bilenko and R. J. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *KDD*, pages 39–48, 2003.
6. S. Chaudhuri, B.-C. Chen, V. Ganti, and R. Kaushik. Example-driven design of efficient record matching queries. In *Proceedings of the 33rd international conference on Very large data bases*, pages 327–338. VLDB Endowment, 2007.
7. W. W. Cohen. Data integration using similarity joins and a word-based information representation language. *ACM Transactions on Information Systems (TOIS)*, 18(3):288–321, 2000.
8. N. N. Dalvi, V. Rastogi, A. Dasgupta, A. D. Sarma, and T. Sarlós. Optimal hashing schemes for entity matching. In *WWW*, pages 295–306, 2013.
9. A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. Duplicate record detection: A survey. *IEEE Trans. Knowl. Data Eng.*, 19(1):1–16, 2007.
10. A. Guttman. *R-trees: a dynamic index structure for spatial searching*, volume 14. ACM, 1984.
11. M. A. Jaro. Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida. *Journal of the American Statistical Association*, 84(406):414–420, 1989.
12. A. McCallum, K. Nigam, and L. H. Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 169–178. ACM, 2000.
13. M. Michelson and C. A. Knoblock. Learning blocking schemes for record linkage. In *Proceedings of the National Conference on Artificial Intelligence*, volume 21, page 440. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006.
14. H. B. Newcombe. *Handbook of record linkage: methods for health and statistical studies, administration, and business*. Oxford University Press, Inc., 1988.
15. L. Shu, A. Chen, M. Xiong, and W. Meng. Efficient spectral neighborhood blocking for entity resolution. In *Data Engineering (ICDE), 2011 IEEE 27th International Conference on*, pages 1067–1078. IEEE, 2011.
16. S. Tejada, C. A. Knoblock, and S. Minton. Learning domain-independent string transformation weights for high accuracy object identification. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 350–359. ACM, 2002.
17. S. E. Whang and H. Garcia-Molina. Incremental entity resolution on rules and data. *VLDB J.*, 23(1):77–102, 2014.
18. S. E. Whang, D. Menestrina, G. Koutrika, M. Theobald, and H. Garcia-Molina. Entity resolution with iterative blocking. In *SIGMOD Conference*, pages 219–232, 2009.