# CampER: An Effective Framework for Privacy-Aware Deep Entity Resolution


Yuxiang Guo
Zhejiang University
Hangzhou, China
guoyx@zju.edu.cn

Lu Chen
Zhejiang University
Hangzhou, China
luchen@zju.edu.cn

Zhengjie Zhou
Zhejiang University
Ningbo, China
zhengjiezhou@zju.edu.cn

Baihua Zheng
Singapore Management University
Singapore
bhzheng@smu.edu.sg

Ziquan Fang
Zhejiang University
Hangzhou, China
zqfang@zju.edu.cn

Zhikun Zhang
Stanford University
Palo Alto, USA
zhikun@stanford.edu

Yuren Mao
Zhejiang University
Ningbo, China
yuren.mao@zju.edu.cn

Yunjun Gao
Zhejiang University
Hangzhou, China
gaoyj@zju.edu.cn



## ABSTRACT

Entity Resolution (ER) is a fundamental problem in data preparation. Standard deep ER methods have achieved state-of-the-art effectiveness, assuming that relations from different organizations are centrally stored. However, due to privacy concerns, it can be difficult to centralize data in practice, rendering standard deep ER solutions inapplicable. Despite efforts to develop rule-based privacy-preserving ER methods, they often neglect subtle matching mechanisms and have poor effectiveness as a result. To bridge effectiveness and privacy, in this paper, we propose CampER, an effective framework for privacy-aware deep entity resolution. Specifically, we first design a training pair self-generation strategy to overcome the absence of manually labeled data in privacy-aware scenarios. Based on the self-constructed training pairs, we present a collaborative fine-tuning approach to learn the match-aware and uni-space individual tuple embeddings for accurate matching decisions. During the matching decision-making process, we first introduce a cryptographically secure approach to determine matches. Furthermore, we propose an order-preserving perturbation strategy to significantly accelerate the matching computation while guaranteeing the consistency of ER results. Extensive experiments on eight widely-used benchmark datasets demonstrate that CampER not only is comparable with the state-of-the-art standard deep ER solutions in effectiveness, but also preserves privacy.



## CCS CONCEPTS

• **Information systems** → **Entity resolution**.




*KDD '23, August 6–10, 2023, Long Beach, CA, USA*



ACM ISBN 979-8-4007-0103-0/23/08…$15.00
https://doi.org/10.1145/3580305.3599266

## KEYWORDS

entity resolution, representation learning, similarity measurement

**ACM Reference Format:**
Yuxiang Guo, Lu Chen, Zhengjie Zhou, Baihua Zheng, Ziquan Fang, Zhikun Zhang, Yuren Mao, and Yunjun Gao. 2023. CampER: An Effective Framework for Privacy-Aware Deep Entity Resolution. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23), August 6–10, 2023, Long Beach, CA, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3580305.3599266


## 1 INTRODUCTION

In the era of information explosion, massive data has been increasingly generated and collected by different organizations, urged to be integrated to advance data analysis and decision-making. Entity Resolution (ER), also known as record linkage or data matching, aims to determine whether two tuples from disparate relations refer to the same real-world entity, which is crucial for information integration [6], cross-organizational decision-making [13], etc.

ER has been extensively studied, using specific rules [11, 44], crowd-sourcing [14, 47], etc. Recently, the approaches [10, 18, 28, 35, 48, 54] motivated by deep learning, especially those based on pre-trained language models (PLMs) [28, 48], gain remarkable effectiveness and have achieved new state-of-the-art accuracy on many ER benchmarks. However, all of these standard deep methods are based on a strict and not-so-realistic assumption, i.e., *the two relations involved in an ER task are stored in a centralized setting*. As shown in Figure 1(a), standard deep ER methods require the relation $\mathcal{D}$ from Walmart and $\mathcal{D}'$ from Amazon to be pooled into a central data site, as the *tuple pair* $(t, t')$ with $t \in \mathcal{D}$ and $t' \in \mathcal{D}'$ is essential and necessary. To be more specific, standard deep ER typically comprises two components: (i) Pair-wise Learning (*PL*), which learns a high-quality representation for each cross-organizational *tuple pair* via pair-wise interaction, such as the similarity comparison [10, 35], self-attention computation [18, 28], etc., and (ii) Binary Classification (*BC*), which classifies each *tuple pair* as a match or not based on its representation. Nonetheless, because of data privacy concerns (e.g., GDPR [39]), it is impossible to pool the data

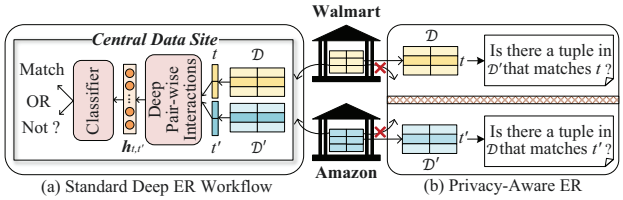
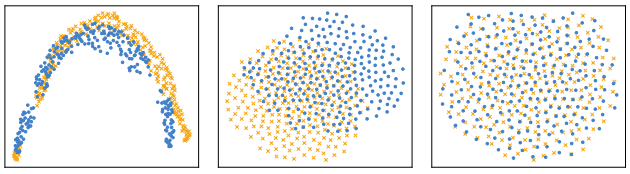



**Figure 1: An example of ER across Walmart and Amazon.**



(a) No fine-tuning (b) Independent fine-tuning (c) Collaborative fine-tuning

**Figure 2: Visualizations of *matchable* tuples from Walmart (blue dots) and Amazon (yellow crosses) with UMAP [31]. Note each visualized tuple from Walmart (resp. Amazon) has a match in Amazon (resp. Walmart), while any two tuples inside Walmart (resp. Amazon) are not matched. Ideally, each blue dot (resp. yellow cross) is close to a yellow cross (resp. blue dot), while yellow crosses (resp. blue dots) are scattered.**

owned by different organizations into one site, which means that the tuple pairs $(t, t')$ cannot be formed, as shown in Figure 1(b). Consequently, all of the above privacy-unaware deep methods are no longer applicable.

To address the lack of privacy consideration in existing ER methods, recent studies [16, 20, 22] have extended the standard rule-based ER solutions by secure techniques (e.g., garbled circuits [53], homomorphic encryption [2], etc.) to securely compute the predefined matching rules without exposing the raw data. While these approaches do offer privacy, they show poor effectiveness, especially on complicated and semantic-rich datasets (to be verified via experiments in Section 4.2). This is because these rule-based solutions are often far less expressive than the deep ER methods. In a nutshell, none of the existing ER solutions can achieve promising ER effectiveness and meanwhile guarantee privacy.

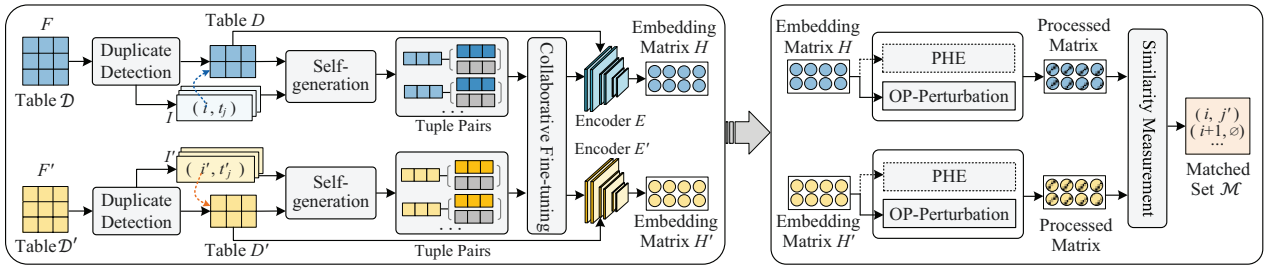
To bridge effectiveness and privacy for ER, in this paper, we aim to introduce a privacy-aware deep ER framework based on PLMs to achieve both promising matching effectiveness and data privacy. This is highly non-trivial, since we need to completely refactor both phases of previous standard deep ER workflow (i.e., *PL* and *BC* mentioned above), to take privacy into consideration: (i) on contrary to standard *PL*, each organization has to learn the *individual* tuple embeddings without any cross-organizational pair-wise interaction, and (ii) instead of making matching decisions via *BC*, two organizations determine the matches through the similarity measurement between their tuple embeddings in a privacy-aware manner. Three key challenges arise, as stated below.

*Challenge I: How to learn the match-aware individual tuple embeddings without manually labeled cross-organizational tuple pairs?* Despite the powerful semantic expression abilities of PLMs, the individual tuple embeddings directly derived from PLMs are not match-aware. For illustration, we plot in Figure 2(a) the tuple embeddings for the Walmart-Amazon dataset when directly adopting pre-trained RoBERTa [30] as the encoder. As we can see, almost all the tuples are mapped into a narrow area to produce high similarity. This hampers the effective matching decision through similarity measurement, as we expect only matched tuple pairs to be close to each other. Hence, each organization needs to fine-tune the PLM to reshape the embedding space in such a way that matched pairs are close to each other while non-matched pairs are far apart. However, in the privacy-aware scenario, there are no labeled pairs available for fine-tuning, since each organization is banned from accessing the tuples from the other organization to form the labeled pairs. To this end, we design a self-generation strategy for each organization to construct training pairs independently, enabling match-aware fine-tuning while avoiding cross-organizational data access.

*Challenge II: How to embed tuples of different organizations into a uni-space in a privacy-aware manner?* As it is difficult to centralize the relations from different organizations, an intuitive approach to guarantee data privacy is that each organization fine-tunes its local PLM independently to learn the tuple embeddings. However, in this way, tuples in different relations are embedded into different vector spaces, which cannot be compared effectively [29]. As we can see in Figure 2(b), the embedding spaces of Walmart and Amazon via independently fine-tuning are not completely aligned, thus misleading the similarity measurement. Although several embedding re-alignment techniques [4, 5] have been proposed, they require the locally learned embeddings to be centrally aggregated for the re-alignment. This has a high risk of privacy leakage, as embeddings can be inverted to partially recover some of the raw inputs [45]. To address this challenge, we design a collaborative fine-tuning strategy with differential privacy, based on the self-generated tuple pairs, to ensure that the PLMs of both organizations are consistent after fine-tuning. Hence, the collectively fine-tuned PLMs embed tuples of both organizations into a uni-space, as shown in Figure 2(c), to support effective similarity measurement.

*Challenge III: How to safely and consistently make the final matching decision through similarity measurement?* After acquiring the match-aware and uni-space tuple embeddings, two organizations should collectively make the final matching decision through similarity measurement. However, raw embeddings can also be partially inverted to the inputs [45], meaning that it is necessary to pre-process the tuple embeddings before similarity measurement to protect privacy. However, similarity computation using the pre-processed embeddings may produce results that are different from the results generated by raw embeddings. To solve this challenge, we first introduce a partially homomorphic encryption-based approach to enable secure and consistent similarity measurement. Although this approach offers a cryptographic security guarantee, it suffers from poor efficiency [16]. To this end, we further present an order-preserving perturbation mechanism to speed up the similarity computation while ensuring that the perturbation has *zero* impact on the final matching decision.

To surmount all the three challenges, we present an effective framework for privacy-aware deep entity resolution, termed CampER, which learns the match-aware and uni-space tuple embeddings for each organization with *zero* manually labeled tuple pairs, and enables two organizations to make final matching decisions via privacy-aware similarity measurement without exposing each one's raw embeddings. Our key contributions are as follows:




(a) Collaborative Match-Aware Representation Learning (b) Privacy-Aware Similarity Measurement

**Figure 3: Framework of CampER. (a) Collaborative Match-Aware Representation Learning enables the match-aware representations of tuples in table $D$ and table $D'$, and makes sure that they are in a uni-space. (b) Privacy-Aware Similarity Measurement allows the two organizations to find the matched pairs through the embedding matrices with privacy-preserving.**

- *Effective ER framework.* We propose CampER [1], an effective entity resolution framework. To the best of our knowledge, it is the first deep-learning framework for privacy-aware ER, which completely subverts the convention of standard deep ER methods.
- *Collaborative representation learning.* We present i) a training pair self-generation strategy, which enables each organization to construct matches and non-matches by using only its own data, and ii) a collaborative fine-tuning approach with designed hard negative sampling, which learns the match-aware and uni-space tuple representations for effective similarity measurement.
- *Privacy-aware similarity measurement.* We introduce a cryptographically secure similarity measurement for the final matching decision. Moreover, we propose an order-preserving perturbation strategy to speed up the matching decision process, and prove our perturbation mechanism has *zero* impact on ER results.
- *Extensive experiments.* We conduct extensive experiments on eight widely-used benchmark datasets. The results demonstrate the superiority of CampER with significant effectiveness over existing private and even standard (non-private) ER solutions and notable privacy considerations that are missing from all the non-private ER solutions.

## 2 PROBLEM FORMULATION

Let $\mathcal{D}$ and $\mathcal{D}'$ be two relational tables owned by different organizations $F$ and $F'$ respectively. We assume that these tables have the same schema $\mathcal{D}(A_1, A_2, \ldots, A_n)$ and $\mathcal{D}'(A_1, A_2, \ldots, A_n)$. If not, schemata of $\mathcal{D}$ and $\mathcal{D}'$ can be matched using private schema matching techniques [41]. Each tuple $t_i \in \mathcal{D}(resp.t'_j \in \mathcal{D}')$ is attached to a unique ID $i$ (resp. $j$ ) and refers to an entity in the real world.

Definition 1. *(Privacy-Aware Entity Resolution). Given two relational tables $\mathcal{D}$ and $\mathcal{D}'$ owned by organizations F and F' respectively, we assume F and F' are semi-honest [20], i.e., they follow the protocol honestly but are also curious about each other's data. Privacy-Aware Entity Resolution is a collaborative procedure for F and F' to discover the matched ID set $\mathcal{M} = \left\{(i, j) \mid t_i \in \mathcal{D} \wedge t'_j \in \mathcal{D}' \wedge m(i, j) = 1\right\}$, where function $m(i, j)$ outputs 1 if tuples $t_i$ and $t'_j$ refer to the same real-world entity, or 0 otherwise. During the collaborative procedure, each of the organizations, say F, is not willing to expose the tuple in his/her table $\mathcal{D}$ to the other organization, say F', if that tuple's ID does not appear in the matched ID set $\mathcal{M}$.*

The key notion in Definition 1 that is different from standard ER is that the tuples in relations $\mathcal{D}$ and $\mathcal{D}'$ should be locally stored. In addition, for each organization, *the tuples not in the output matched set $\mathcal{M}$* are the unique assets that should be protected [16]. Therefore, the existing deep ER workflow based on cross-organizational tuple pairs is no longer applicable in this realistic scenario. Although the rule-based ER solutions can be extended by secure techniques, they show poor effectiveness, as mentioned in Section 1.

## 3 CAMPER FRAMEWORK

In this section, we present an effective framework, termed CampER, for privacy-aware deep entity resolution. We start with an overview of the framework, followed by details on its two components.

### 3.1 Overview of CampER

As shown in Figure 3, CampER comprises two major components: *Collaborative Match-Aware Representation Learning* (CMRL), and *Privacy-Aware Similarity Measurement* (PASM). Given two tables $\mathcal{D}$ and $\mathcal{D}'$ owned by $F$ and $F'$ respectively, $F$ and $F'$ use the same PLM $E$ and $E'$ as their initial local encoders.

(i) ***Collaborative Match-Aware Representation Learning.*** First, each organization, say $F$, detects the duplicates in his/her table $\mathcal{D}$, and obtains a deduplicated version $D$ and a *dup-link set* $I$ which will be defined in Section 3.2.1. Then, the two organizations construct matched/non-matched pairs via the training pair self-generation strategy. Thereafter, they collaboratively fine-tune their respective local encoder using the generated pairs. During this procedure, they iteratively perform local fine-tuning (Section 3.2.3) and collaborative parameter exchange (Section 3.2.4).

(ii) ***Privacy-Aware Similarity Measurement.*** After the collaborative representation learning, the tuple embedding matrices $\boldsymbol{H}$ and $\boldsymbol{H'}$ are generated by the fine-tuned encoders $E$ and $E'$, respectively. The raw embeddings are pre-processed using either the cryptographically secure solution which is based on partially homomorphic encryption (PHE), or the proposed order-preserving (OP)-perturbation strategy. After that, $F$ and $F'$ communicate with each other based on the processed matrices for similarity measurement. Finally, the matched $ID$ set $\mathcal{M}$ is obtained.

[1] Source code is available at https://github.com/ZJU-DAILY/CampER

## 3.2 Collaborative Match-Aware Representation Learning

Unlike the previous standard ER solutions [10, 28, 35, 46] that learn the representations of cross-organizational tuple pairs, CMRL is proposed to learn the match-aware and uni-space embeddings of individual tuples for each organization.

*3.2.1* ***Duplicate Detection.*** We introduce the duplicate detection to CMRL as the first step, and propose a novel data structure called *dup-link set* to maintain the detection results. Specifically, two organizations detect the duplicates (i.e., tuples referring to the same real-word entities [25]) in their respective dataset. For clarity, we take the organization $F$ as an example. $F$ examines his/her original table $\mathcal{D}$ to find all the duplicates. If $t_i \in \mathcal{D}$ and $t_j \in \mathcal{D}$ are duplicates, $F$ removes one of the two tuples ($t_j$ for example) from $\mathcal{D}$, and adds the tuple $t_j$ and the *ID* of $t_j$'s duplicate tuple $t_i$ into the *dup-link set*. The *dup-link set* $I$ is defined as:

$$I = \{(i, t_j) \mid t_i \in D \wedge t_j \in (\mathcal{D} - D) \wedge m(i, j) = 1\} \tag{1}$$

where $D$ is the deduplicated version of $\mathcal{D}$, and $m(i, j)$ is defined in Definition 1. Here, $i$ is an index linked to $t_i \in D$, indicating that the tuples $t_i$ and $t_j$ are duplicates. Note that, $I$ is an empty set if $\mathcal{D}$ is originally duplicate-free. Since duplicate detection can be regarded as the self-ER on the table $\mathcal{D}$ that does not incur any privacy concern, any standard ER solutions mentioned in Section 1 can be easily applied to obtain high accuracy of the detection result. CMRL treats the duplicate detection as a black box, as it is not the focus of this paper. Note that, it is tolerable to have a small number of duplicates remaining in $D$, but $F$ can achieve a perfect precision of duplicate detection by human verification.

The advantages of introducing duplicate detection to CMRL are two-fold. First, it is beneficial to the self-generation strategy to be presented next. Second, it guarantees one-to-one matches, which is helpful for the final matching decision-making (Section 3.3.1).

*3.2.2* ***Self-generation.*** In order to produce match-aware tuple representations, it is necessary to provide the matched and non-matched pairs as training samples. However, it is impossible to get the labeled pairs $\mathbb{L} = \{(t, t') | t \in D, t' \in D'\}$ in the privacy-aware scenarios, as mentioned in Section 1. As a solution, we propose a self-generation strategy that allows each organization to construct the training pairs independently to avoid manual labeling across different organizations. For simplicity, we take $F$ as an example.

**Matched pair generation.** We first detail the matched pair self-generation strategy. For each specific tuple $t_i \in D$ (call it anchor tuple), the organization $F$ tries to generate a matched tuple pair $(t_i, t_i^m)$ by itself. An intuitive idea is to apply data augmentation techniques to $t_i$ to form the augmented tuple $t_i^a$ as the matched tuple $t_i^m$. However, it is hard to guarantee that $t_i^a$ is a truly matched tuple of $t_i$ [34], as the augmentation operators may distort the tuple $t_i$ so much that the augmented tuple $t_i^a$ no longer refer to the same entity as $t_i$. That would degrade the accuracy of ER through our preliminary experimental verification. To this end, we design a safe strategy that will not introduce any false matched pairs.

We first propose a mirror-based matched pair generation strategy (MMG). Specifically, $F$ generates the matched tuple by pairing the anchor tuple $t_i \in D$ with its mirror tuple $t_i$, i.e., $t_i$ itself is a matched tuple of $t_i$. Recall that, tuples refer to the same real-world entity are matched. Since $t_i$ and its mirror tuple are guaranteed to refer to the same entity, the MMG is safe.

Although MMG would not introduce any false matched pairs, the information contained in such pairs is rather limited. Hence, we further present a duplicate-based matched pair generation strategy (DMG). $F$ can generate the matched pair via the newly proposed data structure *dup-link set* $I$. Specifically, for each $t_i \in D$, if $\exists (i, t_j) \in I$, then $t_i$ and $t_j$ are duplicates and $(t_i, t_j)$ form a matched pair. Thus, the matched tuple $t_i^m$ for the anchor $t_i$ is generated as:

$$t_i^m = \begin{cases} t_i & \text{if } \nexists (i, t_j) \in I \\ t_j & \text{if } \exists (i, t_j) \in I \end{cases} \tag{2}$$

We want to highlight that DMG presented above is safe, as the duplicates by definition refer to the same real-world entity. As mentioned before, MMG is also safe. In a word, matched pair generation will not introduce any false-matched pairs.

**Non-matched pair generation.** Next, we explain how to generate the non-matched pairs. For each anchor tuple $t_i \in D$, $F$ tries to generate non-matched pairs $(t_i, t_i^u)$ by itself. Recall that, the table $D$ has been deduplicated, which means that every two tuples in $D$ refer to different entities. Thus, $F$ regards each $t_j \in D$ ($j \neq i$) as a non-match tuple of $t_i$. Accordingly, for each anchor tuple $t_i \in D$, $F$ can generate a set of non-matched tuples, in the form of non-matched set $\mathcal{N}_i^{\mathbf{u}}$ as:

$$\mathcal{N}_i^{\mathbf{u}} = \{t_{ik}^u \mid t_{ik}^u \in D / \{t_i\}\} \tag{3}$$

**Discussion.** As mentioned before, it is possible that the table $D$ still has a few duplicates, since we do not assume the recall of deduplication is 1. We analyze the impact of duplicates in $D$ on the effectiveness of our approach in the experiments (see Section 4.5). We want to highlight that, even with a few duplicates, our non-matched pair generation approach is still effective, as there is a candidate set of non-matched tuples for each anchor tuple.

*3.2.3* ***Local Fine-tuning.*** After generating matched/non-matched tuple pairs, $F$ (resp. $F'$) fine-tunes the local encoder by pulling together matched pairs and pushing apart non-matched pairs.

We take $F$ as an example. A tuple $t = \{A_i, v_i\}_{i \in [1,n]}$ ($v_i$ is the value of the attribute $A_i$) is serialized to a sentence as:

$$S(t) ::= [COL] A_1 [VAL] v_1 \ldots [COL] A_n [VAL] v_n \tag{4}$$

where [COL] and [VAL] are special tokens. The local PLM takes each sentence $S(t)$ as input and allocates a vector $\boldsymbol{u} \in \mathbb{R}^d$ to each token in the sentence $S(t)$, where $d$ is the dimension of the output vector. Following previous studies [40], we apply mean-pooling to obtain the embedding $\boldsymbol{h} \in \mathbb{R}^d$ for each tuple $t$.

Given a tuple $t_i$ and its embedding $\boldsymbol{h}_i$ output by the PLM, each organization adapts the contrastive loss [15] and minimizes the loss to increase the similarity between $\boldsymbol{h}_i$ and the embedding $\boldsymbol{h}_i^m$ of its matched tuple $t_i^m$, and to decrease the similarity between $\boldsymbol{h}_i$ and each embedding $\boldsymbol{h}_{ik}^u$ of its non-matched tuple $t_{ik}^u$ sampled from the non-matched set $\mathcal{N}_i^{\mathbf{u}}$. The contrastive loss is formulated below:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}_i = -\frac{1}{N} \sum_{i=1}^{N} \log \frac{e^{\text{sim}(\boldsymbol{h}_i, \boldsymbol{h}_i^m)/\tau}}{e^{\text{sim}(\boldsymbol{h}_i, \boldsymbol{h}_i^m)/\tau} + \sum_k e^{\text{sim}\left(\boldsymbol{h}_i, \boldsymbol{h}_{ik}^u\right)/\tau}} \tag{5}$$

where $N$ is the number of tuples in $D$, $\tau$ is a temperature scaling parameter, and $\text{sim}(\cdot)$ is a similarity measure implemented by dot product. $F$ minimizes $\mathcal{L}$ to perform parameters local updates through the back-propagation. The same goes for $F'$.

We adopt the momentum training strategy [15] to save the GPU memory cost. Specifically, each organization maintains a pair of encoders $E = (E_l, E_m)$, i.e., an online encoder $E_l$ to encode the anchor tuple $t_i$, and a momentum encoder $E_m$ to encode the generated matched and non-matched tuples ($t_i^m$ and $\{t_{ik}^u\}$). The online encoder $E_l$ is updated by gradient-descent, while the momentum encoder $E_m$ is updated by the momentum mechanism via Eq. 6.

$$\boldsymbol{\theta}_m \leftarrow \alpha \cdot \boldsymbol{\theta}_m + (1-\alpha) \cdot \boldsymbol{\theta}_l, \alpha \in [0,1) \tag{6}$$

where $\boldsymbol{\theta}_l$ and $\boldsymbol{\theta}_m$ refer to the parameter of the online encoder $E_l$ and the momentum encoder $E_m$, respectively. Note $E_l$ and $E_m$ share the same initial parameters but differ at the end of fine-tuning, and the online encoder $E_l$ is used to obtain the final tuple embeddings [15].

**Hard negative sampling.** As mentioned before, given an anchor tuple $t_i$, the non-matched (negative) tuples can be simply sampled from the set $\mathcal{N}_i^{\mathbf{u}}$. However, not all negative tuples are informative for fine-tuning. To tackle this, we design a hard negative sampling method to force the encoder to focus on the informative non-matched pairs.

For each anchor tuple $t_i$, the most informative negative tuples are those close to $t_i$ in the current embedding space. Those pairs that are mapped nearby but should be far apart are essential to the fine-tuning. To this end, each organization samples the negative tuples according to the similarity between the embeddings of the anchor tuple $t_i$ and all the other tuples, i.e., only the top-$k$ negative tuples are sampled and their embeddings $\{\boldsymbol{h}_{ik}^u\}_k$ are used in Eq.5.

Definition 2. *(Top-k negative sampling). Given a table D, the current local encoder E, and the similarity measure* $\text{sim}(\cdot)$*, the top-k negative sampling for the anchor tuple $t_i$ returns the k tuples $\{t_i^u\}$ with the highest similarity scores* $\text{sim}(\boldsymbol{h_i}, \boldsymbol{h_i^u})$*, where $\boldsymbol{h_i}$ and $\boldsymbol{h_i^u}$ are the respective embeddings of $t_i$ and $t_i^u$ generated by the current encoder E, and $t_i^u \in N_i^{\mathbf{u}}$.*

Note that, the top-$k$ negative tuples for a specific anchor $t_i$ would be dynamically changed with the fine-tuning process.

*3.2.4* ***Collaborative Strategy.*** The local tuning process reshapes the vector space of each organization, but cannot guarantee the uni-space requirement (see Figure 2(b)). This is because the independent fine-tuning performed by an organization only relies on its local data, resulting in different parameters of each organization's online encoder at the end of fine-tuning. Even for two tuples that are exactly the same, $F$ and $F'$ may obtain different embeddings through their respective encoder.

To achieve the uni-space property, we aim to guarantee the consistency of two organization's encoders at the end of fine-tuning, so that they can embed their tuples in the same way and achieve the uni-space property [37]. Inspired by FedAvg [32], a well-known federated learning algorithm, we propose a collaborative strategy to fine-tune the online encoders $E_l$ of $F$ and $E_l'$ of $F'$ through parameter exchange. Specifically, $F$ and $F'$ perform local updates based on their own data first, and then exchange and average the parameters of $E_l$ and $E_l'$ to unified ones, which will be used as the initial parameters for the subsequent local updates. At the end of collaborative tuning, $E_l$ and $E_l'$ are eventually the same.

This approach enables collaborative fine-tuning without raw data exchange between $F$ and $F'$, but certain private information might still be divulged to some extent by analyzing the parameters exchanged. Since the model parameters would contain the information of the training samples, one organization, say $F$, may perform a differential attack to judge whether a specific tuple $t' \in D'$ has participated in the tuning process [1]. If the tuple $t'$ dose not appear in the matching set, privacy is leaked, according to Definition 1. Therefore, it is necessary to mitigate the impact of a single tuple on the model parameters. To this end, we further require two organizations to clip the local gradients based on their $l_1$-norm with a threshold $\delta$ during local updates, and apply $(0, \lambda)$–Laplacian noise to the local-updated parameters before parameter exchange to achieve $\epsilon$-differential privacy (as defined in Appendix A). Note that, $\epsilon$ is called the privacy budget.

Theorem 1. *The privacy budget for each organization to perform R rounds of collaborative fine-tuning is $\epsilon = R\epsilon_s = \frac{2\eta\delta R}{\lambda}$, where $\epsilon_s$ is the single-round privacy budget, $\eta$ is the learning rate, and R is the collaborative training round.*

Proof. Please refer to Appendix B.1. □

## 3.3 Privacy-Aware Similarity Measurement

Since it has a high risk of privacy leakage for measuring similarity by raw embeddings, we present PASM to enable two organizations to make the matching decision without exposing raw embeddings.

*3.3.1* ***Criterion of Similarity Measurement.*** After the collaborative fine-tuning, $F$ and $F'$ obtain their respective tuple embeddings through the tuned encoders $E_l$ and $E_l'$, respectively. Let $|D| = N$ and $|D'| = M$. We denote the embedding matrices of $D$ and $D'$ as $\boldsymbol{H} \in \mathbb{R}^{N\times d}$ and $\boldsymbol{H'} \in \mathbb{R}^{M\times d}$, respectively. The $i$-th row of $\boldsymbol{H}$ (resp. $j$-th row of $\boldsymbol{H'}$) is the embedding $\boldsymbol{h}_i$ (resp. $\boldsymbol{h}_j'$) of $t_i \in D$ (resp. $t_j' \in D'$). The similarity matrix $\boldsymbol{S} = \boldsymbol{H}\boldsymbol{H'}^T \in \mathbb{R}^{N\times M}$, and each element in $\boldsymbol{S}$ is $s_{ij} = \text{sim}(\boldsymbol{h}_i, \boldsymbol{h}_j')$ ($i \in [1, N]$, $j \in [1, M]$).

Recall that both $D$ and $D'$ have been deduplicated, meaning the final matches are one-to-one matches. Since the tuple embeddings after CMRL are match-aware, if $t_i$ and $t_j'$ are matched, $\boldsymbol{h}_i$ and $\boldsymbol{h}_j'$ should be the most similar to each other. We adopt the bi-directional top-1 criterion to make the final matching decision, formally presented in Eq. 7. Specifically, if tuple $t_{i^*}$ and $t_{j^*}'$ are mutually the most similar tuple of each other, they can be viewed as a match. Note that, the bi-directional comparison can preclude many false matching and has been widely used in previous studies [4, 12].

$$j^* = \underset{1\le j\le M}{\text{argmax}} \left\{s_{i^*j}\right\} \wedge i^* = \underset{1\le i\le N}{\text{argmax}} \left\{s_{ij^*}\right\} \tag{7}$$

*3.3.2* ***Cryptographically Secure Solution.*** As mentioned above, the core idea of similarity measurement is to find the top-1 similar embedding for both organizations in bi-direction. However, neither of the organizations is allowed to send the embedding matrix in cleartext to the other one for measuring the similarity, as the raw embeddings can still result in privacy leakage [45].

To protect privacy, we develop a partially homomorphic encryption (PHE) based similarity measurement. A PHE scheme [2] is a

probabilistic asymmetric encryption scheme that permits additive (additive HE) or multiplicative (multiplicative HE) computation in the space of ciphers. In this paper, we focus on Additive HE and consider the Paillier cryptosystem [36], since CampER adopts dot product to measure the similarity of tuple embeddings, which suits additive HE well. Specifically, given $x_1$ and $x_2$ as two floating-point values, the corresponding ciphertexts are $[\![x_1]\!]$ and $[\![x_2]\!]$. Additive HE has the properties: $[\![x_1]\!] \oplus [\![x_2]\!] = [\![x_1 + x_2]\!]$, and $x_1 \otimes [\![x_2]\!] = [\![x_1 \cdot x_2]\!]$, where $\oplus$ and $\otimes$ are homomorphic addition and scalar multiplication operations of encrypted values respectively. Given a matrix $\boldsymbol{X}$, we denote the ciphertext of $\boldsymbol{X}$ as $[\![\boldsymbol{X}]\!]$, which is obtained by encrypting each element of $\boldsymbol{X}$ by Paillier. The following theorem provides the opportunity for $F$ and $F'$ to perform secure similarity measurement while ensuring the consistency of the computation result.

Theorem 2. *Assume an embedding matrix $\boldsymbol{H} \in \mathbb{R}^{N\times d}$ from $F$, and an embedding matrix $\boldsymbol{H'} \in \mathbb{R}^{M\times d}$ from $F'$. Then $[\![\boldsymbol{S}]\!] = [\![\boldsymbol{H}]\!]\boldsymbol{H'}^T$, where the similarity matrix $\boldsymbol{S} = \boldsymbol{H}\boldsymbol{H'}^T \in \mathbb{R}^{N\times M}$.*

Proof. Please refer to Appendix B.2. □

Based on the theorem presented above, we can now detail the cryptographically secure solution for similarity measurement. Algorithm 1 in Appendix C summarizes the pseudocode. We assume that $F$ is responsible for the key (i.e., the public key $pk$ and private key $sk$) generation by Paillier [36]. After key generation, $F$ encrypts each value in embedding matrix $\boldsymbol{H}$ using the public key $pk$ to obtain $[\![\boldsymbol{H}]\!]$, and sends $[\![\boldsymbol{H}]\!]$ to $F'$ for computation. Once receiving $[\![\boldsymbol{H}]\!]$, $F'$ computes $[\![\boldsymbol{H}]\!]\boldsymbol{H'}^T$, which is equal to the encrypted similarity matrix $[\![\boldsymbol{S}]\!]$, according to Theorem 2. Then, $F'$ sends the $[\![\boldsymbol{S}]\!]$ back to $F$. After receiving $[\![\boldsymbol{S}]\!]$, $F$ decrypts it to $\boldsymbol{S}$ using the private key $sk$, and finds the bi-directional-top-1 $ID$s via Eq. 7. Then, it derives the matching set $\mathcal{M}$ and shares $\mathcal{M}$ with $F'$. During the data exchange, on the one hand, $F'$ obtains $[\![\boldsymbol{H}]\!]$ from $F$ but is not able to get any raw embedding from $[\![\boldsymbol{H}]\!]$, as she/he cannot decrypt $[\![\boldsymbol{H}]\!]$ without the private key $sk$. On the other hand, $F$ acquires the similarity matrix $\boldsymbol{S}$ by the private key $sk$, but cannot infer $\boldsymbol{H'}$, since each element of $\boldsymbol{S}$ $\text{sim}(\boldsymbol{h}, \boldsymbol{h'}) = \boldsymbol{h} \cdot \boldsymbol{h'} = u_1u'_1 + u_2u'_2 + \cdots + u_du'_d$ has $d$ unknown values (i.e., $u'_1, u'_2, \ldots, u'_d$). In addition, the matching result $\mathcal{M}$ obtained through this way is the same as that obtained by computation on raw embeddings $\boldsymbol{H}$ and $\boldsymbol{H'}$.

*3.3.3* ***Order-Preserving Perturbation Strategy.*** Although the PHE-based approach guarantees cryptographic security, it is highly inefficient and impractical in real applications [16]. Adding perturbation to the embeddings, however, is a commonly used efficient mechanism with experimentally verified defense ability [56]. Hence, we turn to perturb the raw embedding matrix with injected noise to offer a trade-off between efficiency and privacy. However, the noise injected into the raw embeddings may change the results of similarity measurement, resulting in matching decisions that are different from those derived from the raw embeddings. To tackle this, we propose an order-preserving perturbation strategy to guarantee zero impact on the matching results.

Specifically, given an embedding matrix $\boldsymbol{H'} \in \mathbb{R}^{M\times d}$ , we propose to generate the perturbed embedding matrix $\widetilde{\boldsymbol{H'}}$ as follows:

$$\widetilde{\boldsymbol{H'}} = \boldsymbol{H'} + \boldsymbol{r'} \bigotimes c_M \tag{8}$$

where $\boldsymbol{r'} \in \mathbb{R}^{1\times d}$ is a random noise vector ( zero-mean Gaussian noise in our experiment), and the operator $(\cdot \bigotimes c_M)$ produces a noise matrix ($\in \mathbb{R}^{M\times d}$) by repeating the noise vector $\boldsymbol{r'}$ $M$ times. Given another embedding matrix $\boldsymbol{H} \in \mathbb{R}^{N\times d}$, the raw similarity matrix $\boldsymbol{S} = \boldsymbol{H}\boldsymbol{H'}^T$, and the noisy similarity matrix $\widetilde{\boldsymbol{S}} = \boldsymbol{H}\widetilde{\boldsymbol{H'}}^T$, we present the order-preserving property below.

Theorem 3. *For each vector $\boldsymbol{h}_i \in \boldsymbol{H}$, if $\widetilde{\boldsymbol{h'}_j} \in \widetilde{\boldsymbol{H'}}$ is the $k$-th most similar vector to $\boldsymbol{h}_i$ among all the vectors in $\widetilde{\boldsymbol{H'}}$, then $\boldsymbol{h'}_j \in \boldsymbol{H'}$ is also the $k$-th most similar vector to $\boldsymbol{h_i}$ among all the vectors in $\boldsymbol{H'}$.*

Proof. Please refer to Appendix B.3 □

Nevertheless, the order-preserving property is uni-directional, as stated in Theorem 4.

Theorem 4. *Given a vector $\widetilde{\boldsymbol{h'}_j} \in \widetilde{\boldsymbol{H'}}$, although the vector $\boldsymbol{h}_i \in \boldsymbol{H}$ is the $k$-th most similar vector to $\widetilde{\boldsymbol{h'}_j}$ among all the vectors in $\boldsymbol{H}$, it may not be the $k$-th most similar vector to $\boldsymbol{h'}_j$.*

Proof. Please refer to Appendix B.4. □

Based on the proposed order-preserving perturbation mechanism, we present an efficient similarity measurement. Algorithm 2 in Appendix C summarizes the pseudocode. For organization $F'$, $\boldsymbol{H'}$ is perturbed to $\widetilde{\boldsymbol{H'}}$ via Eq. 8 and $\widetilde{\boldsymbol{H'}}$ is sent to $F$. Then, $F$ receives $\widetilde{\boldsymbol{H'}}$ and calculates the noisy similarity matrix $\widetilde{\boldsymbol{S}} = \boldsymbol{H}\widetilde{\boldsymbol{H'}}^T$. Based on $\widetilde{\boldsymbol{S}}$, $F$ computes the $ID$ set $P = \{(i, j)|i \in [1, N], j = \text{argmax}\{\widetilde{s_{ij}}\}\}$ that captures the $ID$s of tuple embeddings that are most similar to each tuple $t_i \in D$. Note that, this result is consistent with that derived from the raw similarity matrix $\widetilde{\boldsymbol{S}}$, owing to the order-preserving property stated in Theorem 3. Since the matching results should be determined by the bi-directional top-1 criterion, while the perturbation strategy is uni-directional order-preserving, $F$ needs to repeat the procedure performed by $F'$ before, i.e., perturbing $\boldsymbol{H}$ and sending the noisy $\widetilde{\boldsymbol{H}}$ to $F'$. Also, the set $P$ is sent to $F'$. Thereafter, $F'$ calculates the noisy similarity matrix $\widetilde{\boldsymbol{S'}} = \boldsymbol{H'}\widetilde{\boldsymbol{H}}^T$, and obtains the $ID$ set $Q = \{(w, z)|w \in [1, M], z = \text{argmax}\{\widetilde{s'_{wz}}\}\}$. Then, $F'$ derives the final matched $ID$ set $\mathcal{M}$ by $P$ and $Q$: for each $(i, j) \in P$, if $\exists(w, z) \in Q$ that $j = w \wedge i = z$, $F'$ appends $(i, j)$ to the matching set $\mathcal{M}$. Finally, $\mathcal{M}$ is shared with $F$.

**Discussion.** The operator of adding noise is much more efficient than the modular exponentiations that are required in PHE-based solution [43]. Furthermore, our perturbation strategy has *zero* impact on the matching result due to its order-preserving property. Although the perturbation strategy would sacrifice certain privacy compared with the cryptographically secure method, we will verify its effectiveness in mitigating advanced inversion attacks [45] even under the white-box setting in Section 4.4. While the experimentally verified defense capability of perturbation-based methods is widely acknowledged, formulating formal privacy guarantees still poses an unsolved challenge [56], which we leave as future work.

## 4 EXPERIMENTS

In this section, we conduct comprehensive experiments to verify the effectiveness and privacy of CampER. We also report the runtime of all approaches and test the scalability of CampER, while the results are presented in Appendix E due to space limitation.

**Table 1: Statistics of datasets used in experiments.**

| Type | Dataset | #Tuple | #Match | #Tuple* | #Match* |
|---|---|---|---|---|---|
| Structured | DBLP-ACM (DA) | 2,471-2,260 | 2,220 | 2,471-2,260 | 2,220 |
| | Amazon-Google (AG) | 1,295-2,150 | 1,167 | 1,288-1,986 | 995 |
| | Walmart-Amzaon (WA) | 1,688-5,249 | 962 | 1,682-5,139 | 847 |
| | DBLP-Scholar (DS) | 2,576-10,694 | 5,347 | 2,504-7,818 | 2,379 |
| | Fodors-Zagats (FZ) | 293-238 | 110 | 293-238 | 110 |
| Dirty | DBLP-ACM (DA-d) | 2,471-2,260 | 2,220 | 2,471-2,260 | 2,220 |
| | DBLP-Scholar (DS-d) | 2,576-10,694 | 5,347 | 2,504-7,818 | 2,379 |
| | Walmart-Amzaon (WA-d) | 1,688-5,249 | 962 | 1,682-5,139 | 847 |

[1]"#Match*" ("#Tuple*") denotes # of matches (tuples) after deduplication.

## 4.1 Experimental Setup

**Datasets.** We conduct experiments on eight widely-used ER benchmark datasets [35]. Since the datasets are partially labeled in pairs for the standard ER task, many tuples are not included in the ground truth and cannot be evaluated. To be fair, we filter out these tuples not included in the ground truth to form new tables for each dataset, with statistics summarized in Table 1.

**Competitors.** The following competitors are evaluated.

- **BF** [42] is a privacy-preserving ER solution widely adopted [13]. We follow [42] using the Dice-coefficient based rules to measure the similarity, and report results under two different similarity thresholds 0.7 and 0.8, denoted as BF(7) and BF(8).
- **EmbDI** [4] is an unsupervised non-private ER method which requires the information from both relations to learn the tuple embeddings. We make a private extension named **EmbDI*** that learns the tuple embeddings of one relation at a time.
- **ZeroER** [51] is an unsupervised non-private ER approach. It uses a variant of a Gaussian Mixture Model to learn different distributions of matched and non-matched tuple pairs.
- **Auto-FuzzyJoin (AuoFJ)** [27] is a non-private fuzzy join method without supervision. It automatically programs fuzzy-join that meets the desired precision and meanwhile maximizes the recall.
- **DeepMatcher (DM)** [35] is a supervised non-private ER framework, which consists of three modules: attribute embedding, attribute similarity representation and classifier.
- **Ditto** [28] is the SOTA supervised non-private ER approach that fine-tunes a pre-trained language model with labeled tuple pairs.

Note that, in the evaluation of supervised DM and Ditto, each dataset is split into the training, validation, and test sets by the ratio of 3:1:1, following the previous studies [28, 35]. For fair comparisons with supervised methods, we evaluate CampER on the same test sets, denoted as CampER(t). For fair comparisons with unsupervised methods, we evaluate CampER on the whole datasets, denoted as CampER(w). Unless otherwise specified, we report the results on the whole datasets as the performance of CampER .

**Implementation Details.** The implementation details and settings of CampER and all competitors are presented in Appendix D.

**Duplicate Detection.** Since the duplicate detection is not our focus in this work, we utilize the matched pairs in the ground truth to infer the duplicates, so that we can obtain the deduplicated tables $D$ and $D'$ and the detection results $I$ and $I'$. Specifically, if both $(t_i, t'_j)$ and $(t_i, t'_k)$ are matched pairs in the provided ground truth, then $t'_j$ and $t'_k$ are duplicates. Thus, organizations $F$ and $F'$ focus on the deduplicated datasets $D$ and $D'$, respectively.

**Table 2: Overall ER results (F1-score) of different approaches.**

| | Approaches | DA | AG | WA | DS | FZ | DA-d | DS-d | WA-d |
|---|---|---|---|---|---|---|---|---|---|
| | Non-Private ER | | | | | | | | |
| w/o S | ZeroER | 0.982 | 0.435 | 0.672 | 0.927 | 0.955 | 0.607 | 0.551 | 0.331 |
| w/o S | AutoFJ | 0.968 | 0.434 | 0.655 | 0.858 | 0.857 | 0.898 | 0.780 | 0.419 |
| w/o S | EmbDI | 0.989 | 0.712 | 0.747 | 0.949 | **0.991** | 0.985 | 0.948 | 0.737 |
| w/ S | DM | 0.985 | 0.705 | 0.689 | 0.930 | 1.0 | 0.978 | 0.914 | 0.462 |
| w/ S | Ditto | 0.990 | 0.743 | 0.840 | 0.938 | 1.0 | 0.986 | 0.913 | 0.829 |
| | Private ER | | | | | | | | |
| w/o S | BF(7) | 0.846 | 0.475 | 0.310 | – | 0.911 | 0.734 | – | 0.293 |
| w/o S | BF(8) | 0.927 | 0.315 | 0.187 | – | 0.881 | 0.692 | – | 0.153 |
| w/o S | EmbDI* | 0.152 | 0.077 | 0.023 | 0.055 | 0.056 | 0.133 | 0.043 | 0.022 |
| | **CampER**(w) | **0.990** | **0.734** | **0.832** | **0.950** | 0.977 | **0.987** | **0.952** | **0.832** |
| | **CampER**(t) | 0.989 | 0.739 | 0.829 | 0.964 | 0.977 | 0.991 | 0.966 | 0.826 |

[1] "w/ S" ("w/o S") denotes the approaches with (without) supervision.
[2] For fairness, we compare CampER(w) with unsupervised approaches, and compare CampER(t) with supervised approaches.
[3]The symbol "–" indicates it fails to produce any result after 15 hours.

**Metrics.** Following most ER works [28, 46], we evaluate the ER results using F1-score, which is the harmonic mean of precision $Pre$ and recall $Rec$ computed as $\frac{2(Pre \times Rec)}{(Pre+Rec)}$.

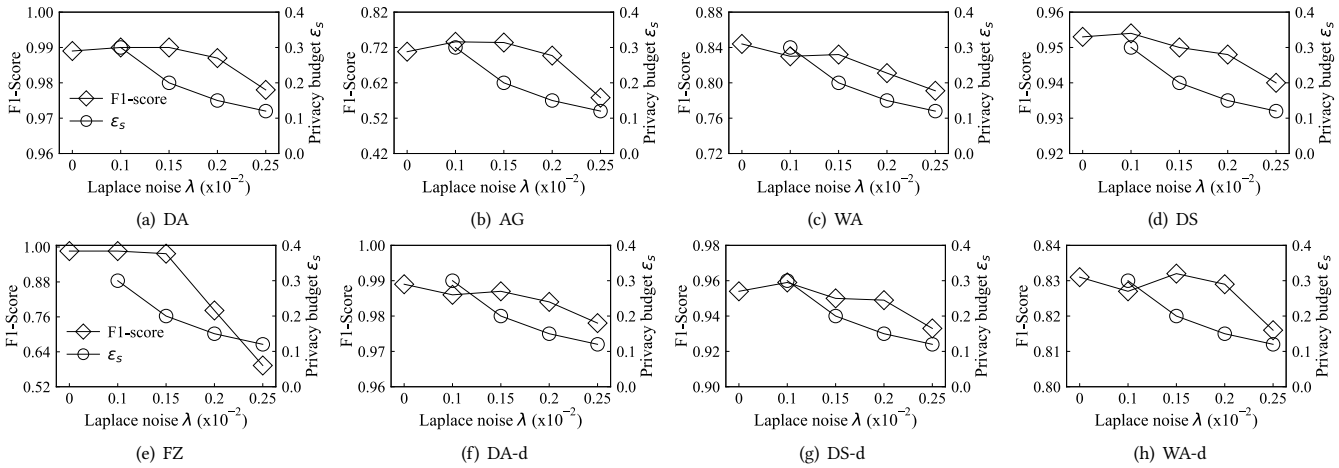
## 4.2 Overall Performance

Table 2 summarizes the overall ER performance by F1-score of CampER and its competitors. Results in **bold** are the best on the whole datasets; underlined results are the best on the test sets.

**CampER vs. Private ER**. It is observed that CampER(w) significantly outperforms BF and yields an average F1-score improvement of more than 2 ×. This is because the rule-based BF is not able to capture the subtle matching mechanisms, especially on the datasets with rich semantics (e.g. AG and WA), as discussed in Section 1. In contrast, CampER is capable of capturing the semantics through fine-tuning the powerful PLMs. We also observe that the similarity threshold of BF has a direct impact on F1-score, and it is not easy to find a proper one (e.g., between similarity thresholds 0.7 and 0.8, DA prefers the latter setting while others prefer the former setting). In contrast, the bi-directional top-1 criterion used by CampER does not require any thresholds. Compared with EmbDI*, CampER(w) has an outstanding improvement in F1-score. The reason is that EmbDI* learns the tuple embeddings of each table independently by random walks, which embeds the tuples in different tables into different spaces. In contrast, CampER achieves uni-space property by CMRL, yielding promising ER results.

**CampER vs. non-private ER.** We first focus on the comparison between CampER(w) and unsupervised (w/o S) methods. It is observed that CampER(w) almost outperforms all these competitors (i.e., ZeroER, AutoFJ, and EmbDI). Especially on the semantic-rich datasets AG, WA, and WA-d, CampER(w) brings nearly 10% improvement on average over the best competitor EmbDI.

We next compare CampER(t) with supervised (w/ S) approaches. The first observation is that CampER(t) outperforms DM by more than 14% on average. This is because the network architecture used by DM is not so expressive in capturing the semantics of words in tuples as the PLM used in CampER. The second observation is that CampER(t) achieves comparable F1-scores with Ditto, and even outperforms it on DS, DA-d, and DS-d. The superiority of




(a) DA (b) AG (c) WA (d) DS

(e) FZ (f) DA-d (g) DS-d (h) WA-d

**Figure 4: F1-score and single-round privacy budget $\epsilon_s$ of CampER with different strengths of noise.**

**Table 3: F1-score of CampER and its variants without the key module. Results in bold are the best.**

| Approaches | DA | AG | WA | DS | FZ | DA-d | DS-d | WA-d |
|---|---|---|---|---|---|---|---|---|
| CampER | **0.990** | **0.734** | **0.832** | **0.950** | 0.977 | **0.987** | **0.950** | **0.832** |
| CampER w/o LF | 0.602 | 0.146 | 0.185 | 0.141 | 0.624 | 0.569 | 0.234 | 0.176 |
| CampER w/o CS | 0.982 | 0.667 | 0.509 | 0.627 | **0.986** | 0.985 | 0.711 | 0.522 |
| CampER w/o DMG | **0.990** | 0.679 | 0.817 | 0.910 | 0.977 | **0.987** | 0.895 | 0.756 |
| CampER w/o HNS | 0.986 | 0.693 | 0.824 | 0.939 | 0.930 | 0.984 | 0.936 | 0.813 |

CampER on DS and DS-d is due to the large number of duplicates in these datasets. These duplicates can be generated as informative matched tuples by DMG for training. We can find that without any supervision and any pair-wise interaction, CampER(t) only falls behind the non-private SOTA by a minimal margin of 2.3% at most.

## 4.3 Ablation Study

We evaluate the effect of different modules on the performance of CampER. The results are listed in Table 3.

**CampER vs. CampER w/o local fine-tuning (LF)**. CampER w/o LF denotes that two organizations obtain the individual tuple embeddings directly from the PLM without any fine-tuning. As observed, the F1 drops more than 64% on average compared with CampER. This is because the naively derived embeddings from the PLMs are crowded in a small space and similar to each other. This justifies the necessity for the match-aware fine-tuning process.

**CampER vs. CampER w/o collaborative strategy (CS)**. CampER w/o CS denotes that two organizations perform LF without collaborative parameter exchange. It is observed that the F1 decreases by nearly 18% on average without CS. This is because CS guarantees the uni-space property of tuple embeddings of different organizations, which is beneficial for effective similarity measurement. Note FZ is an exception, as on easily-matching datasets like FZ, the injected noise during parameter exchange may cancel out the improvements brought by CS.

**CampER vs. CampER w/o duplicate-based matched pair generation (DMG)**. CampER w/o DMG denotes that two organizations only use the mirror-based matched pair generation strategy. Since the datasets DA, FZ, and DA-d are originally duplicate-free, removing the DMG has no impact on their F1. On other datasets, however, we can observe that DMG brings more than 5% F1 improvement. This confirms that the duplicates in each table are safe and informative for the matched pair generation, leading to a higher F1-score.

**CampER vs. CampER w/o hard negative sampling (HNS)**. CampER w/o HNS denotes that CampER selects non-matched tuples randomly from the candidate set $\mathcal{N}_i^{\mathbf{u}}$. We find a drop by at most 5.6% when removing HNS. This is because HNS is able to guide the fine-tuning to focus on the hard pairs, which leads to a higher F1-score under the same number of negative samples.

## 4.4 Privacy Study

We validate the privacy awareness of CampER throughout the whole process of the framework.

**Privacy of CMRL.** In CMRL, two organizations add Laplace noise to the local parameters to achieve the $\epsilon$-differential privacy. We can achieve better privacy protection (i.e., a smaller privacy budget $\epsilon$) by increasing $\lambda$ (i.e., the strength of Laplacian noise). However, strong noise will decrease the effectiveness of the model [50]. Thus, $\lambda$ should be selected based on the trade-off between privacy and model performance. We explore the influence of $\lambda$ on the F1-score and the single-round privacy budget $\epsilon_s$, with results shown in Figure 4. The first observation is that, the difference of the F1 is quite marginal when the noise is relatively small (i.e., between 0 and 0.0015). However, when $\lambda$ increases to 0.002, the F1 begins to drop. The second observation is that, the privacy budget $\epsilon_s$ drops with the growth of $\lambda$, hence a better privacy protection. Based on these results, we set $\lambda$=0.015 to achieve a better privacy protection without sacrificing the effectiveness much. Accordingly, the privacy budget $\epsilon_s$ of each training round is 0.2, and the total privacy budget $\epsilon$ (30 rounds) is 6 according to the composition theorem [7], which is acceptable in common practice [1, 55].

**Privacy of PASM.** Since the PHE-based solution provides cryptographically security [2], we conduct experiments to validate the effectiveness of the proposed OP-Perturbation strategy against the embedding inversion attack. We utilize the advanced embedding attack model [45] to perform the white-box inversion of the tuple embeddings with/without the random perturbation. We vary the strength $\gamma$ of the injected $(0, \gamma)$-Gaussian noise, and evaluate the

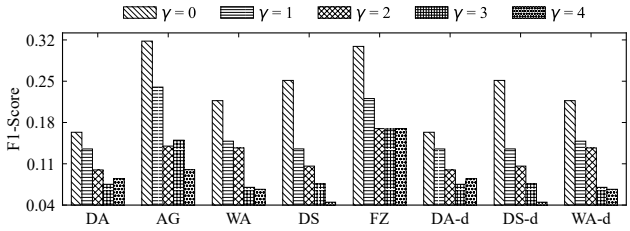
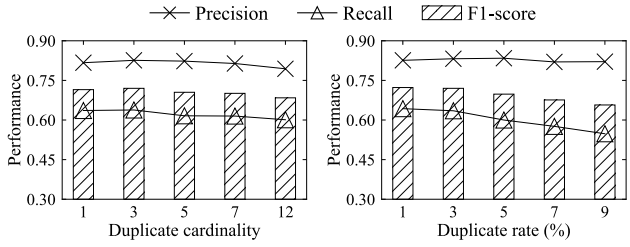



**Figure 5: F1-score of the inversion attack with different $\gamma$.**




(a) AG: Varying $dc$ ($dr = 1$) (b) AG: Varying $dr$ ($dc = 1$)

**Figure 6: Impact of duplicates on the ER performance**

F1-score of the inversion results, as shown in Figure 5. Note that, the higher the F1-score, the more the words in the original tuple are inverted successfully via the noisy tuple embedding, which means poorer privacy. As we can see, if no noise is added to the raw tuple embeddings (i.e., $\gamma = 0$), it has a relatively high probability to be inverted to the raw inputs (e.g. more than 30% on AG and FZ). However, with the noise, the F1 of inversion attack declines significantly, up to 84% on DS. With the further growth of $\gamma$, F1-score tends to keep stable.

## 4.5 Duplicates Impact Analysis

The duplicates remaining in the datasets have impacts on CampER in two stages, i.e., the non-matched pair generation, and the one-to-one matching. We define the duplicate rate $dr$ as the ratio of tuples (termed as d-tuples) that have (one or more) duplicates, and the duplicate cardinality $dc$ as the average number of duplicates owned by each d-tuple. We focus on the AG dataset since the results on other datasets are similar.

First, we duplicate the tuples that do not appear in the matched set, which ensures that the one-to-one matching stage would not be affected. We fix the $dr$ to 1% and vary the duplicate cardinality $dc$. The performance of CampER is shown in Figure 6(a). As we can see, the performance of CampER slightly drops with the growth of $dc$. However, the performance degradation is tolerable. Even when $dc$ increases to 12, which means that half of $k$ ($k = 24$ by default in our experiment) selected non-matched tuples via HNS are false, the F1 still achieves a relatively high value. It shows a certain extent of the robustness of CampER against a few duplicates.

Then, we randomly duplicate the tuples. We fix $dc$ to one, and vary the duplicate rate $dr$. The results are shown in Figure 6(b). The first observation is that the precision of CampER keeps stable when $dr$ is changed, which indicates that the model fine-tuning is not affected when the duplicate cardinality is small. The second observation is that, with the growth of $dr$, the recall has a certain decline. This is because the extra matches introduced by duplicates cannot be found due to the adopted one-to-one matching criterion.

# 5 RELATED WORK

**Entity Resolution.** Entity Resolution (ER) aims to identify whether two tuples from different relations refer to the same real-world entity. Early studies exploit rules [11, 44], crowd-sourcing [14, 47], and machine learning [3, 24] to perform ER. Recently, the deep learning techniques have been widely used in the ER task [10, 18, 26, 28, 35, 46, 48, 54], and have achieved a promising ER accuracy. However, all of these deep methods are privacy-unaware and not applicable to cases where data are locally stored with each organization. Recent study [38] tries to synthesize privacy-preserving ER datasets. However, the ER matcher is designed based on tuple pair and still cannot be applied to the cross-organizational scenarios. In contrast, our CampER performs deep ER and preserves data privacy.

Privacy-Preserving Entity Resolution (Private ER for short) [13] aims to accomplish ER and meanwhile protect privacy. Several avenues of research have been pursued in performing private ER [16, 19–22, 42]. However, all of these methods are based on the predefined matching rules related to specific attributes [13], such as Euclidean distance-based rules [16], q-grams-based rules [42], etc. These solutions show poor effectiveness when handling the data with rich semantics and subtle information, which is a common shortcoming of rule-based approaches. In contrast, CampER is the first attempt to apply deep learning to the privacy-aware entity resolution, which achieves promising effectiveness compared with the existing private ER approaches.

**Federated Learning.** Federated learning [33], a collaborative machine learning technique, has been employed in various fields [17, 50, 52]. Federated learning aims to build models collaboratively by utilizing the locally stored data in a privacy-preserving manner. Inspired by federated learning, CampER designs a collaborative match-aware representation learning approach for two organizations to learn their respective tuple embeddings through parameter exchange. In addition, to prevent certain privacy leakage due to the potential differential attack, we also apply differential privacy techniques to achieve $\epsilon$-DP.

# 6 CONCLUSIONS

In this paper, we propose CampER, an effective framework for privacy-aware deep entity resolution. We first propose a collaborative match-aware representation learning approach, which enables two organizations to learn the match-aware and uni-space tuple embeddings, in order to support the effective similarity measurement. We then present a privacy-aware similarity measurement approach to make the final matching decision. Comprehensive experiments confirm that CampER achieves both promising ER effectiveness and data privacy. In the future, we plan to derive certain privacy guarantees for our order-preserving perturbation strategy, and explore the possibility of designing communication compression methods to reduce the cost of the collaborative fine-tuning phase of CampER to improve efficiency.

# 7 ACKNOWLEDGMENTS

This work was supported in part by the National Key Research and Development Program of China under Grant No. 2021YFC3300303, and the NSFC under Grants No. (62025206, 61972338, and 62102351). Yunjun Gao is the corresponding author of the work.

# REFERENCES


[1] Martín Abadi, Andy Chu, Ian J. Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep Learning with Differential Privacy. In *CCS*. 308–318.

[2] Abbas Acar, Hidayet Aksu, A. Selcuk Uluagac, and Mauro Conti. 2018. A Survey on Homomorphic Encryption Schemes: Theory and Implementation. *ACM Comput. Surv.* 51, 4 (2018), 79:1–79:35.

[3] Mikhail Bilenko and Raymond J. Mooney. 2003. Adaptive duplicate detection using learnable string similarity measures. In *SIGKDD*. 39–48.

[4] Riccardo Cappuzzo, Paolo Papotti, and Saravanan Thirumuruganathan. 2020. Creating embeddings of heterogeneous relational datasets for data integration tasks. In *SIGMOD*. 1335–1349.

[5] Muhao Chen, Yingtao Tian, Mohan Yang, and Carlo Zaniolo. 2017. Multilingual Knowledge Graph Embeddings for Cross-lingual Knowledge Alignment. In *IJCAI*. 1511–1517.

[6] Xin Luna Dong and Theodoros Rekatsinas. 2018. Data Integration and Machine Learning: A Natural Synergy. In *SIGMOD*. 1645–1650.

[7] Cynthia Dwork. 2006. Differential Privacy. In *ICALP*, Vol. 4052. 1–12.

[8] Cynthia Dwork. 2008. Differential Privacy: A Survey of Results. In *TAMC*, Vol. 4978. 1–19.

[9] Cynthia Dwork and Jing Lei. 2009. Differential privacy and robust statistics. In *STOC*. 371–380.

[10] Muhammad Ebraheem, Saravanan Thirumuruganathan, Shafiq R. Joty, Mourad Ouzzani, and Nan Tang. 2018. Distributed Representations of Tuples for Entity Resolution. *PVLDB*. 11, 11 (2018), 1454–1467.

[11] Wenfei Fan, Xibei Jia, Jianzhong Li, and Shuai Ma. 2009. Reasoning about Record Matching Rules. *PVLDB*. 2, 1 (2009), 407–418.

[12] Congcong Ge, Pengfei Wang, Lu Chen, Xiaoze Liu, Baihua Zheng, and Yunjun Gao. 2022. CollaborEM: A Self-supervised Entity Matching Framework Using Multi-features Collaboration. *To appear in TKDE* (2022).

[13] Aris Gkoulalas-Divanis, Dinusha Vatsalan, Dimitrios Karapiperis, and Murat Kantarcioglu. 2021. Modern Privacy-Preserving Record Linkage Techniques: An Overview. *IEEE Trans. Inf. Forensics Secur.* 16 (2021), 4966–4987.

[14] Chaitanya Gokhale, Sanjib Das, AnHai Doan, Jeffrey F. Naughton, Narasimhan Rampalli, Jude W. Shavlik, and Xiaojin Zhu. 2014. Corleone: hands-off crowdsourcing for entity matching. In *SIGMOD*. 601–612.

[15] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. 2020. Momentum Contrast for Unsupervised Visual Representation Learning. In *CVPR*. 9726–9735.

[16] Xi He, Ashwin Machanavajjhala, Cheryl J. Flynn, and Divesh Srivastava. 2017. Composing Differential Privacy and Secure Computation: A Case Study on Scaling Private Record Linkage. In *CCS*. 1389–1406.

[17] Yaochen Hu, Di Niu, Jianming Yang, and Shengping Zhou. 2019. FDML: A Collaborative Machine Learning Framework for Distributed Features. In *SIGKDD*. 2232–2240.

[18] Di Jin, Bunyamin Sisman, Hao Wei, Xin Luna Dong, and Danai Koutra. 2021. Deep Transfer Learning for Multi-source Entity Linkage via Domain Adaptation. *PVLDB*. 15, 3 (2021), 465–477.

[19] Dimitrios Karapiperis, Aris Gkoulalas-Divanis, and Vassilios S. Verykios. 2017. Distance-Aware Encoding of Numerical Values for Privacy-Preserving Record Linkage. In *ICDE*. 135–138.

[20] Dimitrios Karapiperis, Aris Gkoulalas-Divanis, and Vassilios S. Verykios. 2018. FEDERAL: A Framework for Distance-Aware Privacy-Preserving Record Linkage. *TKDE*. 30, 2 (2018), 292–304.

[21] Dimitrios Karapiperis and Vassilios S. Verykios. 2015. An LSH-Based Blocking Approach with a Homomorphic Matching Technique for Privacy-Preserving Record Linkage. *TKDE*. 27, 4 (2015), 909–921.

[22] Basit Khurram and Florian Kerschbaum. 2020. SFour: A Protocol for Cryptographically Secure Record Linkage at Scale. In *ICDE*. 277–288.

[23] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.

[24] Pradap Konda, Sanjib Das, Paul Suganthan G. C., AnHai Doan, Adel Ardalan, Jeffrey R. Ballard, Han Li, Fatemah Panahi, Haojun Zhang, Jeffrey F. Naughton, Shishir Prasad, Ganesh Krishnan, Rohit Deep, and Vijay Raghavendra. 2016. Magellan: Toward Building Entity Matching Management Systems. *PVLDB* 9, 12 (2016), 1197–1208.

[25] Ioannis K. Koumarelas, Thorsten Papenbrock, and Felix Naumann. 2020. MDedup: Duplicate Detection with Matching Dependencies. *PVLDB*. 13, 5 (2020), 712–725.

[26] Bing Li, Yukai Miao, Yaoshu Wang, Yifang Sun, and Wei Wang. 2021. Improving the Efficiency and Effectiveness for BERT-based Entity Resolution. In *AAAI*. 13226–13233.

[27] Peng Li, Xiang Cheng, Xu Chu, Yeye He, and Surajit Chaudhuri. 2021. Auto-FuzzyJoin: Auto-Program Fuzzy Similarity Joins Without Labeled Examples. In *SIGMOD*. 1064–1076.

[28] Yuliang Li, Jinfeng Li, Yoshihiko Suhara, AnHai Doan, and Wang-Chiew Tan. 2020. Deep Entity Matching with Pre-Trained Language Models. *PVLDB*. 14, 1 (2020), 50–60.

[29] Xiao Liu, Haoyun Hong, Xinghao Wang, Zeyi Chen, Evgeny Kharlamov, Yuxiao Dong, and Jie Tang. 2022. SelfKG: Self-Supervised Entity Alignment in Knowledge Graphs. In *WWW*. 860–870.

[30] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *CoRR* abs/1907.11692 (2019).

[31] Leland McInnes, John Healy, and James Melville. 2018. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426* (2018).

[32] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *AISTATS*, Vol. 54. 1273–1282.

[33] H Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Agüera y Arcas. 2016. Federated learning of deep networks using model averaging. *arXiv preprint arXiv:1602.05629* (2016).

[34] Zhengjie Miao, Yuliang Li, and Xiaolan Wang. 2021. Rotom: A Meta-Learned Data Augmentation Framework for Entity Matching, Data Cleaning, Text Classification, and Beyond. In *SIGMOD*. 1303–1316.

[35] Sidharth Mudgal, Han Li, Theodoros Rekatsinas, AnHai Doan, Youngchoon Park, Ganesh Krishnan, Rohit Deep, Esteban Arcaute, and Vijay Raghavendra. 2018. Deep learning for entity matching: A design space exploration. In *SIGMOD*. 19–34.

[36] Pascal Paillier. 1999. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In *EUROCRYPT*, Vol. 1592. 223–238.

[37] Shichao Pei, Lu Yu, Guoxian Yu, and Xiangliang Zhang. 2020. REA: Robust Cross-lingual Entity Alignment Between Knowledge Graphs. In *KDD*. 2175–2184.

[38] Xuedi Qin, Chengliang Chai, Nan Tang, Jian Li, Yuyu Luo, Guoliang Li, and Yaoyu Zhu. 2022. Synthesizing Privacy Preserving Entity Resolution Datasets. In *ICDE*. 2359–2371.

[39] General Data Protection Regulation. 2016. Regulation EU 2016/679 of the European Parliament and of the Council of 27 April 2016. *Official Journal of the European Union* (2016).

[40] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *EMNLP-IJCNLP*. 3980–3990.

[41] Monica Scannapieco, Ilya Figotin, Elisa Bertino, and Ahmed K. Elmagarmid. 2007. Privacy preserving schema and data matching. In *SIGMOD*. 653–664.

[42] Rainer Schnell, Tobias Bachteler, and Jörg Reiher. 2009. Privacy-preserving record linkage using Bloom filters. *BMC Medical Informatics Decis. Mak.* 9 (2009), 41.

[43] Mohsin Shah, Weiming Zhang, Honggang Hu, and Nenghai Yu. 2019. Paillier Cryptosystem based Mean Value Computation for Encrypted Domain Image Processing Operations. *TOMM* 15, 3 (2019), 76:1–76:21.

[44] Rohit Singh, Venkata Vamsikrishna Meduri, Ahmed K. Elmagarmid, Samuel Madden, Paolo Papotti, Jorge-Arnulfo Quiané-Ruiz, Armando Solar-Lezama, and Nan Tang. 2017. Generating Concise Entity Matching Rules. In *SIGMOD*. 1635–1638.

[45] Congzheng Song and Ananth Raghunathan. 2020. Information Leakage in Embedding Models. In *CCS*. 377–390.

[46] Jianhong Tu, Ju Fan, Nan Tang, Peng Wang, Chengliang Chai, Guoliang Li, Ruixue Fan, and Xiaoyong Du. 2022. Domain Adaptation for Deep Entity Resolution. In *SIGMOD*. 443–457.

[47] Jiannan Wang, Tim Kraska, Michael J. Franklin, and Jianhua Feng. 2012. CrowdER: Crowdsourcing Entity Resolution. *PVLDB*. 5, 11 (2012), 1483–1494.

[48] Pengfei Wang, Xiaocan Zeng, Lu Chen, Fan Ye, Yuren Mao, Junhao Zhu, and Yunjun Gao. 2022. PromptEM: Prompt-tuning for Low-resource Generalized Entity Matching. *PVLDB*. 16, 2 (2022), 369–378.

[49] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. HuggingFace's Transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771* (2019).

[50] Chuhan Wu, Fangzhao Wu, Yang Cao, Yongfeng Huang, and Xing Xie. 2021. Fedgnn: Federated graph neural network for privacy-preserving recommendation. *arXiv preprint arXiv:2102.04925* (2021).

[51] Renzhi Wu, Sanya Chaba, Saurabh Sawlani, Xu Chu, and Saravanan Thirumuruganathan. 2020. ZeroER: Entity Resolution using Zero Labeled Examples. In *SIGMOD*. 1149–1164.

[52] Qian Yang, Jianyi Zhang, Weituo Hao, Gregory P. Spell, and Lawrence Carin. 2021. FLOP: Federated Learning on Medical Datasets using Partial Networks. In *SIGKDD*. ACM, 3845–3853.

[53] Andrew Chi-Chih Yao. 1986. How to generate and exchange secrets. In *SFCS*. 162–167.

[54] Dezhong Yao, Yuhong Gu, Gao Cong, Hai Jin, and Xinqiao Lv. 2022. Entity Resolution with Hierarchical Graph Attention Networks. In *SIGMOD*. 429–442.

[55] Lei Yu, Ling Liu, Calton Pu, Mehmet Emre Gursoy, and Stacey Truex. 2019. Differentially Private Model Publishing for Deep Learning. In *S&P*. 332–349.

[56] Zhikun Zhang, Min Chen, Michael Backes, Yun Shen, and Yang Zhang. 2022. Inference Attacks Against Graph Neural Networks. In *USENIX*. 4543–4560.

# A RELATED PRIVACY DEFINITIONS

Differential privacy (DP) was proposed by Dwork for the privacy leakage problem of statistical databases [7], and has become a de-facto standard for data privacy. Under this definition, the results are not sensitive to the changes of an individual tuple, thus providing any individual with plausible deniability that her/his tuple was in the database.

Definition A.1. *($\epsilon$-Differential Privacy). An algorithm $\mathcal{A}$ satisfies $\epsilon$-differential privacy if and only if for any pair of neighboring databases $\mathcal{D}_1, \mathcal{D}_2 \in \mathbb{D}$ that differ in exactly one tuple, any possible output $O$ of $\mathcal{A}$ satisfies the property presented in Equation A.1.*

$$\frac{\Pr[\mathcal{A}(\mathcal{D}_1) = O]}{\Pr[\mathcal{A}(\mathcal{D}_2) = O]} \leq e^{\epsilon} \tag{A.1}$$

Note that, $\epsilon$ is a constant, called the privacy budget. A smaller $\epsilon$ indicates a better privacy protection.

**Laplace mechanism** [8]. Laplace mechanism is a widely-adopted approach for enforcing $\epsilon$-differential privacy, which injects random noise following a Laplace distribution into the query answers. The magnitude of noise is calibrated to the sensitivity of the query that is defined as follows:

$$\Delta f = \max_{\mathcal{D}_1, \mathcal{D}_2} \|q(\mathcal{D}_1) - q(\mathcal{D}_2)\|_1 \tag{A.2}$$

where $\|.\|_1$ refers to the $l_1$-norm, max is to take the maximum over all pairs of neighboring databases $\mathcal{D}_1$ and $\mathcal{D}_1$, and $q(\mathcal{D})$ refers to a query on the relation $\mathcal{D}$.

Given a database $\mathcal{D} \in \mathbb{D}$, a query $q : \mathbb{D} \to \mathbb{R}$, and a privacy budge $\epsilon > 0$, Laplace mechanism $\mathcal{A}$ returns $\mathcal{A}(\mathcal{D})$ to achieve $\epsilon$-differential privacy as follows.

$$\mathcal{A}(\mathcal{D}) = q(\mathcal{D}) + \text{Lap}(0, \epsilon/\Delta f) \tag{A.3}$$

where $\text{Lap}(\mu, b)$ is a Laplace distribution which has the location parameter of $\mu$ and the scale parameter of $b$.

**Composition mechanism** [9]. A charming feature of DP is that it allows a combination of a series of DP algorithms and ensures the composition still satisfies DP. In this work, we only focus on the sequential composition of DP.

Property A.1. *(Sequential Composition). Let each algorithm $\mathcal{A}_i$: $\mathbb{D} \to \mathbb{R}$ satisfies $\epsilon_i$-DP. The sequence of $\{\mathcal{A}_i\}$ satisfies $(\Sigma_i \epsilon_i)$-DP.*

# B PROOF OF THEOREMS

## B.1 Proof of Theorem 1

Proof. Consider two neighboring relations $D_1$ and $D_2$ differing in only one single tuple. Let $\boldsymbol{\theta}_l$ be the initial parameters of $E_l$, $\boldsymbol{\theta}_{l1}$ be the parameters of encoder $E_l$ after a single local training on $D_1$, and $\boldsymbol{\theta}_{l2}$ be the parameters of encoder $E_l$ after a single local training on $D_2$. The sensitivity of $\boldsymbol{\theta}_{l1}$ and $\boldsymbol{\theta}_{l2}$ is $\Delta f = \max|\boldsymbol{\theta}_{l1} - \boldsymbol{\theta}_{l2}| = \max|(\boldsymbol{\theta}_l - \eta \boldsymbol{g}_{c1}) - (\boldsymbol{\theta}_l - \eta \boldsymbol{g}_{c2})| = \max \eta |\boldsymbol{g}_{c1} - \boldsymbol{g}_{c2}| = 2\delta\eta$, where $\delta$ is the clipping threshold. Since each organization adds the zero-mean Laplacian noise of strength $\lambda$ to the parameters after each single local updating, the privacy budget of a single round collaborative training is $\epsilon_s = \frac{\Delta f}{\lambda}$. According to the composition theorem [8], the privacy budget of $R$ rounds of collaborative training is $\epsilon = R\epsilon_s = \frac{R\Delta f}{\lambda} = \frac{2\eta\delta R}{\lambda}$. □

## B.2 Proof of Theorem 2

Proof. To prove Theorem 2, we only need to prove that given two tuple embeddings $\boldsymbol{h} = (u_1, u_2, \ldots, u_d) \in \boldsymbol{H}$ and $\boldsymbol{h'} = (u'_1, u'_2, \ldots, u'_d) \in \boldsymbol{H'}$, the similarity score $\text{sim}(\llbracket \boldsymbol{h} \rrbracket, \boldsymbol{h'}) = \llbracket \text{sim}(\boldsymbol{h}, \boldsymbol{h'}) \rrbracket$.

We denote the encrypted version of the tuple embedding $\boldsymbol{h}$ by Paillier [36] as $\llbracket \boldsymbol{h} \rrbracket = (\llbracket u_1 \rrbracket, \llbracket u_2 \rrbracket, \ldots, \llbracket u_d \rrbracket)$, and the similarity measurement based on $\llbracket \boldsymbol{h} \rrbracket$ and $\boldsymbol{h'}$ is conducted as follows:

$$\begin{aligned} &\text{sim}(\llbracket \boldsymbol{h} \rrbracket, \boldsymbol{h'}) = \llbracket \boldsymbol{h} \rrbracket \cdot \boldsymbol{h'} \\ &= u'_1 \otimes \llbracket u_1 \rrbracket + u'_2 \otimes \llbracket u_2 \rrbracket + \cdots + u'_d \otimes \llbracket u_d \rrbracket \\ &= \llbracket u_1 u'_1 \rrbracket + \llbracket u_2 u'_2 \rrbracket + \ldots \llbracket u_d u'_d \rrbracket \\ &= \llbracket u_1 u'_1 + u_2 u'_2 + \cdots + u_d u'_d \rrbracket = \llbracket \text{sim}(\boldsymbol{h}, \boldsymbol{h'}) \rrbracket \end{aligned} \tag{A.4}$$

□

## B.3 Proof of Theorem 3

Proof. To prove Theorem 3, we only need to prove that, $\forall \boldsymbol{h'_p}, \boldsymbol{h'_q} \in \boldsymbol{H'}$ and their corresponding noisy versions $\widetilde{\boldsymbol{h'_p}}, \widetilde{\boldsymbol{h'_q}} \in \widetilde{\boldsymbol{H'}}$ generated by Equation 8, if $\text{sim}(\boldsymbol{h}_i, \widetilde{\boldsymbol{h'_p}}) > \text{sim}(\boldsymbol{h}_i, \widetilde{\boldsymbol{h'_q}})$, then $\text{sim}(\boldsymbol{h}_i, \boldsymbol{h'_p}) > \text{sim}(\boldsymbol{h}_i, \boldsymbol{h'_q})$. We assume that the noise vector in Equation 8 is $\boldsymbol{r'}$, then we have $\widetilde{\boldsymbol{h'_p}} = \boldsymbol{h'_p} + \boldsymbol{r'}$, and $\widetilde{\boldsymbol{h'_q}} = \boldsymbol{h'_q} + \boldsymbol{r'}$. If $\text{sim}(\boldsymbol{h}_i, \widetilde{\boldsymbol{h'_p}}) > \text{sim}(\boldsymbol{h}_i, \widetilde{\boldsymbol{h'_q}})$, i.e., $\boldsymbol{h}_i \cdot \widetilde{\boldsymbol{h'_p}} > \boldsymbol{h}_i \cdot \widetilde{\boldsymbol{h'_q}}$, then $\text{sim}(\boldsymbol{h}_i, \boldsymbol{h'_p}) = \boldsymbol{h}_i \cdot (\widetilde{\boldsymbol{h'_p}} - \boldsymbol{r'}) > \boldsymbol{h}_i \cdot (\widetilde{\boldsymbol{h'_q}} - \boldsymbol{r'}) = \text{sim}(\boldsymbol{h}_i, \boldsymbol{h'_q})$. □

## B.4 Proof of Theorem 4

Proof. Similar to the proof of Theorem 3, we aim to prove that, $\forall \boldsymbol{h}_p, \boldsymbol{h}_q \in \boldsymbol{H}$, $\text{sim}(\boldsymbol{h}_p, \widetilde{\boldsymbol{h'_j}}) > \text{sim}(\boldsymbol{h}_q, \widetilde{\boldsymbol{h'_j}})$ is not sufficient to guarantee $\text{sim}(\boldsymbol{h}_p, \boldsymbol{h'_j}) > \text{sim}(\boldsymbol{h}_q, \boldsymbol{h'_j})$. Although we have $\text{sim}(\boldsymbol{h}_p, \widetilde{\boldsymbol{h}'_j}) = \boldsymbol{h}_p \cdot (\boldsymbol{h'_j} + \boldsymbol{r'}) > \text{sim}(\boldsymbol{h}_q, \widetilde{\boldsymbol{h}'_j}) = \boldsymbol{h}_q \cdot (\boldsymbol{h'_j} + \boldsymbol{r'})$, we cannot guarantee that $\boldsymbol{h}_p \cdot \boldsymbol{r'} < \boldsymbol{h}_q \cdot \boldsymbol{r'}$, as the noise $\boldsymbol{r'}$ is randomly generated and the embeddings $\boldsymbol{h}_p$ and $\boldsymbol{h}_q$ are any two vectors in $\boldsymbol{H}$. Hence, $\boldsymbol{h}_p \cdot (\boldsymbol{h'_j} + \boldsymbol{r'}) > \boldsymbol{h}_q \cdot (\boldsymbol{h'_j} + \boldsymbol{r'})$ is not sufficient to deduce $\boldsymbol{h}_p \cdot \boldsymbol{h'_j} > \boldsymbol{h}_q \cdot \boldsymbol{h'_j}$. □

# C ALGORITHM

The implementation procedures of the PHE-based similarity measurement is shown in Algorithm 1.

**Algorithm 1:** PHE-based Algorithm

**Organization** $F$:
generate the private key $sk$ and the public key $pk$
$\llbracket H \rrbracket \leftarrow \textbf{Enc}(H, pk)$ //encrypt the embedding matrix $\boldsymbol{H}$ by $pk$
send the encrypted embedding matrix $\llbracket H \rrbracket$ to $F'$
**Organization** $F'$:
receive the encrypted embedding matrix $\llbracket H \rrbracket$ from $F$
$\llbracket S \rrbracket \leftarrow \llbracket H \rrbracket H'^T$
send the encrypted similarity matrix $\llbracket S \rrbracket$ to $F$
**Organization** $F$:
receive the encrypted similarity matrix $\llbracket S \rrbracket$ from $F'$
$S \leftarrow \textbf{Dec}(\llbracket S \rrbracket, sk)$ //decrypt $\llbracket S \rrbracket$ by $sk$
compute the matched $ID$ set $\mathcal{M}$ by $S$ and share with $F'$

The implementation procedures of the proposed perturbation-based similarity measurement is shown in Algorithm 2.

**Algorithm 2:** Perturbation-based Algorithm

**Organization** $F'$:
$\widetilde{\boldsymbol{H}'} \leftarrow$ perturb $\boldsymbol{H}'$ // Equation 8
send the noisy embedding matrix $\widetilde{\boldsymbol{H}'}$ to $F$
**Organization** $F$:
receive the noisy embedding matrix $\widetilde{\boldsymbol{H}'}$ from $F$
$\widetilde{S} \leftarrow \boldsymbol{H}\widetilde{\boldsymbol{H}'}$ //compute the noisy similarity matrix
**foreach** $i \in \{0, 1, ..., N\}$ **do**
  $j \leftarrow \text{argmax}\left\{\widetilde{s_{ij}}\right\}$ // Equation 7
  $P \leftarrow P.append(i, j)$
$\widetilde{\boldsymbol{H}} \leftarrow$ perturb $\boldsymbol{H}$ // Equation 8
send the noisy embedding matrix $\widetilde{\boldsymbol{H}}$ and $P$ to $F'$
**Organization** $F'$:
receive the noisy embedding matrix $\widetilde{\boldsymbol{H}}$ and $P$ from $F'$
$\widetilde{S'} \leftarrow \boldsymbol{H}'\widetilde{\boldsymbol{H}}$ //compute the noisy similarity matrix
**foreach** $w \in \{0, 1, ..., M\}$ **do**
  $z \leftarrow \text{argmax}\left\{\widetilde{s'_{wz}}\right\}$ // Equation 7
  $Q \leftarrow Q.append(w, z)$
**foreach** $(i, j) \in P$ **do**
  **if** *exits* $(w, z) \in Q$ *that* $j = w \wedge i = z$ **then**
    $\mathcal{M}.append(i, j)$
send $\mathcal{M}$ to $F$

# D IMPLEMENTATION DETAILS

We implement CampER in PyTorch and Transformers library [49]. We use RoBERTa [30] as the backbone structure of our model following the previous research [28], and the dimension $d$ of output vector is 768. In all the experiments, the max sequence length is set to 256; the batch size is set to 32; the temperature parameter $\tau$ is set to 0.08; and the momentum coefficient $\alpha$ is set to 0.9999. We use Adam [23] as the optimization algorithm, and set the learning rate $\eta$ to 3e-5. The clipping threshold of gradient matrices is set to 5, and the privacy budget $\epsilon_s$ for a single round is set to 0.2. We varied the number $k$ of negative sampling in the range of {12, 16, 20, 24, 28} on different datasets and found that $k$=24 achieves the best F1-score on almost all the datasets. Therefore, $k$ is set to 24 by default. We use the perturbation-based algorithm during PASM by default, as it is efficient. All experiments were conducted on a server with an Intel(R) Xeon(R) Silver 4216, 2.10GHz CPU, a NVIDIA A100 40G GPU, and 128GB memory. The programs are all implemented in Python.

We report the reproduced results of all competitors by our re-implementation with their publicly available source codes. Next, we detail the settings of competitors. (i)For BF [42], we utilizes the hash functions HMAC-SHA-1 to hash each q-gram of values of selected attributes to a Bloom filter. We select the informative attributes for each datasets. Specifically, *title*, *author*, *venue*, and *year* are selected for datasets DA, DA-d, DS, and DS-d; *title*, *category*, *brand*, *price*, *modelno* are selected for dataset WA and WA-d; *title*, *manufacturer*, and *price* are selected for dataset AG; *name*, *addr*, *city*, and *phone* are selected for dataset FZ. (ii)For EmbDI [4], we use the global approach by pooling two relations and training a common space, and follow all the settings in [4]. We extend EmbDI to EmbDI* by training embeddings one relation at a time, as mentioned in [4]. (iii)For ZeroER [51], we follow all the settings in [51]. (v)For Auto-FuzzyJoin (AuoFJ) [27], we follow the previous research [27] to set the desired threshold of precision $\mu$ to 0.9. Since AuoFJ is a uni-directional join approach, we perform AuoFJ from both directions, and report the average results of bi-directional join. (vi)For DeepMatcher (DM) [35], we report the results of Hybrid strategy, since it performs the best according to [35]. (v)Ditto [28], We use RoBERTa as the PLM and use three optimization operators following the previous work [28].

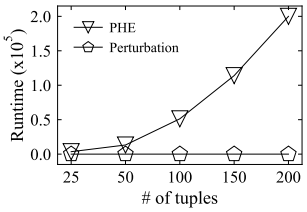



**Figure D.1: Runtime of PASM**

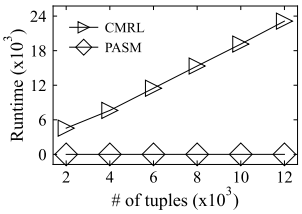



**Figure D.2: Scalability test**

**Table D.1: Runtime of different approaches (in minutes).**

| Approaches | DA | AG | WA | DS | FZ | DA-d | DS-d | WA-d |
|---|---|---|---|---|---|---|---|---|
| CampER | 102.44 | 75.23 | 68.71 | 197.12 | 21.65 | 104.26 | 199.03 | 68.53 |
| BF | 555.74 | 116.91 | 563.79 | – | 2.02 | 511.32 | – | 784.92 |
| EmbDI* | 3.04 | 1.47 | 3.92 | 5.76 | 0.44 | 3.02 | 5.71 | 4.10 |
| ZeroER | 19.16 | 16.11 | 24.43 | 4.52 | 0.67 | 16.99 | 40.17 | 38.46 |
| AutoFJ | 38.41 | 4.91 | 27.87 | 36.19 | 2.03 | 16.18 | 28.35 | 26.67 |
| EmbDI | 2.36 | 1.15 | 3.64 | 5.75 | 0.33 | 2.04 | 5.49 | 3.80 |
| DM | 15.27 | 7.98 | 12.77 | 26.59 | 1.09 | 16.92 | 28.66 | 12.55 |
| Ditto | 17.26 | 8.97 | 11.54 | 23.40 | 1.20 | 15.36 | 23.74 | 12.00 |

[1]The symbol "–" indicates it needs the runtime more than 15 hours.

# E RUNTIME AND SCALABILITY STUDIES

We report the runtime of different approaches in Table D.1. As we can see, CampER incurs shorter time than BF in almost all the datasets. As expected, CampER requires longer runtime than the other standard ER methods, as standard ER does not incur any collaboration overhead. While CampER's runtime is higher than that of non-private methods, the privacy improvements achieved by CampER well justify the overhead. In addition, we show the runtime (in seconds) of different strategies during PASM w.r.t. the size of tuples in Figure D.1. It is observed that PHE-based solution takes nearly 55 hours to perform secure similarity measurements with only 200 tuples. In contrast, the perturbation-based approach only needs several seconds. This confirms that perturbation is much more efficient than the complex modular exponentiations that are required in PHE-based solution. In practice, it is worth choosing the perturbation-based approach to achieve higher computational efficiency. Finally, we verify the scalability of CampER, and report the runtime of two phases (i.e., CMRL and perturbation-based PASM) w.r.t. the number of tuples on DA in Figure D.2. It is observed that the runtime increases linearly to the growth of the dataset size, demonstrating a good scalability.