Privacy-Preserving Deep Learning Based Record Linkage

Thilina Ranbaduge Ⓡ, Dinusha Vatsalan Ⓡ, and Ming Ding Ⓡ, Senior Member, IEEE

Abstract—Deep learning-based linkage of records across different databases is becoming increasingly useful in data integration and mining applications to discover new insights from multiple data sources. However, due to privacy and confidentiality concerns, organisations often are unwilling or allowed to share their sensitive data with any external parties, thus making it challenging to build/train deep learning models for record linkage across different organisations’ databases. To overcome this limitation, we propose the first deep learning-based multi-party privacy-preserving record linkage (PPRL) protocol that can be used to link sensitive databases held by multiple different organisations. In our approach, each database owner first trains a local deep learning model, which is then uploaded to a secure environment and securely aggregated to create a global model. The global model is then used by a linkage unit to distinguish unlabelled record pairs as matches and non-matches. We utilise differential privacy to achieve provable privacy protection against re-identification attacks. We evaluate the linkage quality and scalability of our approach using several large real-world databases, showing that it can achieve high linkage quality while providing sufficient privacy protection against existing attacks.

Index Terms—Differential privacy, deep neural networks, data integration, Bloom filter encoding.

I. INTRODUCTION

ORGANISATIONS in many business domains increasingly produce large databases with millions of records, which might contain detailed and sensitive information about people, such as customers, patients, taxpayers, or travellers. Often such databases need to be shared and integrated to facilitate advanced analytics and processing. However, due to privacy and confidentiality concerns, organisations are not allowed or are unwilling to share their databases in plain text for data linkage purposes [1].

Privacy-preserving record linkage (PPRL) aims to develop techniques that facilitate the linking of sensitive databases without sharing data among the involved organisations [1], [2]. This process is often challenging because no unique entity identifiers, such as social security numbers, are available in the databases to be linked. Therefore, quasi-identifying attributes such as names and addresses, are used to identify records that are similar and likely belong to the same entity [3]. Such quasi-identifiers are however often not allowed to be shared between organisations due to privacy and confidentiality concerns.

A popular approach to link sensitive data in a privacy-preserving way is to encode quasi-identifying values, such as using Bloom filters (BFs), that allows fuzzy matching by calculating approximate similarities on the encoded values in order to identify matches and non-matches [1], [2]. Traditional PPRL applications commonly use a naïve threshold-based classifier to classify encoded record pairs as matches if their corresponding approximate similarity scores are above a user-defined similarity threshold [2].

Recent record linkage literature has shown that supervised classifiers, such as deep learning techniques, can yield significantly high accuracy of linkage [4], [5], [6]. Mudgal et al. [7] explored the design space for deep learning solutions for entity matching and studied the accuracy-efficiency trade-offs. Their experiments have shown that though deep learning does not bring any additional advantage over clean structured data, deep learning models can significantly outperform traditional record linkage approaches on textual data and where the data is prone to errors and variations. As real data often contains errors and variations [1], the use of deep learning can benefit the linkage of databases, especially in Big Data applications [8].

However, deep learning brings in several challenges for PPRL. First, supervised techniques, such as deep learning, require sufficient amounts of training data to train a linkage model, which is challenging due to the lack of ground truth links [7]. Further, generating training data with manual labelling is costly and time-consuming, leading to limited available training data. Second, training a global linkage model across multiple databases (held by different parties) requires exchanging the training data of individual parties among each other or with a trusted linkage unit, which raises additional security and privacy concerns.

Moreover, the use of a distributed learning setting, such as federated learning, in the PPRL context might not overcome the problem of exchanging sensitive data entirely due to the privacy risk involved in the linkage process. This is because the database owners need to reveal their unlabelled records to calculate record pair similarities with other database owners’ unlabelled records. Thus, privacy techniques need to be applied to individual records instead of the local models as performed in federated learning. To the best of our knowledge, deep learning has not been studied for PPRL so far, while its advantages have been effectively utilized in non-PPRL applications.

© 2023 The Authors. This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/
In this work, we aim to investigate the application of deep learning in the PPRL context. The main contributions of this work are: (1) we propose a novel and first PPRL protocol that can be used to link multiple databases with limited training data using a deep learning model, (2) we utilise differentially private BF encoding in the linkage process to provide privacy guarantees for entities associated with the records in the databases, (3) we theoretically prove the privacy guarantees provided by our approach, and (4) empirically evaluate the linkage quality, efficiency, and privacy guarantees using nine data sets from different domains, which shows that our approach can defend against a recently proposed PPRL attack method [9], while achieving a high linkage quality compared to several baselines.

Outline. The rest of the paper is organised as follows: We review related work in the following section and provide preliminaries in Section III. In Section IV, we describe our proposed protocol for PPRL based on deep learning classifiers and analyse the privacy guarantees of our protocol in Section V. We present and discuss the results of our experimental study in Section VI. Finally, we conclude and point out directions of future research in Section VII. Table I introduces the common notation we will use in this paper.

II. RELATED WORK

A long line of research has been conducted in privacy-preserving record linkage (PPRL) over the past three decades [1], [2], [8]. As surveyed in [10], the existing PPRL techniques can be categorised into four generations.

The first generation of PPRL techniques focused on linking records that have the exact same quasi-identifiers (QIDs) across data sets, supporting only exact matching. The second generation considered fuzzy matching of QIDs to account for errors and variations in the QIDs across different data sets. The third generation considered the scalability of the PPRL process to linking large databases as an important aspect in addition to the support for fuzzy matching [8], [11]. The fourth generation of techniques focused on data-driven technologies by developing more advanced methods tailored to Big Data. A key challenge of Big Data is to improve the linkage performance in the presence of data errors by using machine learning techniques [10].

Over the last decade, several deep learning-based techniques have been proposed to link entities in databases [4], [5], [12], [13]. Kooli et al. [5] studied the use of deep Neural Networks (DNN) for linking records where record pairs are classified into matches or non-matches based on word embeddings using a DNN model. DeepER is another approach, that achieves high linkage quality by using recurrent neural networks (RNNs) [4]. The aim is to convert each record to a distributed representation (i.e., a feature vector), which can effectively capture similarities between records. [13] uses deep learning for active and transfer learning to reduce the cost of manual labelling required for improving the accuracy of linking records. This approach allows one to learn a transferable model from a high-resource setting to a low-resource one, and to further adapt to the target data set, active learning is incorporated that carefully selects a few informative examples to fine-tune the transferred model.

However, these existing deep learning-based linkage techniques do not apply to PPRL. To the best of our knowledge, no work has so far considered addressing the privacy constraints in deep learning-based linkage. Supervised learning classifiers, such as deep learning, that are trained on sensitive data sets, can be vulnerable to privacy attacks, especially membership or attribute inference attacks when utilised in PPRL [4], [13].

Nóbrega et al. [14] recently proposed the use of blockchain technologies to gain accountability in a PPRL protocol. The Blockchain technology was employed to implement a semi-trusted party and allows the detection of misbehaving parties in the PPRL protocol. While the use of blockchain techniques is a novel contribution to PPRL, Christen et al. [15] showed [14] is susceptible to privacy attacks and sensitive values can be re-identified easily.

Further, several works have developed provable privacy-preserving data encoding algorithms [6], [16], [17]. For example, ε-differentially private BF encoding algorithms have been studied [6], [16], [17]. Training models on such BF encoded data with differential privacy guarantees makes the models resilient against inference attacks that aim to learn about the individuals used in the training data set [6]. [18] recently proposed a novel encoding technique for PPRL based on autoencoders that transform BF’s into numerical vectors.

Few studies have also investigated the federated learning model by training supervised learning models with privacy guarantees through adding differential privacy noise to the gradients [19]. A successfully used method for differential private federated learning is applying differential privacy in the form of a differentially private stochastic gradient descent (DP-SGD) optimiser [19]. DP-SGD can be prohibitively slow to train, thus a GPU infrastructure might be needed for efficient training.

In our work, we use differentially private BF encoding with horizontal federated learning for applying deep learning-based classification for PPRL. We add DP noise into BF’s instead of gradients in the local model. To the best of our knowledge, this is the first work to address the challenges of applying deep learning for PPRL with improved linkage quality and provable privacy guarantees.
III. BACKGROUND

We now explain how QID values in a record are encoded into a Bloom filter and the application of differential privacy in Bloom filters.

A. Bloom Filters

Bloom filter (BF) encoding was proposed by Schnell et al. [20] for PPRL because BFs can be used to efficiently calculate approximate similarities between records. A BF [21] \( b \) is a bit vector of length \( l = |b| \) where initially all bits are set to 0. Each data information element in a set \( s \in S \) is transformed into \( l \) bits using \( k > 1 \) hash functions, where each hash function outputs an index value between 0 and \( l - 1 \). These index values are then used to toggle the corresponding bits in vector \( b \) to 1. In PPRL, the set \( s \) is generally generated as \( q \)-grams, i.e., substrings of consecutive characters with a length \( q \), from one or more QID values from each record in a database, as shown in Fig. 1, where various methods have been proposed to encode strings [2], [20], [22] as well as numerical and sequence data values [6], [23]. It has however been shown that BF encoding can be vulnerable to privacy attacks [9], [24]. As we show in Section VI, sensitive values that occur frequently in an encoded database can lead to frequent bit patterns in BFs that can be identified [1], and even individual frequent \( q \)-grams can be found using pattern mining techniques [24].

B. Differential Privacy

Differential privacy [25] is a privacy definition that guarantees the outcome of a calculation is insensitive to any particular record in the data set. Differential privacy requires the output of a data analysis mechanism to be approximately the same if any single record is replaced with a new one. In order to obtain this privacy guarantee, the algorithm employed to compute the result of the analysis must contain some form of randomness such that the probability of obtaining a particular outcome \( e \in O \) from database \( D \) is associated to any pair database-outcome \((D,e)\). Formally:

Definition 1 (Neighbouring Databases): Databases \( D \in \mathcal{D} \) and \( D' \in \mathcal{D} \) over a domain \( \mathcal{D} \) are called neighbouring databases if they differ in exactly one record.

Definition 2 (Differential Privacy [26]): A randomised algorithm \( \mathcal{A} \) is \( \epsilon \)-differentially private if for all neighbouring databases \( D \) and \( D' \), and for all sets \( O \) outputs, we have

\[
Pr[\mathcal{A}(D) \in O] \leq exp(\epsilon) \cdot Pr[\mathcal{A}(D') \in O],
\]

where \( Pr[\cdot] \) denotes the probability of an event.

In the context of BF, random noise can be added to BFs to guarantee differential privacy. The most commonly used method to add random noise to BFs for differential privacy guarantees is using the randomised response method that flips certain bit positions (from 0 to 1 or from 1 to 0) in each BF with a certain probability. Different mechanisms have been introduced for adding random noise to BFs [6], [17].

Bloom and flip (known as BLIP) is a method that flips bit values at certain positions in a BF with a bit flip probability \( p \). Formally, for a given bit flipping probability \( p \), a bit \( b[i] \) in a BF \( b \) at position \( i \) is flipped according to:

\[
b[i] = \begin{cases} 
1 & \text{if } b[i] = 0 \text{ with probability } p, \\
0 & \text{if } b[i] = 1 \text{ with probability } p, \\
b[i] & \text{with probability } 1 - p.
\end{cases}
\]

Schnell and Borgs [17] applied the RAPPOR bit flipping method proposed by Erlingsson et al. [27]. Assuming again a flip probability \( p \), the bit value \( b[i] \) at position \( i \) is flipped according to:

\[
b[i] = \begin{cases} 
1 & \text{with probability } \frac{p}{2}, \\
0 & \text{with probability } \frac{p}{2}, \\
b[i] & \text{with probability } 1 - p.
\end{cases}
\]

For example, if the flip probability is set to \( p = 0.1 \) for a BF of length \( l = 1,000 \) bits, then around 100 bits will be randomly selected and flipped using (1), while the rest are unchanged. On the other hand, according to (2) in [17] bits are not flipped based on their original state in the BF \( b \), rather around 100 randomly selected bits are either set to 0 or 1 with equal probability \( p/2 \). Table II shows the estimated number of bits flipped in a BF with different numbers of 1-bits for different flip probabilities.

| Table II | Estimated Number of 1-Bits in a Bloom filter (BF) of Length \( l = 1000 \) Bits after Applying the Two Bit Flipping Approaches [16], [27] for Different Fill Percentages, FP, (Number of 1-Bits) in the BF and Different Flip Probabilities, \( p \) |
| --- | --- | --- | --- | --- | --- |
| FP (%) | 25 (250 1-bits) | 50 (500 1-bits) | 75 (750 1-bits) | 75 (750 1-bits) |
| \( p \) | 0.01 | 0.05 | 0.1 | 0.01 | 0.05 | 0.1 | 0.01 | 0.05 | 0.1 |
| Eq. 1 [16] | 262 | 275 | 300 | 500 | 500 | 500 | 500 | 500 | 712 | 725 | 700 |
| Eq. 2 [27] | 256 | 262 | 275 | 500 | 500 | 500 | 725 | 737 | 725 |
the privacy-preserving record linkage (PPRL) process attempts to accurately classify each record pair \((r_i, r_j)\) as belonging to either \(M\) or \(U\) \([3], [28]\) while preserving privacy of each entity \(e\) in \(D_A\) and \(D_B\).

The proposed approach consists of two main phases, known as training and classification. Similar to a horizontal federated learning setting, we allow different DOs to train a classification model collaboratively in the training phase. This classification model is then sent to a linkage unit (LU) to classify the unlabelled record pairs in the differentially private BF databases sent to it by the DOs into matches and non-matches. We next describe each phase in more details.

### A. Training Phase

The use of training data plays an important role in our approach. In order to train the model we need some pairs for which we know the true label. However, the availability of a large amount of training data in PPRL is not always possible. Below we discuss some methods DOs can use to generate some training data.

One way of obtaining this information is by manually reviewing some of the pairs. Another way of getting a training data set is to utilise any additional information that is available. For example, when linking a data set to a publicly available population register an official id might be available for some of the records in the data set. For these records, the true match status can be determined. Lastly, a set of records with similar quasi-identifiers that have been linked in a previous linkage project can be used as training data.

Here, we would like to emphasise that the DOs do not require to have the same training data representing the same set of entities. Thus, the DOs do not need to share their training data as they can individually generate and/or use their own training data in the training process. However, we assume that the characteristics of records in the training data of each DO be similar to each other, such as data quality issues, range and domain of attribute values, number of missing values, etc. This would allow the encoded BFs in the training data of DOs to have similar similarity distributions.

In our approach, we assume each DO has a small training data set. We assume the record pairs available in this training data set contain the same characteristics as the unlabelled data set allowing us to calculate the same set of features over the records. The main steps of the training phase are illustrated in Fig. 2.

As outlined in Algorithm 1, each DO first encodes the record pairs in its training data set into Bloom filters (BFs) (line 2). We assume the training data consist of record pairs \((r_i, r_j)\) and their true labels, \(t\), as match or non-match. As we explained in Section III-A, in lines 3 and 4, each record pair \((r_i, r_j)\) is encoded into BFs, \(b_i\) and \(b_j\), respectively, using the function \(\text{genBloomFilter}()\).

As we explained in Section III and experimentally validate in Section VI, in lines 5 and 6 the function \(\text{addDPNoise}()\) adds differentially private noise into the generated BFs \(b_i\) and \(b_j\) and creates two new BFs \(b_i'\) and \(b_j'\), respectively. Following Definition 2, we calculate the bit flip probability, \(p\), as,

\[
p = \frac{1}{1 + e^{\epsilon/2nk}}
\]

(3)

to achieve \(\epsilon\)-differential privacy for a given privacy budget \(\epsilon\), where \(n\) is the maximum number of q-grams from a record that are hash-mapped into the BF and \(k\) is the number of hash functions. We provide more details on (3) in Section V.

We follow (3) for flipping the bits in a BF. In Section V we theoretically prove that this provides \(\epsilon\)-differential privacy guarantees. Fig. 3 shows the corresponding flip probabilities \(p\) values for different \(\epsilon\). It is important to note that, due to the randomisation in the \(\text{addDPNoise}()\) function, the bit values in different positions are flipped in each BF in the pair \((b_i, b_j)\) in the bit flipping process.

In line 8 we iterate over each feature function \(f\) in the list \(\mathcal{F}\) to generate a feature vector \(\mathcal{F}\) for the noise added BF pair \((b_i', b_j')\). We use different similarity functions, such as Hamming distance, Dice similarity, Jaccard similarity, etc, as feature functions where each similarity/distance value computed between \((b_i', b_j')\) is added as a feature into \(\mathcal{F}\). Once the feature vector \(\mathcal{F}\) is generated, we append the corresponding ground-truth label \(t\) to \(\mathcal{F}\) and create the train data list \(\mathcal{V}\) (line 11).

In line 12, each DO will use the list \(\mathcal{V}\) to train a deep learning classifier \(C\) using the \(\text{trainModel}()\) function. Recently, different neural network architectures have been proposed to consider different types of embedding structures in entity resolution literature [4]. The utilisation of different deep learning models and their suitability in the record linkage process has been studies in [29].
In our approach, we use a Multi-Layer Perceptron (MLP) for the local models which is a feedforward neural network. We follow a similar MLP architecture used in [29] for local models. In our approach, all DOs agree on the parameter settings to be used for the local models. We provide the details of the hyperparameter selection of individual local models in Section VI. Finally, once the local models C are generated by all the DOs the local models are aggregated into a final global model CG as we illustrate in Fig. 2.

Following a horizontal federated learning setting [30], we utilise a secure aggregator (SA) in this phase to combine the local models sent by the DOs. The use of SA does not allow the LU to learn information about the local models as we explain in Section V. Following [31], SA uses a federated averaging approach to take the average of all weights sent by DOs. The average of the weights is regarded as the new set of weights for the global model. The final global model CG is then sent to the LU to be used in the classification phase.

To estimate the complexity of Algorithm 1, we assume each DO had |T| record pairs in their training data set and each record has \(n_q\) q-grams. For encoding each record into a BF and then adding DP noise into these BFs in lines 3 to 6, the algorithm has a complexity of \(O(|T| \cdot l(k \cdot n_q + 1))\). In lines 8 to 11, the algorithm generates a feature vector \(F\) for each BF pair which has a complexity of \(O(|T| \cdot |F|)\). We assume the algorithm takes a constant time complexity of \(O(c)\) to train the local model in line 12. Thus, the overall time complexity of Algorithm 1 is \(O(|T| (l(k \cdot n_q + 1) + |F|) + c)\).

### B. Classification Phase

As illustrated in Fig. 4, the DOs first encode their databases following the same parameter setting used in the training phase. Each DO then adds differentially private noise to their encoded database \(D^\epsilon\). As in Algorithm 1, the DOs use the functions genBloomFilter() and addDPNoise() to encode each attribute value in the list of attributes \(A\) of each record in their database into BFs and add differentially private noise to these generated BFs, respectively. The noise-perturbed encoded databases are then sent to the LU.

We outline the classification process for two encoded databases in Algorithm 2. In line 2 the LU first applies a blocking technique [11] to the encoded databases to generate candidate record pairs. We assume this blocking technique to be a black box as any appropriate private blocking technique can be used [1].

![Algorithm 2: Classification by the LU.](image)

**Algorithm 2:** Classification by the LU.

**Input:**
- \(-D_A^e:\) Alice’s encoded database
- \(-D_B^e:\) Bob’s encoded database
- \(-C_G:\) The global model
- \(-F:\) Feature functions

**Output:**
- \(-M:\) Classified match record pairs

1: \(M = [], B = \{\}\) \hspace{1cm} // Initialise variables
2: \(B = genBlocks(D_A^e, D_B^e)\) \hspace{1cm} // Generate blocks
3: \(\text{foreach} (r_i.id, r_j.id, b_i, b_j) \in B \text{ do:} \hspace{1cm} //\text{Loop over blocks}\)
4: \(\mathcal{F} = [\]\hspace{1cm} // Initialise feature vector
5: \(\text{foreach} f \in F \text{ do:} \hspace{1cm} //\text{Loop over each function}\)
6: \(s = f(b_i, b_j)\) \hspace{1cm} // Compute the feature value
7: \(\mathcal{F}.add(s)\) \hspace{1cm} // Add the value to the feature vector
8: \(m = classify(C_G, \mathcal{F})\) \hspace{1cm} // Classify the record pair
9: \(\text{if} m == \text{match do:}\)
10: \(M.add((r_i.id, r_j.id))\) \hspace{1cm} // Add the record pair to matches
11: return \(M\)
Fig. 4. Overview of the classification phase with two DOs. The steps within the dashed box are performed by the LU.

[2], [11]. The LU then iterates over each candidate record pair and applies the feature generation step by using the same list of feature functions \( F \) used in the training phase (lines 5 to 7).

In line 8, each feature vector \( \mathcal{F} \) is then classified using the global model \( C_G \) it received from the secure aggregator to classify each candidate record pair as a match or non-match. If the candidate record pair is classified as a match then the corresponding record identifier pair \( (r_i, \text{id}, r_j, \text{id}) \) is added to the set of matches \( M \) (line 10). Once all the record pairs are classified, the LU sends \( M \) to the DOs (line 11).

We assume the time complexity for blocking each database \( D \) in line 2 of Algorithm 2 is \( O(|D|) \). Assuming that each block in \( B \) contains \( n \) BF pairs, the classification of each BF pair (in lines 3 to 8) has a \( O(|B| \cdot n \cdot |F|) \) complexity.

V. PRIVACY ANALYSIS

Two possible methods can be used for preserving privacy in the training phase: 1) adding differential privacy noise to the encoded Bloom filters (BFs) and training the local models on the noise-added BFs to make the trained local models robust against inference attacks when shared, or 2) adding differential privacy noise to the trained model weights and exchanging only the noise-added model weights.

In the context of PPRL, the linkage of unlabelled records requires exchanging or sharing the (encoded) records between the DOs or with a third party (linkage unit), and thus the (encoded) records need to be privatised in any case. Hence, in our approach, method (1) is more appropriate, because for linking or classifying record pairs using the trained model, the similarities between records from different DOs need to be computed and used by the LU.

Moreover, training the model using training data and classifying unlabelled record pairs need to be conducted on the same feature space, i.e., training the model on non-perturbed BFs and classifying the perturbed BFs of unlabelled record pairs, would not learn correctly and perform accurate linkage. Thus, we apply differential privacy noise to the BFs and train the deep learning model, which is then used by the LU to classify the (differentially private) BFs of the unlabelled record pairs.

As proposed in BLIP [16], each bit in a BF can be flipped with a probability \( p = 1/(1 + e^{\epsilon/2g}) \), where \( \epsilon \) is the privacy budget and \( g \) is a scaling factor. In general, \( g \) can be defined based on the maximum number of bits two neighbouring BFs differ [17]. As shown in Fig. 5, specifically, the BFs are perturbed by using the randomised response method that flips the bits with a probability of \( \frac{1}{1 + e^{\epsilon/2g}} \) to achieve \( \epsilon \)-differential privacy, with \( g = nk \) where \( k \) is the number of hash functions used to hash-map tokens (e.g., \( g \)-grams for string attributes or neighbouring values for numerical attributes) into the BFs, and \( n \) is the maximum possible number of tokens in any record in a data set (assuming \( n < l \), where \( l \) is the length of the BFs).

\[
\text{Theorem 1. Flipping the bits in BFs with } \frac{1}{1 + e^{\epsilon/2g}} \text{ probability, where } k \text{ is the number of hash functions and } n \text{ is the maximum possible number of tokens in any record in a data set, makes the BFs } \epsilon \text{-differentially private.}
\]

\[
\text{Proof. Let us assume two Bloom filters (BFs) of any two records } r \text{ and } r' \text{ can at maximum differ by } n \times k \text{ bit positions (an example is illustrated in Fig. 5), when there are no collision between bits during hash-mapping, where } n \text{ is the maximum number of tokens in any record and } k \text{ is the number of hash functions.}
\]

\[
\text{Let } \hat{b} \text{ and } \hat{b}' \text{ be the BFs produced by records } r \text{ and } r' \text{ respectively before noise is applied. Let } A : \{0, 1\}^l \rightarrow \{0, 1\}^l \text{ be a random noise function such that } A(x) = x \text{ with probability } e^{-\epsilon/2nk}/1 + e^{\epsilon/2nk}, \text{ and } A(x) = 1 - x \text{ with probability } 1/(1 + e^{\epsilon/2nk}), \text{ where } x \in \{0, 1\}.
\]

\[
Pr[BF(r, \epsilon) = \hat{b}] = \prod_{\beta=1}^{l} Pr[A(b_{\beta}) = \hat{b}_{\beta}]
\]

\[
Pr[BF(r', \epsilon) = \hat{b}'] = \prod_{\beta=1}^{l} Pr[A(b'_{\beta}) = \hat{b}'_{\beta}]
\]

(4)

Note that the two BFs \( b, b' \in \{0, 1\}^l \) can only differ in at most \( 2 \times n \times k \) positions, corresponding to \( b_{h_i(r_j)} = 1 \) for all \( 1 \leq i \leq k \) and \( 1 \leq j \leq n \), and \( b_t = 0 \) for all \( t \neq h_i(r_j) \), while \( b'_{h_{i}(r_{j})} = 1 \) for all \( 1 \leq i \leq k \) and \( 1 \leq j \leq n \), and \( b'_t = 0 \) for all \( t \neq h_{i}(r_{j}) \). We can then simplify the ratio in (4) by considering only the corresponding \( n \times k \) positions, because all terms \( b_t = 0 \)

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
TABLE III
OVERVIEW OF THE DATA SETS USED IN THE EXPERIMENTAL EVALUATION

| Data set               | Domain        | Num Records | Ground Truth | Training data | Missing values (%) | Provenance |
|------------------------|---------------|-------------|--------------|---------------|--------------------|------------|
| DBLP-ACM (Clean)       | Publication   | 2,616 - 2,294 | 2,224        | 7,417         | 0 - 0              | Real       |
| DBLP-ACM (Dirty)       | Publication   | 2,616 - 2,294 | 2,224        | 7,417         | 49 - 47            | Real       |
| DBLP-Scholar (Clean)    | Publication   | 2,616 - 64,263 | 2,324    | 17,223        | 7 - 22              | Real       |
| DBLP-Scholar (Dirty)    | Publication   | 2,616 - 64,263 | 2,324    | 17,223        | 48 - 61            | Real       |
| iTune-Amazon           | Music         | 6,907 - 55,923 | 132         | 321           | 0 - 0              | Real       |
| Music-Brainz           | Music         | 3,827 - 3,861 | 1,219        | 3,565         | 24 - 23            | Real       |
| Amazon-Amazon          | E-commerce    | 1,363 - 3,226 | 1,300        | 3,571         | 8 - 10             | Real       |
| NCVR                   | Demographic   | 222,251 - 224,061 | 148,036  | 425,731        | 3 - 4              | Real       |
| European Census        | Demographic   | 25,343 - 24,613 | 24,043     | 71,867        | 3 - 3              | Synthetic  |

if \( t \neq h_i(r_j) \) and all terms \( b_i = 0 \) if \( t \neq h_i(r'_j) \), for all \( 1 \leq i \leq k \) and \( 1 \leq j \leq n \).

This ratio is maximised when all \( n \times 2k \) bit positions are different in both BFs, and therefore \( 2nk \) bits in either \( b \) or \( b' \) need to be flipped (maximum ratio).

\[
e^{-\epsilon} \leq \frac{Pr[BF(r, \epsilon) = \overline{b}]}{Pr[BF(r', \epsilon) = \overline{b}]} = \prod_{\beta=1}^{2nk} \frac{e^{\epsilon/2nk}}{1 + e^{\epsilon/2nk}} \leq e^{\epsilon} \tag{5}
\]

Bounding the above ratio, we get

\[
-\epsilon \leq \ln \left( \frac{Pr[BF(r, \epsilon) = \overline{b}]}{Pr[BF(r', \epsilon) = \overline{b}]} \right) \leq \epsilon \tag{6}
\]

VI. EXPERIMENTS

We conducted experiments to validate the effectiveness of the proposed approach. We first describe the experimental setup, including the data sets, baselines and parameter setting, and then we discuss in detail the results that we have obtained.

A. Experimental Setup

1) Data Sets: Table III shows the details of the data sets used in the experiments. DBLP-ACM and DBLP-Scholar consist of links between academic publications [4]. We used the attributes: authors’ names, publication name, venue, and year as the linkage attributes. For these two data sets, we use two variations Clean and Dirty for experiments, where Clean data sets contain records without any data quality issues, while Dirty data sets embody various data quality problems, such as missing values, misspellings, and variations of values [4].

The iTune-Amazon data set contains music data from iTunes and Amazon. We used the attributes song name, artist name, album name, genre, and price. The MusicBrainz data set is based on real records of songs from the MusicBrainz database [32]. We used the attributes song title, artist name, album name, and year in the linkage process.

We used the North Carolina Voter Registration (NCVR) database (http://dl.ncsbe.gov/data/) with one snapshot from April 2014 and a second one from June 2014. We extracted pairs of records that correspond to the same voter but had name and/or address changes over time. We used a synthetic European census database (https://ec.europa.eu/eurostat/cros/content/job-training_en) generated to represent real observations of the decennial census. The database contains personal details of fictitious people. We used the attributes first name, last name, street address, and city for both NCVR and European census databases.

Following the ER literature [4], each data set is split into the training, validation, and test sets using a ratio of 3:1:1. For each of the data sets, we performed K-fold cross-validation with \( K = 5 \). We report the average results along with the standard deviation.

2) Performance Evaluation Metrics: We evaluated scalability using runtime, and linkage quality using precision, recall, and F-measure [1]. We consider \( TP \), \( FP \), and \( FN \) as the number of true matches, false matches, and false non-matches, respectively [3]. Precision (P) can be computed as \( P = TP/(TP + FP) \) which measures the number of true matched record pairs against the total number of matched record pairs generated by a particular approach; and recall (R) can be computed as \( R = TP/(TP + FN) \) which measures the number of true matched record pairs against the total number of record pairs in the linked ground truth data [2]. F-measure (F) can be computed as \( F = (2 \times P \times R)/(P + R) \) which provides the harmonic mean between precision and recall.

Further, we also used a novel measure called \( F^* \)-measure [34] in our evaluation because recent research has shown that F-measure is not suitable for measuring linkage quality in record linkage due to the relative importance given to precision and recall, which depends upon the number of predicted matches. The \( F^* \)-measure \( (F^*) \) is calculated as \( F^* = \frac{TP}{TP + FP + FN} \), which corresponds to the number of true matches against the number of matches which are either misclassified or are correctly classified [34].

As baselines, we compared our approach (referred to as \( DP-DL-BF \)) with four PPRL techniques. (1) We used a BF-based PPRL technique (named as \( BF \)), proposed by Schnell et al. [20] which uses a similarity threshold to classify record pairs encoded into BFs as matches and non-matches. (2) For the second baseline approach (named as \( DP-BF \)), we followed the BLIP approach [17], where we added differentially private noise into BFs before the linkage based on (2). (3) We used the recently proposed autoencoder-based BF technique approach [18] (named as \( AE-BF \)) as the third baseline. This technique trains
a mapping function based on autoencoders that transform the BF encodings of one DO into the vector space of the other DO. This transformation guarantees the comparability of numerical vectors. (4) We applied the same deep learning technique in the classification step of the first baseline approach BF to make the fourth baseline, named DL-BF. In this approach, we did not include any DP noise in BF encodings.

3) Parameter Values: For DL-BF and DP-DL-BF approaches, we used three hidden layers with 21, 42, and 84 neurons, respectively. We adopted an Adam optimiser with a learning rate of 0.002. We conducted 50 epochs with a batch size of 5 in the training step. In the training step, we randomly sampled 50% of records from the training data for each database owner to train their local models. Table IV shows the similarity and distance functions we used as the list of feature functions, \( F \), in Algorithms 1 and 2, where these functions are commonly used in PPR and pattern matching for binary data \( [1, 2, 33] \).

Following \([17, 27]\), we set bit flip probability \( p = [0.01, 0.05, 0.1] \) in Algorithms 1 and 2 to ensure the same amount of DP noise is added in BFs and set the number of hash functions \( k = [10, 20, 30] \). We used the q-gram length \( q = 2 \), the BF length \( l = 1,000 \) bits, and double hashing \([11]\) to encode q-grams into BFs. For the AE-BF approach, we set the parameters according to the authors’ suggestions and set the encoding dimension to 256. To select the best similarity threshold value in the classification of BF, DP-BF, and AE-BF, we conducted a parameter sensitivity analysis. Following \([20]\), in this analysis we set the threshold ranging from 0.1 to 1.0, in 0.1 steps and measured precision and recall values for each threshold value. As shown in Fig. 6, we set the similarity threshold to 0.7 as it provides the highest F-measure value for BF, DP-BF, and AE-BF. We adopted a Hamming distance-based locality sensitive hashing technique \([11]\) for blocking.

4) Experiment Environments: We implemented BF, DB-BF, and our approach in Python 3.7. We used TensorFlow to implement our deep learning model \([35]\). For AE-BF we used the code provided by the authors \([18]\). All experiments were performed on a 64-bit Intel Core i9 chip, with eight cores running 16 threads at speeds of up to 2.4 GHz, along with 64 GB of memory, and running Windows 10. To facilitate repeatability, the data sets and the programs are available from the authors.

B. Results and Discussion

1) Comparison of Runtime: Table V shows the runtime of all approaches on different data sets. As expected, our approach consumes more runtime compared to the BF approach because of the addition of differentially private noise in BFs and the independent training of the local models by each DO. DP-BF and DL-BF need a similar runtime due to the independent noise addition and local model training by each DO, respectively, as classification of unlabelled record pairs by the LU requires a small amount of additional runtime for feature generation for the record pairs generated by the blocking technique. Thus, the use of an efficient blocking technique can reduce the additional overhead of runtime required by the LU to a minimum.

However, we noted for all the data sets AE-BF is the slowest approach compared to all the other approaches. This is because AE-BF conducts independent training of encoders in the encoding phase and the LU computes a mapper model for linking the encoded vectors. The data sets are linked by applying the mapper to an encoded data set of a DO and searching for the nearest neighbours of the output in the encoded data of another DO. This requires additional runtime to identify the matches of the candidate pairs.

2) Linkage Quality: Table VI shows the linkage quality (averaged over all parameters) of all approaches on different data sets. As can be seen, DL-BF achieves the best precision and recall values among all approaches. Our approach, DP-DL-BF achieves a similar linkage quality as DL-BF, but, shows a slightly lower recall. This is because of the noise addition in BFs, which leads to true match record pairs being classified as non-matches in the classification phase. However, BF, DP-BF, and AE-BF resulted in lower linkage quality (25–39% in terms of F- and F*-measures) compared to the deep learning-based approaches, indicating that deep learning models can classify encoded record pairs with higher accuracy than the naïve threshold-based classification technique.

We have noted that AE-BF performs slightly better compared to the BF approach achieving higher precision. However, we noted that when each record contains a larger number of q-grams the AE-BF approach tends to reduce recall as the mapper model fails to correctly identify the matches. However, our approach outperformed BF, DP-BF, and AE-BF by 12–20% in terms of F- and F*-measures for data sets with longer attribute values, such as iTune-Amazon, Music-Brainz, and Amazon-Google.

Further, our approach managed to achieve a high linkage quality (20–30% in terms of F- and F*-measures) even when the data set contains many missing values, such as DBLP-ACM-D, DBLP-Scholar-D, and Music-Brainz) compared to BF, DP-BF, and AE-BF. This shows our approach is robust with linkage quality even when data sets contain missing values and longer attribute values. Moreover, we have seen that the AE-BF method

| Similarity functions | Distance functions |
|----------------------|--------------------|
| Jaccard [1], Dice [1], Cosine [2], Russell-Rao [33], Yule [33], Sokal-Sneath [33], Sokal-Michener [33], Rogers-Tanimoto [33] | Hamming [2], Bray-Curtis [33], Jensen-Shannon [33], Kulinski [33], Minkowski [33], Squared Euclidean [33], Weighted Minkowski [33]|

Fig. 6. Precision and recall values of BF, DP-BF, and AE-BF for the DBLP-ACM-C (left) and NCVR (right) data sets.
TABLE V
AVERAGE RUNTIME RESULTS (IN SECONDS) FOR LINKING THE DIFFERENT DATA SETS

| Method       | DBLP-ACM-C | DBLP-ACM-D | DBLP-Scholar-C | DBLP-Scholar-D | iTune-Amazon | MusicBrainz | Amazon-Google | NCVR | European Cens. |
|--------------|------------|------------|----------------|----------------|--------------|-------------|---------------|------|----------------|
| BF [20]      | 6          | 6          | 70             | 71             | 91           | 16          | 11            | 462  | 58             |
| DP-BF [17]   | 8          | 9          | 75             | 75             | 94           | 18          | 13            | 473  | 73             |
| AE-BF [18]   | 725        | 725        | 1,024          | 1,024          | 2,475        | 946         | 845           | 21,025| 1,556         |
| DL-BF        | 8          | 9          | 77             | 78             | 96           | 18          | 14            | 475  | 75             |
| DP-DL-BF     | 10         | 12         | 92             | 91             | 103          | 25          | 19            | 491  | 82             |

The best f-measure results are shown in bold.

TABLE VI
AVERAGE LINKAGE QUALITY RESULTS (P: PRECISION, R: RECALL, F: F-MEASURE, AND F*: F*-MEASURE) OF DIFFERENT METHODS ON DIFFERENT DATA SETS

| Data set            | BF       | DP-BF    | AE-BF    | DL-BF    | DP-DL-BF |
|---------------------|----------|----------|----------|----------|----------|
|                      | P / R / F / P* | P / R / F / P* | P / R / F / P* | P / R / F / P* | P / R / F / P* |
| DBLP-ACM-C          | 0.94 / 0.60 / 0.82 / 0.69 | 0.94 / 0.60 / 0.82 / 0.77 | 0.94 / 0.84 / 0.88 / 0.79 | 0.96 / 0.82 / 0.88 / 0.79 | 0.96 / 0.79 / 0.86 / 0.75 |
| DBLP-ACM-D          | 0.85 / 0.56 / 0.68 / 0.51 | 0.82 / 0.45 / 0.58 / 0.41 | 0.86 / 0.72 / 0.63 / 0.46 | 0.96 / 0.72 / 0.82 / 0.70 | 0.93 / 0.73 / 0.82 / 0.69 |
| DBLP-Scholar-C      | 0.88 / 0.47 / 0.61 / 0.44 | 0.89 / 0.59 / 0.43 / 0.28 | 0.88 / 0.46 / 0.40 / 0.43 | 0.98 / 0.65 / 0.78 / 0.64 | 0.99 / 0.80 / 0.61 / 0.44 |
| DBLP-Scholar-D      | 0.58 / 0.32 / 0.40 / 0.25 | 0.57 / 0.25 / 0.34 / 0.21 | 0.59 / 0.30 / 0.40 / 0.25 | 0.98 / 0.58 / 0.72 / 0.57 | 0.99 / 0.54 / 0.62 / 0.53 |
| iTune-Amazon        | 0.68 / 0.43 / 0.52 / 0.35 | 0.55 / 0.36 / 0.43 / 0.27 | 0.73 / 0.43 / 0.42 / 0.37 | 0.82 / 0.67 / 0.64 / 0.52 | 0.81 / 0.50 / 0.62 / 0.45 |
| MusicBrainz         | 0.90 / 0.89 / 0.89 / 0.60 | 0.96 / 0.51 / 0.66 / 0.49 | 0.92 / 0.67 / 0.89 / 0.80 | 0.99 / 0.84 / 0.90 / 0.83 | 0.99 / 0.72 / 0.82 / 0.73 |
| Amazon-Google       | 0.68 / 0.66 / 0.67 / 0.50 | 0.55 / 0.46 / 0.50 / 0.33 | 0.68 / 0.60 / 0.63 / 0.47 | 0.85 / 0.69 / 0.76 / 0.61 | 0.85 / 0.63 / 0.72 / 0.57 |
| NCVR                | 0.89 / 0.69 / 0.78 / 0.63 | 0.91 / 0.42 / 0.57 / 0.40 | 0.92 / 0.69 / 0.79 / 0.65 | 0.86 / 0.84 / 0.84 / 0.74 | 0.80 / 0.81 / 0.80 / 0.67 |
| European Cens.      | 0.82 / 0.66 / 0.73 / 0.57 | 0.80 / 0.49 / 0.60 / 0.43 | 0.85 / 0.69 / 0.76 / 0.61 | 0.99 / 0.82 / 0.89 / 0.81 | 0.96 / 0.82 / 0.89 / 0.79 |

is more sensitive to the parameter settings and does not perform well when BFs are encoded with more attribute values.

Further, the results in Table VI show that the addition of DP noise in BFs can render a threshold-based classifier less effective as precision and recall drop by at most 33% and 43%, respectively. This indicates that the use of differential privacy in practical linkage projects with threshold-based classifiers requires careful fine-tuning of BF encoding and threshold parameters to achieve an acceptable level of linkage quality.

3) Resistance to Privacy Attacks: As shown in literature BFs are susceptible to cryptanalysis attacks [1]. The majority of existing attacks on BFs exploit information about the frequencies of bit patterns in sets of BFs and the corresponding q-gram frequencies [9]. In our privacy evaluation, we assume the LU is honest-but-curious [2] and DOs do not collude with the LU. Thus, the LU follows the protocol steps while trying to learn sensitive data of a DO [1], [2].

To evaluate privacy, we used the frequency-based cryptanalysis attack by Christen et al. [9]. This attack aligns frequent BFs and plain text values in a public database to conduct re-identification of the most frequent values encoded in these BFs. We assume this attack is launched by the LU in the classification phase on the BFs that the LU receives from all DOs. To simulate the worst-case scenario, we used the data set of one DO as the plain-text data while the encoded BFs of another DO as the encoded data.

We evaluate the re-identification accuracy in terms of the percentages of (1) correct guesses with 1-to-1 matching (1-1 corr), (2) correct guesses with 1-to-many (1-m corr) matching, (3) wrong guesses (Wrong), and (4) no guesses (No), where these four percentages sum to 100. We considered the re-identification accuracy of the attack based on identifying the 10, 20, 50, and 100 most frequent plain text attribute values from the public data set [9].

Fig. 7 shows the re-identification results for the DBLP-ACM-C (top row), NCVR (middle row), and European Census (bottom row) data sets. As can be seen, without DP noise (p = 0) in BFs (as in BF and DL-BF), the attack identified frequent attribute values correctly for different attribute combinations. For the DBLP-ACM-C data set the attack correctly identified the 4 most frequent author names correctly. For NCVR, the attack correctly identified the 10 most frequent first names correctly while for European Census the attack re-identified the top 20 first names correctly. Further, as our privacy results show, the attack failed to correctly identify the attribute values encoded in BFs when bits are flipped according to DP noise (p > 0). This is because these encoded BFs contain different bit patterns, leading the cryptanalysis attack unable to identify any frequent BFs.

These results suggest that DP-BF and DP-DL-BF have low re-identification risk compared to BF and DL-BF because of differential privacy guarantees provided by DP-BF and DP-DL-BF approaches. By flipping bits in BFs, the attack was not able to recognise the q-grams map to a certain bit position, thus avoiding the 1-to-1 or 1-to-many re-identifications. This is because many of the BFs contain different bit patterns leading each BF not being assigned to a plain text value. Therefore, DL-DP-BF provides adequate privacy against frequency-based privacy attacks, while achieving a high linkage quality as shown in Table VI. It is important to note that, due to the transformation of BFs into numerical vectors, we were unable to run this cryptanalysis attack on AE-BF.

Further, we noted that as we increase the number of attributes the attack did not re-identify plain text values as not enough frequency information is available to identify q-grams that are
encoded in the BFs. Hence, conducting such cryptanalysis attacks upon BFs that are encoded with q-grams from different attributes was unlikely to be successful.

C. Ablation Study

To evaluate the robustness of our approach, we ran experiments with different numbers of hash functions and different multi-party linkage scenarios including 2, 3, 5, 7, and 10 DOs.

Fig. 8 shows the runtime required for the training and classification phases of our approach with different flip probability ($p$), number of hash functions ($k$), and number of DOs. As can be seen, the classification phase consumes more runtime compared to the training phase due to the additional runtime required by the LSH blocking technique. Thus, the efficiency of the classification technique can be improved by using a more efficient blocking technique [1].

The runtime did not change much with different $p$ values. This is because BFs are processed in the same way, according to (2), with different $p$ values in the noise addition step. However, we noted the runtime increases with larger $k$ values as each q-gram in an attribute value needs to be hashed by large number of hash functions. We have seen the runtime required for the training phase increases linearly with an increasing number of DOs while runtime for classification by the LU increases exponentially.

Tables VII and VIII show linkage quality results of our approach with various $p$ and $k$ values. As shown in Table VII, the linkage quality decreases slightly (5–12% in terms of F- and F*-measures) as $p$ grows. This trend can especially be seen in the data sets with longer attribute values, such as DBLP-Scholar.
Music-Brainz, and Amazon-Google. Such longer attribute values result in BFs being filled with more 1-bit values. Thus, a large $p$ value can flip a high number of bits in BFs as we show in Table II leading to a reduction in the recall. As shown in Table VIII, we noted that our approach resulted in similar precision and recall with different $k$. This is because the hashing of q-grams does not affect the training of the classification model in the training phase.

Table IX displays the linkage quality results for different numbers of DOs involved in the linkage process. As can be seen, precision decreases by 2–15% while recall drops by 15–30% when more databases are to be linked. Also, when linking databases with missing values, such as DBLP-ACM-D, and DBLP-Scholar-D, the recall drops only by 5–7% while precision does not drop significantly even with more DOs involved in the linkage process. This suggests that the local model training and aggregation allow our approach to compute a more robust global model that potentially leads to a more accurate classification. However, a larger $p$ value can result in the overall linkage quality decreasing, while the privacy of BFs is increased. Thus, a multi-party linkage project requires careful optimisation of the $p$ value to achieve high linkage quality and privacy.

Limitations and Future Work. Our approach has several limitations. First, our approach assumes the DOs have training data with similar characteristics. However, in real applications it might not be trivial to determine such training data with
similar characteristics for multiple DOs. Thus, we also plan to investigate how transfer learning can be used to address the problem of limited training data in PPRL and aim to investigate how to generate training data in a federated learning setting. Second, we assume a trusted aggregator in the training step to combine the individual models generated by DOs into a global classification model. In future work, we aim to use local DP with an untrustworthy aggregator in our approach to make our approach more feasible in a practical linkage project. Third, the performance of deep learning models changes with different hyperparameter settings. As we have not evaluated our approach against different hyperparameter settings, we left the complexity and performance analysis of local models in our approach for future research.

VII. CONCLUSION

We proposed the first deep learning-based PPRL protocol that can be used to link sensitive databases held by different database owners (DOs). The DOs collaboratively train a deep learning model on deferentially private (DP) Bloom filters (BFs) which can then be used to classify unlabelled BF pairs as matches and non-matches. An empirical evaluation shows that our approach can achieve high linkage quality while providing strong privacy protection against privacy attacks.

REFERENCES

[1] P. Christen, T. Ranbaduge, and R. Schnell, Linking Sensitive Data. Berlin, Germany: Springer, 2020.
[2] D. Vatsalan, P. Christen, and V. Verykios, “A taxonomy of privacy-preserving record linkage techniques,” Inf. Syst., vol. 38, no. 6, pp. 946–969, 2013.
[3] P. Christen, Data Matching. Berlin, Germany: Springer, 2012.
[4] M. Ebraheem et al., “Distributed representations of tuples for entity resolution,” VLDB Endowment, vol. 11, no. 11, pp. 1454–1467, 2018.
[5] N. Kooli, R. Allesiardo, and E. Pigneul, “Deep learning based approach for entity resolution in databases,” in Proc. Asian Conf. Infell. Infell Database Syst., Springer, 2018, pp. 3–12.
[6] W. Xue, D. Vatsalan, W. Hu, and A. Seneviratne, “Sequence data matching and beyond: New privacy-preserving primitives based on bloom filters,” IEEE Trans. Inf. Forensics Secur., vol. 15, pp. 2973–2987, Mar. 2020.
[7] S. Mudgal et al., “Deep learning for entity matching: A design space exploration,” in Proc. Int. Conf. Manage. Data, 2018, pp. 19–34.
[8] D. Vatsalan, Z. Sehlit, P. Christen, and E. Rahm, “Privacy-preserving record linkage for Big Data: Current approaches and research challenges,” in Handbook of Big Data Technologies. Berlin, Germany: Springer, 2017, pp. 851–895.
[9] P. Christen, T. Ranbaduge, D. Vatsalan, and R. Schnell, “Precise and fast cryptanalysis for bloom filter based privacy-preserving record linkage,” IEEE Trans. Knowl. Data Eng., vol. 31, no. 11, pp. 2164–2177, Nov. 2019.
[10] A. Gkoulalas-Divanis, D. Vatsalan, D. Karapistoris, and M. Kantarcıoğlu, “Modern privacy-preserving record linkage techniques: An overview,” IEEE Trans. Inf. Forensics Secur., vol. 16, pp. 4966–4987, Sep. 2021.
[11] D. Karapistoris and V. S. Verykios, “An LSH-based blocking approach with a homomorphic matching technique for privacy-preserving record linkage,” IEEE Trans. Knowl. Data Eng., vol. 27, no. 4, pp. 909–921, Apr. 2015.
[12] V. Di Cicco et al., “Interpreting deep learning models for entity resolution: An experience report using LIME,” in Proc. 2nd Int. Workshop Exploiting Artif. Intell. Techn. Data Manage., 2019, pp. 1–4.
[13] J. Kasai et al., “Low-resource deep entity resolution with transfer and active learning,” 2019, arXiv: 1906.08042.
[14] T. Nórega, C. E. S. Pires, and D. C. Nascimento, “Blockchain-based privacy-preserving record linkage: Enhancing data privacy in an untrusted environment,” Elsevier Inf. Syst., vol. 102, 2021, Art. no. 101826.
[15] P. Christen, R. Schnell, T. Ranbaduge, and A. Vidanage, “A critique and attack on “blockchain-based privacy-preserving record linkage,”” Inf. Syst., vol. 108, 2022, Art. no. 101930.
[16] M. Alaggaan, S. Gamb, and A. Kermarrec, “BLIP: Non-interactive differentially-private similarity computation on bloom filters,” in Proc. Symp. Self-Stabilizing Syst., Springer, 2012, pp. 202–216.
[17] R. Schnell and C. Borges, “Randomized response and balanced Bloom filters for privacy preserving record linkage,” in Proc. IEEE 16th Int. Conf. Data Mining Workshops, 2016, pp. 218–224.
[18] V. Christen, T. Hantschel, P. Christen, and E. Rahm, “Privacy-preserving record linkage using autoencoders,” Int. J. Data Sci. Analytics, vol. 15, pp. 347–357, 2023.
[19] M. Abadi et al., “Deep learning with differential privacy,” in Proc. ACM SIGSAC Conf. Comput. Commun. Secur., 2016, pp. 308–318.
[20] R. Schnell, T. Bachteler, and J. Reiher, “Privacy-preserving record linkage using Bloom filters,” BMC Med. Inform. Decis. Mak., vol. 9, no. 1, 2009, Art. no. 41.
[21] M. Alaggaan, “Space/time trade-offs in hash coding with allowable errors,” Commun. ACM, vol. 13, no. 7, pp. 422–426, 1970.
[22] S. Vaiwarsi, T. Ranbaduge, and P. Christen, “Accurate and efficient privacy-preserving string matching,” Int. J. Data Sci. Analytics, vol. 14, pp. 191–215, 2022.
[23] D. Vatsalan and P. Christen, “Scalable privacy-preserving record linkage for multiple databases,” in Proc. 23rd ACM Int. Conf. Infell. Knowl. Manage., Shanghai, 2014, pp. 1795–1798.
[24] P. Christen, A. Vidanage, T. Ranbaduge, and R. Schnell, “Pattern-mining based cryptanalysis of Bloom filters for privacy-preserving record linkage,” in Proc. Pacific-Asia Conf. Knowl. Discov. Data Mining, 2018, pp. 530–542.
[25] C. Dwork et al., “The algorithmic foundations of differential privacy,” Found. Trends Comput. Sci., vol. 9, no. 3/4, pp. 211–407, 2014.
[26] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor, “Our data, ourselves: Privacy via distributed noise generation,” in Proc. Annu. Int. Conf. Theory Appl. Cryptographic Techn., Springer, 2006, pp. 486–503.
[27] U. Erlingsson, V. Pihur, and A. Korolova, “RAPPOR: Randomized aggregatable privacy-preserving ordinal response,” in Proc. ACM SIGSAC Conf. Comput. Commun. Secur., 2014, pp. 1054–1067.
[28] I. Fellegi and A. Sunter, “A theory for record linkage,” J. Amer. Stat. Assoc., vol. 64, no. 328, pp. 1183–1210, 1969.
[29] N. Kooli, R. Allesiardo, and E. Pigneul, “Deep learning based approach for entity resolution in databases,” in Proc. Asian Conf. Infell. Infell Database Syst., Springer, 2018, pp. 3–12.
[30] P. Kairouz et al., “Advances and open problems in federated learning,” Found. Trends Mach. Learn., vol. 14, no. 1/2, pp. 1–210, 2020.
[31] B. McMahan et al., “Communication-efficient learning of deep networks from decentralized data,” in Proc. Int. Conf. Artif. Intell. Statist., 2017, pp. 1273–1282.
[32] D. G. Leipzigt, “Benchmark datasets for entity resolution,” 2021. [Online]. Available: https://dbs.uni-leipzig.de/research/projects/object-matching/benchmark-datasets-for-entity-resolution
[33] S. Choi, S. Cha, and C. Tappert, “A survey of binary similarity and distance measures,” J. Systems Cybern. Inform., vol. 8, no. 1, pp. 43–48, 2010.
[34] D. J. Hand, P. Christen, and N. Kirzielle, “F*: An interpretable transformation of the F-measure,” Mach. Learn., vol. 110, no. 3, pp. 451–456, 2021.
[35] M. Abadi et al., “TensorFlow: A system for large-scale machine learning,” in Proc. USENIX Symp. Operating Syst. Des. Implementation, 2016, pp. 265–283.
[36] N. Kirzielle, P. Christen, and T. Ranbaduge, “TransER: Homogeneous transfer learning for entity resolution,” in Proc. 25th Int. Conf. Extending Database Technol., 2022, pp. 2:118–2:130.