A Proofs and Analysis of Privacy

A.1 Proof of Differential Privacy (DP)

The privacy risk of FedSim incurred by similarities can also be formalized by a strong metric *differential privacy* [15]. This metric, however, requires more additional noise, thus seriously affecting the accuracy. In the beginning, we introduce some background of traditional Gaussian mechanism (Theorem 3) [15] as $\varepsilon \in (0, 1)$.

Theorem 3. *([15]) For any $(\varepsilon, \delta) \in (0, 1)$, the Gaussian mechanism $M(x) = f(x) + N(0, \sigma^2 I)$ with $\sigma = \Delta\sqrt{2\log(1.25/\delta}$.*

According to [5], the traditional Gaussian mechanism can be extended to Theorem 4 to support $\varepsilon > 1$.

Theorem 4. *([5]) Let $f : \mathbb{X} \to \mathbb{R}^d$ have global L_2 sensitivity Δ. Suppose $\varepsilon > 0$ and $0 < \delta < 1/2 - e^{-3\varepsilon}/\sqrt{2\pi\varepsilon}$. If the mechanism $M(x) = f(x) + N(0, \sigma^2 I)$ is (ε, δ)-DP, then $\sigma \geq \Delta/\sqrt{2\varepsilon}$.*

Theorem 3 suggests that the Gaussian mechanism as $\varepsilon > 1$ (Theorem 3) requires different scale of noise compared to the Gaussian mechanism as $\varepsilon \in (0, 1)$ (Theorem 4). Specifically, to achieve (ε, δ)-DP, Theorem 3 requires $\sigma = \Theta(1/\varepsilon)$, while Theorem 4 requires $\sigma = \Omega(1/\sqrt{\varepsilon})$. Notably, Theorem 4 indicates the smallest scale σ of noise required for (ε, δ)-DP, thus **further implying the lowest ε that can be proved from the given scale of noise ($\sigma \geq \Delta/\sqrt{2\varepsilon}$)**. Focusing on the case of FedSim, given the dataset $\mathbf{k}^S$ to be protected, we restate procedure $\mathcal{G}$ as follows.

Then, we prove the differential privacy of $\mathcal{G}$ in both cases when $\varepsilon \in (0, 1)$ and $\varepsilon > 1$ according to Gaussian mechanisms.

Procedure $\mathcal{G}$: Take $\mathbf{k}^S$ as the input. Each $k_i^P \in \mathbf{k}^P$ is linked to multiple $k_j^S \in \mathbf{k}^S$; the similarities between k_i^P and k_j^S are calculated by $s_{ij} = -\frac{\|k_i^P - k_j^S\|_2 + \mu_0}{\sigma_0} + N(0, \sigma^2)$, where $N(0, \sigma^2)$ refers to Gaussian distribution with variance σ^2. Output the similarities $\mathbf{s}$.

Theorem 5. *Let $\varepsilon, \delta \in (0, 1)$. Suppose the size of $\mathbf{k}^P$ is $n \times \beta$. Procedure $\mathcal{G}$ is (ε, δ)-DP if $\sigma \geq \Delta_{\mathcal{G}}\sqrt{2\ln(1.25/\delta)}/\varepsilon$, where $\Delta_{\mathcal{G}} = n \cdot \max\left\{\left|\frac{1+\mu_0}{\sigma_0}\right|, \left|\frac{-1+\mu_0}{\sigma_0}\right|\right\}$.*

Proof. The change of a single of one Bloom filter on party S may change the distance by 1; therefore, the sensitivity of $\|k_i^P - k_j^S\|_2 = 1$. The sensitivity of a single similarity s_{ij}, which is a linear transformation of $\|k_i^P - k_j^S\|_2$, is

$$\max\left\{\left|\frac{1+\mu_0}{\sigma_0}\right|, \left|\frac{-1+\mu_0}{\sigma_0}\right|\right\} \tag{3}$$

Since each $k_j^S \in \mathbf{k}^S$ might be linked to all n instances in $\mathbf{k}^P$, modifying one instance in $\mathbf{k}^S$ results in the change of at most n similarities. The sensitivity of procedure $\mathcal{G}$ can be derived as

$$\Delta_{\mathcal{G}} = n \cdot \max\left\{\left|\frac{1+\mu_0}{\sigma_0}\right|, \left|\frac{-1+\mu_0}{\sigma_0}\right|\right\} \tag{4}$$

According to Theorem 3, if $\sigma \geq \Delta_{\mathcal{G}}\sqrt{2\ln(1.25/\delta)}/\varepsilon$, procedure $\mathcal{G}$ is differential privacy (ε, δ)-differential privacy according to Gaussian mechanism. □

Theorem 1. *Suppose $\varepsilon > 0$ and $0 < \delta < 1/2 - e^{-3\varepsilon}/\sqrt{2\pi\varepsilon}$. Suppose the size of $\mathbf{k}^P$ is $n \times \beta$. If procedure $\mathcal{G}$ is (ε, δ)-DP, then $\sigma \geq \Delta_{\mathcal{G}}/\sqrt{2\varepsilon}$, where $\Delta_{\mathcal{G}} = n \cdot \max\left\{\left|\frac{1+\mu_0}{\sigma_0}\right|, \left|\frac{-1+\mu_0}{\sigma_0}\right|\right\}$.*

Proof. According to Equation 4, it holds

$$\Delta_{\mathcal{G}} = n \cdot \max\left\{\left|\frac{1+\mu_0}{\sigma_0}\right|, \left|\frac{-1+\mu_0}{\sigma_0}\right|\right\}$$

According to Theorem 4, if $\mathcal{G}$ is (ε, δ)-DP, then $\sigma \geq \Delta_{\mathcal{G}}/\sqrt{2\varepsilon}$. □

A.2 Proof of Theorem 1

Theorem 2. *Given a finite set of perturbed similarities s_{ij} $(i \in Q)$ between $|Q|$ bloom-filters k_i^P $(i \in Q)$ in party P and one Bloom filter k_j^S in party S, if an attacker knows the scaling parameters μ_0, σ_0 and follows the procedure of the attack method, the probability of the attacker's predicted Bloom filter $\hat{k}_j^S$ equaling the real Bloom filter k_j^S is bounded by a constant τ. Formally,*

$$\Pr\left[\hat{k}_j^S = k_j^S \middle| \{s_{ij}|i \in Q\}, \{k_i^P|i \in Q\}, \mu_0, \sigma_0, \mathcal{A}\right] \leq \tau$$

where constant $\tau = \operatorname{erf}\left(\frac{\sqrt{\sigma^2+1}}{2\sqrt{2}\sigma\sigma_0}\right)$; $\operatorname{erf}(\cdot)$ is the error function, i.e., $\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}dt$; event $\mathcal{A}$: attackers follow the given attack method.

Proof. According to the attack method, the attacker can predict based on any $|Q|$ Bloom filters in party P and their corresponding distances. We denote the negative distance between k_i^P and k_j^S as $l_{ij} = -\operatorname{dist}(k_i^P, k_j^S)$. The attacker's predicted value of l_{ij} is denoted as $\hat{l}_{ij}$. Then, we have

$$\begin{aligned}
&\Pr\left[\hat{k}_j^S = k_j^S \middle| \{s_{ij}|i \in Q\}, \{k_i^P|i \in Q\}, \mu_0, \sigma_0, \mathcal{A}\right] \\
=\ &\Pr\left[\hat{k}_j^S = k_j^S, \{\hat{l}_{ij} = l_{ij}|i \in Q\} \middle| \{s_{ij}|i \in Q\}, \{k_i^P|i \in Q\}, \mu_0, \sigma_0, \mathcal{A}\right] \\
=\ &\Pr\left[\hat{k}_j^S = k_j^S \middle| \{\hat{l}_{ij} = l_{ij}|i \in Q\}, \{k_i^P|i \in Q\}, \mu_0, \sigma_0, \mathcal{A}\right] \\
&\cdot \Pr\left[\{\hat{l}_{ij} = l_{ij}|i \in Q\} \middle| \{s_{ij}|i \in Q\}, \{k_i^P|i \in Q\}, \mu_0, \sigma_0, \mathcal{A}\right] \\
=\ &\Pr\left[\hat{k}_j^S = k_j^S \middle| \{\hat{l}_{ij} = l_{ij}|i \in Q\}, \{k_i^P|i \in Q\}, \mu_0, \sigma_0, \mathcal{A}\right] \\
&\cdot \Pr\left[\{\hat{l}_{ij} = l_{ij}|i \in Q\} \middle| \{s_{ij}|i \in Q\}, \mathcal{A}\right] \\
=\ &\Pr\left[\hat{k}_j^S = k_j^S \middle| \{\hat{l}_{ij} = l_{ij}|i \in Q\}, \{k_i^P|i \in Q\}, \mu_0, \sigma_0, \mathcal{A}\right] \\
&\cdot \prod_{i \in Q} \Pr\left[\hat{l}_{ij} = l_{ij} \middle| s_{ij}, \mathcal{A}\right]
\end{aligned} \tag{5}$$

The first equation holds because of the assumption that the attacker predicts k_j^S through the most likely distance $\hat{l}_{ij}$, and correctly predicting the $|Q|$ correct distances is the prerequisite of predicting the correct Bloom filter k_j^S, i.e., $\{\hat{k}_j^S = b_j^S\} \subseteq \{\hat{l}_{ij} = l_{ij}|i \in Q\}$. The second equation holds because of the definition of conditional probability. The third equation holds because the attacker predicts distances independently based on perturbed similarities, indicating that $\{\hat{l}_{ij} = l_{ij}|i \in Q\}$ is **conditionally independent** with μ_0, σ_0 and $\{k_i^P|i \in Q\}$ given $\mathcal{A}$. The fourth equation holds because all the $\hat{l}_{ij}$ $(i \in Q)$ are predicted independently by the attacker from the corresponding s_{ij}, indicating that each $\hat{l}_{ij} = l_{ij}$ is **conditionally independent** with $\{s_{kj}|k \in Q, k \neq i\}$.

According to the attack method, the attacker first predicts the similarity $\hat{\rho}_{ij}$ based on Maximum a Posteriori (MAP) estimation with one experiment s_{ij}. With Bayes' theorem, we have

$$p(\rho_{ij}|s_{ij}, \mathcal{A}) \propto p(s_{ij}|\rho_{ij}, \mathcal{A})p(\rho_{ij}|\mathcal{A}) \tag{6}$$

Since $\rho_{ij} \sim N(0, 1)$ and $s_{ij} \sim N(\rho_{ij}, \sigma^2)$,

$$p(\rho_{ij}|s_{ij}, \mathcal{A}) \sim N\left(\frac{s_{ij}}{\sigma^2+1}, \frac{\sigma^2}{\sigma^2+1}\right) \tag{7}$$

Note that the posterior distribution is also a Gaussian distribution. This property of Gaussian distribution is also known as *conjugate distribution* [38]. With Equation 7, ρ_{ij} is estimated by

$$\hat{\rho_{ij}} = \arg\max_{\rho_{ij}} p(\rho_{ij}|s_{ij}, \mathcal{A}) = \frac{s_{ij}}{\sigma^2+1} \tag{8}$$

Then, $\hat{\rho_{ij}}$ is scaled back to distance $\hat{l_{ij}}'$ with known scaling parameters μ_0, σ_0, i.e.,

$$\hat{l_{ij}}' = \sigma_0\hat{\rho_{ij}} + \mu_0 \tag{9}$$

Since the distances between the Bloom filters are integers, predicting the correct distance, i.e., $\hat{l}_{ij} = l_{ij}$, implies that

$$l_{ij} = \sigma_0 \rho_{ij} + \mu_0, |\hat{l_{ij}}' - l_{ij}| \leq \frac{1}{2} \tag{10}$$

With Equation 8, 9 and 10, we can evaluate $\Pr[\hat{l}_{ij} = l_{ij} | s_{ij}, \mathcal{A}]$ by

$$\begin{aligned} \Pr[\hat{l}_{ij} = l_{ij} | s_{ij}] &= \Pr\left[|\hat{l_{ij}}' - l_{ij}| \leq \frac{1}{2} \middle| s_{ij}, \mathcal{A} \right] \\ &= \Pr\left[\left| \rho_{ij} - \frac{s_{ij}}{\sigma^2 + 1} \right| \leq \frac{1}{2\sigma_0} \middle| s_{ij}, \mathcal{A} \right] \end{aligned} \tag{11}$$

Note that we already know the probability density function $p(\rho_{ij} | s_{ij}, \mathcal{A})$ in Equation 7. By shifting the mean of the distribution, we have

$$p\left(\rho_{ij} - \frac{s_{ij}}{\sigma^2 + 1} \middle| s_{ij}, \mathcal{A} \right) \sim N\left(0, \frac{\sigma^2}{\sigma^2 + 1} \right) \tag{12}$$

Considering Equation 11 and 12, $\Pr[\hat{l}_{ij} = l_{ij} | s_{ij}]$ can be calculated by a simple integral

$$\Pr[\hat{l}_{ij} = l_{ij} | s_{ij}, \mathcal{A}] = \int_{\frac{1}{-2\sigma_0}}^{\frac{1}{2\sigma_0}} \frac{\sqrt{\sigma^2 + 1}}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}(\sigma^2+1)} dx = \operatorname{erf}\left(\frac{\sqrt{\sigma^2 + 1}}{2\sqrt{2}\sigma\sigma_0} \right) \tag{13}$$

From Equation 5, 11, and 13, we have

$$\begin{aligned} & \Pr\left[\hat{k}_j^S = k_j^S \middle| \{s_{ij} | i \in Q\}, \{k_i^P | i \in Q\}, \mu_0, \sigma_0, \mathcal{A} \right] \\ = \ & \Pr\left[\hat{k}_j^S = k_j^S \middle| \{\hat{l}_{ij} = l_{ij} | i \in Q\}, \{k_i^P | i \in Q\}, \mu_0, \sigma_0, \mathcal{A} \right] \\ & \cdot \prod_{i \in Q} \Pr\left[\hat{l}_{ij} = l_{ij} \middle| s_{ij}, \mathcal{A} \right] \\ \leq \ & \prod_{i \in Q} \Pr[\hat{l}_{ij} = l_{ij} | s_{ij}, \mathcal{A}] = \left[\operatorname{erf}\left(\frac{\sqrt{\sigma^2 + 1}}{2\sqrt{2}\sigma\sigma_0} \right) \right]^{|Q|} \end{aligned} \tag{14}$$

The inequality in the fourth line of Equation 14 holds because the first probability is always less or equal to 1. This relaxation implies that the properties of Bloom filters (e.g., size, value) does not affect the proved bound.

According to the property of error function, $0 < \operatorname{erf}(x) < 1$ for any $x > 0$. Thus, for any set Q, we have

$$\Pr\left[\hat{k}_j^S = k_j^S \middle| \{s_{ij} | i \in Q\}, \{k_i^P | i \in Q\}, \mu_0, \sigma_0, \mathcal{A} \right] \leq \operatorname{erf}\left(\frac{\sqrt{\sigma^2 + 1}}{2\sqrt{2}\sigma\sigma_0} \right) \tag{15}$$

□

Interestingly, Equation 14 implies that a larger $|Q|$ leads to a lower success rate of the attack, which is reasonable due to the noise added to each s_{ij}. Specifically, because of the added noise in Equation 2, the attacker cannot accurately predict the real similarities in the first step of the attack. Though the attacker has more predicted real similarities $\hat{\rho}_i$ to infer k_j^S in the last step of the attack, these additional $\hat{\rho}_i$ bring more noise than information, thus leading to a higher success rate. In conclusion, under the intuitive greedy attack model, the optimal choice of the attacker is to launch an attack through a single Bloom filter and its similarity by setting $|Q| = 1$, the success rate of which can be bounded by a small constant.

A.3 Comparison between Two Privacy Metrics

As observed from both Theorem 5 and Theorem 1, the noise scale σ derived from differential privacy increases linearly by n. A such large scale of noise would seriously affect the performance of FedSim. Taking `house` dataset as an example, $\mu_0 = -46237.78, \sigma_0 = 21178.86$. Letting $\delta = 10^{-5}$, we can

derive the values of τ and ε under different noise scales σ as Figure 7. As observed from Figure 7, setting $\sigma = 4$, the differential privacy parameter $\varepsilon = 2.96 \times 10^9$ implies that there is almost no privacy guarantee at all. Nonetheless, the attacking success rate $\tau = 1.94 \times 10^{-5}$ suggests that the privacy risk from certain attacks can be very low.

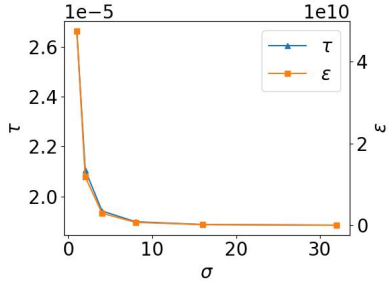


Figure 7: Values of τ and ε under different noise scales σ

The main difference between the two metrics is the accumulation of privacy risk over various similarities linked to the same k_j^S. In differential privacy, the overall privacy loss is a linear summation of the privacy loss from each similarity. Nevertheless, in our proposed metric, the attacking success rate τ decreases as the number $|Q|$ of similarities used in the attack increases according to Equation 14. In summary, **differential privacy that protects against all possible attacks is impractical in this scenario; as an alternative, our analysis suggests that the privacy risk against certain attacks can be significantly reduced by adding Gaussian noise**.

B Advanced Attacks on FedSim

B.1 Advanced Attacks

Since FedSim is the first VFL approach indicating that the proper use of similarities can benefit the training process, the attacks on the similarities remain unexplored. In this paper, we have proposed a greedy attack method and the corresponding defense as a start. Notably, more advanced attacks are non-trivial but possible and their corresponding defense mechanisms are desirable. We provide three potential vulnerabilities of FedSim for future attack design.

1) Background information of ρ_{ij}: In step (1) of the greedy attack method, we assume the attack adopts a Gaussian prior distribution to perform estimation, which is true when the attacker has no knowledge of the distribution of distances ρ_{ij}. Nonetheless, if some prior knowledge of ρ_{ij} is known to the attacker, one may carefully design a better prior than unit Gaussian and achieve higher predictive accuracy. For example, if knowing $\rho_{ij} < 100$, the attacker can truncate the prior distribution by setting $\Pr[\rho_{ij} \geq 100] = 0$. **2) Non-greedy attack:** The greedy attack method assumes the attacker predicts Bloom filters with the predicted values of distances. However, deriving an accurate value of each distance may not be necessary. It remains an open direction to design a method that directly exploits the distribution of distances to predict the target Bloom filter. **3) Correlation between shared information:** In Section 5.1, we list three shared information and discuss their privacy risk respectively. Nevertheless, the privacy of each piece of shared information does not imply the privacy of all three pieces of shared information. For example, if shared similarities $\mathbf{s}$ are correlated with shared intermediate results $\mathbf{c}_i{}_{i=1}^m$ in SplitNN, disclosing both could much more damaging than disclosing each of them. The effect of such correlation, which remains unexplored, will be left as our future work.

C Extension to Multiple Parties

As the effect of linkage on VFL receives increasing attention [39], to the best of our knowledge, all the existing VFL approaches with non-exact linkage [22, 27, 39, 40] focus on the two-party setting. Similar to these approaches, we also mainly focus on the two-party setting which has many real-world applications (e.g., bank and fintech company [60]). In this section, we explain the major challenges of

multi-party VFL with non-exact linkage and present a simple extension of FedSim to the multi-party setting.

The multi-party setting of VFL is far more than a trivial extension of the two-party setting because the linkage among parties can be complex. For example, consider a three-party VFL of an e-bank, an e-shopping company, and a delivery company. The e-bank and e-shopping company are linked by customers' *names*; the e-shopping company and the delivery company are linked by *transaction IDs*; the e-bank and the delivery company are linked by *address*. In this case, we need to select two "best" common features from *name*, *transaction ID*, and *address* to link the three parties and determine a proper order for the linkage. Although in the exact linkage on the same common feature [7, 26, 60], the choice and order of common features do not affect the linkage result, the result of non-exact linkage can be significantly affected by both the choice and order of common features. How to handle these data non-exactly linked as a circle (or even a complicated graph) is still an open problem of VFL.

Additionally, even assuming that all the parties are linked on the same common feature and the order of linkage is fixed, performing one-to-many linkage between each pair of parties suffers an efficiency issue. Specifically, one intuitive approach is to link all possible pairs across the parties and expand $\mathbf{o}_i$ from $K \times l_m$ to size $K^{n-1} \times l_m$, where n is the number of parties (both primary and secondary). This approach, leading to K^{n-2} times more training cost than the cost of the two-party FedSim, is not scalable on parties.

Fortunately, our empirical study on five real-world datasets suggests that, **compared to secondary parties, the primary party who owns labels usually also holds more important or at least comparable features**. Specifically, we first link each data record in the primary party with the most similar data record in the secondary party (similar to Top1Sim). Then, each data record in both the primary party and the secondary party has a label. Finally, we train the same model on the primary party and the secondary party, respectively, and summarize the performance in Table 2. Compared with the secondary party, we observe that the primary party has much better performance (thus much more important features) on `house`, `hdb`, and `song`. Meanwhile, the primary party has slightly lower but comparable performance (thus comparable features) on `bike` and `game`.

Table 2: Performance on a single party (either primary or secondary) on real-world datasets

Dataset	house (RMSE)	bike (RMSE)	hdb (RMSE)	game (Accuracy)	company (RMSE)
Primary	58.31±0.28	272.83±1.50	29.75±0.15	85.27±0.29%	37.08±0.61
Secondary	150.27±0.17	265.79±0.06	134.82±0.10	88.21±0.06%	225.07±0.02
Relative diff.[1]	**61.20**%	-2.65%	**77.94**%	-3.34%	**83.53**%

[1] Positive means primary outperforms secondary; negative means secondary outperforms primary. The value means relative difference, i.e., $|\text{primary} - \text{secondary}|/\text{secondary}$.

This observation implies that the linkage between the primary party and the other parties is the most vital in practice. Under this observation, to overcome the scalability issue of multi-party FedSim, we propose an intuitive approach that extends FedSim to the multi-party setting. Specifically, we first perform one-to-one linkage for all the secondary parties, then perform one-to-many linkage between the linked secondary parties and the primary party. During the training process, each secondary party holding a local model performs training according to multi-party SplitNN. This approach is empirically evaluated in Appendix F.3.

D Applications

In this section, we discuss what datasets FedSim performs well on and how common such datasets exist in real-world applications. As we have explained in Appendix F.6, the main assumption of FedSim is

the similarity between identifiers is related to the similarity between data records.

If this assumption holds, FedSim probably outperforms Exact/Top1Sim. The potential improvement can be estimated by our proposed metric Δ. If this assumption does not hold, e.g., the identifier

is unique ID, the experiment in Appendix F.6 indicates that FedSim has the close performance to Exact/Top1Sim under a small K. Setting a large K is likely to significantly reduce the performance due to overfitting.

This assumption commonly holds in real-world applications. Since VFL is a rather recent paradigm with fewer real applications, we investigate the applications of a traditional area "record linkage" which has many existing applications. The application scope of record linkage reflects the application scope of VFL, because all the VFL algorithms require the data to be linked (either on ID or other features) before training.

German Record Linkage Center (GRLC) published an article [4] to summarize its completed linkage projects since 2011. We summarize the information of identifiers in Table 3.

Table 3: The identifiers of the completed projects in GRLC since 2011 (excluding two papers written in German that we are unable to translate)

Application	Reference	Identifier	Satisfy the Assumption
Employment	[2]	name, birthday, street, house number, etc.	✓
	[16]	address	✓
	[41]	address	✓
	[49]	name, address	✓
	[9]	ID	✗
	[58]	ID	✗
Commercial	[48]	name, address	✓
	[3]	company name	✓
Migration	[31]	gender, age, address, municipality, etc.	✓
Medical	[20]	name, date of birth	✓
Education	[19]	ID	✗
Number of applications satisfying the assumption			**8/11 (72.7%)**

Two observations can be made from these projects. First, among the 11 completed projects, eight projects require linking datasets without user ID. This implies that, in the majority (around 8/11) of real applications, there does not exist a shared user ID. Second, the similarity of some fields can reflect the similarity of the property, such as address (5/11 projects) in GPS or string format. Even for the field whose own similarity does not reflect record property, such as name, birth date, and postal code, the similarity of the quasi-identifier containing these fields (2/11 projects) can reflect the property. This is because records with more matched fields are more likely to belong to the same user, especially when considering typos that widely exist in practice. Therefore, the similarity of shared features is related to the property of records in many (around 8/11) real-world cases, which supports our main assumption.

E Experimental Details

Dataset. The basic information of these datasets is summarized in Table 4. Specifically, in `house` dataset, party P contains housing data in Beijing collected from *lianjia* [44], and party S contains renting data in Beijing collected from *Airbnb* [1]. Two parties are linked by longitude and latitude and the task is to predict the housing price. In `taxi` dataset, party P contains taxi trajectory data in New York from TLC [52], and party S contains bike trajectory data in New York from Citi Bike [8]. Two parties are linked by the longitude and latitude of the source and destination of the trajectory and the task is to predict the time of the trip along this trajectory. In `hdb` dataset, party P contains HDB resale data in Singapore collected from Housing and Development Board [23], and party S contains recent rankings and locations of secondary schools in Singapore collected from salary.sg [45]. Two parties are linked by longitude and latitude and the task is to predict the HDB resale prices. In `game` dataset, party P contains game data (e.g., prices) and the number of owners (#owners) from Steam [11]. party S contains game ratings and comments from RAWG [24]. Two parties are linked by game titles and the task is to classify games as popular (#owners $>$ 20000) or unpopular (#owners $\leq$ 20000) based on ratings, prices, etc. In `song` dataset, party P contains timbre values of songs extracted from *million song dataset* [6], and party S contains the basic information of songs extracted

from FMA dataset [12]. Two parties are linked by the titles of songs and the task is to predict the publication years of songs.

Table 4: Basic information of datasets

Dataset	Type	Party P			Party S			Identifiers		Task
		#samples	#ft.	ref	#samples	#ft.	ref	#dims	type	
sklearn	Syn	60,000	50	[10]	60,000	50	[10]	5	float	bin-cls
frog	Syn	7,195	19	[13]	7,195	19	[13]	16	float	multi-cls
boone	Syn	130,064	40	[14]	130,064	40	[14]	30	float	bin-cls
house	Real	141,050	55	[44]	27,827	25	[1]	2	float	reg
taxi	Real	200,000	964	[52]	100,000	6	[8]	4	float	reg
hdb	Real	92,095	70	[23]	165	10	[45]	2	float	reg
game	Real	26,987	38	[11]	439,999	86	[24]	1	string	bin-cls
company	Real	77,225	91	[47]	220,583	157	[64]	1	string	reg

Note: "#ft." means number of features; "Syn" means synthetic; "Real" means real-world; "bin-cls" means binary classification; "multi-cls" means multi-class classification; "reg" means regression.

These eight datasets have a wide variety in multiple dimensions. 1) **Matching of common features**: Due to the property of common features, some datasets can be exactly matched (e.g., `syn`, `frog`, `boone` without noise), some datasets can partially exactly matched (e.g, `game`, `song`), and some datasets can only be soft matched (e.g., `house`, `taxi`, `hdb`). 2) **Similarity metric**: These datasets cover three commonly used similarity metrics including Euclidean-based similarity, Levenshtein-based similarity, and Hamming-based similarity. 3) **Task**: The selected datasets cover three common tasks including binary classification, multi-class classification, and regression.

Hyperparameters. The number K of neighbors is chosen from $\{50, 100\}$. The sizes of hidden layers are chosen from $\{100, 200, 400\}$ and remain consistent across all VFL algorithms. The learning rate is chosen from $\{3 \times 10^{-4}, 10^{-3}, 3 \times 10^{-3}, 1 \times 10^{-2}\}$ and weight decay is chosen from $\{10^{-5}, 10^{-4}\}$. Since FedSim and AvgSim process K times more samples per batch compared to other algorithms, their batch sizes are $1/K$ times smaller than the others. The batch size of FedSim and AvgSim are chosen from $\{32, 128\}$, while the batch sizes of other algorithms are set to 4096. The training is stopped at the best performance on the validation set.

Hardwares. The training of all the experiments is conducted on a machine with four A100 GPUs, two AMD EPYC 7543 32-Core CPUs, and 504GB of memory. The linkage of all the experiments is performed on another machine with two Intel Xeon Gold 6248R CPUs and 377GB memory. Additional 500GB swap space is needed to link large datasets like `song`.

Licenses. Our codes are based on Python 3.8 and some public python packages. No existing codes are included in FedSim. We will release our codes under Apache V2 license[1].

The datasets that we use in our experiments have different licenses as summarized in Table 5. All of them can be used for analysis but only some of them can be shared or used commercially.

F Additional Experiments

F.1 Additional Experiment on Privacy

Due to the page limit, we present the experiment of Section 5 on `company` in Figure 8. We can make similar observations to Section 5 that FedSim is robust to noise on similarities.

F.2 Effect of Hyperparameter K

In this subsection, we aim to illustrate that the baselines cannot achieve good performance even by carefully tuning the number of neighbors K, while FedSim remains stable with large K and

[1] `https://www.apache.org/licenses/LICENSE-2.0`

Table 5: Licenses of datasets

License	Dataset	Analyze	Adapt	Share	Commercial
CC BY 4.0[a]	[11, 64]	✓	✓	✓	✓
CC BY-NC-SA 4.0[b]	[44]	✓	✓	✓	✗
CC BY-SA 4.0[g]	[47]	✓	✓	✓	✓
CC0 1.0[c]	[1, 13, 14, 45]	✓	✓	✓	✓
Singapore Open Data License[d]	[23]	✓	✓	✓	✗
NYCBS Data Use Policy[e]	[8]	✓	✓	✓	✓
All rights reserved	[24, 52]	✓	✗	✗	✗

[a] `https://creativecommons.org/licenses/by/4.0/`
[b] `https://creativecommons.org/licenses/by-nc-sa/4.0/`
[c] `https://creativecommons.org/publicdomain/zero/1.0/`
[d] `https://data.gov.sg/open-data-licence`
[e] `https://www.citibikenyc.com/data-sharing-policy`
[f] `http://millionsongdataset.com/faq/`
[g] `https://creativecommons.org/licenses/by-sa/4.0/`

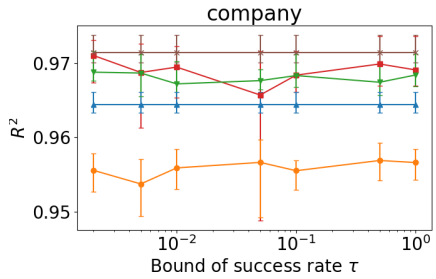


Figure 8: Performance with different scale of noise on similarities

consistently outperforms the baselines. With other hyperparameters fixed, we present the performance under different K in Figure 9.

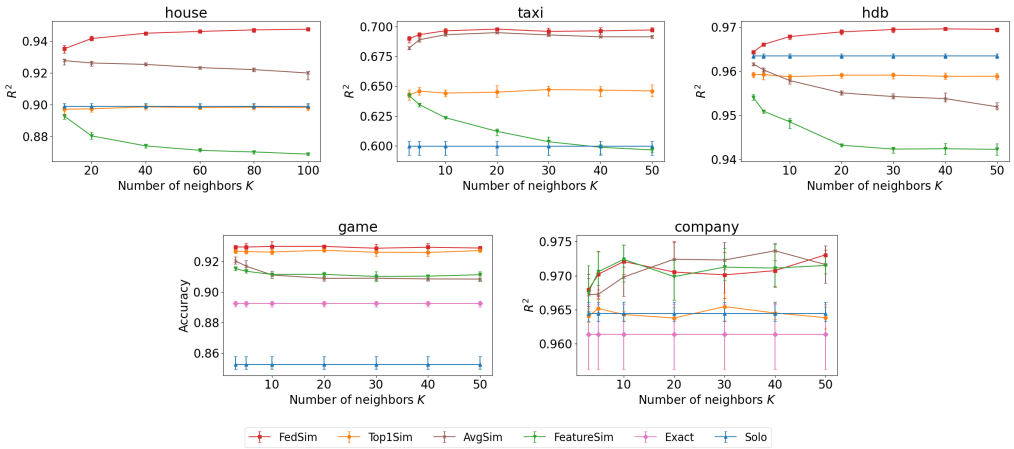


Figure 9: Performance with different K

As we expect, the performance of Top1Sim remains steady since the pairs with the largest similarities remain the same. The performance of AvgSim and FeatureSim drops as K increases since they cannot handle the redundant data properly. On the contrary, with increasing K, the performance of FedSim increases and then remains steady, which means FedSim can effectively exploit useful information in these additional data. In conclusion, **FedSim is robust to large** K; setting a relatively large K achieves the best performance for FedSim.

F.3 Multi-Party FedSim

In this subsection, we evaluate FedSim in multi-party setting on `house`. Specifically, we equally split the dataset for secondary parties into three parties by features. In Table 6, we report the performance when the party P conducts VFL with each combination of three secondary parties $\{S_1, S_2, S_3\}$. It can be observed that FedSim with more secondary parties produces better performance, which shows that FedSim can effectively utilize the data of more parties.

Table 6: Performance of Four-Party FedSim on house

Secondary parties	S_1, S_2, S_3	S_1, S_2	S_1, S_3	S_2, S_3
R^2	0.9475	0.9470	0.9481	0.9456
Secondary parties	S_1	S_2	S_3	Solo
R^2	0.9469	0.9114	0.9457	0.8989

F.4 Time Consumption of FedSim

Linkage. We present the time for linkage in Table 7. All the datasets organically contain either float or string identifiers. For float identifiers, Euclidean similarities are calculated; for string identifiers, edit similarities are calculated. Meanwhile, all the identifiers can be converted to Bloom filters according to [29] for privacy. The Hamming similarities between Bloom filters are calculated. As can be observed from Table 7, linking Bloom filters for privacy is generally more time-consuming than linking raw float/string features. This indicates that a more efficient PPRL approach is desired. We leave this topic as our future study.

Table 7: Time for record linkage (min)

Type of Identifiers	**house**	**taxi**	**hdb**	**game**	**company**
Float/String	3.48	12.95	0.98	3.62	13.20
Bloom Filter [29]	145.20	594.27	258.23	41.38	19.87

Training. FedSim has more parameters compared with other one-to-one baselines, thus leading to a longer training time. For each dataset, we report the training time per epoch in Table 8, and present the number of parameters of each model in Table 9. Note that the soft linkage procedure (including similarity calculation and k-nearest-neighbor search), as a preprocessing step, is not included in the training time.

Two observations can be made from Table 8. First, FedSim, AvgSim, and FeatureSim require **longer but acceptable training time** compared to other baselines. This is because these three approaches train K times more samples than other approaches, thus costing approximately K times longer time. Second, FedSim requires **similar training time** compared to AvgSim and FeatureSim in each epoch. Although FedSim contains around K times more parameters (mostly in merge model) than AvgSim and FeatureSim, the trained samples in the merge model are also K times fewer than the original batch size, because each d_i^P, together with its K neighbors $\mathbf{d}_i^S$, is regarded as one 2D sample in the merge model. Therefore, the per-epoch training time of FedSim is similar to that of AvgSim and FeatureSim.

Table 8: Training time (seconds) per epoch of different VFL algorithms on real-world datasets

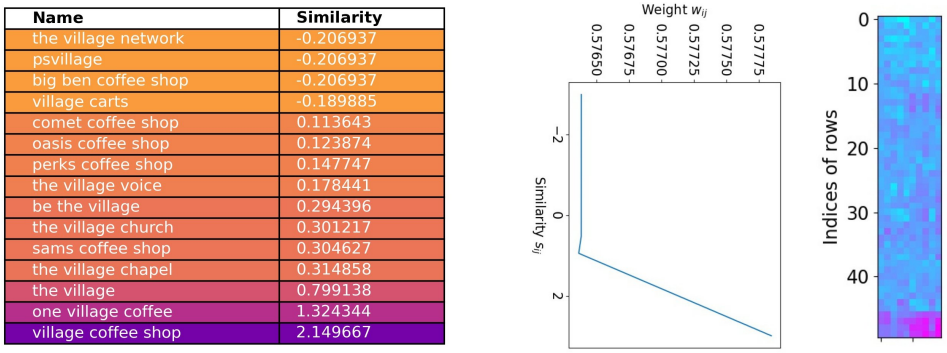
Models	**house**	**taxi**	**hdb**	**game**	**company**
Exact	-	-	-	<1	4
FeatureSim	15	51	5	1	23
AvgSim	6	35	4	1	15
Top1Sim	<1	1	<1	<1	3
FedSim	9	38	6	4	62

Table 9: Number of parameters of each model

Models	house	taxi	hdb	game	company
Exact	-	-	-	63,301	85,401
Featuresim	76,201	116,801	76,201	32,801	45,201
Avgsim	76,001	116,701	136,601	63,301	75,701
Top1sim	76,001	11,971	76,001	63,301	170,601
Fedsim	3,469,806	1,846,490	1,869,806	1,826,506	494,474

F.5 Visualization of the Similarity Model and Merge Model

To provide more insights into FedSim, we visualize the similarity model and the merge model as Figure 10, both of which are extracted from the converged FedSim on `company` dataset. For the data records linked with "the village coffee shop" (left), we highlight similar records with deeper colors. For the similarity model (middle), which has one-dimensional input and one-dimensional output, we plot the model in a two-dimensional coordinate system when scaled similarities range in [-3,3]. For the merge model (right), we evaluate the feature importance of each feature in $\mathbf{o}'_t(50 \times 10)$ by *integrated gradients* [51], which is widely used [43, 46] in model explanation. Larger vertical indices indicate larger similarities due to the sort gate. The deeper colors indicate higher importance and the lighter colors indicate lower importance.



Name	Similarity
the village network	-0.206937
psvillage	-0.206937
big ben coffee shop	-0.206937
village carts	-0.189885
comet coffee shop	0.113643
oasis coffee shop	0.123874
perks coffee shop	0.147747
the village voice	0.178441
be the village	0.294396
the village church	0.301217
sams coffee shop	0.304627
the village chapel	0.314858
the village	0.799138
one village coffee	1.324344
village coffee shop	2.149667

Figure 10: Visualization of FedSim on a data batch of `company` dataset. The left figure includes identifiers ("company name") similar to "the village coffee shop". The middle figure shows the mapping from similarities to weights in the similarity gate. The right figure displays the feature importance calculated by *integrated gradients* [51] of each input feature (50×10) of merge gate.

Three observation can be made from Figure 10. First, "the village coffee shop" is soft-linked with three categories of companies: (a) exactly matched "village coffee shop", (b) other coffee shops, and (c) other companies unrelated to the coffee shop (e.g., the village network). Intuitively, (a) and (b) tends to have more similar feature values (e.g., number of employees) with "the village coffee shop" than (c). Therefore, when training "the village coffee shop", the linked pairs with (a) and (b) should have a larger effect than (c). Second, as expected, in the similarity model, the record in (a) is granted a dominant weight, the records in (b) are granted moderate-to-high weights, and the records in (c) are granted small weights. Third, in the merge model, the linked records with high similarities, as an input feature for the merge model, have a more significant effect on the merge model. These observations imply that the weight gate and merge gate can effectively capture the key data records and filter useless records according to the similarities.

F.6 Independent Common Features

In this subsection, we study a special case where there is no mutual information between common features and unique features. We generate a synthetic dataset `sklearn-random` similarly to `sklearn` dataset. The only difference is that `sklearn-random` generates random common features, while `sklearn` selects existing features as common features. In this case, the similarity between common features is independent of the other features. Thus, we expect FedSim to perform the same as

Top1Sim/Exact baseline at a reasonable K. After training FedSim with CNN merge model five times, the results are summarized in Table 10.

Table 10: Performance on random common features

Method	Accuracy
Solo	60.76±0.41
Top1Sim/Exact	92.15±0.33
Combine	94.70± 0.58
FedSim ($K = 50$)	87.79± 0.77
FedSim ($K = 30$)	**92.80± 0.61**

From Table 10, we observe that the performance of FedSim is similar to Exact and Combine when $K = 30$ though dropping lower when $K = 50$ due to the noise. This demonstrates the basic assumption behind FedSim: **the similarity between identifiers is related to the similarity between records**. FedSim outperforms baselines when the relationship is tight (e.g. identifiers are GPS), and provides similar performance to baselines when the relationship is weak or non-existent (e.g. identifiers are hash value). Notably, by properly setting hyperparameters, FedSim will not degrade under baselines because `Exact` and Top1Sim are special cases of FedSim when setting the number of neighbors K to 1 with a linear merge model.

F.7 Different Similarity Metrics

In this subsection, we study how different similarity metrics affect the performance of FedSim. For numeric identifiers, we test Euclidean similarity and Hamming similarity; for string identifiers, we test edit similarity and Hamming similarity. The performance of FedSim is presented in Table 11. We observe that the dataset with Hamming similarity is usually lower than other metrics because generating Bloom filters introduces random noise. The only exception is the `company` dataset with long strings of company names as identifiers. In these names, words can be dislocated or shuffled, thus failing the edit similarity. For example, "Kentucky bank" and "bank of Kentucky" with a large edit distance can be recognized as similar by Bloom filters built from q-grams. In general, the similarity metric does not significantly affect the performance.

Table 11: Performance of FedSim on different similarity metrics

Similarity Metric	house (numeric)	bike (numeric)	hdb (numeric)	game (string)	company (string)
Euclidean/Edit	42.12±0.23	235.67±0.27	27.13±0.06	92.88±0.11%	40.04±2.18
Hamming	50.03±1.09	238.81±0.50	28.29±0.22	92.70±0.48%	37.08±0.61

G Limitations

We discuss the limitation of FedSim in the following two aspects.

Accuracy. As discussed in Appendix D and Appendix F.6, the performance boosting of FedSim is based on an assumption that *the similarity between identifiers is related to the similarity between data records*. This assumption may not hold in a small portion of real applications (e.g., identifiers are randomly generated IDs). In these cases, FedSim has no improvement on baselines and can even be outperformed by baselines if K is too large (see Appendix F.6). Further improvements on these cases are left as our future work.

Privacy. As stated in Section 5, we propose an intuitive attack and the corresponding defense. Nonetheless, this approach does not fully guarantee the privacy of FedSim. We further discuss three directions of advanced attacks in Appendix B. The design of these advanced attacks is non-trivial, thus we leave these advanced attacks as well as their defense as our future work.

Scalability. In Appendix C, we propose an extension of FedSim to the multi-party setting under the assumption that the primary party holds more important or comparable features than secondary

parties. In this case, FedSim is scalable on parties since secondary parties can be linked by one-to-one mappings without significant performance loss. Nevertheless, if this assumption no longer holds, performing one-to-many linkage between each pair of parties is infeasible due to the expensive computational cost. Designing a non-exact VFL model for multiple parties remains an open problem in VFL.

H Discussion

Potential negative societal impact. `FedSim` might be adapted to a linkage attack method, which could threaten the privacy of released data. Since we provide an effective approach to exploit information through fuzzy matched identifiers (e.g. GPS locations), this kind of identifier should be paid special attention to privacy when releasing a dataset.

Consent of dataset. As discussed in Appendix E, all the datasets that we are using are publicly available for analysis. Therefore, we naturally obtain consent of using these datasets by following their licenses.

Personally identifiable information. To the best of our knowledge, the datasets that we are using do not contain personally identifiable information or offensive contents.

Extension to other VFL algorithms FedSim makes a significant step toward practical VFL by enabling VFL on a wider range of applications that requires fuzzy matching. Although `FedSim` is designed based on SplitNN, the idea of working directly with similarities can also be developed in other VFL frameworks. Unfortunately, the framework of FedSim cannot be directly applied to other VFL algorithms, e.g., logistic regression and tree-based algorithms. This is because FedSim requires the similarity model and merge model to be trained together with the main VFL model, which requires the VFL algorithm to be neural-network-based. To adapt FedSim to other VFL algorithms, a possible direction is to exchange the intermediate information in these algorithms like exchanging gradients in SplitNN. We leave this topic as our future work.

Relation to other topics. Besides VFL, our main insight that *the similarity between identifiers is related to the similarity between data records* has also been recognized in many other topics. For example, k-nearest neighbors (k-NN) algorithm [42] predicts by averaging the most similar samples. Graph neural networks (GNN) [21] adopt a loss function that encourages neighboring nodes to have similar representations. Semi-supervised learning [65] can benefit from matching similar instances. FedSim is the first approach that exploits this insight in VFL.

I Additional Background

FEDERAL [29] is a PPRL framework that theoretically guarantees the indistinguishability of bloom filters. Suppose the size of bloom filters is set to N. For strings, it generates q-grams and encodes q-grams to bloom filters of size N by composite cryptographic hash functions. For numeric values, it first generates N random numbers r_i $(i \in [1, N])$ and sets a threshold t. Next, it determines N hash functions $h_i(x)$

$$h_i(x) = \begin{cases} 1, & x \in [r_i - t, r_i + t] \\ 0, & otherwise \end{cases} \quad (i \in [1, N])$$

Each numeric value is hashed by all the functions $h_i\,(i \in [1, N])$ and is converted to a bloom filter of size N. These bloom filters are used to calculate similarities on an honest-but-curious third party. They prove that all the bloom filters have similar numbers of ones if we properly set the size of bloom filters. Formally, denoting ω as the number of ones in a bloom filter, we have $\Pr[\omega \leq (1-\epsilon)E[\omega]] < \delta$, where $\epsilon < 1$ and $\delta < 1$ are tolerable deviations. Therefore, attackers cannot infer whether the numeric values are large or small given the bloom filters. This method is adopted in our experiment on hamming-based similarities.