An Overview of Big Data Issues in Privacy-Preserving Record Linkage

Dinusha Vatsalan[1], Dimitrios Karapiperis[2](✉), and Aris Gkoulalas-Divanis[3]

[1] Data61, CSIRO, Sydney, Australia
dinusha.vatsalan@data61.csiro.au
[2] School of Science and Technology, Hellenic Open University, Patras, Greece
dkarapiperis@eap.gr
[3] IBM Watson Health, Cambridge, MA, USA
gkoulala@us.ibm.com

Abstract. Nearly 90% of today's data have been produced only in the last two years! These data come from a multitude of human activities, including social networking sites, mobile phone applications, electronic medical records systems, e-commerce sites, etc. Integrating and analyzing this wealth and volume of data offers remarkable opportunities in sectors that are of high interest to businesses, governments, and academia. Given that the majority of the data are proprietary and may contain personal or business sensitive information, Privacy-Preserving Record Linkage (PPRL) techniques are essential to perform data integration. In this paper, we review existing work in PPRL, focusing on the computational aspect of the proposed algorithms, which is crucial when dealing with Big data. We propose an analysis tool for the computational aspects of PPRL, and characterize existing PPRL techniques along five dimensions. Based on our analysis, we identify research gaps in current literature and promising directions for future work.

Keywords: Privacy-Preserving Record Linkage · Entity resolution

1 Introduction

In the era of information explosion, massive amounts of data, coming from various sources, need to be integrated to facilitate data analysis for businesses, governments, and academia. Record linkage, also known as *entity resolution* or *data matching*, is the process of resolving whether two records that belong to disparate data sets, refer to the same real-world entity. Record linkage is a two-step process. The goal of the first step, known as *blocking*, is to formulate as many as possible matching pairs and, simultaneously, maintain the number of non-matching pairs as small as possible. In the second step, termed as *matching*, the distances between the pairs formed during the blocking step are calculated. Privacy-Preserving Record Linkage (PPRL) investigates how to perform the steps described above in a secure manner, by respecting the privacy of the individuals who are represented in the data. For this reason, input records undergo a data masking process that embeds them into a space, where the underlying data is kept private (Fig. 1).

In this paper, we adapt the taxonomy proposed for PPRL in [69] and – inspired from analysis tools that are commonly used in business – such as SWOT and PEST [23,54], we develop an analysis tool that focuses on the computational aspects of PPRL techniques. We describe the proposed analysis tool for the computational aspects of PPRL in Sect. 2. In Sect. 3, we use this tool to characterize, analyze, review and compare existing PPRL techniques with respect to their computational aspects. Last, in Sect. 4, we discuss a number of gaps that we identified in current literature, along with some promising directions for future research.

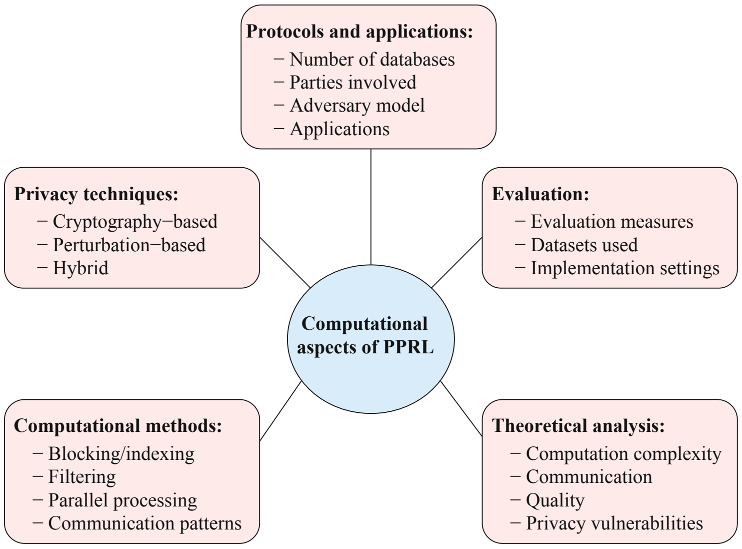

Fig. 1. An analysis tool consisting of five dimensions used to analyze and characterize computational aspects of PPRL techniques.

2 Analysis Tool

In this section, we present our proposed analysis tool for the computational aspects of PPRL techniques. It consists of five dimensions, each of which includes several topics. These dimensions are: (1) protocols and applications, (2) privacy techniques, (3) computational methods, (4) theoretical analysis, and (5) evaluation. In what follows, we describe each of these dimensions in detail.

2.1 Protocols and Applications

This dimension includes the protocol settings and application areas of PPRL techniques. The PPRL protocol is determined by the number of databases to be linked, the parties involved, and the considered adversarial model. The application areas of PPRL specify the different computation aspects, such as volume, flow, real-time vs batch processing, dynamic nature, and sensitivity of errors and variations in the data.

Number of databases. The computational aspect of PPRL is associated with the number of databases that have to be linked using the PPRL protocol. The naïve comparison space required for PPRL has an exponential growth with the number of databases. Existing techniques for PPRL can be categorized into *two databases linking* and *multiple databases linking*, where the latter received a lot of attention recently, due to the increasing demand of supporting Big Data applications [74,76]. The computational challenges and privacy risk in terms of collusion between parties, with the aim to learn another party's data, increase with the number of databases to be linked. Further, the variations and different schemas used in different databases, result in the linkage quality challenge, which requires advanced techniques.

Parties involved. Different PPRL protocols use different types of parties for the linkage. The database owners typically participate in the protocol with the use of an external linkage unit for facilitating the linkage. The linkage unit conducts linkage of the encoded records from the database owners. Some PPRL protocols use more than one linkage unit, which leads to additional privacy risks. Linkage unit-based approaches are computationally more efficient that other PPRL approaches, especially for linking multiple large databases, since the protocols without linkage unit need more complex techniques to make sure that the database owners cannot infer any information from the data that is exchanged among them [69]. In addition, a global authority might be used in some protocols for managing or providing encoding keys and protocol parameters to the parties of the protocol. Finally, a researcher or an external party may be involved in the protocol to obtain access to some attributes of the records that are identified as matching by the protocol, for conducting further analysis.

Adversary model. PPRL protocols generally assume either the *honest-but-curious model* (HBC), or the *malicious model* [21,22,46]. In the HBC model, parties are curious in that they try to find out as much as they can about the other party's inputs, through inference attacks or collusions, while following the protocol [22,46]. Inference attacks can be performed on encoded records based on some background information, such as frequency distribution, to re-identify the records. Collusion is a privacy risk of some parties colluding among them to learn other parties' sensitive information [46]. The protocol is secure in the HBC perspective if and only if all parties involved obtain no new knowledge at the end of the protocol, above what they would have learned from the output. In contrast to HBC parties, in the malicious model the parties behave arbitrarily in terms of refusing to participate in a protocol, not following the protocol, choosing arbitrary values for their data inputs, or aborting the protocol at any time [45]. PPRL techniques under the malicious model are computationally expensive and privacy evaluation of PPRL techniques under this model is difficult compared to the HBC model, due to many potentially unpredictable ways of malicious parties to deviate from the specified steps of the protocol [9,21,46]. Since PPRL techniques for HBC models are not realistic for real-world applications and PPRL for malicious models are computationally expensive, more advanced models have been recently proposed in cryptography [46]. Two of them are the *accountable computing* and the *covert model*, where the former allows honest parties to detect the misbehavior of an adversary with high probability [2], and the latter provides accountability for privacy compromises without the excessive complexity and cost of the malicious model [27].

Applications. PPRL is increasingly being required in several application areas, including in healthcare, national security, crime and fraud detection, business, governments, social sciences and population informatics. For example, data from several sources including different hospitals, pharmacies, and travel data, need to be linked for outbreak detection or clinical trials in healthcare applications [8,48], while linking the social security databases, law enforcement agencies databases, the police databases, and Internet service providers databases allows identifying crimes and frauds in national security and crime and fraud detection applications [28,55,78]. The computational requirements of PPRL techniques depend on the application area they are developed for, including the type and size of data, required output, and other application-specific constraints.

2.2 Privacy Techniques

The privacy techniques used for encoding data, processing data, and comparing and classifying data in PPRL can be categorized as follows:

Cryptographic-based techniques. These employ computationally expensive secure multi-party computation (SMC) techniques, such as homomorphic encryptions, Yao-based protocols, secret sharing, secure scalar product, and secure vector operations [46]. Although these techniques are provably secure and highly accurate, they are not efficient and scalable enough to be used for linking large databases.

Perturbation-based techniques. These are computationally efficient methods, allowing PPRL to scale to large databases. Increasing the privacy of perturbation-based techniques, however, results in accuracy loss, and vice versa. Some of the widely used perturbation techniques include generalization (such as k-anonymity, value generalization hierarchies, and binning), noise addition techniques (such as random and differential privacy), embedding techniques, and probabilistic data structure-based approaches (such as Bloom filters and count-min sketches). Perturbation-based techniques, especially Bloom filters, have increasingly been used in several real PPRL applications in recent times [60].

Hybrid PPRL approaches. These use perturbation-based techniques to efficiently remove highly non-matching records from the comparison space and then apply cryptographic-based techniques on the resulting records to achieve high quality linkage results without excessive computation [26].

2.3 Computational Methods

Several computational methods have been proposed in the literature aiming to reduce the exponential comparison (or search) space required by naïve PPRL and to speedup the linkage process. These methods of optimization are largely orthogonal, so that they can be combined to achieve maximal efficiency.

Blocking approaches. Blocking is defined on selected attributes to partition the records in a database into several blocks, such that comparison can be restricted to the records of the same block. Numerous blocking techniques have been used for record linkage [11] and for PPRL, with the additional challenge in the case of PPRL of preserving privacy in the blocking step [69]. Blocking improves the runtime of linkage, but it still involves unnecessary comparisons that limit its performance. Block processing is the approach of restructuring a collection of generated blocks to be compared and classified in the next step, so that unnecessary comparisons are pruned [32,59].

Filtering approaches. Filtering is an optimization for a particular comparison function which optimizes the evaluation of a specific similarity measure for a predefined similarity threshold to be met by matching records. It utilizes filtering or indexing techniques to eliminate sets of records that cannot meet the similarity threshold for the selected similarity measures [10,16].

Parallel processing approaches. Parallel linkage aims at improving the execution time proportionally to the number of processors [15,40,41]. This can be achieved by partitioning the entire set of record pairs to be compared, and conducting the comparison of the different partitions in parallel on different processors. A special case would be to utilize a blocking approach to compare the records in different blocks in parallel. Two approaches have been used so far for parallel linkage: (1) utilizing graphics processing units (GPUs) [19,63], and (2) using Hadoop and its MapReduce framework [37,42,77].

Communication patterns. Different communication patterns have different computation and communication complexities. With the increasing number of databases, the comparison space remains very large, even when a blocking or indexing technique is used [56,72,74]. Improved communication patterns can reduce the exponential growth (for larger number of databases to be linked) down to a smaller value. Such improved communication patterns include sequential, ring-by-ring, tree-based, and hierarchical patterns. Some of these patterns have been recently used for PPRL on multiple databases [74].

2.4 Theoretical Analysis

The dimension of theoretical analysis of PPRL techniques includes analysis of complexity, quality, and privacy vulnerabilities to allow for comparison and assessment of their expected scalability to large databases, quality of linkage results, and privacy guarantees.

Computation and communication complexity. The overall computational efforts and cost of communication required in the PPRL process are generally measured using the big O-notation [53]. For example, given that n is the number of records in a database, $O(n)$ represents linear complexity, $O(n^2)$ quadratic complexity, and $O(c^n)$ exponential complexity, where $c > 1$.

Quality of linkage. The quality of linkage is theoretically analyzed in terms of fault-tolerance of the linkage technique to data errors and variations, whether the matching records are identified across all databases or it allows identifying matching records across subsets of databases, trade-offs with privacy and complexity, step-wise quality (i.e., preprocessing quality, blocking quality, and matching quality), and final linkage quality.

Privacy vulnerabilities. The privacy vulnerabilities that a PPRL technique is susceptible to, provide a theoretical estimate of the privacy guarantees of the technique. These include *frequency attacks*, where the list of encoded values is matched with the frequency distribution of a list of unencoded values, *dictionary attacks*, where a list of unencoded values are matched with the list of encoded values by applying different encoding functions on the unencoded values, and encoding-specific attacks, such as *cryptanalysis attacks* specific to Bloom filter encoding, where – depending upon the parameter setting – iterative mapping of individual encoded values back to their original values is possible using a constrained satisfaction solver. Another vulnerability associated with linkage unit-based approaches and/or multiple-databases linking is *collusion* between parties, where parties involved in a PPRL protocol may work together to find out another party's data.

2.5 Evaluation

The linkage outcomes need to be evaluated in terms of scalability, linkage quality, and privacy. This dimension includes evaluation measures, datasets, and implementation settings.

Evaluation measures. The scalability and linkage quality can be evaluated using standard evaluation measures, such as runtime, memory consumption, communication size, speedup, reduction ratio, pairs completeness, pairs quality, precision, recall, and the F-measure [10,69,73]. However, linkage quality evaluation requires access to truth data, which can be rarely accommodated in PPRL applications. Consequently, sample evaluation or evaluation on synthetic/perturbed datasets is typically used to assess linkage quality. Various measures have been used to quantify the privacy protection of PPRL techniques, including information theory-based entropy and information gain measures [17,31,64], as well as disclosure risk-based measures [18,25,65,70,73]. However, no standard measures for privacy evaluation have been used in the literature.

Datasets. Experimental evaluation of PPRL techniques on several datasets is important to gain reliable evidence of the techniques' performance. Due to the difficulties of obtaining real-world data that contain personal information, synthetically generated or perturbed databases are typically used. Several tools are available to generate or corrupt data [13,24,66]. However, to evaluate PPRL techniques with regard to their expected performance in real-world applications, evaluations should ideally be done on databases that exhibit real-world properties and error characteristics.

Implementation. The implementation techniques that have been used to prototype a PPRL technique and the settings used for conducting its experimental evaluation, determine the complexity and scalability results. Further, some scalability measures, such as runtime, memory size, and communication size, are platform dependent. Comparing different techniques requires conducting experimental evaluation in the same platform and settings.

3 Literature Review

In this section, we review existing literature and categorize PPRL techniques along the dimensions of computational methods, which include: (1) blocking/indexing techniques, (2) block processing techniques, (3) filtering techniques, (4) parallel processing, and (5) improved communication patterns.

3.1 Blocking Techniques

Numerous blocking strategies [10] have been developed for record linkage and PPRL. Standard blocking groups records according to blocking criteria (known as *blocking key*), to partition all records into disjoint blocks. The blocking key values (BKVs) are the values of a selected attribute (e.g., zipcode), or the result of a function on one or several attribute values (e.g., the concatenation of the first two letters of last name and year of birth). Other blocking approaches include sorted neighborhood that sorts records according to a sorting key and only compares neighboring records within a certain window, and canopy clustering that results in overlapping clusters [10]. Multi-pass blocking is utilized to improve recall, where records are blocked according to different blocking keys, at the cost of a larger number of comparisons. In the following, we review blocking approaches for PPRL along the dimensions of the proposed tool.

Al-Lawati et al. [1] proposed a secure blocking protocol for linking two databases with the use of a linkage unit that assumed a *HBC* adversary model. Token-based blocking was used to improve computation efficiency. The linkage unit matches the records based on the computed TF-IDF distances of the hash signatures, using the *Jaccard* coefficient. The proposed blocking approach consists of three methods: *simple blocking*, *record-aware blocking*, and *frugal third party blocking* [1]. Simple blocking arranges hash signatures in overlapping blocks where the similarity of a pair may be computed more than once if they are in more than one common block. Record-aware blocking solves this issue by using an identifier with every hash signature to indicate the record it belongs to. Frugal third party blocking, uses a secure set intersection (SSI) protocol to reduce the cost of transferring the whole databases to the third party, by first identifying the hash signatures that occur in both databases.

Inan et al. [26] proposed a hybrid approach for PPRL of two databases under the *HBC* adversary model, by combining efficient generalization and expensive cryptographic privacy techniques. A blocking approach based on value generalization hierarchies is used, and the record pairs that need more detailed comparison to determine the match status are compared in a computationally expensive *SMC* computation step, using cryptographic techniques. The cost is reduced in the blocking step by reducing the number of candidate record pairs that need to be compared using SMC techniques.

A secure blocking based on phonetic encoding algorithms was presented by Karakasidis et al. [29]. A *three-party* (two database owners and a linkage unit) setting in a *HBC* model is assumed. The basic idea is to encode the values of a BKV (e.g. last name) with a phonetic function, such as Soundex or Metaphone [10]. All records with the same phonetic code are assigned to the same block. This approach uses a secure version of *edit distance* on *Bloom filters.* The experimental study, conducted on a synthetic dataset (generated using *Febrl* [12]), showed that the approach outperforms the original edit distance algorithm in terms of complexity (due to the secure blocking component), while preserving privacy, and also offers almost the same matching performance.

Karakasidis et al. [31] proposed three noise addition techniques for improving the privacy of [29]. In the first method, fake values are added to the datasets, such that both the attribute values and the Soundex values exhibit uniform distributions. This increases the complexity due to excessively oversized datasets. The second method overcomes this drawback by modifying the frequency of attribute values, such that all Soundex values occur equally frequent. However, some attribute values are removed where the corresponding Soundex values have more than the average number of attribute values, and therefore true matches might be missed in the linkage process. The third method adds fake values, where each Soundex value reflects at least k attribute values. Parameter k is tunable to adjust the number of fake records added in order to balance the trade-off between complexity and privacy. This work was experimentally evaluated using a real Australian telephone database, and the results indicated that in terms of information gain, using a phonetic-based fake injection blocking approach can offer adequate privacy for PPRL.

A generalization-based k-anonymous private blocking approach using a reference table was proposed in [30] for linking two databases using a linkage unit. Initially, clusters of size k are generated for the set of reference values that are shared by the database owners, using the k-nearest neighbor clustering algorithm with the Dice coefficient metric. Then, each database owner assigns their set of BKVs to the respective clusters. The resulting clusters are sent to the linkage unit that identifies and merges the corresponding clusters to generate candidate record pairs. Clusters contain at least k reference values, making inference attacks using the reference values difficult. Experiments conducted using a real Australian telephone book as the reference table and synthetic data generated using *Febrl* [12], as datasets to be linked, validate that this approach provides k-anonymity guarantees, while reducing the number of candidate record pairs.

Vatsalan and Christen in [71] proposed an approach that utilizes local sorted neighborhood clustering for improved performance in the blocking phase, to generate k-anonymous clusters based on reference values. Each database owner sorts a shared set of reference values and then inserts their records into the sorted list according to their sorting keys. Initial Sorted Neighborhood Clusters are determined such that each cluster contains one reference value and a set of database records. To offer k-anonymity, the initial clusters are merged into larger blocks containing at least k database records based on similarity or size constraints. These clusters are then sent to a linkage unit to identify and merge similar clusters across the databases, based on the reference values in the clusters. Experiments conducted on real Australian telephone and North Carolina voters databases, show the improved performance of the approach compared to [30], in terms of runtime and blocking quality. A variant of [71], involving a two-party setting without a linkage unit, was presented in [75]. In this approach, the two database owners generate their reference values independently. Each database owner sorts its reference values, inserts its records into the sorted list, builds initial clusters (with one reference value and its associated records), and merges these clusters to guarantee k-anonymity. Afterwards the database owners exchange their reference values, which are then merged together and sorted. In order to find candidate pairs between the sources, a sorted neighborhood method with a sliding window w is applied on the reference values. The window size w determines the number of reference values originating from each data source in a window. The clusters of the reference values that fall into the same window are determined as candidate blocks, which need to be compared in the next step. An evaluation study, conducted in [73] using real and synthetic datasets, showed that this approach outperforms several existing blocking approaches in terms of runtime and privacy, with no loss in blocking quality.

Durham investigated the use of Locality-Sensitive Hashing (LSH) [20] for private blocking of records encoded as Bloom filters [17]. She proposed the use of a family of hash functions (Minhash for Jaccard, or Hamming LSH for Hamming distances) to generate keys that are used to partition the records in a database, so that similar records are grouped into the same block [39]. A Minhash function permutes the bits in a Bloom filter and selects the first index position in the permuted Bloom filter that is set to 1. By applying ϕ Minhash functions, ϕ index positions are obtained, which are concatenated to generate the final Minhash key for the Bloom filter. The HLSH hash functions select the bit value of a Bloom filter at a random position. In the same way as Minhash, ϕ HLSH functions are applied on a record's Bloom filter and the values of the ϕ selected bits are concatenated to obtain the final hash key. LSH provides guaranteed accuracy while being efficient. However, it requires data dependent parameters to be tuned effectively and it can be applied only to specific encodings, such as Bloom filters.

Karapiperis et al. proposed a private blocking approach for linking multiple databases based on LSH [34–36]. This approach uses L independent hash tables, each consisting of key-bucket pairs, where keys represent the blocking keys and buckets host a linked list aimed at grouping similar records. Each hash table is assigned a set of K hash functions, generated by a linkage unit and sent to all the database owners to populate their set of blocks. This approach was later extended by proposing a frequent pairs scheme (FPS) [38] for reducing the number of comparisons, while maintaining a high level of recall. FPS achieves high blocking quality by identifying similar record pairs that exhibit many LSH collisions, and then performs distance calculations only for those pairs. Empirical results showed significant improvement in running time due to a drastic reduction of candidate pairs by FPS, while achieving high blocking quality [35]. Based on these methods, the authors have developed LSHDB [33], which uses LSH to efficiently block the masked records, and store the produced blocking structures on disk for further use. LSHDB achieves very fast response times, which makes it ideal for online settings, thanks to the utilization of efficient algorithms and the employment of flexible and robust noSQL systems for storing the data.

Ranbaduge et al. [56] proposed a private blocking approach for multiple databases without a linkage unit, based on a single-bit tree data structure. The single-bit tree is iteratively constructed by all database owners to store records' Bloom filters, such that similar records are placed into the same tree leaf. At each iteration, the set of Bloom filters in a tree node is recursively split based on selected bit positions, which are agreed upon by all parties. This requires a communication step among all parties in each iteration of the algorithm. Another drawback of this approach is that it might miss true matches due to the recursive splitting of Bloom filters. This limitation was addressed in [57], using a multi-bit tree [43] data structure combined with canopy clustering. Multi-bit trees were used to split the database records (encoded into Bloom filters) individually by the database owners into small mini-blocks, which are then merged into larger blocks according to privacy and computational requirements, using a canopy clustering technique [14].

A communication-efficient private blocking approach for multiple databases using a linkage unit, was proposed in [58]. In this approach, local blocks are generated individually by each database owner, using a private blocking technique. A block representative, in the form of a min-hash signature [7], is then generated for each block and sent to the linkage unit. The linkage unit applies global blocking using LSH to identify the candidate block sets from all databases, based on the similarity between block representatives. Local blocking enables the database owners to generate their blocks with more flexibility and control, without any iterative communication among them. This approach outperforms existing private blocking approaches for multiple databases in terms of scalability, privacy, and blocking quality [58].

3.2 Block Processing Techniques

Several block processing methods have been used for record linkage and PPRL, in order to process the generated blocks in an efficient and effective way to reduce the number of required comparisons, while improving blocking quality [51,52,59]. In this section we review these block processing techniques.

Wang et al. [79] introduced an iterative block processing technique for deduplicating a single database. The comparison results of blocks are propagated to subsequent blocks to avoid repeated comparisons. This approach was later extended in [39] for record linkage using LSH.

Two categories of block processing methods were used by Papadakis et al. [51] for deduplication. The first category includes block purging and block scheduling methods, which operate at the coarse level of processing individual blocks. The second category of comparison-refinement methods, such as comparison propagation, duplicate propagation, comparison pruning, and comparison scheduling, operate at a finer level of individual comparisons within blocks.

The concept of meta-blocking was introduced by Papakadis et al. in [52] for record linkage, to restructure a collection of blocks to reduce the number of comparisons. In their approach, a block collection is provided as input to a supervised classifier to identify promising comparisons based on block-feature vectors. The drawback of this approach is the selection of suitable features and requirement of training data to achieve accurate pruning of record comparisons.

Meta-blocking has been recently studied for the PPRL of two databases using a linkage unit [32]. A sorted neighborhood blocking based on reference values is used along with multi-sampling transitive closure as a processing technique to prune records based on redundant assignments to blocks. Experimental results show the efficacy of the approach in terms of recall and computational cost.

Recently, a general meta-blocking technique for PPRL on multiple databases was proposed by Ranbaduge et al. [59]. Their approach uses a graph structure to schedule the comparison of blocks with the aim of minimizing the number of repeated and superfluous comparisons between records, where the former is comparison of duplicate record pairs and latter is comparison of records with non-matching record pairs. The experimental results of their approach on real datasets, show that up to five orders of magnitude reduction in the number of record comparisons can be achieved compared to existing approaches.

3.3 Filtering Techniques

Several filtering approaches have been proposed in record linkage and PPRL literature to speed up the linkage process. The proposed optimizations, include the use of different filters, such as length and prefix filters, and dynamic inverted indexes [5]. Several filtering approaches also utilize the characteristics of similarity measures for metric spaces to reduce the search space, such as the triangle inequality [80]. For PPRL, filtering approaches need to be adapted to the comparison of encoded records, such as Bloom filters. In what follows, we describe several filtering approaches that have been proposed for PPRL.

Token-based similarity functions, such as the Jaccard and Dice coefficients, allow the application of a simple *length filter* to reduce the comparison space. This is because the minimal similarity can only be achieved if the lengths (for example, the number of bits set to 1 in Bloom filters) of the two records do not deviate too much. Formally, for records r_i, r_j, with $|r_i| \leq |r_j|$, it holds that: $Jacc\_sim(r_i, r_j) \geq s_t \Rightarrow |r_i| \geq \lceil s_t \cdot |r_j| \rceil$ and $Dice\_sim(r_i, r_j) \geq \frac{2min(|r_i|,|r_j|)\cdot(1-s_t)}{s_t}$.

For example, two records cannot satisfy a Jaccard similarity threshold $s_t = 0.8$ if their lengths differ by more than 20%, and a Dice similarity threshold $s_t = 0.8$ if the length difference is at least 50% of the length of the smaller record. Hence for a similarity threshold of 0.8, the length filter would prune record pairs that do not meet the length condition without comparing in detail.

Vatsalan and Christen used such a length filter for Dice coefficient similarity for Bloom filter-based PPRL in a two-party setting without using a linkage unit [67]. In their approach, certain bit positions (depending on privacy criteria) from the Bloom filters are iteratively exchanged between the two database owners to classify the pairs as matches, non-matches, and possible matches. For the possible matches in an iteration, more bits are revealed in the next iteration until the maximum number of bits are revealed, or all pairs are classified as matches or non-matches. A length filtering phase is used in addition to phonetic blocking to filter record pairs that are potentially non-matches, without revealing any bit positions for these records' Bloom filters.

The privacy-preserving version of PPJoin (called P4Join), proposed by Sehili et al. [63], utilizes three filters to reduce the Bloom filter-based comparison space: the length filter, a prefix filter, and a position filter. The *prefix filter* excludes Bloom filter pairs that have an insufficient overlap in the bit positions set to 1, in order to satisfy a predefined threshold, and this overlap test can be limited to only the prefix bit positions of the Bloom filters. The *position filter* of P4Join can avoid the comparison of two records even if their prefixes overlap, depending on the prefix positions where the overlap occurs. However, these filtering approaches achieve only a small improvement for PPRL, since the filter tests incur significant cost. Moreover, Bloom filter encoding for PPRL should ideally have 50% of their bits set to 1, in order to make them less vulnerable to frequency attacks [49], thereby constituting filtering less effective.

The use of multi-bit trees was proposed for fast similarity search in large databases of chemical *fingerprints* (encoded into Bloom filters) [3,43]. A multi-bit tree is a binary tree used to iteratively assign fingerprints to its nodes based on match bits. A match bit refers to a specific position of the bit vector, and can be 1 or 0: it indicates that all fingerprints in the associated subtree share the specified match bit. When building up the multi-bit tree, one match bit, or multiple such bits, are selected in each step, so that the number of unassigned fingerprints can be roughly split in half. The split is continued as long as the number of fingerprints per node does not fall under a limit. The match bits can then be used for a query fingerprint, to determine the maximal possible similarity for subtrees when traversing the tree and can thereby eliminate many fingerprints to compare. The multi-bit tree-based approach was extended by Bachteler et al. [3] to partition the fingerprints according to their lengths, such that all fingerprints with the same length belong to the same partition. To apply the length filter, the search for similar fingerprints using a Jaccard similarity measure is restricted to the partitions meeting the length criterion of $Jacc\_sim(r_i, r_j) \geq s_t$. Query efficiency is further improved by organizing all fingerprints of a partition within a multi-bit tree. Experimental evaluations showed that the multi-bit tree approach is very effective and performs equally or superior to blocking approaches, such as canopy clustering and sorted neighborhood [3,62].

Several metric space-based PPRL approaches have been proposed in the literature. One of the main properties that a metric or distance function for metric spaces has to satisfy is the triangle inequality. Distance functions for metric spaces satisfying this property include the Euclidean distance, edit distance, Hamming distance and Jaccard coefficient (but not Dice coefficient) [80]. The triangle inequality has been used for private comparison and classification in PPRL, using reference values [50,68], as well as a filtering technique to reduce the comparison space for similarity search and record linkage [4,6]. The triangle inequality allows to eliminate the computation of distance between two objects, based on their distances to a reference object or pivot.

3.4 Parallel Processing

The utilization of GPUs that provide thousands of cores within a single machine to speed-up similarity computations is a relatively new approach for parallel processing [19]. At the same time, Hadoop provides programming frameworks, such as MapReduce, Spark, and Flink, that allow developing programs to be automatically executed in parallel on Hadoop clusters [42,77]. In the following, we review two existing parallel PPRL techniques based on these two approaches.

A GPU-based parallel PPRL approach using the P4Join filtering is described in [63]. The approach sorts the records encoded into Bloom filters according to the number of bits set to 1, and partitions the set of Bloom filters into equi-sized blocks, such that multiple of such blocks fit into the GPU memory. Pairs of blocks are then continuously loaded into the GPU for parallel comparison. Length filtering and prefix filtering are applied to remove pairs of blocks that do not meet the filtering criterion, to reduce the the number of comparisons. Experimental evaluation results show that the approach improved runtime by a factor of 20, even with a low-profile graphics card (Nvidia GeForce GT 540M).

Several record linkage approaches have utilized the Hadoop-based MapReduce framework for parallel processing [42,77]. The Map tasks read the input data and assign each record to a block according to its blocking key value. Then, the records are redistributed among the Reduce tasks, such that all records with the same blocking key value are sent to the same Reduce task. Comparison is then performed in parallel by the Reduce tasks. The load balancing problem with highly skewed block sizes for parallel processing, is addressed in [42].

Karapiperis and Verykios proposed a parallel PPRL approach for linking two databases with a linkage unit using MapReduce [37]. The approach uses a

LSH-based blocking in the Map phase and determines the Minhash signature for each record encoded into a Bloom filter. These signatures are fragmented into several pieces and the Bloom filters are redistributed such that all Bloom filters with the same Minhash fragment value are assigned to the same Reduce task for comparison. The approach thus leads to a replicated redistribution of Bloom filters according to the number of fragments and a Bloom filter may have to be compared at several Reduce tasks. To overcome this problem, an alternative approach of chaining two MapReduce jobs was proposed, where the first job outputs the pairs of records' identifiers in the Reduce phase. In the second job, duplicate record pairs are grouped at the same Reducer to be compared only once. The evaluation of this approach in [37] shows the efficiency of parallel processing. However, the study was limited to a few nodes and only 300K records.

3.5 Improved Communication Patterns

Most PPRL techniques use the naïve all-to-one communication, where all database owners send their encoded records to a linkage unit to conduct the linkage. A few PPRL techniques for linking multiple databases use a ring communication, where encoded records are sent from one database owner to another, following a ring pattern [69]. Another communication pattern is all-to-all communication, where each database owner sends encoded records to all other database owners [44]. Last, some PPRL techniques require communication in several steps, or iteratively in many rounds [56], making them impractical for real applications.

Several query tree representations have been used for optimizing multi-way join queries [47,61], and can be adapted for efficient processing of multi-party PPRL. Schneider and DeWitt [61] studied query processing plans with different types of structures: left-deep, right-deep, and bushy. Left-deep and right-deep trees use a base table as the inner and outer operand, respectively, of each join in the plan, while in bushy trees both inputs to a join may themselves result from joins. For PPRL, the concepts used in deep trees can be adapted to improve efficiency. However, only one recent study investigated such improved communication patterns for linking multiple databases in PPRL.

Recent work by Vatsalan et al. [74] proposed two improved communication patterns for reducing the number of comparisons for PPRL on multiple databases using counting Bloom filters (CBFs). In [74], the parties are grouped into rings and a secure summation protocol is used to generate a CBF for each set of parties' records encoded into Bloom filters. The comparison of records is conducted: (1) sequentially by a LU, such that only the matches of a ring are compared with the candidate record sets of the next ring, or (2) symmetrically, without a LU, where matches are identified for each individual ring in the first phase and then, using the matches from individual rings, the matches from all rings are identified in the second phase. The computational complexity of MP-PPRL techniques is exponential in the number of records per database (n^p, assuming n records in each of the p databases). These improved communication patterns reduce this exponential growth with p down to the ring size r (with $r < p$).

4 Conclusions

Computational aspects of Privacy-preserving record linkage (PPRL) are crucial for making PPRL viable for linking large collections of disparate data sources, especially for Big data applications. In this paper we proposed an analysis tool for analyzing, reviewing, and comparing existing computational methods for PPRL and then conducted an extensive survey using the proposed tool. Such an analysis tool allows identifying research gaps in the current literature and promising directions for future work in PPRL.

References

1. Al-Lawati, A., Lee, D., McDaniel, P.: Blocking-aware private record linkage. In: IQIS, pp. 59–68 (2005)
2. Aumann, Y., Lindell, Y.: Security against covert adversaries: efficient protocols for realistic adversaries. J. Cryptology **23**(2), 281–343 (2010)
3. Bachteler, T., Reiher, J., Schnell, R.: Similarity Filtering with Multibit Trees for Record Linkage. Tech. Rep. WP-GRLC-2013-01, German Record Linkage Center (2013)
4. Barros, J.E., French, J.C., Martin, W.N., Kelly, P.M., Cannon, T.M.: Using the triangle inequality to reduce the number of comparisons required for similarity-based retrieval. In: Electronic Imaging: Science & Technology, pp. 392–403 (1996)
5. Bayardo, R.J., Ma, Y., Srikant, R.: Scaling up all pairs similarity search. In: WWW, Canada, pp. 131–140 (2007)
6. Berman, A., Shapiro, L.G.: Selecting good keys for triangle-inequality-based pruning algorithms. In: IEEE Workshop on Content-Based Access of Image and Video Database, pp. 12–19 (1998)
7. Broder, A.Z.: On the resemblance and containment of documents. In: Compression and Complexity of Sequences, pp. 21–29. IEEE (1997)
8. Brook, E., Rosman, D., Holman, C.: Public good through data linkage: measuring research outputs from the western Australian data linkage system. Aust NZ J. Public Health **32**, 19–23 (2008)
9. Canetti, R.: Security and composition of multiparty cryptographic protocols. J. Cryptol. **13**(1), 143–202 (2000)
10. Christen, P.: Data Matching - Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection. Data-Centric Systems and Application. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31164-2
11. Christen, P.: A survey of indexing techniques for scalable record linkage and deduplication. IEEE TKDE **24**(9), 1537–1555 (2012)
12. Christen, P., Gayler, R., Hawking, D.: Similarity-aware indexing for real-time entity resolution. In: ACM CIKM, Hong Kong, pp. 1565–1568 (2009)
13. Christen, P., Pudjijono, A.: Accurate synthetic generation of realistic personal information. In: Theeramunkong, T., Kijsirikul, B., Cercone, N., Ho, T.-B. (eds.) PAKDD 2009. LNCS (LNAI), vol. 5476, pp. 507–514. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-01307-2_47
14. Cohen, W.W., Richman, J.: Learning to match and cluster large high-dimensional data sets for data integration. In: ACM SIGKDD, Edmonton, pp. 475–480 (2002)

15. Dal Bianco, G., Galante, R., Heuser, C.A.: A fast approach for parallel deduplication on multicore processors. In: ACM Symposium on Applied Computing, pp. 1027–1032 (2011)
16. Dey, D., Mookerjee, V., Liu, D.: Efficient techniques for online record linkage. IEEE Trans. Knowl. Data Engin. **23**(3), 373–387 (2010)
17. Durham, E.: A framework for accurate, efficient private record linkage. Ph.D. thesis, Faculty of the Graduate School of Vanderbilt University, Nashville, TN (2012)
18. Elliot, M., Hundepool, A., Nordholt, E., Tambay, J., Wende, T.: Glossary on statistical disclosure control. In: Joint UNECE/Eurostat Work Session on Statistical Data Confidentiality (2005)
19. Forchhammer, B., Papenbrock, T., Stening, T., Viehmeier, S., Draisbach, U., Naumann, F.: Duplicate Detection on GPUs. In: Database Systems for Business, Technology, and Web, pp. 165–184 (2013)
20. Gionis, A., Indyk, P., Motwani, R.: Similarity search in high dimensions via hashing. In: VLDB, pp. 518–529 (1999)
21. Goldreich, O.: Foundations of Cryptography: Basic Applications, vol. 2. Cambridge University Press, Cambridge (2004)
22. Hall, R., Fienberg, S.: Privacy-preserving record linkage. In: PSD, Corfu, Greece, pp. 269–283 (2010)
23. Hill, T., Westbrook, R.: Swot analysis: it's time for a product recall. Long Range Plann. **30**(1), 46–52 (1997)
24. Hoag, J., Thompson, C.: A parallel general-purpose synthetic data generator. ACM SIGMOD **36**, 19–24 (2007)
25. Hundepool, A., et al.: Handbook on statistical disclosure control. A Network of Excellence in the European Statistical System in the field of Statistical Disclosure Control (2010)
26. Inan, A., Kantarcioglu, M., Bertino, E., Scannapieco, M.: A hybrid approach to private record linkage. In: IEEE ICDE, Cancun, Mexico, pp. 496–505 (2008)
27. Jiang, W., Clifton, C., Kantarcıoğlu, M.: Transforming semi-honest protocols to ensure accountability. Data Knowl. Eng. **65**(1), 57–74 (2008)
28. Jonas, J., Harper, J.: Effective counterterrorism and the limited role of predictive data mining. Policy Anal. **584** (2006)
29. Karakasidis, A., Verykios, V.S.: Secure blocking+secure matching = secure record linkage. JCSE **5**, 223–235 (2011)
30. Karakasidis, A., Verykios, V.S.: Reference table based k-anonymous private blocking. In: ACM SAC, Riva del Garda, pp. 859–864 (2012)
31. Karakasidis, A., Verykios, V.S., Christen, P.: Fakling. In: Garcia-Alfaro, J., Navarro-Arribas, G., Cuppens-Boulahia, N., de Capitani di Vimercati, S. (eds.) DPM/SETOP -2011. LNCS, vol. 7122, pp. 9–24. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28879-1_2
32. Karakasidis, A., Koloniari, G., Verykios, V.S.: Scalable blocking for privacy preserving record linkage. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 527–536. ACM (2015)
33. Karapiperis, D., Gkoulalas-Divanis, A., Verykios, V.: LSHDB: a parallel and distributed engine for record linkage and similarity search. In: ICDM Demo, pp. 1–4 (2016)
34. Karapiperis, D., Gkoulalas-Divanis, A., Verykios, V.: Distance-aware encoding of numerical values for privacy-preserving record linkage. In: ICDE, pp. 135–138 (2017)

35. Karapiperis, D., Verykios, V.: An LSH-based blocking approach with a homomorphic matching technique for privacy-preserving record linkage. TKDE **27**(4), 909–921 (2015)
36. Karapiperis, D., Verykios, V.: FEDERAL: a framework for distance-aware privacy-preserving record linkage. TKDE **30**(2), 292–304 (2018)
37. Karapiperis, D., Verykios, V.S.: A distributed framework for scaling up lsh-based computations in privacy preserving record linkage. In: ACM BCI, pp. 102–109 (2013)
38. Karapiperis, D., Verykios, V.S.: A fast and efficient hamming LSH-based scheme for accurate linkage. KAIS **49**(3), 1–24 (2016)
39. Kim, H., Lee, D.: Harra: fast iterative hashed record linkage for large-scale data collections. In: EDBT, Lausanne, Switzerland, pp. 525–536 (2010)
40. Kim, H., Lee, D.: Parallel linkage. In: ACM CIKM, pp. 283–292 (2007)
41. Kirsten, T., Kolb, L., Hartung, M., Groß, A., Köpcke, H., Rahm, E.: Data partitioning for parallel entity matching. VLDB **3**(2) (2010)
42. Kolb, L., Thor, A., Rahm, E.: Dedoop: efficient deduplication with hadoop. VLDB **5**(12), 1878–1881 (2012)
43. Kristensen, T.G., Nielsen, J., Pedersen, C.N.: A tree-based method for the rapid screening of chemical fingerprints. Algorithms Mol. Biol. **5**(1), 9 (2010)
44. Lai, P., Yiu, S., Chow, K., Chong, C., Hui, L.: An efficient Bloom filter based solution for multiparty private matching. In: International Conference on Security and Management, p. 7 (2006)
45. Lindell, Y., Pinkas, B.: An efficient protocol for secure two-party computation in the presence of malicious adversaries. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 52–78. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-72540-4_4
46. Lindell, Y., Pinkas, B.: Secure multiparty computation for privacy-preserving data mining. JPC **1**(1) (2009)
47. Lu, H., Shan, M.C., Tan, K.L.: Optimization of multi-way join queries for parallel execution. In: VLDB, pp. 549–560 (1991)
48. Malin, B.A., El Emam, K., O'Keefe, C.M.: Biomedical data privacy: problems, perspectives, and recent advances. JAMIA **20**(1), 2–6 (2013)
49. Mitzenmacher, M., Upfal, E.: Probability and Computing: Randomized Algorithms and Probabilistic Analysis. Cambridge University Press, Cambridge (2005)
50. Pang, C., Gu, L., Hansen, D., Maeder, A.: Privacy-preserving fuzzy matching using a public reference table. In: McClean, S., Millard, P., El-Darzi, E., Nugent, C. (eds.) Intelligent Patient Management. Studies in Computational Intelligence, vol. 189, pp. 71–89. Springer, Heidelberg (2009).https://doi.org/10.1007/978-3-642-00179-6_5
51. Papadakis, G., Ioannou, E., Palpanas, T., Niederee, C., Nejdl, W.: A blocking framework for entity resolution in highly heterogeneous information spaces. IEEE Trans. Knowl. Data Eng. **25**(12), 2665–2682 (2013)
52. Papadakis, G., Papastefanatos, G., Koutrika, G.: Supervised meta-blocking. Proc. VLDB Endowment **7**(14), 1929–1940 (2014)
53. Papadimitriou, C.: Computational Complexity. Wiley, Hoboken (2003)
54. Peng, G.C.A., Nunes, M.B.: Using pest analysis as a tool for refining and focusing contexts for information systems research. In: Research Methodology for Business and Management Studies, Lisbon, Portugal, pp. 229–236 (2007)
55. Phua, C., Smith-Miles, K., Lee, V., Gayler, R.: Resilient identity crime detection. IEEE TKDE **24**(3), 533–546 (2012)

56. Ranbaduge, T., Christen, P., Vatsalan, D.: Tree based scalable indexing for multi-party privacy-preserving record linkage. In: AusDM (2014)
57. Ranbaduge, T., Vatsalan, D., Christen, P.: Clustering-based scalable indexing for multi-party privacy-preserving record linkage. In: Cao, T., Lim, E.-P., Zhou, Z.-H., Ho, T.-B., Cheung, D., Motoda, H. (eds.) PAKDD 2015. LNCS (LNAI), vol. 9078, pp. 549–561. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-18032-8_43
58. Ranbaduge, T., Vatsalan, D., Christen, P., Verykios, V.: Hashing-based distributed multi-party blocking for privacy-preserving record linkage. In: Bailey, J., Khan, L., Washio, T., Dobbie, G., Huang, J.Z., Wang, R. (eds.) PAKDD 2016. LNCS (LNAI), vol. 9652, pp. 415–427. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-31750-2_33
59. Ranbaduge, T., Vatsalan, D., Christen, P.: Scalable block scheduling for efficient multi-database record linkage. In: ICDM. Barcelona (2016)
60. Randall, S.M., Ferrante, A.M., Boyd, J.H., Semmens, J.B.: Privacy-preserving record linkage on large real world datasets. JBI **50**, 205–212 (2014)
61. Schneider, D.A., DeWitt, D.J.: Tradeoffs in processing complex join queries via hashing in multiprocessor database machines. In: VLDB, pp. 469–480 (1990)
62. Schnell, R.: An efficient privacy-preserving record linkage technique for administrative data and censuses. Stat. J. IAOS **30**(3), 263–270 (2014)
63. Sehili, Z., Kolb, L., Borgs, C., Schnell, R., Rahm, E.: Privacy preserving record linkage with PPJoin. In: BTW Conference, Hamburg (2015)
64. Shannon, C., Weaver, W.: The Mathematical Theory of Communication, vol. 19. University of Illinois Press, Urbana (1962)
65. Sweeney, L.: Computational disclosure control: A Primer on Data Privacy Protection. Ph.D. thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science (2001)
66. Tran, K.N., Vatsalan, D., Christen, P.: GeCo: an online personal data generator and corruptor. In: ACM CIKM, San Francisco, pp. 2473–2476 (2013)
67. Vatsalan, D., Christen, P.: An iterative two-party protocol for scalable privacy-preserving record linkage. In: AusDM, CRPIT, vol. 134, Sydney (2012)
68. Vatsalan, D., Christen, P., Verykios, V.S.: An efficient two-party protocol for approximate matching in private record linkage. In: AusDM, Ballarat (2011)
69. Vatsalan, D., Christen, P., Verykios, V.S.: A taxonomy of privacy-preserving record linkage techniques. JIS **38**(6), 946–969 (2013)
70. Vatsalan, D.: Scalable and approximate privacy-preserving record linkage. Ph.D. thesis, Research School of Computer Science, The Australian National University (2014)
71. Vatsalan, D., Christen, P.: Sorted nearest neighborhood clustering for efficient private blocking. In: Pei, J., Tseng, V.S., Cao, L., Motoda, H., Xu, G. (eds.) PAKDD 2013. LNCS (LNAI), vol. 7819, pp. 341–352. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-37456-2_29
72. Vatsalan, D., Christen, P.: Scalable privacy-preserving record linkage for multiple databases. In: ACM CIKM, Shanghai (2014)
73. Vatsalan, D., Christen, P., O'Keefe, C.M., Verykios, V.S.: An evaluation framework for privacy-preserving record linkage. JPC (2014)
74. Vatsalan, D., Christen, P., Rahm, E.: Scalable privacy-preserving linking of multiple databases using counting bloom filters. In: IEEE ICDMW, Barcelona, Spain (2016)

75. Vatsalan, D., Christen, P., Verykios, V.S.: Efficient two-party private blocking based on sorted nearest neighborhood clustering. In: ACM CIKM, San Francisco, pp. 1949–1958 (2013)
76. Vatsalan, D., Sehili, Z., Christen, P., Rahm, E.: Privacy-preserving record linkage for big data: current approaches and research challenges. In: Zomaya, A.Y., Sakr, S. (eds.) Handbook of Big Data Technologies, pp. 851–895. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-49340-4_25
77. Vernica, R., Carey, M.J., Li, C.: Efficient parallel set-similarity joins using MapReduce. In: Proceedings of ACM SIGMOD, pp. 495–506 (2010)
78. Wang, G., Chen, H., Atabakhsh, H.: Automatically detecting deceptive criminal identities. Commun. ACM **47**(3), 70–76 (2004)
79. Whang, S.E., Menestrina, D., Koutrika, G., Theobald, M., Garcia-Molina, H.: Entity resolution with iterative blocking. In: ACM SIGMOD, Providence, Rhode Island, pp. 219–232 (2009)
80. Zezula, P., Amato, G., Dohnal, V., Batko, M.: Similarity Search: The Metric Space Approach, vol. 32. Springer, New York (2006). https://doi.org/10.1007/0-387-29151-2