Datenbank Spektrum (2019) 19:31–40
https://doi.org/10.1007/s13222-019-00305-y

SCHWERPUNKTBEITRAG

ScaDS Research on Scalable Privacy-preserving Record Linkage

Martin Franke[1] · Marcel Gladbach[1] · Ziad Sehili[1] · Florens Rohde[1] · Erhard Rahm[1]

Received: 8 October 2018 / Accepted: 23 January 2019 / Published online: 6 February 2019

Abstract
Privacy-preserving record linkage (PPRL) supports the matching and integration of person-related data, e.g., on patients or customers without compromising privacy. It is based on the encoding of sensitive attribute values needed for matching and often involves trusted parties for linkage. We report on recent research results from the Big Data center ScaDS Dresden/Leipzig to improve the efficiency, scalability and quality of PPRL, and to apply PPRL in the medical domain. In particular, we present the use of pivot-based filtering techniques and LSH (locality-sensitive hashing)-based blocking to reduce the number of comparisons. Furthermore, we report on parallel linkage implementations based on Apache Flink supporting scalability to millions of records.

Keywords Record Linkage · Privacy · Data Integration · Blocking · Metric Space · LSH · Apache Flink

1 Introduction

Data integration is one of the main areas of research at the Big Data center ScaDS (Competence Center for **Sca**lable **D**ata **S**ervices and Solutions) Dresden/Leipzig. A crucial task in each data integration workflow is record linkage that aims at linking records referring to the same real-world entity, such as patients [5]. Typically, there is a lack of global and unique identifiers, therefore the linkage can only be achieved by comparing attributes that help identify entities, such as name and birth date for entities representing persons. Record linkage is complicated by the increasing need to protect person-related data (or other sensitive data) in order to ensure the privacy of persons and to avoid revealing critical information. For example, in medical research data from several sources, e.g., hospitals and other health care providers, has to be linked to investigate possible correlations between certain diseases without revealing the identity of patients.

Privacy-Preserving Record Linkage addresses this problem by providing techniques to link records while preserving the privacy of represented entities allowing the combination of data from different sources for improved data analysis and research. In particular, PPRL aims at solving the following challenges:

- A high degree of *privacy* has to be ensured by suitable encoding of sensitive data, i.e., attribute values, such that the identity of persons can not be identified but linkage is still possible. Furthermore, all parties that are involved in the linkage process should have no access to analysis-relevant data, e.g., health-related information.
- Despite the use of encoded attribute values, PPRL must achieve a high linkage *quality* by avoiding false or missing matches. A high precision (avoidance of false matches) is especially important, e.g., to avoid that information about different patients is combined.
- Finally, a high *efficiency* with *scalability* to large data volumes and also potentially many data sources are needed.

At ScaDS, we concentrated on improving the performance and scalability of PPRL in the last years while using well-known approaches for ensuring privacy and high linkage quality, in particular the encoding of records as bit vectors (Bloom filters) [2]. We investigated blocking and filtering approaches to reduce the number of comparisons

Martin Franke
franke@informatik.uni-leipzig.de

Marcel Gladbach
gladbach@informatik.uni-leipzig.de

✉ Ziad Sehili
sehili@informatik.uni-leipzig.de

Florens Rohde
florens.rohde@uni-leipzig.de

Erhard Rahm
rahm@informatik.uni-leipzig.de

1 Database Group, Institute of Computer Science, University of Leipzig, Leipzig, Germany

for Bloom filters, namely locality-sensitive hashing (LSH) and metric space filtering. Furthermore, we developed parallel PPRL schemes based on the distributed processing framework Apache Flink [4] to gain further performance speedup. Since obtaining a high linkage quality becomes more difficult for larger datasets, we also investigated an improved selection of match candidates. Finally, we started to investigate PPRL use case scenarios in medical applications to identify current weak points and limitations.

In this overview paper we summarize the results of our PPRL research. In the next section, we provide background information on PPRL and assumed configurations. In Sect. 3 we present blocking and filtering methods that are essential to make PPRL approaches scalable to real world datasets and use cases. At first, we investigate the use of locality-sensitive hashing (LSH) for blocking on encoded person-related records and report evaluation results. We also analyze a distributed version using Apache Flink that provides enhanced scalability up to several millions of records. Then, we present filtering approaches for PPRL, especially for metric space distance functions where the triangle inequality property can be used to filter out dissimilar pairs of records. A pivot-based technique that partitions the metric space over a set of objects (pivots) showed a significant improvement of performance compared to previous filtering techniques. We also discuss parallel versions of the pivot approach for improved scalability. In Sect. 4 we introduce a post-processing step into the PPRL workflow in order to increase linkage quality, in particular when dealing with large datasets containing many record pairs with high similarities. Finally, we report on our initial efforts to apply PPRL in practice (Sect. 5) before we conclude and discuss future work.

2 Background

Record linkage (RL) deals with the problem of finding records that represent the same real-world entity. It has been addressed by numerous approaches and several surveys [10, 18, 5]. RL mainly focuses on achieving a high linkage quality and scalability to large datasets. Linkage quality is largely affected by the quality of the input data which often contain typographical errors and heterogeneity in structure [16, 5]. To weaken such problems a preprocessing step to clean and standardize the input data is usually conducted [15, 24]. The second problem, scalability to large datasets, is addressed by several blocking and filtering techniques [6] that alleviate the inherent quadratic complexity when every record has to be compared with every other record. PPRL adds a third challenge that is protecting the privacy of represented entities. For this purpose secure protocols and encoding (encryption) techniques are used to minimize the risk of disclosing sensitive information [30, 31]. These techniques should not only impede the re-identification of persons but also have to preserve the (dis)similarity between records such that the similarity between encoded records corresponds to the similarity of their clear-text counterparts.

Like most previous approaches in PPRL we focus on the use of Bloom filters for encoding records attributes. Furthermore, we deploy PPRL under a three-party protocol where encoded records are sent to a trusted linkage unit performing the actual linkage. The details about these frequently used approaches are outlined in the following subsections.

2.1 Adversary Model

PPRL protocols generally assume a *Honest-but-Curious* cooperation scheme between the involved parties. Honest-but-Curious implies, that each party follows the prescribed steps of the protocol (honest), in the sense that the data holders pre-process and encode the same attributes using the same parameters e.g., the lengths of q-grams and Bloom filter and the used hash functions. In addition, involved parties are not allowed to send fake records, because such a behaviour allows the re-identification of others data without participating in the linkage process. Nevertheless, the parties may try to know as much as possible about the others data (curious), e.g. the LU may analyze the Bloom filters to decode them, but they neither deviate from the protocol nor do they collude with each other to gain information.

2.2 Bloom Filter encoding

The use of Bloom filters [2] for PPRL has been proposed by Schnell and colleagues [25] and has become the most popular encoding scheme for PPRL in research as well as in real applications [19, 20, 30, 31]. A Bloom filter (BF) is a bit vector of fixed size m where initially all bit positions are set to zero. k cryptographic hash functions are used to hash a set of record features into the bit vector. Usually, identifying attributes, i.e., attributes that may allow the re-identification of a person, like first name, last name and date of birth, are tokenized into substrings of length q (q-grams) to build the set of record features. Then, *each* hash function is applied on *each* q-gram of the set and returns a position $p \in [0, m-1]$. Fig. 1 illustrates the approach for two records using bi-grams (q-grams of length $q = 2$) on the first name attribute. Given that identical q-grams are mapped to the same bit positions, a high overlap of q-grams leads to similar Bloom filters making them suitable for determining the record similarity (see below).

There are different possibilities to create Bloom filters, in particular the use of a separate Bloom filter for each

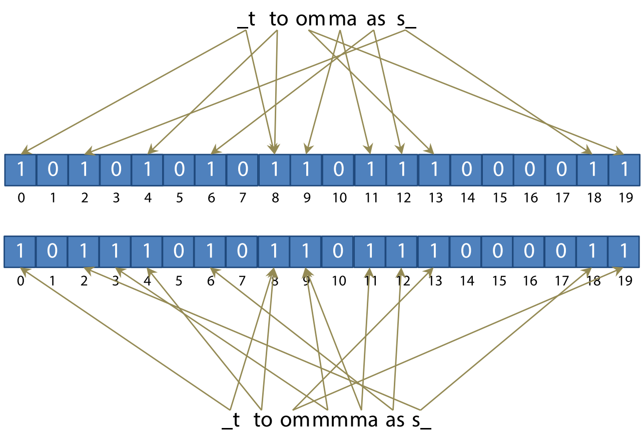

Fig. 1 Bit vector (Bloom filter) encoding of two names *tomas* and *tommas*, each tokenized to bi-grams, using $k = 2$ hash functions and bit vectors of length $l = 20$

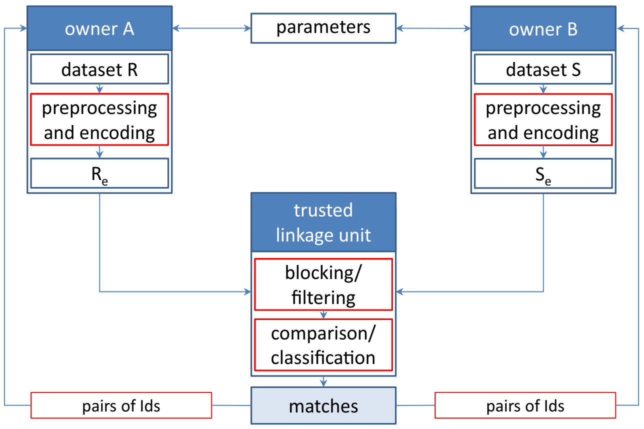

Fig. 2 Three-party protocol for PPRL.

identifying attribute [25] or a combined Bloom filter for all identifying attributes. We will focus on the latter approach, called *Cryptographic Longterm Key* (CLK), since it has been shown to be more secure than the use of attribute-level Bloom filters [26].

However, in general Bloom filters carry the risk of frequency analysis, since the frequency of clear text q-grams will be reflected in the Bloom filters. For example the bigrams, *sc*, *ch*, *hm*, *mi*, *id* and *dt*, of the most frequent German name *Schmidt* are expected to set the same bit positions in several Bloom filters, allowing their re-identification. The same strategy is applied on another attribute like first name and the combination of these two disclosed attributes may lead to the re-identification of persons. The risk of such attacks can be reduced by using hardening techniques [27, 3] that typically introduce errors in Bloom filters but also lead to a reduced match quality.

2.3 PPRL Process

Fig. 2 illustrates the three-party protocol which is commonly deployed [30]. We basically consider two data holders A and B owning datasets R and S respectively, and a trusted third party, a so-called linkage unit (LU), that carry out linkage step. Three-party protocols inherently support scalability, since the LU can apply efficient blocking and filter techniques and utilize large-scale computer resources, as a shared-nothing HDFS-cluster with tens of worker nodes. Furthermore, this approach can be extended to support matching for more than two data owners [31].

The overall PPRL process consists of several steps run at the data holders and the LU that we briefly describe in the following.

Pre-processing: First, the data holders exchange some parameters about the attributes to use in linkage, and how to clean and standardize them. In particular, they exchange the encoding parameters, e.g., Bloom filter length, set of attributes to use in the match process. Each data holder then conducts a data cleaning and standardization step. This step is essential to achieve a high linkage quality, since real-world data is considered to be dirty [15, 24]. After that, data holders encode (mask) their records into Bloom filters as described above. At the end of this step the data holders A and B send their set of encoded records, R_e and S_e respectively, to the linkage unit.

Blocking/Filtering: To avoid that every record of the first source has to be compared with every record of the second source, blocking or filtering techniques are used to exclude obviously dissimilar record pairs from detailed comparisons [6]. Although, these two methods have the same goal, they operate differently and could even be used in combination:

The basic idea of **blocking** is to group records in blocks such that only records within the same block are compared with each other. One or more blocking key values (BKV) are generated for each record, whereby records with the same blocking key value are assigned to the same block [6]. Each corresponding blocking key represents a specific, potentially complex, criterion, that records must satisfy in order to be considered as match candidates. A blocking key is defined by a function which takes as input one or more record attributes. These record attributes may be different from the attributes used in the comparison step (similarity calculation).

If only one blocking key is used, the resulting blocks are disjoint. However, a single blocking criterion might be too restrictive and can thus lead to many false-negatives. As a consequence, often multiple blocking keys are used to increase the probability for records to share at least one blocking key. The drawback of using multiple blocking keys is that it leads to overlapping (non-disjoint) blocks hence to duplicate candidates.

Beside the number of blocking keys, PPRL processes allow the generation of BKV either already at the data owners on unencoded attribute values or at the LU on encoded attributes. For example, one could use the Soundex value of last name [23] as a blocking key and send a corresponding block id to the LU together with the Bloom filter. At the LU, only records with the same BKV need to be compared with each other. A drawback of this approach is that the data owners have to agree on the blocking key beforehand. Furthermore, sending the Soundex code to the LU might disclose some private information. A more promising approach is thus to directly determine the blocks using encoded data at the LU and we investigate the use of LSH [13, 9] for this purpose.

Typically, blocking shows a trade-off between linkage quality and performance depending on the type and number of blocking keys used. In general, using restrictive blocking criteria will lead to many small blocks and thus fewer candidates, but at the same time the chance to miss true matches increases. On the other hand, if less restrictive criteria are selected, the number of blocks decreases while the block sizes and the number of candidates increase. As a consequence, more candidates need to be compared leading to fewer false-negatives but also less efficiency. Generally, the higher the number of blocking keys the higher the number of resulting blocks and consequently the number of (duplicate) candidates.

On the other side **filter techniques** exploit the properties of the similarity functions and the match similarity threshold (see below) to filter out dissimilar record pairs that cannot reach or exceed the threshold [31]. These techniques are conducted by the LU, and other than blocking, they do not affect the recall of the linkage process because they found *all* pairs of records (matches) having a similarity value above the predefined threshold.

Both methods, blocking and filtering, will be investigated in Sect. 3. The output of both methods is a set of candidate record pairs C where $C = \{(r,s) \mid r \in R_e, s \in S_e\}$.

Similarity calculation and match classification: Several similarity functions can be used to compare pairs of records and derive the pairs of matching records. Previous PPRL methods on Bloom filters mostly apply either Jaccard (1) or Dice (2) similarity:

$$sim_{jac}(r,s) = \frac{|r \wedge s|}{|r \vee s|} \tag{1}$$

$$sim_{dice}(r,s) = \frac{2 \times |r \wedge s|}{|r| + |s|} \tag{2}$$

In these formulas the cardinality or *length* of records refers to the number of set bits. The similarity functions are thus easy to compute for bit vectors and return a value between 0 and 1. For the example of Fig. 1, the Jaccard and Dice similarities are $11/12 = 0.92$ and $22/23 = 0.96$, respectively.

To identify matching records, PPRL methods mostly follow a simple threshold-based approach such that two records r and s are considered to represent the same real-world entity if their similarity $sim(r,s)$ meets or exceeds a threshold t, i.e., $sim(r,s) \geq t$ [30, 31].

Finally, the LU returns the pairs of IDs of matching records back to the data holders, that are now able to link their data of interest, e.g., medical data, for matching records.

3 Improving Scalability

At ScaDS we followed two strategies, namely blocking and filtering, to mitigate the quadratic complexity of the matching process. Furthermore, we implemented parallel versions of both strategies to make the linkage process scalable to very large datasets.

3.1 Blocking Methods

In the following we first explain Soundex, a simple blocking method used as a baseline in our evaluation. Then we introduce the more secure and powerful LSH blocking and its parallelization.

3.1.1 Phonetic Blocking

A frequently used blocking approach in the record linkage domain is phonetic blocking, e.g., based on the Soundex function [23, 5]. A phonetic encoding function, as Soundex, is typically applied on name attributes and aims to produce the same output for input values with a similar pronunciation (even with typographical variations or errors). For instance, the Soundex value for both names 'Sara' and 'Sara**h**' is S600. However, since the first letter of the attribute value is preserved in the Soundex code, typographical variations at the beginning of a name, e.g., 'Zarah' (Z600) vs. 'Sarah' (S600), can not be compensated.

3.1.2 LSH-based Blocking

Locality-sensitive hashing (LSH) was proposed for solving the nearest neighborhood problem in high-dimensional data spaces [13]. The basic idea of LSH is to apply a set of hash functions on the objects of interests, i.e., bit vectors. These hash functions are drawn from a family $\mathcal{F}$ of hash functions which are sensitive to a certain distance measure d, e.g., Hamming or Jaccard distance. For each hash function in $\mathcal{F}$ it is ensured that the probability of a collision, i.e., same output value for two different input values, is much higher

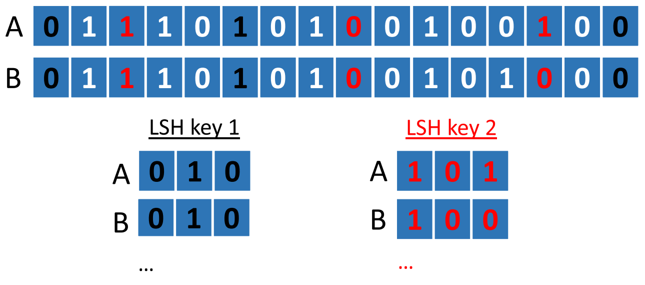

Fig. 3 HLSH blocking using $\Lambda = 2$ blocking keys of size $\Psi = 3$. The first (black) HLSH key uses bit positions $\{0, 5, 15\}$, while the second (red) uses $\{2, 8, 13\}$. Bit vectors A and B agree on the first HLSH key and are assigned to the same block. In contrast, for the second HLSH key the values for A and B differ ('101', '100') and thus A and B are assigned to different blocks.

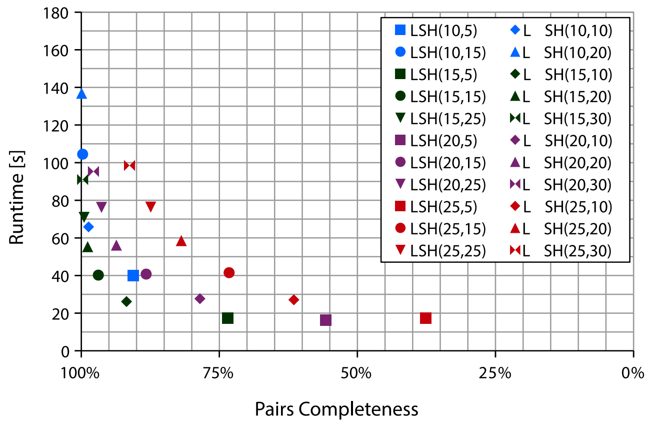

Fig. 4 Evaluation of different HLSH parameter settings, denoted as LSH(Ψ, Λ), on a datasets containing one million moderately corrupted records.

for objects with a small distance (high similarity) than for objects with greater distance (low similarity).

LSH is used as probabilistic blocking approach for PPRL by applying hash functions $f \in \mathcal{F}$ on bit vectors (Bloom filters) [9]. For this purpose the hash family $\mathcal{F}_{\mathcal{H}}$ that is sensitive to the Hamming distance can be used (HLSH). Each function $f_i \in \mathcal{F}_{\mathcal{H}}$ returns the bit value at position i in the bit vector [9]. For instance, applying the function $f_7 \in \mathcal{F}_{\mathcal{H}}$ on the bit vector 11011001 would return the bit value on position 7 and therefore 1. In order to group similar records, a blocking key is constructed by using Ψ such hash functions which are drawn randomly from the $\mathcal{F}$. Then, the output values of these Ψ hash functions are concatenated to obtain the blocking key. Consequently, the parameter Ψ is also called the blocking key length. Due to the probabilistic nature of LSH, it is possible that two bit vectors with distance smaller or equal to d_1 may produce different blocking keys, namely if the bit value(s) at one or several of the Ψ positions are different. Due to dirty data, Λ blocking keys are therefore generated to increase the probability that two similar bit vectors share at least one blocking key value. In Fig. 3, the HLSH-based blocking scheme is illustrated for two bit vectors.

In order to achieve a high linkage quality, efficiency as well as scalability to large datasets the two parameters Ψ and Λ need to be carefully selected. Since Ψ specifies the length of each LSH key, a higher value for Ψ increases the probability that only bit vectors with a high similarity are assigned to the same block. Hence, a higher Ψ will lead to smaller blocks and thus fewer intra-block comparisons while a lower Ψ will instead produce larger blocks but also decreases the probability that two similar bit vectors are missed due to erroneous data. On the other hand, the higher Λ, the higher is the probability that two similar bit vectors share a blocking key. However, at the same time the number of blocks and thus the number of candidates that need to be processed increases which decreases the scalability of LSH-based blocking.

Parallel HLSH-based Blocking: In our work [12], we focused primarily on the empirical evaluation of different LSH parameter settings using datasets with different sizes and corruption levels (medium and high). Another aim of this work was to improve the scalability of PPRL by parallelizing the PPRL process using the LSH-based blocking approach based on the Hamming distance (HLSH). For this purpose, we used Apache Flink as state-of-the-art distributed processing framework to enable the utilization of large shared-nothing clusters to speed up PPRL proportional to the number of CPUs in the cluster.

For datasets containing one million records with a moderate corruption level several LSH parameter configurations, denoted as LSH(Ψ, Λ), achieved high quality as well as high efficiency in terms of pairs completeness [5] and execution time. In fact, all settings where Ψ ranges from 10 to 20 and Λ ranges from 10 to 30 are able to achieve good results for moderately corrupted data (see Fig. 4). The

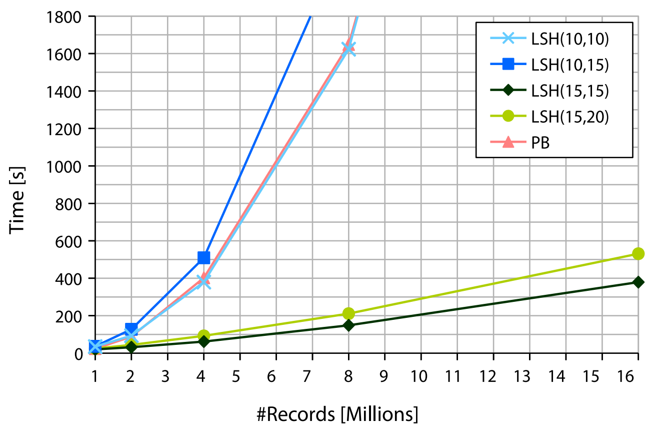

Fig. 5 Runtimes for HLSH and phonetic blocking (PB) on moderately corrupted records using a cluster with 16 workers.

parallel approach also showed almost linear speedup behavior for a cluster with 16 worker nodes and running Apache Flink (Fig. 5). It could also clearly outperform Soundex-based blocking where the block sizes for frequent names increase thereby limiting scalability.

In conclusion, our evaluation showed the high efficiency and effectiveness of the parallel LSH-based PPRL approach, even for large datasets and corrupted data. Moreover, LSH-based blocking clearly outperformed phonetic blocking approaches based on Soundex in terms of both linkage quality and runtimes . It remains an open challenge how to (automatically) find an optimal LSH parameter setting in real-world applications where the data quality and the true match status is unknown and the inspection of actual attribute values is not possible due to privacy constraints.

3.2 Filtering Methods

Filtering strategies utilize properties of the similarity function as well as the chosen similarity threshold to identify pairs that can not satisfy the match criterion. Such approaches have already been applied for regular record linkage (similarity joins) [17] and can be adapted for PPRL. We have studied two approaches that we will discuss in the following. We start with P4Join, an adaptation of the PPJoin algorithm for Bloom filters that applies several filters. We then consider filter techniques for metric space distance functions, in particular pivot-based approaches.

3.2.1 P4Join

PPJoin [33] is a fast implementation for similarity joins utilizing several filters. Its adaptation for Bloom filters, called P4Join, has been proposed in [28]. Similar to PPJoin, P4Join applies several filters to exclude record pairs from the similarity computation, in particular the length, prefix and position filters. The length filter for example excludes all pairs r, s from further consideration if their lengths are very different. More concretely, for Jaccard similarity function and a threshold t it holds:

$$sim_{jac}(r,s) \geq t \Longrightarrow |r| \geq \lceil t \times |s| \rceil, \tag{3}$$

if $|r| \leq |s|$. For $t = 0.9$, this allows excluding all pairs of records that differ more than 10% in the number of set bits from further processing. The evaluation in [28] showed a high effectiveness for the length filter because it allows pruning sets of records instead of one record pair at a time. Still, the runtime improvements of P4join were relatively modest (only a factor of 2 compared to no filtering) since the prefix and position filters did not lead to significant savings. However, P4Join and the match comparisons for bit vectors could be effectively run on GPUs thereby achieving order-of-magnitude runtime improvements [28].

3.2.2 The Metric Space Approach

A metric space $\mathcal{M}(\mathcal{U}, d)$ consists of a set $\mathcal{U}$ of data objects and distance metric d to compute the distances between the objects of $\mathcal{U}$. The distance function d must satisfy several properties, in particular the triangle inequality: $\forall r,s,p \in \mathcal{U} : d(r,s) \leq d(r,p) + d(p,s)$ [34], which can be utilized for filtering pairs of records to find matches. In contrast to Dice similarity, Jaccard similarity is a metric distance function that can thus make use of the triangle inequality. Furthermore, it has been shown that using the Jaccard similarity with threshold t can be translated into an equivalent Hamming distance d_h [33] which is defined as:

$$d_h(r,s) = |r \vee s| - |r \wedge s| = r \; XOR \; s. \tag{4}$$

The Hamming distance thus corresponds to the number of differently set bit positions, e.g., it has value 1 for the example in Fig. 1.

Instead of using the Jaccard threshold t to determine matching records for a query bit vector q, we use the Hamming distance and determine a corresponding maximum distance or radius $rad(q)$ [29]:

$$rad(q) = |q| \frac{1-t}{t} \tag{5}$$

To determine matching record pairs we use the encoded records from the first source R_e to form the metric space and determine for each query record q from the second source S_e the records $r \in R_e$ that lie within the query radius $rad(q)$:

$$sim_j(q,r) \geq t \Leftrightarrow d_h(q,r) \leq rad(q) \tag{6}$$

Finding these record pairs can be fastened by applying a pivot-based approach as explained next.

Pivot-based filtering:

The pivot-based PPRL approach works in two steps: an indexing step and similarity search. The indexing involves the selection of a set P of m pivots or reference points from the first dataset R_e. Then, each record $r \in R_e$ is assigned to its closest pivot $p_i \in P$ and the distance $d_h(r, p_i)$ is stored. Furthermore, the maximal distance between each pivot p_i and the records assigned to it, $rad(p_i)$, is precomputed as illustrated in Fig. 6.

In the second step similarity search is performed for each record q of the second source S_e to find the matches in R_e. Here, two filters are applied to reduce the number of

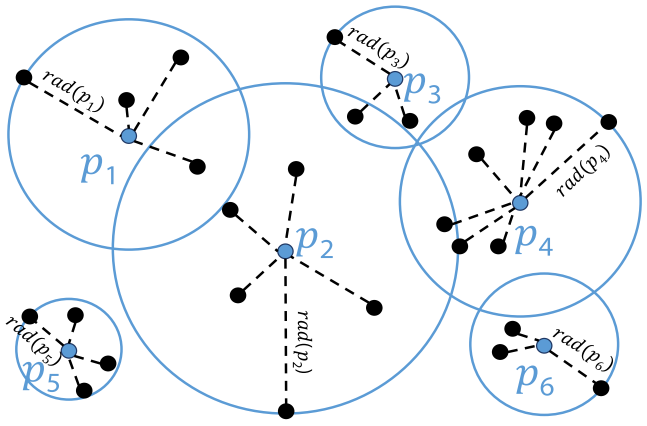

Fig. 6 Objects in metric space (black point) are partitioned over a set of pivots (blue point) with their corresponding radii

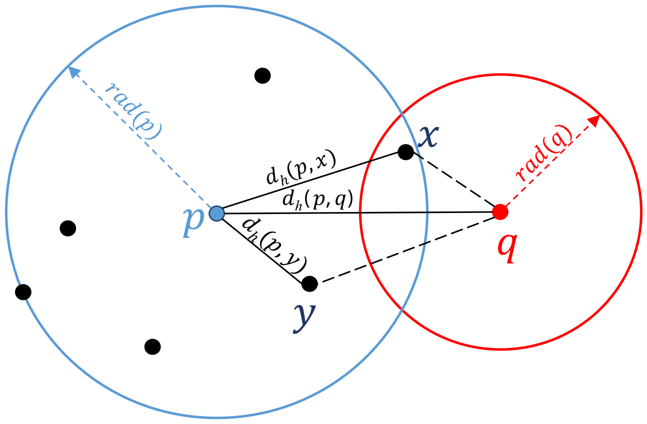

Fig. 7 Utilization of the triangle inequality in metric space. Object y cannot lie within the search radius of query object q since the difference between $d(p,q)$ and $d(p,y)$ exceeds $rad(q)$.

comparisons. First, q is compared with each pivot p_i so that only pivots satisfying:

$$d_h(q, p_i) \leq rad(q) + rad(p_i) \qquad (7)$$

are considered further. This way we can save all comparisons with the records assigned to the excluded pivots. In a second stage, for each of the remaining pivots p_i we use the triangle inequality to exclude all records r from further consideration for which it holds:

$$d_h(p_i, q) - d_h(p_i, r) > rad(q). \qquad (8)$$

Fig. 7 illustrates this filter condition allowing to exclude the consideration of record y but not x. The remaining records, such as x in Fig. 7, are compared with q using the similarity function.

The effectiveness of the pivot usage significantly depends on the number of pivots and how pivots are selected. In general there are only heuristics to find "good" pivots. In particular, the pivot radii should have a minimal overlap in order to limit the number of pivots to consider for matching. This condition is generally satisfied by pivots on the "edge" of the metric space [22]. To find pivots we apply two iterative state-of-the-art algorithms with runtime complexity $\mathcal{O}(n \cdot m)$ where $n = |R_e|$ and $m = |P|$:

- *Maximum Separation (ms)* aims at maximizing the sum of distances between pivots. After picking the first pivot randomly the following ones are chosen by selecting the element having the largest sum of distances to all previously selected pivots [22].
- *Farthest-First-Traversal (fft)* tries to select "corners" in the metric space as pivots. In each iteration, for every element the smallest distance to all previously selected pivots is calculated. The element for which this distance is the largest, is selected as the next pivot [22].

In [29] we analyzed the performance of the pivot-based approach using the maximum separation technique (ms) to choose the pivots. The evaluation showed that this method needs substantially lower runtime and has a higher reduction ration compared to alternate filter methods like P4Join [28] or Multibit-tree [1].

Distributed pivot-based filtering: To further improve the linkage time and scalability to large datasets we developed a distributed algorithm [14] using the pivot-based metric space approach. The method works in two main phases both executed in parallel on a Shared Nothing cluster with partitioned datasets: A preprocessing to determine the pivots and a matching. The preprocessing phase consists of several steps: First, local pivots on each partition of the first dataset are determined. From the union of the local pivots a global selection of the final pivots is performed. For both, the local and the global steps to determine pivots, different strategies can be applied, in particular the mentioned Farthest-First-Traversal (fft) and Maximum Separation (ms) algorithms. The assignment of each record r to the nearest pivot p is also conducted in the preprocessing phase. In the matching phase for each query record q from the second source a set of relevant pivots are determined using Eq. (7). Then, the final comparison is performed for each candidate pairs satisfying Eq. (8).

In [14] we evaluated the parallel PPRL approach with pivot-based filtering for different synthetic and real-world data sets using the Apache Flink framework [4]. The synthetic data are generated using the GeCo tool [7] with look-up files for postcodes, cities and names from Germany and from the region around Leipzig. Each dataset is split into two subsets, the first one containing 80% of the original

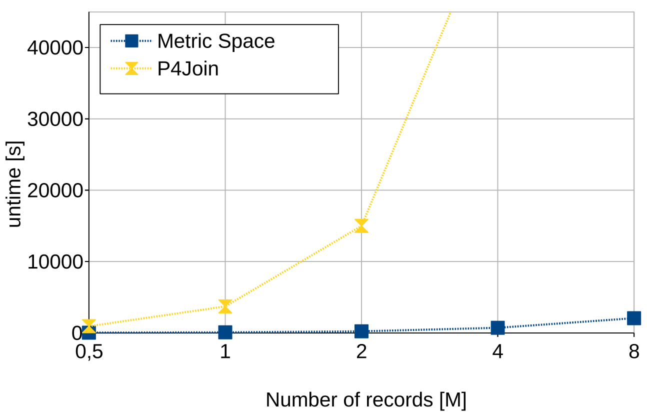

Fig. 8 Performance comparison for synthetic datasets with 0.5 to 8 million records using a cluster of 16 workers. The distributed metric space approach (blue line) clearly outperforms the distributed P4Join implementation (yellow line).

dataset is indexed and the second subset (20%) is used as a set of queries. The experiments showed that the Farthest-First-Traversal (fft) algorithm to find global pivots in combination with a fft or random strategy to find local pivots achieves good filter effects and outperforms other strategies. Another important parameter is the number of pivots which depends on the dataset sizes and the chosen pivot selection. Fig. 8 shows the scalability of the metric space compared P4Join (both parallel) for the synthetic data from the region around Leipzig. Overall, the parallel approach of the metric space showed a good scalability to larger data sizes as well as excellent speedup.

4 Improving Linkage Quality

Obtaining high linkage quality is one of the key challenges of PPRL. Ideally, a PPRL approach should find all matches, despite possible data quality problems in the source databases. On the other hand, false matches should be strictly avoided, as otherwise (medical) conclusions based on incorrect assumptions could be made. Besides data quality there are many factors that significantly influence the final linkage quality, e.g., the encoding method, the blocking or filtering approach and the respective parametrization, as well as the selected similarity threshold that is used in the classification step in order to decide whether a record pair represents a match or a non-match. In our experiments we found that for some datasets only moderate linkage quality can be obtained, even if the records are only slightly corrupted and all parameters are carefully (empirically) selected. The reason behind this phenomenon are datasets where many non-matching record pairs have a high similarity score. For example, datasets containing many families or households tend to have many non-matches with a high similarity, since many persons share their last name and address. The situation becomes even worse if the dataset also is large or of poor quality.

As described in Sect. 2 most PPRL approaches only apply a simple threshold-based classification using a single threshold value. Unlike as in traditional record linkage approaches, in a privacy-preserving context, no supervised machine learning approaches can be utilized, as training data is usually not available. Furthermore, the linkage result cannot be inspected manually since this would give part of the privacy and is also not feasible for millions of records. As a consequence, the simple threshold-based match criterion often leads to situations where a record has more than one record in the other source for which the similarity threshold is reached. For duplicate-free source databases, this would be incorrect and should thus be avoided as otherwise one person would be linked to more than one person of the other source.

To weaken this effect, we proposed in [11] to apply a *post-processing step* after the classification in order to determine the most likely matches so that the final matches represents a 1:1-mapping between the two datasets, where one record is linked to at maximum one record of the other source. Therefore, we analyzed different post-processing strategies, namely computing a stable matching or a maximum weight matching that can be obtained by applying the Gale-Shapley, or the Hungarian algorithm respectively. Moreover, we considered a heuristic symmetric best match approach, that is also know as Max-Both. Max-Both accepts a candidate pair (r, s) as a match, only if s has the maximal similarity with r among all records in the second source and r has the maximal similarity with s in the first source. It turns out that this heuristic approach is very efficient and effective (high precision) and outperforms the two other approaches.

5 Applying PPRL in medical applications

PPRL finds increasing adoption, especially in health care and clinical research and applications [21, 8]. At ScaDS, we have also started to investigate PPRL for medical applications within two projects that we discuss briefly.

5.1 SMITH

The collaborative project SMITH (**S**mart **M**edical **I**nformation **T**echnology for **H**ealthcare) is part of the Medical Informatics Initiative Germany. It was initiated by the university hospitals in Leipzig, Jena and Aachen and is complemented by the university hospitals Halle, Hamburg, Essen, Bonn, Düsseldorf, and Rostock as well as several industrial partners and ScaDS. The four year development and networking phase started in 2018. The main goal of the project is to advance and harmonize the IT infrastructure in the participating sites and enable data exchange between them for healthcare and research using interoperability standards [32]. Therefore data integration center (DICs) will be established at each participating university hospital.

PPRL methods will be part of the ID management for patients as identifying patient-related attributes are not allowed to leave their respective context (e.g. clinical or research). It is planned to employ multiple PPRL Coding Services as well as a central PPRL Matching Service for each university hospital. Some patients will likely be registered in multiple hospitals and therefore PPRL is also needed on this level e.g. to avoid duplicates in research studies. New features for PPRL have to be implemented in order to satisfy the specific requirements of this project. This includes continuous matching (matching new patients without the linkage of the complete sources again) and multiparty matching (building clusters of matches of more than two sources).

5.2 Mainzelliste

Mainzelliste is an open-source software for identity management for multi-site medical projects and applications [20]. Its core functionalities, pseudonymization and de-pseudonymization of patient data, are accessible via a RESTful interface. The pseudonymization process includes PPRL based on attribute-level Bloom filters. Even though the software has been widely used in real use cases for many years, there was no systematic evaluation of the quality and runtime of its linkage process. We evaluated quality and runtime of the linkage process within the Mainzelliste by using several synthetic datasets of different size and error-rates. We observed that the Mainzelliste achieves a high linkage quality but very poor runtimes that limits the applicability to smaller datasets. This is because there is no blocking or filtering used in the current implementation. We added already a Soundex-like blocking that could improve runtimes significantly. Further improvements such as support of LSH blocking are planned.

6 Conclusions and Outlook

We developed several distributed PPRL approaches with filtering and blocking techniques that showed high efficiency and effectiveness for large synthetic and real-world datasets with up to 16 million records. For filtering, the proposed pivot-based approach for metric space distance functions outperformed other methods while for blocking the use of LSH is most promising. The parallel version of these approaches showed a good speedup and supported scalability to millions of records.

In future work, we plan to simplify the use of PPRL by investigating approaches to automatically find suitable parameter settings. Furthermore, we are currently developing a PPRL toolbox with state-of-the-art encoding and linkage techniques. The toolbox is designed to simplify the comparison of PPRL methods and their practical usability as well as their configuration. Furthermore, we plan to investigate multi-party PPRL approaches with more than two sources where one has to find subset matches, i.e., sets of matching records that are only in a subset of the sources. Finally, we continue the integration of PPRL into practical applications and use cases, especially for the Medical Informatics Initiative within the SMITH consortium.

Acknowledgements This work was partially funded by the German Federal Ministry of Education and Research within the project Competence Center for Scalable Data Services and Solutions (ScaDS) Dresden/Leipzig (BMBF 01IS14014B).

References

1. Bachteler T, Reiher J, Schnell R (2013) Similarity filtering with multibit trees for record linkage. GRLC, Working Paper WP-GRLC-2013-02
2. Bloom B (1970) Space/time trade-offs in hash coding with allowable errors. CACM 13(7):422–426. https://doi.org/10.1145/362686.362692
3. Brown AP, Borgs C, Randall SM, Schnell R (2017) Evaluating privacy-preserving record linkage using cryptographic long-term keys and multibit trees on large medical datasets. BMC Med Inform Decis Mak 17(1):83. https://doi.org/10.1186/s12911-017-0478-5
4. Carbone P et al (2015) Apache Flink: Stream and batch processing in a single engine. IEEE TCDE 36(4):28–38
5. Christen P (2012) Data matching: concepts and techniques for record linkage, entity resolution, and duplicate detection. Springer, Berlin, Heidelberg https://doi.org/10.1007/978-3-642-31164-2
6. Christen P (2012) A survey of indexing techniques for scalable record linkage and deduplication. IEEE Trans Knowl Data Eng 24(9):1537–1555. https://doi.org/10.1109/TKDE.2011.127
7. Christen P, Vatsalan D (2013) Flexible and extensible generation and corruption of personal data. In: ACM CIKM, pp 1165–1168 https://doi.org/10.1145/2505515.2507815
8. Clark DE (2004) Practical introduction to record linkage for injury research. Inj Prev 10(3):186–191. https://doi.org/10.1136/ip.2003.004580
9. Durham EA (2012) A framework for accurate, efficient private record linkage. Faculty of the Graduate School of Vanderbilt University, Nashville, TN, (Ph.D. thesis)
10. Elmagarmid AK, Ipeirotis PG, Verykios VS (2007) Duplicate record detection: a survey. IEEE Trans Knowl Data Eng 19(1):1–16. https://doi.org/10.1109/TKDE.2007.250581
11. Franke M, Sehili Z, Gladbach M, Rahm E (2018) Post-processing methods for high quality privacy-preserving record linkage. In: Data privacy management, Cryptocurrencies and Blockchain technology. Springer, Berlin, Heidelberg, pp 263–278 https://doi.org/10.1007/978-3-030-00305-0_19
12. Franke M, Sehili Z, Rahm E (2018) Parallel privacy preserving record linkage using LSH-based blocking. In: IoTBDS, pp 195–203 https://doi.org/10.5220/0006682701950203

13. Gionis A, Indyk P, Motwani R et al (1999) Similarity search in high dimensions via hashing. In: Proceedings of the 25th VLDB Conference, vol 99, pp 518–529
14. Gladbach M, Sehili Z, Kudraß T, Christen P, Rahm E (2018) Distributed privacy-preserving record linkage using pivot-based filter techniques. In: ICDE-W, pp 33–38 https://doi.org/10.1109/ICDEW.2018.00013
15. Hernández MA, Stolfo SJ (1998) Real-world data is dirty: data cleansing and the merge/purge problem. Data Min Knowl Discov 2(1):9–37. https://doi.org/10.1023/A:1009761603038
16. Herzog TN, Scheuren FJ, Winkler WE (2007) Data quality and record linkage techniques, 1st edn. Springer, Berlin, Heidelberg https://doi.org/10.1007/0-387-69505-2
17. Jiang Y, Li G, Feng J, Li WS (2014) String similarity joins: an experimental evaluation. Proc VLDB Endow 7(8):625–636. https://doi.org/10.14778/2732296.2732299
18. Köpcke H, Rahm E (2010) Frameworks for entity matching: a comparison. DKE 69(2):197–210. https://doi.org/10.1016/j.datak.2009.10.003
19. Kuehni CE, Rueegg CS, Michel G, Rebholz CE, Strippoli MPF, Niggli FK, Egger M, von der Weid NX (2012) Cohort profile: the Swiss childhood cancer survivor study. Int J Epidemiol 41(6):1553–1564. https://doi.org/10.1093/ije/dyr142
20. Lablans M, Borg A, Ückert F (2015) A RESTful interface to pseudonymization services in modern web applications. BMC Med Inform Decis Mak. https://doi.org/10.1186/s12911-014-0123-5
21. Malin BA, Emam KE, O'Keefe CM (2013) Biomedical data privacy: problems, perspectives, and recent advances. J Am Med Inform Assoc 20(1):2–6. https://doi.org/10.1136/amiajnl-2012-001509
22. Mao R, Zhang P, Li X, Liu X, Lu M (2016) Pivot selection for metric-space indexing. Int J Mach Learn Cybern. https://doi.org/10.1007/s13042-016-0504-4
23. Odell M, Russell R (1918) The soundex coding system. US Patents 1261167
24. Rahm E, Do HH (2000) Data cleaning: problems and current approaches. IEEE Data Eng Bull 23(4):3–13
25. Schnell R, Bachteler T, Reiher J (2009) Privacy-preserving record linkage using Bloom filters. BMC Med Inform Decis Mak 9(1):41. https://doi.org/10.1186/1472-6947-9-41
26. Schnell R, Bachteler T, Reiher J (2011) A novel error-tolerant anonymous linking code. GRLC, No. WP-GRLC-2011-02
27. Schnell R, Borgs C (2016) Randomized response and balanced bloom filters for privacy preserving record linkage. In: IEEE ICDMW, pp 218–224 https://doi.org/10.1109/ICDMW.2016.0038
28. Sehili Z, Kolb L, Borgs C, Schnell R, Rahm E (2015) Privacy preserving record linkage with PPJoin. In: Proc. BTW
29. Sehili Z, Rahm E (2016) Speeding up privacy preserving record linkage for metric space similarity measures. Datenbank Spektrum 16(3):227–236. https://doi.org/10.1007/s13222-016-0222-9
30. Vatsalan D, Christen P, Verykios VS (2013) A taxonomy of privacy-preserving record linkage techniques. Inf Syst 38(6):946–969. https://doi.org/10.1016/j.is.2012.11.005
31. Vatsalan D, Sehili Z, Christen P, Rahm E (2017) Privacy-preserving record linkage for big data: current approaches and research challenges. Handb Big Data Technol. https://doi.org/10.1007/978-3-319-49340-4_25
32. Winter A, Stäubert S, Ammon D, Aiche S, Beyan O, Bischoff V, Daumke P, Decker S, Funkat G, Gewehr JE, de Greiff A, Haferkamp S, Hahn U, Henkel A, Kirsten T, Klöss T, Lippert J, Löbe M, Lowitsch V, Maassen O, Maschmann J, Meister S, Mikolajczyk R, Nüchter M, Pletz MW, Rahm E, Riedel M, Saleh K, Schuppert A, Smers S, Stollenwerk A, Uhlig S, Wendt T, Zenker S, Fleig W, Marx G, Scherag A, Löffler M (2018) Smart Medical Information Technology for Healthcare (SMITH). Methods Inf Med 57(1):e92–e105. https://doi.org/10.3414/ME18-02-0004
33. Xiao C, Wang W, Lin X, Yu JX (2008) Efficient similarity joins for near duplicate detection. In: Proceedings of the 17th International Conference on World Wide Web, pp 131–140 https://doi.org/10.1145/1367497.1367516
34. Zezula P, Amato G, Dohnal V, Batko M (2006) Similarity search: the metric space approach. Springer, Berlin, Heidelberg https://doi.org/10.1007/0-387-29151-2