Herausgeber et al. (Hrsg.): Name-der-Konferenz,
0 Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn

Multi-Party Privacy Preserving Record Linkage in Dynamic Metric Space

Ziad Sehili[1] Florens Rohde[2] Martin Franke[3] Erhard Rahm[4]

Abstract: We propose and evaluate several approaches for privacy-preserving record linkage for multiple data sources. To reduce the number of comparisons for scalability we propose a new pivot-based metric space approach that dynamically adapts the selection of pivots for additional sources and growing data volume. Furthermore, we investigate so-called early and late clustering schemes that either cluster matching records per additional source or holistically for all sources. A comprehensive evaluation for different datasets confirms the high effectiveness and efficiency of the proposed methods.

Keywords: Privacy Preserving Record Linkage, Bloom filter, metric space, triangle inequality, Clustering

1 Introduction

Record linkage or entity resolution is the task of finding records in different data sources that describe the same real-world entity, e.g. product or customer. Privacy-preserving record linkage (PPRL) is a special form of record linkage for sensitive data and aims at achieving record linkage while preserving privacy. This kind of record linkage is especially important for person-related record linkage, e.g., for finding matching patient or customer records.

PPRL has received a substantial amount of research interest in the last decade [SBR09, VCV13, Va17]. Most of the proposed approaches aim at improving privacy by matching on encoded record attribute values instead of the original values for identifying attributes, such as person name, address and birth of date. These attributes are called quasi-identifiers (QIDs) as the equality or high similarity in these attributes allows one to find matching persons. Many proposed PPRL schemes also rely on a so-called trusted linkage unit to perform the matching of encoded person records thereby avoiding the need to exchange records between different data owners [VCV13]. Like normal record linkage, PPRL has to achieve a high match quality and scalability to large datasets.

Most proposed PPRL methods focus on the special case of linking two sources only [VCV13]. However, various use cases and data analysis tasks require a PPRL for multiple

[1] Leipzig University, Database Group, Germany sehili@informatik.uni-leipzig.de
[2] Leipzig University, Database Group, Germany rohde@informatik.uni-leipzig.de
[3] Leipzig University, Database Group, Germany franke@informatik.uni-leipzig.de
[4] Leipzig University, Database Group, Germany rahm@informatik.uni-leipzig.de

(≥ 2) sources, e.g., databases from hospitals, clinical studies and census data. Multi-Party PPRL (MP-PPRL) introduces further challenges to be addressed. In particular, the number of record comparisons grows quadratically with the size and the number of sources making scalability a problem. Furthermore, a record may have matches in an arbitrary subset of the data sources not only in one data source. This asks for clustering matching records over multiple sources so that a cluster contains all matches for a specific person. This clustering should utilize that individual sources are often curated and duplicate-free so that every cluster should have at most one record for any data source.

To address these challenges we investigate several novel approaches for MP-PPRL and clustering of encoded records. Specifically, we make the following contributions:

- To reduce the number of comparisons between records we utilize a pivot-based metric space approach [SR16]. We propose an extension of the static approach with a dynamic adaptation of the pivot selection in order to deal with additional data sources and growing data volume.
- We investigate different clustering schemes for multiple parties that either cluster new data sources one after the other (early clustering) or that first determine similar record pairs over all sources before a final clustering is performed (late clustering).
- The presented approaches, PPRL in dynamic metric space and the clustering techniques, are exhaustively evaluated using synthetic and real datasets. We also compare them with baseline approaches.

After a brief discussion or related work we describe basics of PPRL (including the use of Bloom filters to encode quasi-identifiers) and metric space. Section 4 describes the dynamic pivoting approach for multi-party PPRL. The early and late clustering approaches are outlined in Section 5. In Section 6 we evaluate the proposed approaches before we conclude.

2 Related Work

PPRL has been applied in several real health-related use cases, e.g., to compare surgical treatment received by Aboriginal and non-Aboriginal people with non-small cell lung cancer in Australia [Gi16], or to analyze long-term consequences of childhood cancer in Switzerland by linking national data from several cantonal registries [Ku11]. Several surveys [VCV13, Va17, BRF15] categorize the variety of proposed PPRL methods with respect to their challenges of privacy, quality and scalability. The privacy of the represented entities can be provided by using either secure multiparty computation (SMC) to run PPRL between the data owners without involving a linkage unit or by encoding records, e.g. within Bloom filters [SBR09], and then sending the encoded records to a dedicated linkage unit (see Section 3). Although SMC methods for PPRL aim at higher privacy guarantees they are

not applicable in many practical use cases due to their high computational complexity. The linkage quality depends on many factors like the input data, the linkage and classification methods [Ch12], and the encoding technique. While the first factors are not specific to PPRL but also apply to record linkage in general, the effect of encoding records to Bloom filters before comparing them have been studied in [SBR11, Du12]. The authors have shown that the linkage quality of Bloom filters does not drop significantly compared to the original (string) records. We will thus use Bloom filters for encoding records in this paper.

In previous PPRL papers, the scalability problem has been addressed by applying blocking or filtering techniques for Bloom filters to reduce the number of comparisons [Se15, SR16]. Furthermore, the PPRL computations at a linkage unit can be run in parallel [Gl18, FSR18]. All previous blocking and filtering schemes for PPRL are static so that their effectiveness tends to decrease with increasing data volume (e.g. the number of comparisons per block mostly grows quadratically with the block size). In our previous work [SR16] we introduced the use of a pivot-based metric space approach [Ze06] for PPRL with two sources. In contrast to blocking, this approach can reduce the number of comparisons without introducing from recall reductions. The main drawback of this method is its upfront selection of a static set of reference records (pivots) making it inapplicable for dynamically growing number and size of sources. To overcome this problem we advise a new dynamic pivoting scheme. In a non-PPRL setting a related dynamic approach, Sparse Spacial Selection (SSS), has been proposed in [PB07], that dynamically selects reference records from one source depending on the spacial distribution of records in the metric space. We use this method as baseline in our evaluation.

Multi-party record linkage with a clustering of matching records without the privacy prerequisite was studied in [Sa18]. The approach uses a static blocking scheme, determines similar pairs of records from all sources (and keeps them in a similarity graph) before a final clustering is performed (late clustering). [Fr18] shows that PPRL match results for two duplicate-free sources can be improved by a post-processing to determine a 1:1 match mapping. This can be achieved by a Hungarian or Max-Both approach and we will use these methods in our early clustering schemes for multiple parties. [VCR20] investigates clustering techniques for MP-PPRL, but for static blocking instead of our dynamic metric space filtering.

3 Preliminaries

This section starts by introducing the multi-party PPRL process, then presents the Bloom filter encoding scheme that preserves privacy and similarity of records. Furthermore, we explain the use of metric space to improve scalability of the linkage process.

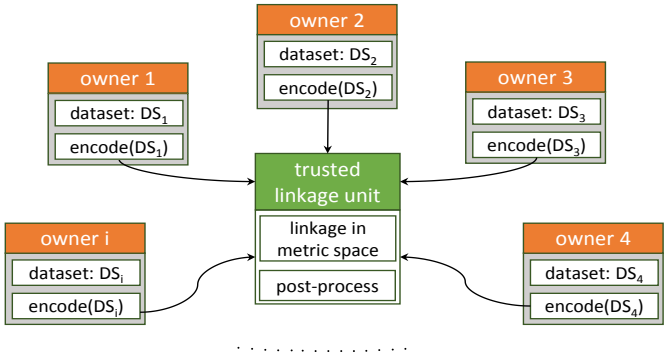

Fig. 1: Multi-Party PPRL: The data owners encode their records and send them to a trusted linkage unit (LU). The LU links the records and determines clusters of matching records

3.1 Multi-Party PPRL

A Multi-Party PPRL process that involves i data holders and a trusted linkage unit (LU) to run the linkage process is depicted in Fig. 1. We assume duplicate-free datasets DS_1, DS_2, ..., DS_i. Quasi-identifying attributes of the records are first encoded to Bloom filters (see below) by their respective data holders to preserve the privacy of the represented persons. The encoded records are sent to the LU where a linkage process using the metric space approach is applied for improved scalability (explained in Sec. 4) and where clusters of similar records are generated. Furthermore, and due to the assumption that the sources are duplicate-free, a post-processing or cleaning step is run to ensure that each cluster contains at most one record from any source. Hence, the size of any clean cluster c is $1 \leq |c| \leq i$. Clustering strategies are discussed in Sec. 5.

3.2 Bloom Filter

The use of Bloom filter [Bl70] to encode records involved in PPRL was introduced in [SBR09]. The encoding scheme of one record takes as input a set of q-grams (bi- or tri-grams) of the relevant attributes (e.g. first and last name, date of birth and address), a bit array of length l with all positions initially set to zero and a set of k independent cryptographic hash functions that return values in $[0, l-1]$. Then each q-gram is mapped to the bit array k-times using the k hash functions by setting the corresponding positions to 1. Figure 2 shows a simple example of two similar names and their encoding to Bloom filters of length $l = 20$ using $k = 2$ hash functions.

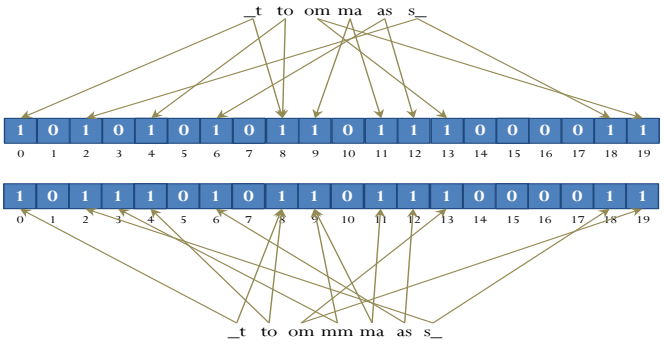

Fig. 2: Bloom filter (bit vector) encoding of two names *tomas* and *tommas*, each tokenized to bi-grams, using $k = 2$ hash functions and bit vectors of length $l = 20$

3.3 PPRL in Metric Space

A metric space $\mathcal{M}(\mathcal{U}, d)$ consists of a set of data objects $\mathcal{U}$ and a metric d to compute the distance between these objects. The main property satisfied by the distance metric d is the *triangle inequality* of the distances: $\forall x, y, z \in \mathcal{U} : d(x, z) \leq d(x, y) + d(y, z)$. This inequality can be used to considerably reduce the search space without discarding any pair of similar records, i.e., without reducing recall [Ze06].

For a data object $q \in U$ the match candidates cannot be outside a radius $rad(q)$ in order to satisfy a maximal distance threshold. The triangle inequality allows one to avoid computing the distance between points x and q based on precomputed distances to a reference point p, also known as pivot. Hence, only objects that satisfy:

$$|d(p, q) - d(p, x)| \leq rad(q) \tag{1}$$

need to be considered as potential matches so that the distance $d(q, x)$ has to be computed to determine whether q and x match.

PPRL processes generally use a similarity function like Jaccard to compare records and a threshold t to classify pairs as similar or non-similar. Since metric spaces rely on metrics (distance function), we can use the Hamming distance, that was shown to be equivalent to the Jaccard similarity [Xi08], to search matching records. The radius $rad(q)$ that includes similar records to q can be inferred from t as [SR16]:

$$rad(q) = |q| \times \frac{1 - t}{t}.$$

where $|q|$ is the number of positions set to 1 in the Bloom filter. Records having a Hamming distance $\leq rad(q)$ are those that have a Jaccard similarity $\geq t$.

For two sources DS_1 and DS_2, the pivot-based metric space approach for PPRL operates in two steps: index building and similarity search. Indexing source DS_1 first requires selecting a set P of m data objects that serve as pivots. One proven approach is to select these pivots from DS_1 in an iterative manner by choosing the objects having the maximum distance to each other [SR16]. Next, each object $x \in DS_1$ is assigned to its closest pivot $p_i \in P$, $elem(p_i) = \{x \in DS_1 : dist(x, p_i) < dist(x, p_j), \forall j \neq i\}$. For every object $x \in DS_1$ the precomputed distance $d(x, p_i)$ to its pivot p_i is stored as well as the radius of a pivot $rad(p_i)$, i.e., the maximal distance for any object assigned to pivot p_i.

For each (query) object $q \in DS_2$, the matching with DS_1 objects involves a similarity search with radius $rad(q)$. This search for match candidates can utilize two filter steps. First, only those pivots p_i need to be considered for which the pivot radius $rad(p_i)$ overlaps with $rad(q)$ since otherwise all of the pivot's objects are outside radius $rad(q)$ and cannot match. Secondly, for the remaining pivots p_i the number of its objects x can be reduced according to the above triangle inequality.

4 MP-PPRL in Dynamic Metric Space

We first explain our approach to dynamically increase the number of pivots for growing data volume and then explain the use of this approach for multi-party PPRL.

4.1 Dynamic Pivoting

The static pivot method with its upfront determination of pivots has two problems: 1) the number of pivots is difficult to determine and dependents on the number and distribution of records. 2) The number of pivots should be adequately increased with growing number and size of sources to be indexed since more records per pivot leads to more comparisons and thus poor scalability. Furthermore, more records per pivot lead to increased pivot radii and higher overlap between the pivots (see below) which in turn can cause that more pivots need to be considered to find the matches of a query record. Hence, the potential to reduce the number of comparisons can be severely reduced when we keep a static set of pivots in the presence of strongly growing data volume, e.g. due to additional data sources.

To overcome these problems for multi-source PPRL, we devise an algorithm to adapt the pivot selection dynamically during the indexation of an additional data source. For this purpose we control the so-called pivot overlap using a parameter α. Note that an indexed record x may be in the radius of several pivots p_i ($dist(x, p_i) < rad(p_i)$) although it is assigned to only one (the closest) pivot. Similar to the idea introduced in [Tr00], the overlap between pivots can thus be determined by the average number of intersections per record.

Algorithm 1: Dynamic Pivot-based indexing

Input : dataset DS;
maximal intersection α;
Output: set P of pivots with their assigned elements

```
1  if DS is the first source then
2      I_c = 0; // intersections records pivots
3      N = 0; // number of read records
4      P = S; // small set S of initial pivots with rad(p_i) = 0, for p_i ∈ P
5  foreach x ∈ DS do
6      N = N + 1;
7      minDist = ∞;
8      bestPivot = null;
9      foreach p_i ∈ P do
10         if dist(x, p_i) ≤ rad(p_i) then
11             I_c = I_c + 1;
12         if dist(x, p_i) < minDist then
13             minDist = dist(x, p_i);
14             bestPivot = p_i;
15     elem(bestPivot) = elem(bestPivot) + {x};
16     if rad(bestPivot) < minDist then
17         rad(bestPivot) = minDist;
18     overlap(P) = I_c / (N × |P|);
19     if overlap(p) > α then
20         run Algorithm 2 // Generate new pivot
```

Let I_c be the sum of the number of pivot intersections for any record x, N the number of indexed objects, and $|P|$ the number of pivots.We define the *overlap factor* of the pivots as:

$$overlap(P) = \frac{I_c}{N \times |P|} \tag{2}$$

The overlap factor thus determines the average number of pivot intersections per record normalized by the total number of pivots. A low overlap value means that the data objects are largely partitioned according to their pivots, e.g., when the radii are relatively narrow. Additional data objects are assigned to the closest pivots so that the radii and thus the overlap between pivots tends to increase. This will also reduce the filter effects and thus increase the number of necessary match comparisons.

We use an adaptation parameter α to control the maximal overlap ratio between pivots, i.e., we increase the number of pivots as soon as this value is exceeded in order to increase the number of pivots for a growing number of data objects. Algorithm 1 specifies the dynamic indexing using parameter α. The algorithm starts with a small set of pivots P (line 4), that

Algorithm 2: Generate new pivot

Input : set of pivots P with their elements;

```
1  choose p ∈ P; // pivot with max cardinality
2  choose p_new = x ∈ elem(p): furthest element from p;
3  foreach p_i ∈ P do
4      foreach e ∈ elem(p_i) do
5          if dist(e, p_new) ≤ dist(e, p_i) then
6              elem(p_new) = elem(p_new) + {e};
7              store dist(e, p_new);
8              update rad(p_new); // if necessary
9              elem(p_i) = elem(p_i) − {e};
10             update rad(p_i); // if necessary
11 P = P + {p_new};
```

can be chosen for example randomly. In the next step, new records of the dataset are read sequentially and compared with the existing pivots. I_c is incremented each time the distance between x and any pivot p_i is smaller than the radius $rad(p_i)$ (line 10 and 11). Finally, x is added to the elements of the closest pivot and the radius of this pivot is possibly updated (lines 15-17). Furthermore, the overlap factor is calculated and compared with α (lines 18-20). If the overlap exceeds the value of α a new pivot p_{new} is generated (Algorithm 2) and all the objects already indexed are re-partitioned over the pivots.

Note that there are different strategies to select a new pivot (line 1 and 2 of Algorithm 2):, e.g., choose the new pivot as the furthest record from the elements of the pivot with the highest cardinality (approach *MaxCard*) or from the pivot having the largest radius (*MaxRadius*). Another approach is to select the next pivot as the most furthest object to the already chosen pivots (*FurthestNode*). Preparatory experiments showed the highest effectiveness for *MaxCard* so that we will use this approach in our evaluation.

4.2 MP-PPRL using Dynamic Pivots

Running MP-PPRL using dynamic pivot based metric space is now straightforward and its steps are shown in Algorithm 3. For a set of data sources DS_i we start by indexing the first source DS_1 using algorithms 1 and 2. For each additional dataset DS_j $j \geq 2$ we run two methods successively: *link*(DS_j) which finds matches M_j between records of DS_j and records already assigned to $p_i \in P$ from former sources. The second method *index*(DS_j) partitions records of DS_j on the existing pivots $p_i \in P$ following the algorithms 1 and 2 which might lead to the generation of additional pivots.

This algorithm outputs pairs of similar records that form a similarity graph. By computing the connected components from this graph we generate initial clusters of matches. These

Algorithm 3: MP-PPRL in dynamic metric space

Input : $D = \{DS_j, j \in 1..i\}$ datasets;
P set of pivots;
Output : C set of clusters;

$P = \emptyset$;
$index(DS_1)$ // using algorithms 1 and 2
for $j \geq 2$ **do**
    $link(DS_j)$;
    $index(DS_j)$;

clusters may still have several records per source and we will apply a post-processing steps to solve this problem.

In what follows we use the term *linkage-iteration* to denote the execution of $index(DS_i)$ on data source DS_i and $link(DS_{i+1})$ on the forthcoming data source DS_{i+1}.

5 Clustering

The goal of clustering is to group all matching records (Bloom filters) based on the previously determined pairs of matching records. Due to the assumption of duplicate-free sources, we have to ensure that every cluster should be *clean*, i.e., include at most one record per source. In this section we outline several approaches for early and late clustering that generate clean clusters during and after the linkage process, respectively.

5.1 Early Clustering

Early clustering algorithms build clean clusters progressively after each linkage-iteration by considering the linkage output of each iteration as a weighted bipartite graph. Based on some predefined criteria, edges are deleted from the graph so that each resulting cluster contains at most one record from any involved source. We describe two representative algorithms, Hungarian and Max-Both, of such a strategy.

Hungarian Algorithm: The Hungarian algorithm [Ku55] is a combinatorial optimization method to solve the assignment problem in polynomial time. The input of the algorithm is a weighted bipartite graph $G(S, T, E_w)$ with S and T consisting of two disjoint sets of vertices representing records from two different sources DS_i and DS_j and a set of weighted edges $E_w = \{(s, t) : s \in S \wedge t \in T\}$ where the weights represent the similarity values for pairs of elements of S and T. The goal of the Hungarian algorithm is to find an assignment with the highest global similarity between the elements of S and T so that each element from each set is linked with at most one element from the other set (1:1 mapping).

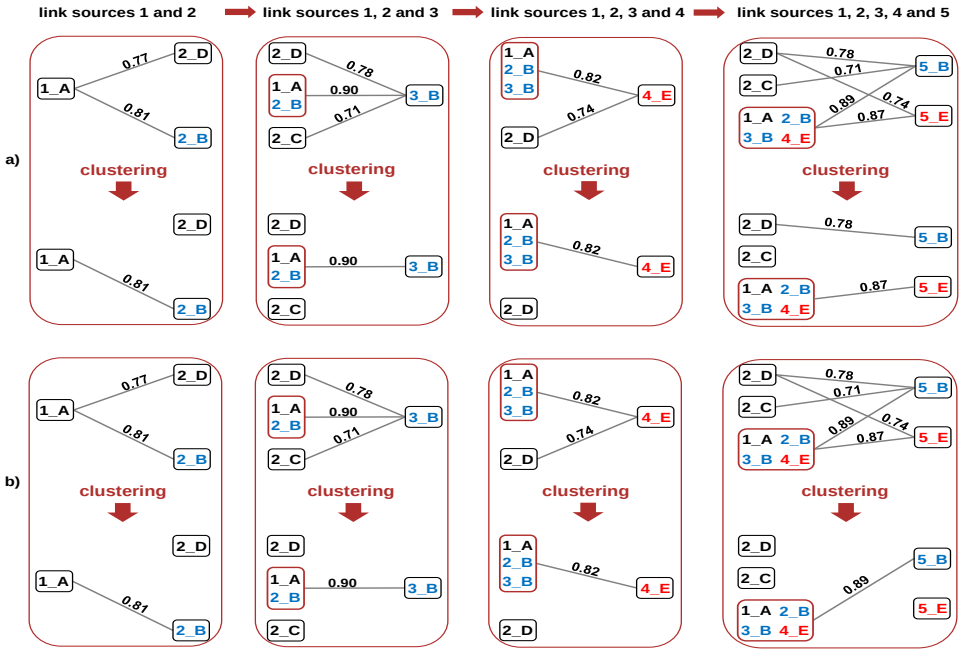

Fig. 3: Clustering the linkage result of 5 sources using the a) Hungarian Algorithm and b) Max-Both

For MP-PPRL the Hungarian algorithm is run after each linkage-iteration of a new dataset. Hence, after the linkage of the first two sources, DS_1 and DS_2 the algorithm is run in a straightforward manner on graph $G(DS_1, DS_2, E_e)$ and a set of clusters $c_i \in C$ with $1 \leq |c_i| \leq 2$ is generated. For each further source DS_i, $i \geq 3$, the linkage step computes the similarities between the elements $t \in DS_i$ and the clusters $c_i \in C$ and generates a new weighted bipartite graph $G(C, DS_i, E_w)$, that serves as an input for the Hungarian algorithm. The top of Fig. 3 shows the linkage of 5 sources and the application of the Hungarian algorithm to cluster the results. A perfect clustering would output the following clusters: $c_1 = \{2\_B, 3\_B, 5\_B\}$, $c_2 = \{4\_E, 5\_E\}$, and the singletons $c_3 = \{1\_A\}$, $c_4 = \{2\_C\}$, $c_5 = \{2\_D\}$ (colored with blue, red and black respectively).

Max-Both: Max-Both was introduced in [MGR02, DR02] to solve the multimapping problem of two sources. Max-Both works in a similar manner as the Hungarian algorithm; in each linkage-iteration it finds the 'best' assignment in a weighted bipartite graph. The only difference is that Max-Both keeps a link (s, t) only if this link is the best for both elements s and t. Therefore, for a bipartite graph $G(S, T, E_e)$ Max-Both starts by selecting for each element $s \in S$ the element $bestMath(s) = t \in T$ so that $sim(s, t) > sim(s, t') : \forall t' \in T$ and $t' \neq t$. On the other side, for each $t \in T$ the element $bestMath(t) = s \in S$ is selected so that $sim(s, t) > sim(s', t) : \forall s' \in S$ and $s' \neq s$. Then a mapping (s, t) is returned only if $bestMatch(s) = t \wedge bestMatch(t) = s$.

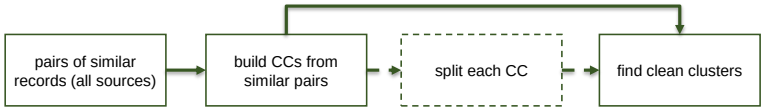

Fig. 4: Steps involved in late clustering

similar pairs

id_1	id_2	sim
1_A	2_D	0.77
1_A	**2_B**	**0.81**
1_A	3_E	0.81
1_A	4_E	0.94
1_A	**5_E**	**0.94**
1_A	**5_B**	**0.81**
2_C	**3_B**	**0.71**
2_C	5_B	0.71
2_B	**3_B**	**1.00**
2_B	4_E	0.76
2_B	5_E	0.76
2_B	5_B	1.00
2_D	**3_B**	**0.78**
2_D	**4_E**	**0.74**
2_D	**5_E**	**0.74**
2_D	**5_B**	**0.78**
3_B	**4_E**	**0.76**
3_B	5_E	0.76
3_B	**5_B**	**1.00**
4_E	**5_E**	**1.00**
4_E	5_B	0.76
4_F	5_E	0.71

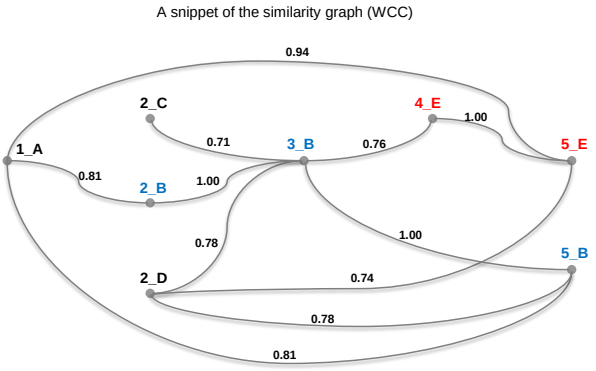

Fig. 5: Transformation of similar pairs to a similarity graph (weighted connected component). For clarity only the edges for the bold entries on the left side are shown.

The Bottom of Fig. 3 displays the output of Max-Both. Due to the simplicity of the input data Max-Both and the Hungarian algorithm return the same results in the first three iterations. In the last iteration, however, Max-Both keeps the best edge (higher similarity) between the large cluster and the singleton 5_*B* and prune all other edges. This leads to a more homogeneous cluster compared to the Hungarian algorithm, which try to maximize the weights by preserving a larger number of edges.

5.2 Late Clustering

In this approach the linkage process is run between records of all sources to generate a similarity graph. Clustering is performed late and uses a weighted connected component (WCC) as input instead of a weighted bipartite graph. The rationale behind this approach is to preserve a global view of the linkage result and avoid pruning edges in early iterations that might be good in later ones. Fig. 4 shows the steps of late clustering. First, the pairs of similar records from all sources are used to build WCCs, an example of such transformation is shown in Fig. 5. Then an optional step to split WCCs is run to reduce the size of large CCs, and thus reduce the complexity for clustering. To split connected components we can use either the method introduced in [Sa18] by deleting so-called weak links (the records connected with such links have at least one higher-similarity link to another record of the respective other source) inside each CC, or use the k-medoid algorithm to group a set of

a) sorting according to the avg. similarity

order	cluster	avg. sim
1	**[2_B, 3_B, 5_B]**	1.0
2	[2_B, 5_B]	1.0
3	[2_B, 3_B]	1.0
4	[3_B, 5_B]	1.0
5	**[4_E, 5_E]**	1.0
6	[1_A, 4_E, 5_E]	0.96
⋮		
67	[2_C, 3_B, 4_E]	0.74
68	[2_D, 5_E]	0.74
69	[2_D, 4_E]	0.74
70	[2_C, 5_B]	0.71
71	[2_C, 3_B]	0.71
72	[4_F, 5_E]	0.71

b) sorting according to the number of edges

order	cluster	# edges
1	**[1_A, 2_B, 3_B, 4_E, 5_B]**	10
2	[1_A, 2_B, 3_B, 4_E, 5_E]	10
3	[1_A, 2_D, 3_B, 4_E, 5_E]	10
4	[1_A, 2_D, 3_B, 4_E, 5_B]	10
5	[1_A, 2_B, 3_B, 5_B]	6
6	[2_B, 3_B, 4_E, 5_B]	6
⋮		
67	[4_E, 5_B]	1
68	**[2_D, 5_E]**	1
69	[2_D, 4_E]	1
70	[2_C, 5_B]	1
71	[2_C, 3_B]	1
72	[4_F, 5_E]	1

Fig. 6: The outputs of the algorithm Sort and Keep Best on the similarity graph of Fig. 5 using (a) avg. similarity and (b) number of edges. Returned clusters for each sorting criteria are green framed.

records into clusters of similar objects. In the following, we present two late clustering methods, graph multicut and SKB (sort and keep best).

Graph Multicut: Multicut is a graph partitioning problem that takes as input a connected graph $G(V, E)$, a weight function $w : E \rightarrow R$ that assigns weights to the edges E, and a set of k pairs $(s_1, t_1), (s_2, t_2), \ldots (s_k, t_k)$. The algorithm tries to find the set of edges $F \subset E$ whose removal disconnect each pair (s_i, t_i) for $i = 1 \ldots k$. A minimum graph multicut returns the set F with the lowest cost. This problem can be defined as a linear program:

$$\begin{aligned} \text{minimize} \quad & \sum_{e \in E} w_e \times d_e \\ \text{subject to} \quad & \sum_{e \in p} d_e \geq 1 \qquad \forall p \in P_{s_i,t_i}, i = 1 \ldots k \\ & d_e \geq 0 \qquad\qquad \forall e \in E \end{aligned} \tag{3}$$

This linear program assigns the smallest positive value d_e to each edge $e \in E$ which is on any path p_i that joins the pairs s_i, t_i (pairs to be disconnected) so that the sum of the assigned values d_i for each path is greater than 1. The edges to be removed (cut edges) are contained in the set $F = \{e \in E : d_e \geq 0.5\}$[GVY93].

This algorithm can be easily adapted to cluster the result of MP-PPRL. Each WCC constitutes a graph $G_i(V_i, E_i)$ of records V_i and weighted edges E_i (similarity values). We consider each pair of records $x, y \in V_i$ from the same source as terminals to be disconnected, and generate clusters by solving linear program 3. For the example graph in Fig. 5 the pair $(5\_E, 5\_B)$ from the fifth source constitutes two terminals to be disconnected. Multicut algorithm assigns to each edge of the paths connecting 5_E and 5_B the smallest possible value d_i then prunes all edges having a value ≥ 0.5. Applying Multicut on the graph of Fig. 5 returns the following clusters: $\{1\_A, 2\_B, 3\_B, 4\_E, 5\_E\}$, $\{2\_D, 4\_F, 5\_B\}$ and

Algorithm 4: Sort and Keep Best (SKB)

Input : WCC weighted connected component;
K sorting criteria;
Output : C set of clean clusters;;
$C = \emptyset$;
$tmp = \emptyset$;// to store records of clean clusters
generate all possible clean clusters c_i;
sort c_i according to K;
foreach $clr \in c_i$ **do**
    if $clr \cap tmp = \emptyset$ **then**
        $C = C + \{clr\}$;
        $tmp = tmp \cup clr$;// add records of clr to tmp
    else
        prune clr;

the singleton $\{2\_C\}$. As we can see, Multicut is a kind of generalisation of the Hungarian algorithm for multiple sources which tries to keep as many edges as possible.

Sort and Keep Best (SKB): Algorithm 4 implements this method. It takes as input a WCC and starts to generate all possible clean clusters contained in it. Based on some criteria, the algorithm sorts the generated clusters. Finally, the clusters are read sequentially and it is checked whether the actual cluster includes records that are contained in better (already processed) clusters or not (line 7). If this is not the case the cluster is added to the set of final clusters and its elements are stored to check forthcoming clusters (line 8-9).

We evaluated two criteria to sort the clusters: 1) Average similarity inside the cluster $Avg\_sim = \sum_{e \in E_c} w_e / |E_c|$, if Avg_sim of two clusters are equal then sort according to $|E_c|$. 2) The number of edges $|E_c|$ between the elements of a cluster, if $|E_c|$ of two clusters are equal then sort according to Avg_sim. Fig. 6 illustrates the output of both sorting criteria on the similarity graph of Fig. 5. Sorting the clusters by Avg_sim leads generally to small but more homogeneous clusters than sorting them by edges. The latter method tends to create large clusters that might include miss-matches.

6 Evaluation

After the description of the experimental setup, we evaluate the performance of the proposed dynamic selection of pivots in metric space for MP-PPRL. In particular, we analyse the influence of the overlap parameter α and present a comparison with Sparse Spacial Selection (SSS), an alternate dynamic pivoting scheme. Furthermore, we conduct a comparative analysis of the match quality and runtimes of the presented early and late clustering strategies.

# sources	dataset	1	2	3	4	5	6	7	8	9	10
3	S-NCVR	58%	17%	25%							
	R-NCVR	20%	50%	30%							
5	S-NCVR	60%	3%	5%	7%	25%					
	R-NCVR	19%	24%	24%	19%	14%					
7	S-NCVR	60%	1%	2%	3%	4%	5%	25%			
	R-NCVR	16%	21%	21%	16%	11%	8%	7%			
10	S-NCVR	61%	1%	1%	1%	1%	2%	2%	3%	3%	25%
	R-NCVR	15%	20%	20%	15%	10%	6%	5%	4%	3%	2%

Tab. 1: Distributions of duplicate records over the sources to be matched for both dataset collections S-NCVR and R-NCVR.

6.1 Experimental Setup

For our experiments we use two collections of datasets with different sizes and number of sources (parties) to simulate a MP-PPRL process. The first collection, S-NCVR, was generated from the publicly available North Carolina Voter Register (NCVR) dataset. Using a snapshot of about 7 million persons several sources have been generated. Duplicate records over the different sources have been created by introducing some modifications (typos) to the original attribute values. The second collection R-NCVR, also obtained from NCVR, represents real data without any modification. Duplicate records over sources arise from real changes or modification in the attributes values of some persons. Table 1 shows the number of sources and the distribution of duplicate records over the sources for both sizes $100,000$ and $500,000$. The two collections have different duplicates distributions, e.g., to link three parties each containing $100,000$ records, the three sources from S-NCVR contains about 60% of singletons, 17% of records are duplicate in two sources and 25% of records are present in all three sources. The distribution of sources from R-NCVR decreases the number of duplicates over sources when the number of sources increase. Note that the duplicates are a mixture of similar and equal records distributed over the sources.

Records of both collections of data have been encoded to Bloom filters of length $1,000$ using between 10 and 30 hash functions. All experiments are conducted single-threaded on a machine with a 4-core $4.00GHz$ CPU and 32 GB of main memory.

6.2 Evaluation of Dynamic Pivoting

6.2.1 Influence of max overlap value α

We use the first collection of datasets R-NCVR to determine the influence of parameter α and to determine effective settings for it. We run MP-PPRL without any clustering method to

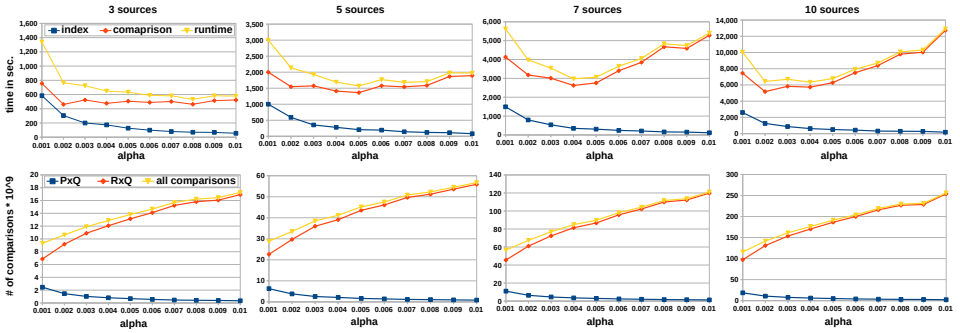

Fig. 7: Runtimes (index and comparison) and number of comparisons w.r.t. the value of α to link 3,5,7 and 10 sources each containing 100, 000 records.

link sets of sources with different configuration w.r.t the number of sources and the size of each source. We varied the value of α between 0.001 and 0.01 and set the similarity threshold to 0.75. For each configuration we evaluate the time needed to index records, i.e., to assign them to their pivots, and to find the matches. Furthermore, we analyze the number of distance computations.

The top of Fig. 7 shows the index, comparison and total runtime to link sets of sources (3, 5, 7 and 10 sources) each containing 100, 000 records. We observe that small α values < 0.002 lead to both high index and comparison times. For $\alpha = 0.001$ the index time, i.e., the time to dynamically generate new pivots and to distribute records on these pivots, represents about 43% of the total runtime when MP-PPRL is run for three sources. The reason of this high index time is that low α values are frequently exceeded leading to the determination of additional pivots and a corresponding reassignment of record to these pivots. Furthermore, the generated pivots need to be compared with all the query records from the second source to check the triangle inequality, which increases comparison time. For $\alpha = 0.001$ the number of dynamically generated pivots from the first source is about 10, 000 pivots that must be compared with the 100, 000 records from the second source before the real linkage begins.

On the other side, higher α values decrease indexing time because fewer pivots are generated. However, the comparison time increases dramatically when $\alpha > 0.005$ and the number of sources to link is greater than five. For such α values a large number of records are distributed on a small number of pivots in the index phase which causes an enlargement of the pivots radii and therefore their ability to exclude pairs from farther comparison using the triangle inequality. This can be shown in the bottom of Fig. 7, where the number of distance computation to link ten sources grows from 97×10^9 to 254×10^9 when α is changed from 0.001 to 0.01.

Another behaviour we can observe is that the comparison time is not always related to the number of comparisons computed. By assigning α small values we reduce in fact the

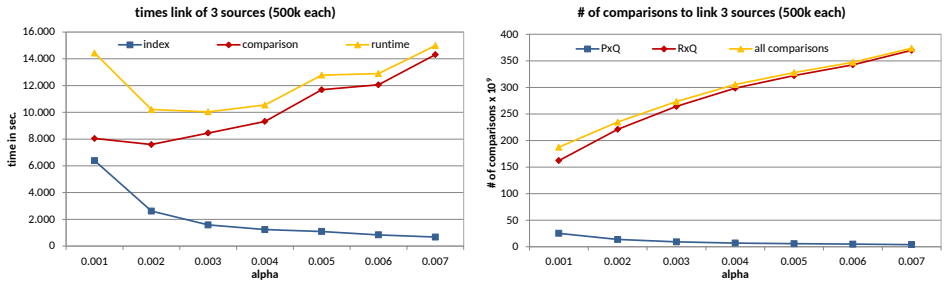

Fig. 8: Runtimes (index and comparison) and number of comparisons w.r.t. the value of α to link 3 sources each containing 500, 000 records.

number of comparisons (queries with pivots + queries records), however, the computation time does not follow this trend. This is due to the overhead of parsing all queries for each pivot and the distance function (XOR) that is known to be very cheap to execute.

To investigate the best α value for large dataset we run MP-PPRL to link a set of three sources each containing 500, 000 records and varying α between 0.001 and 0.007. The right part of Fig. 8 shows the index, comparison and total runtime. For this experiment we observe the same behaviour as for smaller datasets, i.e., very small α values ≤ 0.002 increase both the index and comparison time, and large α values ≥ 0.003 decrease the index time but increase the comparison time for the same reason as mentioned above. As we can see the best runtime (index + comparison) is obtained for α values between 0.002 and 0.004. We use $\alpha = 0.003$ in the following experiments.

A very high value of parameter α, e.g., 1, means that the generation of new pivots will never be triggered which corresponds to a static pivoting approach where the initially selected pivots are not changed any more. The shown curves in Fig. 7 and 8 show that the runtimes and number of comparisons for the highest α values are much worse than for the best settings of α which underlines the high value of the proposed dynamic pivoting approach.

6.2.2 Comparison with Sparse Spatial Selection (SSS) Method

We now compare the performance of our method of dynamically selecting pivots with the the Sparse Spatial Selection method (`SSS`) that was proposed outside the context of PPRL. `SSS` starts with a random record x as pivot. The next pivots are records having a distance $\geq M \times \beta$ to any already selected pivot, being M the maximal distance between any two records in the dataset and β a constant parameter taking values in $[0.35, 0.40]$. We run some initial experiments and found $\beta = 0.54$ the optimal value for our datasets. To compare our method with `SSS`, we run MP-PPRL to link datasets of 3 and 7 sources each containing 500, 000 records from the second collection, `S-NCVR`. Table 2 shows the achieved results for our method (named `overlap`) and `SSS` for different similarity threshold $(0.75 - 0.95)$.

		0.75		0.8		0.85		0.9		0.95	
		SSS	**overlap**	**SSS**	**overlap**	**SSS**	**overlap**	**SSS**	**overlap**	**SSS**	**overlap**
3 src.	**# Pivot_1**	17,064	6,247	17,064	6,247	17,064	6,247	17,064	6,247	17,064	6,247
	# Pivot_2	21,436	7,276	21,436	7,276	21,436	7,276	21,436	7,276	21,436	7,298
	P x Q ($\times 10^9$)	19.25	**6.76**	19.25	**6.76**	19.25	**6.76**	19,25	**6.76**	19.25	**6.77**
	R x Q ($\times 10^9$)	**175.90**	228.83	**37.04**	62.71	**7.69**	14.82	**1.91**	3.91	**0,48**	1.06
	index time	35	**16**	32	**16**	26	**15**	26	**15**	29	**15**
	runtime	179	**153**	102	**72**	85	**57**	57	**46**	**51**	54
7 src.	**# Pivot_1**	16,825	6,279	16,825	6,279	16,825	6,279	16,825	6,279	16,825	6,279
	# Pivot_2	30,742	9,724	30,742	9,725	30,742	9,726	30,742	9,741	30,742	9,835
	P x Q ($\times 10^9$)	74.33	**24.64**	74.33	**24.64**	74.33	**24.64**	74.33	**24.65**	74.33	**24.82**
	R x Q ($\times 10^9$)	**1,235.27**	1,412.03	**265.37**	366.05	**55.57**	85.53	**13.92**	22.77	**3.50**	6.13
	index time	48	**37**	47	**39**	33	**31**	**33**	47	**34**	38
	runtime	1,237	**1,185**	545	**412**	309	**176**	259	**162**	292	**152**

Tab. 2: Comparison of our method (`overlap`) with `SSS` to link 3 and 7 sources from the `S-NCVR` each containing 500,000 records. We investigate the number of pivots generated in the first (# pivot_1) and last (# pivot_2) linkage iteration, the number of comparisons between pivots and queries (P x Q) and between indexed records and queries (R x Q), the index time and the complete runtime in minutes. (best values in bold)

We report the number of generated pivots, the number of comparisons between pivots and queries (P x Q) and between indexed records and queries (R x Q), Furthermore, we consider the index time needed to find adequate pivots and partition records over them and the complete runtime.

We observe that `SSS` generates many more (three times more) pivots as our overlap-based method. To index the first source (for both 3 and 7 sources) `SSS` needs about $17,000$ pivots while our method generates only about $6,000$ pivots for the same source. During the MP-PPRL process `SSS` continues to generate numerous pivots whose number reach $30,000$ to compare 7 sources. While the high number of pivots generally can reduce the number of comparisons between indexed records and queries (as explained before) it leads, however, to a large index time and runtime, and a high number of comparisons between queries and pivots. For both number of sources (3 and 7) `SSS` requires three times more comparisons between pivots and queries as our method (6.76×10^9 vs. 19.25×10^9 for 3 sources). Hence, our methods outperforms `SSS` w.r.t. the quality and number of pivots generated and the runtime of MP-PPRL.

6.3 Comparison of Clustering Methods

6.3.1 Number of sources

We first evaluate the quality of the proposed early and late clustering approaches, Hungarian algorithm (HUNG), Max-Both (MAX-B), graph multicut (M-CUT) and sort and keep best using average similarity (SKB-S) and number of edges (SKB-E) as sorting criterion, as well as of the baseline approach connected components (CC) in terms of precision, recall and f-measure for different number of sources. In the first experiment, we use sources of

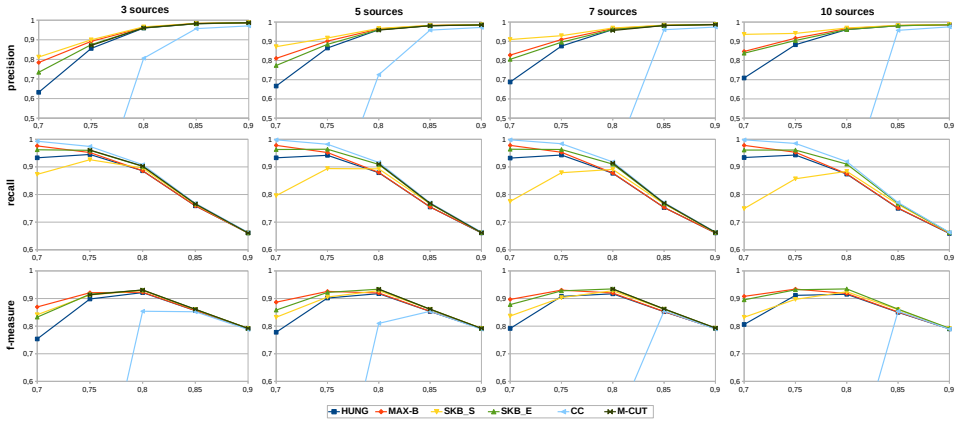

Fig. 9: Quality of the different clustering methods run on 3,5,7 and 10 sources of size 100, 000 from the S-NCVR collection and varying the similarity threshold between 0.7 and 0.9

size 100, 000 and vary the numbers of sources to be linked between 3 and 10. For each clustering algorithm the similarity threshold is varied between 0.7 and 0.9. Fig. 9 shows the results of the experiment using datasets from the S-NCVR collection. As we can see, clusters generated by building connected components from the similarity pairs are generally not usable, especially for low threshold. The precision of such clusters by threshold ≤ 0.75 is about 0, while the recall is not considerably higher than the other clustering methods.

CC is clearly outperformed by the proposed early and late clustering schemes. The best f-measure values are generally achieved for similarity values between 0.75 and 0.8. The Hungarian algorithm generally achieves the lowest f-measure due to its low precision compared to the other approaches. Max-Both by contrast achieves a much better precision and is among the best performing approaches, especially for lower similarity thresholds. The f-measure results for the late clustering approaches are relatively close together. However, graph multicut cannot be applied for low thresholds (≤ 0.7) or high number sources (10 sources) due to its high runtime and memory consumption. The late clustering approach SKB-S, that elects clusters with the highest similarity, achieves the best precision especially for lower similarity. Interestingly, the f-measure does not decrease when the number of sources is increased. This is surprising since it is generally more difficult to correctly find bigger clusters than smaller clusters. This has been possible despite the existence of many large clusters for this synthetic dataset where 25% of all duplicate records belong to the largest clusters with records from all sources (Table 1).

To study the impact of a different duplicate distributions with only few large clusters we run a similar experiment on the second collection of datasets, R-NCVR, for 3 and 10 sources of size 100,000. As we can see in Fig 10 the CC method has again the poorest cluster quality due to a very low precision for lower similarity thresholds ≤ 0.75. Now the late clustering

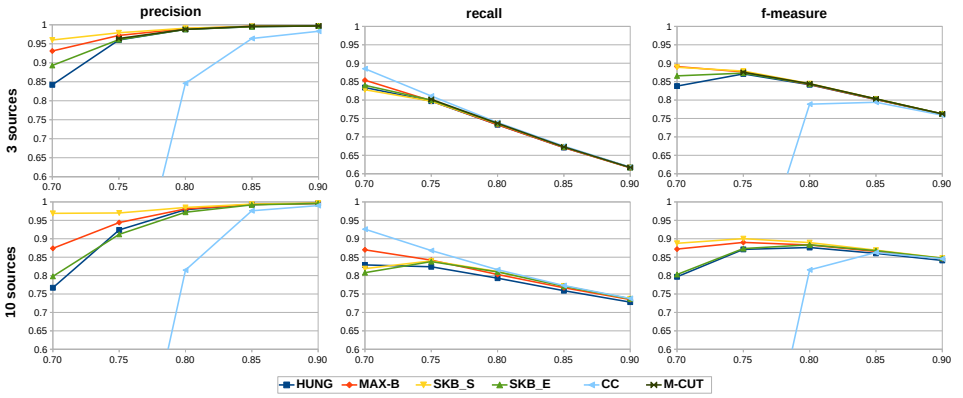

Fig. 10: Quality of the different clustering methods run on 3 sources and 10 sources of size 100, 000 from R-NCVR

method SKB-S performs best since it achieves not only the best precision but can achieve a similarly good recall than the other late clustering methods. Max-Both finds a good balance between precision and recall and performs almost as good as SKB-S. Again, the f-measure for the larger number of sources (10 sources) is similar to the one for only 3 sources.

6.3.2 Size of sources

Figure 11 shows the precision, recall and f-measure results and the total runtime (linkage and clustering) for 5 sources and 7 sources applying a similarity threshold of 0.75. The size of each source is set to 100, 000 and 500, 000 for both number of sources. As we can see, scaling the size of the sources leads to a drop of the quality for all clustering methods. Precision of the Hungarian algorithm drops the most by about 10% from 0.95 to 0.85 followed by Max-Both by about 5%. Only SKB-S shows a small decrease by 1% in precision when scaling up the size of sources from 100, 000 to 500, 000 records. Furthermore, SKB-S is the only method that still manifests a precision above 0.9 for the larger datasets. This is due to its selecting clusters with the highest intra-cluster similarity as clean clusters. However, this also reduces recall since such a high similarity is typically reached by smaller clusters that are thus preferred over larger clusters with lower internal similarity. Hence, SKB-S achieves the lowest recall (0.79) for 5 sources of size 500, 000 records. Note that SKB-E, which promotes large clusters over smaller ones, achieves a similar recall as SKB-S. This is because R-NCVR contains more small clusters of size 2 and 3 than larger ones. Among all algorithms Max-Both returns the best recall for all sizes and number of source. The f-measure of SKB-S and Max-Both are similar with a light advantage for SKB-S.

The right of Fig. 11 shows the runtime to run both linkage and post-processing steps. All four methods achieve similar runtime of about 25 and 50 minutes to process 5 sources and

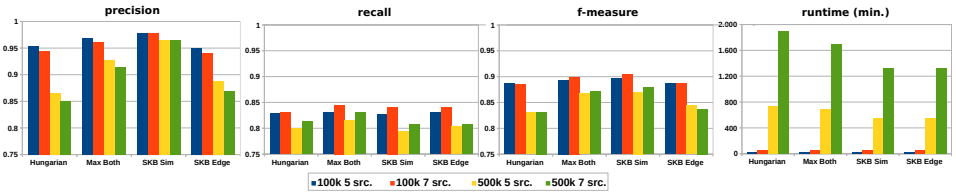

Fig. 11: Quality of the clustering methods and total runtime of linkage process for similarity threshold 0.75 compared with relation to the size and number of sources from `R-NCVR`

7 sources with 100, 000 records each. However, scaling the size of the sources increases the total runtime by a factor of 10 for all methods. This is due the use of metric space for the linkage. As explained in section 3.3 metric space finds all pairs of records that have a similarity above a predefined threshold. We observe, however, that both SKB methods achieve generally the lowest runtime.

7 Conclusions

We studied the use of metric space for multi-party privacy preserving records linkage (MP-PPRL) to efficiently link and cluster records encoded as Bloom filters. We proposed a dynamic pivot-based metric space approach to reduce the number of comparisons that can adapt the number and choice of pivots for a growing number of data sources and thus increasing data volume. The approach is driven by a parameter to control and limit the overlap between the pivots in the metric space. The evaluation showed that this method is very efficient to link multiple sources. Furthermore we presented five early and late clustering methods that create clusters containing at most one element from each source. Early clustering approaches build clusters during the linkage process and late clustering postpone the determination of clusters after all sources have been linked. Our evaluation shows the high scalability and good quality of Max-Both as an early clustering method and SKB-S as a late clustering method.

Despite the effectiveness of the dynamic pivot-based metric space the runtime of the new approaches still increase more than linear with data size. We will thus analyze further runtime improvements such as the adoption of parallel processing on frameworks such as Apache Spark and the combined use of metric space and blocking.

Bibliography

[Bl70] Bloom, Burton H.: Space/Time Trade-offs in Hash Coding with Allowable Errors. Commun. ACM, 13(7):422–426, July 1970.

[BRF15] Boyd, James H.; Randall, Sean M.; Ferrante, Anna M.: Application of Privacy-Preserving Techniques in Operational Record Linkage Centres. In: Medical Data Privacy Handbook. Springer, pp. 267–287, 2015.

[Ch12] Christen, P.: Data Matching - Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection. Springer, 2012.

[DR02] Do, H.; Rahm, E.: COMA: A System for Flexible Combination of Schema Matching Approaches. In: Proc. VLDB conf. pp. 610–621, 2002.

[Du12] Durham, E.A.: A framework for accurate, efficient private record linkage. PhD thesis, Faculty of the Graduate School of Vanderbilt University, Nashville, TN, 2012.

[Fr18] Franke, M.; Sehili, Z.; Gladbach, M.; Rahm, E.: Post-processing Methods for High Quality Privacy-Preserving Record Linkage. In: Data Privacy Management, Cryptocurrencies and Blockchain Technology. 2018.

[FSR18] Franke., M.; Sehili., Z.; Rahm., E.: Parallel Privacy-preserving Record Linkage using LSH-based Blocking. In: Proc. 3rd Int. Conf. on Internet of Things, Big Data and Security (IoTBDS). INSTICC, SciTePress, pp. 195–203, 2018.

[Gi16] Gibberd, A.; Supramaniam, R.; Dillon, A.; Armstrong, B.; O'Connell, D.: Lung cancer treatment and mortality for Aboriginal people in New South Wales, Australia: Results from a population-based record linkage study and medical record audit. BMC Cancer, 16, 12 2016.

[Gl18] Gladbach, M.; Sehili, Z.; Kudrass, T.; Christen, P.; Rahm, E.: Distributed Privacy-Preserving Record Linkage Using Pivot-Based Filter Techniques. In: Proc. ICDE workshops (ICDEW). pp. 33–38, April 2018.

[GVY93] Garg, N.; Vazirani, V. V.; Yannakakis, M.: Approximate Max-Flow Min-(multi)cut Theorems and Their Applications. SIAM Journal on Computing, 25:698–707, 1993.

[Ku55] Kuhn, H.W.: The Hungarian Method for the Assignment Problem. Naval Res. Logist. Quart., 2:83–98, 01 1955.

[Ku11] Kuehni, Claudia E; Rueegg, Corina S; Michel, Gisela; Rebholz, Cornelia E; Strippoli, Marie-Pierre F; Niggli, Felix K; Egger, Matthias; von der Weid, Nicolas X; for the Swiss Paediatric Oncology Group (SPOG): Cohort Profile: The Swiss Childhood Cancer Survivor Study. Int. Journal of Epidemiology, 41(6):1553–1564, 10 2011.

[MGR02] Melnik, S.; Garcia-Molina, H.; Rahm, E.: Similarity flooding: a versatile graph matching algorithm and its application to schema matching. In: Proc.18th Int. Conf. on Data Engineering. pp. 117–128, Feb 2002.

[PB07] Pedreira, Oscar; Brisaboa, Nieves R.: Spatial Selection of Sparse Pivots for Similarity Search in Metric Spaces. In: SOFSEM 2007: Theory and Practice of Computer Science. Springer Berlin Heidelberg, pp. 434–445, 2007.

[Sa18] Saeedi, A.; Nentwig, M.; Peukert, E.; Rahm, E.: Scalable Matching and Clustering of Entities with FAMER. CSIM Quarterly, 16:61–83, 2018.

[SBR09] Schnell, R.; Bachteler, T.; Reiher, J.: Privacy-preserving record linkage using Bloom filters. BMC Medical Informatics and Decision Making, 9(1):41, Aug 2009.

[SBR11] Schnell, R.; Bachteler, T.; Reiher, J.: A Novel Error-Tolerant Anonymous Linking Code. Technical Report WP-GRLC-2011-02, German Record Linkage Center, Duisburg, 2011.

[Se15] Sehili, Z.; Kolb, L.; Borgs, C.; Schnell, R.; Rahm, E.: Privacy Preserving Record Linkage with PPJoin. In: Proc. BTW. pp. 85–104, 2015.

[SR16] Sehili, Z.; Rahm, E.: Speeding up Privacy Preserving Record Linkage for Metric Space Similarity Measures. Datenbank-Spektrum, 16(3):227–236, Nov 2016.

[Tr00] Traina, C.; Traina, A.; Seeger, B.; Faloutsos, C.: Slim-Trees: High Performance Metric Trees Minimizing Overlap between Nodes. In: Advances in Database Technology — EDBT 2000. Springer, pp. 51–65, 2000.

[Va17] Vatsalan, D.; Sehili, Z.; Christen, P.; Rahm, E.: Privacy-preserving record linkage for big data: Current approaches and research challenges. In: Handbook of Big Data Technologies, pp. 851–895. Springer, 2017.

[VCR20] Vatsalan, D.; Christen, P.; Rahm, E.: Incremental clustering techniques for multi-party Privacy-Preserving Record Linkage. Data & Knowledge Engineering, 2020.

[VCV13] Vatsalan, D.; Christen, P.; Verykios, V. S.: A taxonomy of privacy-preserving record linkage techniques. Information Systems, 38(6):946–969, 2013.

[Xi08] Xiao, C.; Wang, W.; Lin, X.; Yu, J. X.: Efficient Similarity Joins for Near Duplicate Detection. In: Proc. 17th Int. Conf. on World Wide Web. pp. 131–140, 2008.

[Ze06] Zezula, P.; Amato, G.; Dohnal, V.; Batko, M.: Similarity search: the metric space approach. Springer, 2006.