SELF-EVOLVING PRIVATE EVOLUTION FOR DIFFERENTIALLY PRIVATE SYNTHETIC TEXT GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Differentially private synthetic text generation enables data sharing and text mining applications while providing formal privacy guarantees. Recent advances leverage large language models (LLMs) as strong generative priors for producing fluent and semantically rich synthetic text. Among these methods, Private Evolution (PE) is a promising approach that incorporates LLM inference APIs to generate and refine synthetic samples without expensive fine-tuning. PE iteratively improves synthetic data by generating variations, evaluating candidates through private voting, and selecting high-quality samples for the next iteration. However, existing PE methods discard these privatized evaluation signals after each iteration, despite their potential utility for future rounds. We propose `SelfEvo-PE`, which reuses available privatized signals as supervision to train lightweight evaluators that guide future evolution steps without additional privacy cost. This mechanism allows `SelfEvo-PE` to perform more evolution steps under the same privacy budget. Our experiments on three datasets show that `SelfEvo-PE` outperforms the conventional PE (Aug-PE) under limited-data and strict-privacy settings, improving FID and MAUVE by up to $\sim$15% and $\sim$20%, respectively.[1]

1 INTRODUCTION

Text is a cornerstone of information that captures human knowledge, communication, decisions, experiences, and interactions across nearly every domain (Gentzkow et al., 2019). However, sharing and utilizing the data is often restricted by privacy concerns, especially in sensitive domains such as healthcare (Nowrozy et al., 2024; Nass et al., 2009), finance (Balch et al., 2024), and legal services (Yin & Habernal, 2022). Synthetic text generation with differential privacy (DP) (Yue et al., 2023) offers a promising path forward by producing semantically similar data while offering formal privacy guarantees (Dwork, 2006). Such synthetic data can enable downstream applications, including data sharing, model training, and research collaboration, without directly exposing the original sensitive records.

Due to its practical importance, differentially private synthetic text generation has emerged as an active and popular research area (Ponomareva et al., 2025; Yue et al., 2023; Xie et al., 2024; Hou et al., 2025; Xu et al., 2026; Tan et al., 2025; Hu et al., 2025; Chien et al., 2026). Most of the proposed methods are based on the development of Large Language Models (LLMs) and their ability to generate fluent and coherent text. There are two main lines of work: (1) **LLM fine-tuning** (Yue et al., 2023; Kurakin et al., 2024; Tan et al., 2025; Xu et al., 2026; Hu et al., 2025) and (2) **LLM inference-only** (Tang et al., 2024; Lin et al., 2024; Xie et al., 2024; Hou et al., 2024; Zou et al., 2025; Hou et al., 2025; Chien et al., 2026). The LLM fine-tuning approach employs differentially private optimization (Abadi et al., 2016) to update the model parameters, which can align the LLM behavior towards the private data distribution such as vocabularies, styles, and topics. However, this approach is computationally expensive and requires significant resources to perform the gradient computations and model updates. In practical scenarios, the local resources available for private data holders may be limited and insufficient to support the fine-tuning process. In contrast, the LLM inference-only approach does not require any LLM fine-tuning and can be performed with online LLM inference APIs, enabling more accessible and efficient solutions for end users such as mobile phone users (Hou et al., 2024; Research, 2025). In practice, the choice of whether to use

[1] Our code is available at `https://anonymous.4open.science/r/selfevo-pe-0133`

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

LLM fine-tuning or inference-only approaches depends on the specific requirements and constraints of the application (Ponomareva et al., 2025; Chien et al., 2026). There are two main categories of inference-only approaches: (1) Private Inference (Tang et al., 2024; Gao et al., 2025; Amin et al., 2024; Mori et al., 2026; Vinod et al., 2026) which applies noise at the token level to ensure DP. Since the privacy cost accumulates over both the number of generated tokens and the number of generated samples, these methods are typically more suitable for producing a small number of synthetic examples, such as protecting demonstration data for in-context learning; (2) Private Evolution (PE) (Lin et al., 2024; Xie et al., 2024; Hou et al., 2025) that applies privacy-preserving selection at the sample level. This allows generation of a large number of synthetic samples with the goal of approximating the private distribution, making PE on par with the LLM fine-tuning approach in terms of scalability. *This paper focuses on the PE approach due to its practicality and accessibility in real-world applications.*

Private Evolution was first introduced by Lin et al. (2024) for image generation using diffusion models and has been adopted by subsequent works (Xie et al., 2024; Hou et al., 2024; 2025; Wang et al., 2025b; Zhu et al., 2026) to generate synthetic text using LLMs. PE performs an evolutionary loop: it generates variations of synthetic samples, evaluates the candidates using private data by a DP voting approach, selects the best ones, and uses them to produce the next generation, illustrated in Fig. 1 (Top). To improve PE, recent works have better guided LLM-based generation toward the private data distribution through in-context learning (Zou et al., 2025), metadata-informed initialization (Chien et al., 2026), and alignment (Hou et al., 2025). Despite these improvements, the core private data voting process of PE remains unchanged: at each iteration, PE queries the private data to evaluate the generated synthetic samples and vote for the best ones for the next generation. The voting results are then discarded after each iteration. This can be a waste, as we found that the voting results from previous iterations contain useful information for predicting which synthetic samples are likely to be high-quality in later iterations (Sec. 3).

In this paper, we explore a new direction of PE: treating the voting results from each iteration as a persistent resource that can be stored, managed, and reused, rather than single-iteration information. Building on this perspective, we propose Self-Evolving Private Evolution (`SelfEvo-PE`), a novel framework that can perform the evolutionary process *without querying the private data at every iteration*. Our key insight is that each private voting round produces differentially private supervision: synthetic samples and their corresponding noisy vote counts. `SelfEvo-PE` leverages this free supervision to train lightweight evaluators that approximate the voting results for newly generated samples. These evaluators then guide subsequent evolution steps without additional privacy cost, illustrated in Fig. 1 (Bottom). With this approach, `SelfEvo-PE` can perform more iterations without incurring any additional privacy cost. By enabling more evolution under the same privacy budget, `SelfEvo-PE` better explores the synthetic data space and improves synthetic data quality. Our contributions can be summarized as follows:

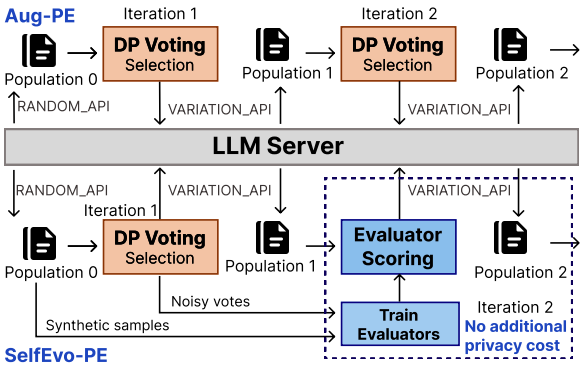

Figure 1: Aug-PE (Top) queries the private data at every iteration. In contrast, SelfEvo-PE (Bottom) reuses the available DP voting results to train evaluators, enabling more rounds without additional privacy cost.

- We propose `SelfEvo-PE`, a novel framework that can perform the evolutionary process without accessing the private data at every iteration, thus saving privacy cost and improving the quality of synthetic data.
- We conduct extensive experiments on three datasets under various privacy budgets. `SelfEvo-PE` outperforms the PE baselines by up to ∼15% and ∼20% in terms of FID and MAUVE, demonstrating its effectiveness under practical scenarios of limited private data and tight privacy budgets.
- We explore the impact of various design choices in `SelfEvo-PE` and provide insights into the trade-offs between privacy cost, quality, and computational efficiency. We analyze

the strengths and weaknesses of SelfEvo-PE and the conventional PE approach under different settings, providing guidance for practitioners to choose the appropriate method.

2 PRELIMINARY

Differential Privacy (Dwork, 2006). A randomized algorithm $\mathcal{M}$ is (ϵ, δ)-differentially private if for any two neighboring datasets $\mathcal{D}$ and $\mathcal{D}'$ that differ by at most one record, and for any subset of outputs $S \subseteq \text{Range}(\mathcal{M})$, the following holds:

$$\Pr[\mathcal{M}(\mathcal{D}) \in S] \leq e^{\epsilon} \cdot \Pr[\mathcal{M}(\mathcal{D}') \in S] + \delta$$

3 METHODOLOGY: SELFEVO-PE

Motivating Observation. At each iteration, PE evaluates synthetic samples using private data in a DP voting manner and selects promising samples to generate the next population. We run Aug-PE on the Congressional Speech dataset (Hou et al., 2025) and take the top 5% samples ranked by noisy vote count at each iteration as a reference set. We then compare their similarity to the next-iteration samples that PE either selects for further generation or discards. As shown in Fig. 2, for all iterations, selected samples consistently exhibit higher similarity to the reference set than unselected samples. Fig. 7 (Appx. C.1) provides other similarity metrics and confirms the same trend. This similarity gap suggests that **previous voting results contain useful signals for identifying high-quality samples in later iterations**.

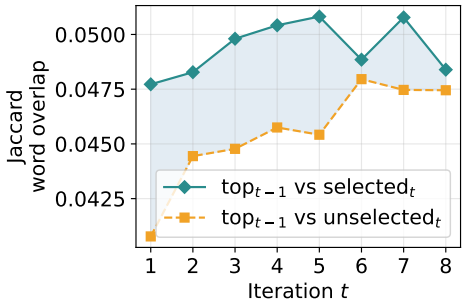

Figure 2: Selected samples exhibit higher similarity to the top samples of the previous iteration than unselected samples.

Overview. Given a private dataset $\mathcal{D}_{\text{priv}}$, our goal is to generate a synthetic dataset $\mathcal{D}_{\text{syn}}$ that approximates the private distribution while satisfying DP. The key idea of SelfEvo-PE is to reuse past private voting results to guide further evolution without additional privacy cost. Starting from an initial synthetic population, SelfEvo-PE alternates between conventional PE iterations that obtain noisy private votes and self-evolving iterations guided by evaluators trained on these votes. Both types of iterations select promising samples and generate variations to form the next population. We describe the main components below.

Initialization (RANDOM_API). The initialization generates an initial synthetic dataset $\mathcal{P}_0$. We reuse the design of Xie et al. (2024), which simply prompts an LLM. To enhance the diversity, we randomly select from a predefined set of keywords specifying writers, tones, and topics.

Private Voting (DP_NN_HISTOGRAM). For each conventional PE iteration, the private voting mechanism scores each synthetic sample in a DP manner (Alg. 1). Each private sample z votes for a single synthetic sample x^*, where x^* is the nearest neighbor in the embedding space among all synthetic samples (Alg. 1, L4-7). Let h be the (clean) histogram of votes for all synthetic samples and let $h(x)$ denote to the vote count for sample x. To ensure (ϵ, δ)-DP, we add noise drawn from a Gaussian distribution $\mathcal{N}(0, \sigma^2)$ to the histogram h, resulting in a noisy histogram $\tilde{h}$ (Alg. 1, L9). The sensitivity of this histogram is 1, as each private sample contributes exactly one vote. The noise multiplier σ is determined by the privacy parameters ϵ and δ and the number of conventional PE iterations performing private voting.

Algorithm 1 DP_NN_HISTOGRAM

1: **Input:** Private dataset $\mathcal{D}_{\text{priv}}$, population P, noise multiplier σ
2: **Output:** Noisy histogram $\tilde{h}$
3: $h(x) \leftarrow 0$ for all $x \in P$
4: **for** each private sample $z \in \mathcal{D}_{\text{priv}}$ **do**
5: $\quad x^\star \leftarrow \arg\min_{x \in P} \text{distance}(z, x)$
6: $\quad h(x^\star) \leftarrow h(x^\star) + 1$
7: **end for**
8: **for** each synthetic sample $x \in P$ **do**
9: $\quad \tilde{h}(x) \leftarrow h(x) + \mathcal{N}(0, \sigma^2)$
10: **end for**
11: **return** $\tilde{h}$

Evaluator Training. Evaluators aim to work as the Private Voting mechanism, scoring a new synthetic sample x within population $\mathcal{P}$. This enables more evolution rounds without incurring any additional privacy cost. The idea is that each private voting round provides information about synthetic

samples and their corresponding noisy scores ($\tilde{h}$). Evaluators are then trained on this data source, which is already differentially private, to predict scores of new synthetic samples. A straightforward method is to train a regression model that takes the embedding of synthetic samples as input and predicts the corresponding noisy scores. However, these counts are noisy and depend on the surrounding neighborhood: close samples compete for votes, so a sample's score can change as the neighborhood evolves even when its content remains unchanged. We address these challenges through learning to rank and density-aware features, and use ensemble learning to stabilize predictions.

Learning to rank. Sample selection requires only knowing which samples are more promising, rather than their exact vote counts. We therefore train the evaluator to preserve the order using the pairwise Learning To Rank (LTR) loss (Burges et al., 2005). For pairs of synthetic samples (x_i, x_j) from the same class:

$$\mathcal{L}_{\text{LTR}} = \sum_{i,j} \mathbb{I}(\tilde{h}(x_i) > \tilde{h}(x_j)) \cdot \log(1 + e^{-(f_\theta(x_i) - f_\theta(x_j))}),$$

where $\mathbb{I}(\cdot)$ is an indicator function. When x_i receives more noisy votes than x_j, the loss encourages $f_\theta(x_i) > f_\theta(x_j)$. This approach can be more robust to noise than regression, as it focuses on relative order rather than absolute values.

Density-aware features. The vote count of a synthetic sample depends on both its quality and the density of its neighborhood. Our feature set first includes the embedding of the sample itself, which is already computed during the private voting mechanism. To capture the sample's neighborhood information, we consider: 1) the average distance to its k nearest neighbors in the embedding space – capturing local density; 2) the total noisy vote count of its neighbors within a globally calibrated radius – representing the overall quality of the neighborhood. The radius is the average k-nearest-neighbor distance across all samples. We consider several values of k and concatenate these features with the sample embedding and its class as the input to the evaluator.

Ensemble learning. To reduce reliance on a single model's errors, we train multiple evaluators on different data splits and average their predictions during inference. In practice, we train small XGBoost models (Chen & Guestrin, 2016), which do not add much computational overhead on top of the conventional PE (Fig. 5, Sec. 4.2.2).

Variation Generation (`VARIATION_API`). Following Aug-PE (Xie et al., 2024), for each selected synthetic sample (the parent), we prompt the LLM to generate several variations (child samples). Each parent and its variations form a *family*, denoted by $\mathcal{F}$. We retain these family relationships for future sample selection.

Sample Selection. Conventional PE iterations rank samples by their noisy vote count $\tilde{h}(x)$, while self-evolving iterations employ a score that balances predicted quality and diversity:

$$s(i) = \mathrm{Z}(f_\theta(x_i)) + \lambda\, \mathrm{Z}(\log(\max\{d_{i,k}\}))\,,$$

where $f_\theta(x_i)$ is the evaluator's prediction, $d_{i,k}$ is the distance from x_i to its k-th nearest neighbor among the synthetic population, and $\lambda \geq 0$ controls the quality-diversity trade-off. Both terms are standardized using the Z-score to ensure comparability. The diversity regularization term favors samples farther from their neighbors. Moreover, early iterations require substantial shifts toward the private distribution, whereas later iterations focus on local refinement. Therefore, we use standard top-K selection, as in Aug-PE (Xie et al., 2024), for all conventional PE iterations and early self-evolving iterations. In later self-evolving iterations, we select only the highest-scoring sample from each family, consisting of a parent and its variations. We call this strategy *Top1PerFamily*. Although it may slow evolution, it encourages stable local refinement and limits excessive distribution shifts driven by evaluator biases. Top1PerFamily has also been shown to improve diversity compared with top-K selection (Yu et al., 2026).

Self-Evolving Private Evolution. Alg. 2 combines the components described above. Starting from a synthetic population $\mathcal{P}_0$ generated by `RANDOM_API` (Alg. 2, L3–8), `SelfEvo-PE` repeats a cycle of one conventional PE iteration followed by τ_{evo} self-evolving iterations for a total of T iterations (if–else condition in Alg. 2, L11,19). Since self-evolving iterations incur no additional privacy cost, this allows roughly $(\tau_{\text{evo}} + 1)$ times as many iterations as conventional PE under the same privacy budget and noise multiplier.

In each conventional PE iteration, `DP_NN_HISTOGRAM` computes noisy vote counts for the current population $\mathcal{P}t$ (Alg. 2, L14). These noisy voting results are accumulated across classes and

Algorithm 2 Self-Evolving Private Evolution – (New steps are highlighted)

Input: Classes C, private dataset $\mathcal{D}_{\text{priv}}$, noise multiplier σ, number of iterations T, number of free-selection iterations T_{free}, variation degree m, number of synthetic samples N, number of self-evolving steps τ_{evo}
Output: Synthetic dataset $\mathcal{D}_{\text{syn}}$
for each class $c \in C$ **do**
  # Initialize population by prompting LLM APIs
  $\mathcal{D}^{(c)}_{\text{priv}} \leftarrow \{x \in \mathcal{D}_{\text{priv}} : y(x) = c\}$
  $N^{(c)} \leftarrow N \cdot |\mathcal{D}^{(c)}_{\text{priv}}|/|\mathcal{D}_{\text{priv}}|$ *# Assuming the class distribution is known, as in previous settings (Lin et al., 2024; Xie et al., 2024). All the baselines in our experiments (Sec. 4) employ this assumption*
  $\mathcal{P}^{(c)}_0 \leftarrow \texttt{RANDOM_API}(m \cdot N^{(c)})$
end for
$\mathcal{Z} \leftarrow \emptyset$ *# Evaluator training set*
for $t = 0$ **to** $T - 1$ **do**
  if $t \bmod (\tau_{\text{evo}} + 1) == 0$ **then**
    # conventional PE iteration
    for each class $c \in C$ **do**
      $\tilde{h}^{(c)}_t \leftarrow \texttt{DP_NN_HISTOGRAM}\big(\mathcal{D}^{(c)}_{\text{priv}}, \mathcal{P}^{(c)}_t, \sigma\big)$ *# the only access to $\mathcal{D}_{priv}$*
      $\mathcal{D}^{(c)}_t \leftarrow \text{TopK}\left(\mathcal{P}^{(c)}_t, \tilde{h}^{(c)}_t, N^{(c)}\right)$
      $\mathcal{Z} \leftarrow \mathcal{Z} \cup \{(x, \tilde{h}^{(c)}_t(x)) : x \in \mathcal{P}^{(c)}_t\}$ *# store data for evaluator training*
    end for
    $\theta \leftarrow \arg\min_\theta \sum_{(x,\tilde{h})\in\mathcal{Z}} \mathcal{L}\big(f_\theta(x), \tilde{h}\big)$ *# train evaluator on accumulated data*
  else
    # self-evolving iteration
    for each class $c \in C$ **do**
      $s^{(c)}_t(x)$ for all $x \in \mathcal{P}^{(c)}_t$ *# score synthetic samples by evaluator*
      if $t < T_{\text{free}}$ **then**
        $\mathcal{D}^{(c)}_t \leftarrow \text{TopK}\left(\mathcal{P}^{(c)}_t, s^{(c)}_t, N^{(c)}\right)$ *# free top-K selection*
      else
        $\mathcal{D}^{(c)}_t \leftarrow \text{Top1PerFamily}\left(\mathcal{F}^{(c)}_t, s^{(c)}_t\right)$ *# select top-1 sample per parent family*
      end if
    end for
  end if
  # apply variation APIs to generate the next population
  for each class $c \in C$ **do**
    $(\mathcal{P}^{(c)}_{t+1}, \mathcal{F}^{(c)}_{t+1}) \leftarrow \texttt{VARIATION_API}(\mathcal{D}^{(c)}_t, m)$ *# generate variants and group them into families*
  end for
end for
$\mathcal{D}_{\text{syn}} \leftarrow \bigcup_{c\in C} \mathcal{D}^{(c)}_{T-1}$
return $\mathcal{D}_{\text{syn}}$

iterations to train the evaluators (Alg. 2, L18). In self-evolving iterations, the trained evaluator scores synthetic samples directly, avoiding additional access to the private data (Alg. 2, L22). The high-scoring samples (either by private voting or evaluator scoring) are selected based on two strategies: top-K selection for PE iterations (Alg. 2, L15) and early self-evolving iterations (Alg. 2, L24), and Top1PerFamily for later self-evolving iterations (Alg. 2, L26). Finally, the selected samples $\mathcal{D}_t$ are used to generate the next population $\mathcal{P}_{t+1}$ through `VARIATION_API` (Alg. 2, L32). The process continues until the maximum number of iterations T is reached, at which point the final synthetic dataset $\mathcal{D}_{\text{syn}}$ is returned. We provide the privacy analysis of `SelfEvo-PE` in Appx. A.1.

4 Experiments and Results

4.1 Overall Evaluation

General Setup. We evaluate `SelfEvo-PE` on three datasets that have been widely used in the literature (Hou et al., 2025; Xie et al., 2024): (1) BioRxiv (Hou et al., 2025) – biomedical research

paper abstracts, (2) Congressional Speech (Hou et al., 2025) – speeches from the US, UK, and Canadian parliaments, and (3) OpenReview (Xie et al., 2024) – peer paper reviews. While previous works use the entire dataset (ranging from 10K to 200K samples) as the private data, we focus on the more challenging and practical setting of limited private data scenarios (a few thousand samples). Our experiments are conducted under three strict privacy budgets $\epsilon \in \{0.2, 0.5, 1.0\}$. We set $\delta = 1/\left(|D_{\text{priv}}| \cdot \ln |D_{\text{priv}}|\right)$, which is a common choice in the DP literature (Yue et al., 2023). All methods generate the same number of synthetic samples ($0.25 \times |\mathcal{D}_{priv}|$) and are evaluated on three widely used metrics: (1) **FID** (Fréchet Inception Distance) (Heusel et al., 2018) – distance between the real and synthetic data distributions, (2) **MAUVE** (Pillutla et al., 2021) – distributional similarity that was designed for text data, and (3) **Classification Accuracy** – the performance of a downstream classifier trained on the synthetic data and evaluated on a held-out test set of real data. For the LLM, we employ Llama 3.2 3B-Instruct (AI, 2024). The implementation details can be found in Appx. B.

Baselines. We consider two lines of differentially private synthetic data generation methods: LLM Fine-tuning and LLM Inference only. For the fine-tuning approach, we consider DP-FT (Yue et al., 2023; Kurakin et al., 2024), CTCL (Tan et al., 2025), and POPri (Hou et al., 2025). DP-FT directly fine-tunes an LLM using DP-SGD (Abadi et al., 2016). CTCL requires fine-tuning a (smaller) language model that generates text samples from topics. Meanwhile, POPri performs multiple rounds of LLM alignment. For the inference-only methods, we consider methods based on the private evolution framework (Lin et al., 2024) including Aug-PE (Xie et al., 2024) and ICL-PE, which employs in-context learning and is adapted from WASP (Zou et al., 2025). The implementation details can be found in Appx. B.2. It is worth noting that **fine-tuning methods are not comparable to inference-only methods as they differ in resource requirements and practical constraints** (Amin et al., 2025; Ponomareva et al., 2025). For example, fine-tuning requires computational resources, access to model weights, and the ability to perform gradient updates, which are not always feasible. In contrast, inference-only methods can be applied to online LLM APIs and mitigate the need for local computational resources, enabling deployment on edge devices, e.g., mobile phones (Research, 2025; Hou et al., 2024).

Dataset	Method	$\varepsilon = 0.2$			$\varepsilon = 0.5$			$\varepsilon = 1.0$		
		FID ↓	MAUVE ↑	Acc ↑	FID ↓	MAUVE ↑	Acc ↑	FID ↓	MAUVE ↑	Acc ↑
BioRxiv $\lvert\mathcal{D}\rvert = 3K$	*Zero-shot*	*($\epsilon = 0$) FID = 46.95 MAUVE = 0.326 Acc = 64.909*								
	CTCL	146.58	0.011	51.455	127.29	0.019	56.424	110.22	0.040	56.909
	DP-FT	33.03	0.631	66.303	28.55	0.732	68.788	26.75	0.779	66.905
	PO-Pri	36.07	0.650	67.030	36.35	0.606	68.303	35.94	0.643	67.152
	Aug-PE	43.63	0.431	67.657	41.91	0.478	68.263	40.62	0.667	69.616
	ICL-PE	42.72	0.442	68.424	41.89	0.438	69.253	40.99	0.541	69.030
	SelfEvo-PE	**40.47**	**0.554**	**68.808**	**37.68**	**0.659**	**70.242**	**34.93**	**0.718**	**71.232**
Congressional Speech $\lvert\mathcal{D}\rvert = 1K$	*Zero-shot*	*($\epsilon = 0$) FID = 99.86 MAUVE = 0.069 Acc = 54.300*								
	CTCL	105.14	0.115	54.300	105.28	0.123	54.300	87.98	0.125	54.300
	DP-FT	72.37	0.223	52.723	50.87	0.459	52.511	34.765	0.910	54.300
	PO-Pri	41.59	0.714	31.233	38.81	0.669	31.233	41.40	0.824	31.233
	Aug-PE	43.60	0.718	54.300	43.95	0.704	54.300	41.09	0.762	54.833
	ICL-PE	46.59	0.503	54.300	45.93	0.537	53.767	42.97	0.604	54.300
	SelfEvo-PE	**41.06**	**0.803**	**55.267**	**41.60**	**0.797**	**54.900**	**39.74**	**0.812**	**55.767**
OpenReview $\lvert\mathcal{D}\rvert = 2K$	*Zero-shot*	*($\epsilon = 0$) FID = 68.09 MAUVE = 0.063 Acc = 35.025*								
	CTCL	84.02	0.030	35.919	75.19	0.034	37.098	78.22	0.041	34.846
	DP-FT	72.75	0.093	36.705	37.17	0.484	39.921	31.91	0.728	41.851
	PO-Pri	113.44	0.023	37.312	110.09	0.030	40.779	112.15	0.032	36.061
	Aug-PE	58.92	0.124	34.644	54.52	0.132	38.325	51.59	0.171	39.052
	ICL-PE	56.79	**0.138**	36.109	53.23	0.169	36.538	49.26	0.242	38.194
	SelfEvo-PE	**56.34**	0.129	**37.908**	**50.97**	**0.195**	**38.718**	**46.17**	**0.248**	**41.172**

Table 1: Comparison of methods across datasets and privacy budgets (ε). The background colors indicate methodology categories: LLM Fine-tuning and LLM Inference only. It is worth noting these two approaches impose different resource demands and practical constraints, making them not directly comparable (Amin et al., 2025; Ponomareva et al., 2025). The bold values represent the best among non-fine-tuning methods. The values of the inference-only methods are averaged over 3 runs, with standard deviations provided in Tab. 2 (Appx. C.2).

Overall Performance. Tab. 1 summarizes the performance of the methods across the datasets and privacy budgets. Among the fine-tuning methods, DP-FT performs better than CTCL, which employs a much smaller language model (140M), compared to the Llama 3.2 3B used by the other

methods. This is also the reason why CTCL performs even worse than the zero-shot baseline. Among the PE baselines, ICL-PE does not show clear improvement over Aug-PE. One possible reason is that the small 3B LLM does not have strong in-context learning ability. Additionally, even with LLM alignment, PO-Pri does not always outperform Aug-PE. Under this setting of limited private data and strict privacy budgets, the alignment data is limited and noisy, which can lead to suboptimal or wrong alignment, consistent with the findings of Zou et al. (2026). **`SelfEvo-PE` consistently improves upon existing inference-only methods**, achieving the lowest FID and highest classification accuracy across all datasets and privacy budgets, and the highest MAUVE in eight of nine settings. Although the two method classes (fine-tuning and inference-only) are not directly comparable, `SelfEvo-PE` achieves higher accuracy on BioRxiv and Congressional Speech across all privacy budgets, while DP-FT often achieves better FID and MAUVE.

4.2 Findings and Analysis

4.2.1 Impact of Noise Reduction and Number of Iterations

Noise Reduction. We calculate the noise multiplier of `SelfEvo-PE` and conventional PE in different settings (ϵ, number of iterations). With $\tau_{evo} = 1$, `SelfEvo-PE` halves the number of iterations that access private data, reducing the required Gaussian noise multiplier by a factor of $\sqrt{2}$ for the same privacy budget (Fig. 3). The absolute difference in noise multiplier σ between two methods increases as the number of evolution steps T increases and ϵ decreases (shown in Fig. 9, Appx. C.3).

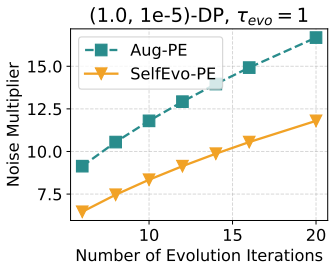

Figure 3: Noise multiplier values of `SelfEvo-PE` and Aug-PE under various configurations of the number of iterations T.

Number of Iterations. Reducing the number of iterations is a straightforward way to decrease the noise multiplier. However, it may also lead to worse synthetic data quality as the evolution process has not fully converged. To compare the two methods under the same noise multipliers, we vary the number of iterations from 2 to 20. As shown in Fig. 10 (Appx. C.3), `SelfEvo-PE` consistently outperforms Aug-PE in both FID and MAUVE across the configurations. This demonstrates **the source of performance gain is not only from noise reduction but also from the extra self-evolving steps**.

4.2.2 Analysis of Evaluators

Performance of Selection with Evaluators. We compare different selection strategies in an offline manner to ensure that they perform on the same set of samples. More specifically, we collect the samples generated by Aug-PE at each iteration and apply several selection strategies: (1) Random selection, (2) Top K on DP histogram (Aug-PE), (3) iterative top K on (noise-reduced) DP histogram and LTR models (`SelfEvo-PE`), and (4) Top K on clean histogram (Oracle, $\epsilon = \infty$). Each synthetic sample x is associated with a number of votes $h(x)$. Intuitively, a more effective selection strategy should yield a selected set $\mathcal{D}_t$ with a higher total vote count $\sum_{x \in \mathcal{D}_t}(h(x))$. As shown in Fig. 4 (left), **the samples selected by `SelfEvo-PE` carry more clean votes than those selected by Aug-PE across all iterations** by around 7%, leading to better end-to-end performance.

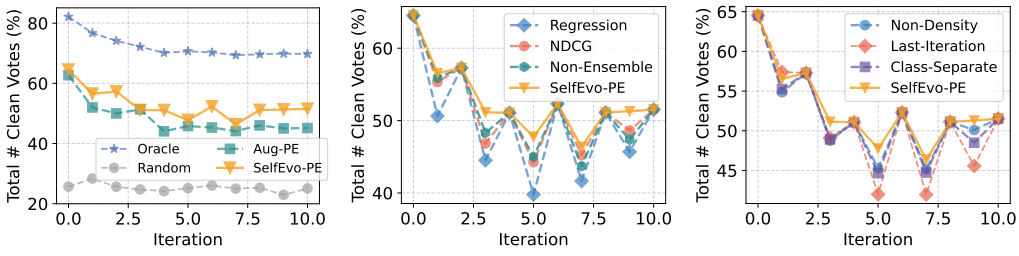

Figure 4: The total votes carried by the selected samples by different selection strategies at $\epsilon = 1.0$. We configure $\tau_{evo} = 1$, so at even-numbered iterations, all LTR methods have equal performance as they all perform top-K selection on DP histogram.

Evaluator Training Strategies. We consider several training strategies for the evaluators: (1) Regression, (2) Listwise LTR (NDCG) (Järvelin & Kekäläinen, 2002), (3) Pairwise LTR (Burges, 2010) without ensemble learning (Non-Ensemble) and (4) Pairwise LTR with ensemble learning (`SelfEvo-PE`). As shown in Fig. 4 (middle), **all the LTR methods perform better than Regression**, likely due to greater robustness to noise, as they consider the relative order of samples rather than the absolute quality scores. Pairwise LTR yields better performance than listwise LTR, as it considers the relative order of each pair at a time while listwise LTR considers the order of the whole list, which can be more sensitive to noise. Additionally, we analyze the training data sources and feature engineering in Fig. 4 (right). It is possible that samples from earlier iterations or from other classes are out of distribution of the current iteration/class. Specifically, we consider four strategies: (1) Non-Density uses only sample embeddings as input, excluding density-aware features; (2) Last-Iteration trains the evaluator using only data from the most recent iteration; (3) Class-Separate trains a separate evaluator for each class using only data from that class; and (4) `SelfEvo-PE` trains a shared evaluator using the full set of features and data from all previous iterations and classes. The results show that combining everything works best. We hypothesize that pairwise LTR with density-aware features can handle the distribution shift well as it can understand the relationship between the samples and the population distribution. Therefore, combining all data yields more training data and leads to better generalization of the evaluators.

Evaluator Compute Overhead. As we implement small XGBoost models as evaluators, **the compute overhead is not significant – 2.2% of the total run time**, illustrated in Fig. 5. Each PE round provides a fixed number of pairs (e.g., $|\mathcal{D}_{syn}|^2$), leading to a linear growth of training data with the number of PE rounds. We observe *the training time grows sublinearly with the training data size*. For instance, the first training round takes 29 seconds, while the second one with double the training data size costs 32 seconds. This is likely because most new training samples are drawn from a similar distribution as the previous rounds, thus increasing the sample counts does not necessarily increase the complexity of tree structures.

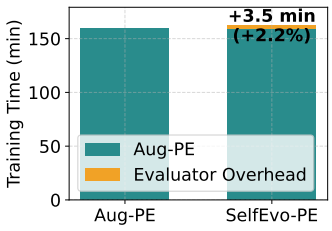

Figure 5: Run time of Aug-PE and `SelfEvo-PE` on a single RTX 6000 GPU. The compute overhead of evaluators is not significant.

4.2.3 Signal-To-Noise Ratio Study

`SelfEvo-PE` is effective in scenarios with limited private data or a need for a large amount of synthetic data. Generally, a higher signal-to-noise ratio (SNR) can lead to better performance of the selection on DP histogram. There are three main factors that significantly affect the SNR: (1) noise multiplier (σ), (2) private data size $|\mathcal{D}_{priv}|$, (3) synthetic data size $|\mathcal{D}_{syn}|$. Here, we focus on the effect of data sizes. Since each private sample casts one vote, the total vote count assigned to all synthetic samples equals $|\mathcal{D}_{priv}|$. Thus, a larger ratio $|\mathcal{D}_{priv}|/|\mathcal{D}_{syn}|$ leads to higher average vote counts per synthetic sample, which reduces the relative impact of noise and improves the SNR. We perform two experiments: one varying the private data size and the other varying the synthetic data size with $\epsilon = 0.5$ (more details in Appx. C.4). In both cases, `SelfEvo-PE` outperforms Aug-PE when the private data size is less than 16× the synthetic data size.

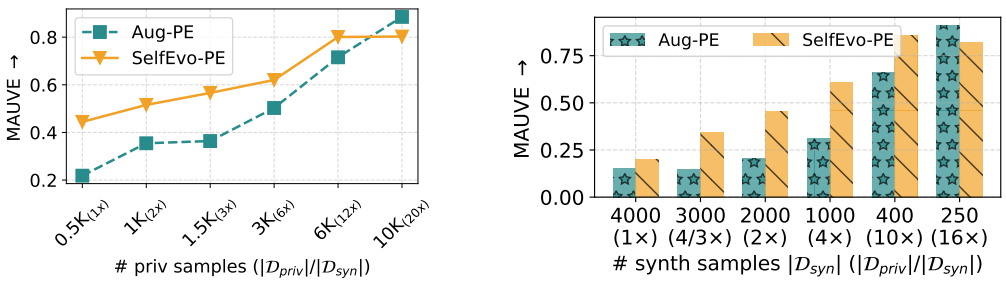

Figure 6: `SelfEvo-PE` performs better than Aug-PE under limited private data scenarios. We vary the private data size (left) and synthetic data size (right) under $\epsilon = 0.5$.

We provide additional ablation studies, hyperparameter sensitivity analyses, and settings in Appx. C.5 & Appx. C.6, including the selection strategy, self-evolving scoring function, and number of self-evolving iterations, as well as the performance of `SelfEvo-PE` *under additional settings: looser privacy budgets and a larger LLM.*

5 Related Works

Differentially Private Fine-Tuning (DP-FT). Differentially private synthetic text data generation has been based on the development of LLMs (Ponomareva et al., 2025). Large-scale pretrained LLMs offer great capabilities in generating fluent, coherent, semantically rich synthetic text. Early methods involve fine-tuning the LLMs with differentially private stochastic gradient descent (DPSGD) (Abadi et al., 2016) to shift the model behavior towards the private data distribution (Mattern et al., 2022; Yue et al., 2023; Kurakin et al., 2024; Zou et al., 2026). Since large-scale pretrained LLMs are generalized for language modeling, they may not be optimal for this task. To strengthen the model's ability for synthetic text generation, Tan et al. (2025) pretrained a small LLM from scratch to perform synthetic text generation from keywords. Meanwhile, recent works (Hu et al., 2025; Xu et al., 2026) explore the use of reinforcement learning to further enhance the quality of synthetic text data. While fine-tuning provides a direct way to steer the model distribution, it requires substantial computational resources as well as access to the LLM weights and permission to modify them. These requirements may not be feasible in many real-world scenarios, especially when the private data is hosted in low-resource environments or when the LLMs are proprietary and hosted by third-party providers.

Private Evolution (PE). To address the limitations of DP-FT, PE has emerged as an alternative approach that can work directly with black-box *inference* API access (Lin et al., 2024; 2025; Xie et al., 2024; Tran et al., 2026; Wang et al., 2025a;b). The original PE framework was proposed for image data generation (Lin et al., 2024) that performs an evolutionary process. It iteratively generates synthetic samples, evaluates their quality by querying the private data, and generates variations of the top-quality samples using foundation models. Over time, the synthetic data distribution shifts towards the private data distribution. Due to its practicality, PE has been extended to text in centralized (Xie et al., 2024) (named Aug-PE) and federated (Hou et al., 2024; 2025) settings. While Aug-PE relies primarily on the instruction-following capabilities of LLMs, recent works improve PE by enhancing variation generation through in-context learning (Zou et al., 2025), alignment (Hou et al., 2025), and multiple LLMs (Zou et al., 2025), as well as improving initialization with metadata (Chien et al., 2026). Despite these improvements, the core private data voting process of PE remains unchanged. We explore a new angle of PE, reusing the privatized voting results to guide the evolutionary process. Due to different aspects, the existing improvements on Aug-PE are orthogonal to our work and can be potentially combined with ours. Beyond differentially private synthetic data generation, PE has also been applied to other tasks such as private model training (Zou et al., 2026) and clustering (Humphries et al., 2026).

6 Conclusion

In this paper, we have introduced `SelfEvo-PE`, a novel method enabling PE to perform more evolutionary rounds without increasing privacy cost. `SelfEvo-PE` represents a new direction for PE: treating differentially private signals as persistent resources that can be stored, managed, and reused, rather than single-round information as in the existing methods. From this perspective, we developed an evaluator-guided self-evolving PE algorithm that is able to reuse the DP signals from previous rounds in an effective and efficient way. Across datasets, `SelfEvo-PE` enhances the performance under the limited-data and strict privacy settings. Although we only experiment with text generation, the idea can potentially be extended to other modalities, such as image and tabular data. While our design of evaluators is computationally efficient (accounting for only 2% of the total computation time), future work can explore more powerful evaluator designs such as uncertainty-aware and adaptive selection strategies. Additionally, another way of leveraging the evaluators is to filter out trivial candidates before private voting, potentially improving the signal-to-noise ratio of the remaining candidates.

REFERENCES

Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS'16, pp. 308–318. ACM, October 2016. doi: 10.1145/2976749.2978318. URL http://dx.doi.org/10.1145/2976749.2978318.

Meta AI. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Kareem Amin, Alex Bie, Weiwei Kong, Alexey Kurakin, Natalia Ponomareva, Umar Syed, Andreas Terzis, and Sergei Vassilvitskii. Private prediction for large-scale synthetic text generation. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 7244–7262, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.425. URL https://aclanthology.org/2024.findings-emnlp.425/.

Kareem Amin, Salman Avestimehr, Sara Babakniya, Alex Bie, Weiwei Kong, Natalia Ponomareva, and Umar Syed. Clustering and median aggregation improve differentially private inference, 2025. URL https://arxiv.org/abs/2506.04566.

Tucker Balch, Vamsi K. Potluru, Deepak Paramanand, and Manuela Veloso. Six levels of privacy: A framework for financial synthetic data, 2024. URL https://arxiv.org/abs/2403.14724.

Borja Balle and Yu-Xiang Wang. Improving the gaussian mechanism for differential privacy: Analytical calibration and optimal denoising, 2018. URL https://arxiv.org/abs/1805.06530.

Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, pp. 89–96, New York, NY, USA, 2005. Association for Computing Machinery. ISBN 1595931805. doi: 10.1145/1102351.1102363. URL https://doi.org/10.1145/1102351.1102363.

Chris J.C. Burges. From ranknet to lambdarank to lambdamart: An overview. Technical Report MSR-TR-2010-82, June 2010. URL https://www.microsoft.com/en-us/research/publication/from-ranknet-to-lambdarank-to-lambdamart-an-overview/.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pp. 785–794. ACM, August 2016. doi: 10.1145/2939672.2939785. URL http://dx.doi.org/10.1145/2939672.2939785.

Eli Chien, Yuzheng Hu, Ryan McKenna, Shanshan Wu, Zheng Xu, and Peter Kairouz. Maple: Metadata augmented private language evolution, 2026. URL https://arxiv.org/abs/2603.19258.

Jinshuo Dong, Aaron Roth, and Weijie J. Su. Gaussian differential privacy, 2019. URL https://arxiv.org/abs/1905.02383.

Cynthia Dwork. Differential privacy. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener (eds.), *Automata, Languages and Programming*, pp. 1–12, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-35908-1.

Fengyu Gao, Ruida Zhou, Tianhao Wang, Cong Shen, and Jing Yang. Data-adaptive differentially private prompt synthesis for in-context learning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=sVNfWhtaJC.

Matthew Gentzkow, Bryan Kelly, and Matt Taddy. Text as data. *Journal of Economic Literature*, 57(3):535–74, September 2019. doi: 10.1257/jel.20181020. URL https://www.aeaweb.org/articles?id=10.1257/jel.20181020.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium, 2018. URL https://arxiv.org/abs/1706.08500.

Charlie Hou, Akshat Shrivastava, Hongyuan Zhan, Rylan Conway, Trang Le, Adithya Sagar, Giulia Fanti, and Daniel Lazar. Pre-text: Training language models on private federated data in the age of llms, 2024. URL https://arxiv.org/abs/2406.02958.

Charlie Hou, Mei-Yu Wang, Yige Zhu, Daniel Lazar, and Giulia Fanti. Private federated learning using preference-optimized synthetic data. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=ZuaU2bYzlc.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

Yuzheng Hu, Ryan McKenna, Da Yu, Shanshan Wu, Han Zhao, Zheng Xu, and Peter Kairouz. Actg-arl: Differentially private conditional text generation with rl-boosted control, 2025. URL https://arxiv.org/abs/2510.18232.

Thomas Humphries, Zinan Lin, and Sergey Yekhanin. Pe-means: Improved differentially private k-means clustering through private evolution, 2026. URL https://arxiv.org/abs/2606.00342.

Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, October 2002. ISSN 1046-8188. doi: 10.1145/582415.582418. URL https://doi.org/10.1145/582415.582418.

Alexey Kurakin, Natalia Ponomareva, Umar Syed, Liam MacDermed, and Andreas Terzis. Harnessing large-language models to generate private synthetic text, 2024. URL https://arxiv.org/abs/2306.01684.

Zinan Lin, Sivakanth Gopi, Janardhan Kulkarni, Harsha Nori, and Sergey Yekhanin. Differentially private synthetic data via foundation model APIs 1: Images. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=YEhQs8POIo.

Zinan Lin, Tadas Baltrusaitis, Wenyu Wang, and Sergey Yekhanin. Differentially private synthetic data via apis 3: Using simulators instead of foundation model, 2025. URL https://arxiv.org/abs/2502.05505.

Justus Mattern, Zhijing Jin, Benjamin Weggenmann, Bernhard Schoelkopf, and Mrinmaya Sachan. Differentially private language models for secure data sharing, 2022. URL https://arxiv.org/abs/2210.13918.

Ryan McKenna, Yangsibo Huang, Amer Sinha, Borja Balle, Zachary Charles, Christopher A. Choquette-Choo, Badih Ghazi, George Kaissis, Ravi Kumar, Ruibo Liu, Da Yu, and Chiyuan Zhang. Scaling laws for differentially private language models, 2025. URL https://arxiv.org/abs/2501.18914.

Junki Mori, Kazuya Kakizaki, Taiki Miyagawa, and Jun Sakuma. Differentially private synthetic text generation for retrieval-augmented generation (rag), 2026. URL https://arxiv.org/abs/2510.06719.

Sharyl Nass, Laura Levit, and Lawrence Gostin. *Beyond the HIPAA Privacy Rule: Enhancing Privacy, Improving Health Through Research*. 02 2009. ISBN 978-0-309-12499-7. doi: 10.17226/12458.

Raza Nowrozy, Khandakar Ahmed, A. S. M. Kayes, Hua Wang, and Timothy R. McIntosh. Privacy preservation of electronic health records in the modern era: A systematic survey. *ACM Comput. Surv.*, 56(8), April 2024. ISSN 0360-0300. doi: 10.1145/3653297. URL https://doi.org/10.1145/3653297.

Krishna Pillutla, Swabha Swayamdipta, Rowan Zellers, John Thickstun, Sean Welleck, Yejin Choi, and Zaid Harchaoui. MAUVE: Measuring the gap between neural text and human text using divergence frontiers. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=Tqx7nJp7PR.

Natalia Ponomareva, Zheng Xu, H. Brendan McMahan, Peter Kairouz, Lucas Rosenblatt, Vincent Cohen-Addad, Cristóbal Guzmán, Ryan McKenna, Galen Andrew, Alex Bie, Da Yu, Alex Kurakin, Morteza Zadimoghaddam, Sergei Vassilvitskii, and Andreas Terzis. How to dp-fy your data: A practical guide to generating synthetic data with differential privacy, 2025. URL https://arxiv.org/abs/2512.03238.

Apple Machine Learning Research. Understanding aggregate trends for Apple Intelligence using differential privacy, April 2025. URL https://machinelearning.apple.com/research/differential-privacy-aggregate-trends.

Bowen Tan, Zheng Xu, Eric P. Xing, Zhiting Hu, and Shanshan Wu. Synthesizing privacy-preserving text data via finetuning *without* finetuning billion-scale LLMs. In *Forty-second International Conference on Machine Learning*, 2025. URL https://arxiv.org/abs/2503.12347.

Xinyu Tang, Richard Shin, Huseyin A Inan, Andre Manoel, Fatemehsadat Mireshghallah, Zinan Lin, Sivakanth Gopi, Janardhan Kulkarni, and Robert Sim. Privacy-preserving in-context learning with differentially private few-shot generation. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=oZtt0pRnOl.

Toan Tran, Arturs Backurs, Zinan Lin, Victor Reis, Li Xiong, and Sergey Yekhanin. Differentially private synthetic data via apis 4: Tabular data, 2026. URL https://arxiv.org/abs/2606.08259.

Vishnu Vinod, Krishna Pillutla, and Abhradeep Guha Thakurta. Invisibleink: High-utility and low-cost text generation with differential privacy, 2026. URL https://arxiv.org/abs/2507.02974.

Haoxiang Wang, Zinan Lin, Da Yu, and Huishuai Zhang. Synthesize privacy-preserving high-resolution images via private textual intermediaries. *arXiv preprint arXiv:2506.07555*, 2025a.

Shuaiqi Wang, Vikas Raunak, Arturs Backurs, Victor Reis, Pei Zhou, Sihao Chen, Longqi Yang, Zinan Lin, Sergey Yekhanin, and Giulia Fanti. Struct-bench: A benchmark for differentially private structured text generation. *arXiv preprint arXiv:2509.10696*, 2025b.

Chulin Xie, Zinan Lin, Arturs Backurs, Sivakanth Gopi, Da Yu, Huseyin Inan, Harsha Nori, Haotian Jiang, Huishuai Zhang, Yin Tat Lee, Bo Li, and Sergey Yekhanin. Differentially private synthetic data via foundation model apis 2: text. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Fangyuan Xu, Sihao Chen, Zinan Lin, Taiwei Shi, Sydney Graham, Pei Zhou, Mengting Wan, Alex Stein, Virginia Estellers, Charles Chen, Morris Sharp, Richard Speyer, Tadas Baltrusaitis, Jennifer Neville, Eunsol Choi, and Longqi Yang. Dp-rft: Learning to generate synthetic text via differentially private reinforcement fine-tuning, 2026. URL https://arxiv.org/abs/2602.18633.

Ying Yin and Ivan Habernal. Privacy-preserving models for legal natural language processing. In *Proceedings of the Natural Legal Language Processing Workshop 2022*, pp. 172–183. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.nllp-1.14. URL http://dx.doi.org/10.18653/v1/2022.nllp-1.14.

Minseo Yu, Yerin Yoon, Jueun Lee, Heonseok Ha, and Ho Bae. Differentially private synthetic data generation with diversity via APIs, 2026. URL https://openreview.net/forum?id=csD5GiGGFc.

Xiang Yue, Huseyin A. Inan, Xuechen Li, Girish Kumar, Julia McAnallen, Hoda Shajari, Huan Sun, David Levitan, and Robert Sim. Synthetic text generation with differential privacy: A simple and practical recipe, 2023. URL https://arxiv.org/abs/2210.14348.

Xuancheng Zhu, Guoshun Nan, Han Zhang, Ben Niu, Yang Yue, Zixu Wang, Yilian Liu, Min Lei, and Xiaofeng Tao. Selpe: Progressive selection for private structured text synthesis, 2026. URL https://arxiv.org/abs/2606.22817.

Tianyuan Zou, Yang Liu, Peng Li, Yufei Xiong, Jianqing Zhang, Jingjing Liu, Xiaozhou Ye, Ye Ouyang, and Ya-Qin Zhang. Contrastive private data synthesis via weighted multi-PLM fusion. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=oRdfFS7xO5.

Tianyuan Zou, Zinan Lin, Sivakanth Gopi, Yang Liu, Ya-Qin Zhang, Robert Sim, Xin Deng, and Sergey Yekhanin. PE-SGD: Differentially private deep learning via evolution of gradient subspace for text. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=713ywmTZHv.

Appendix

Due to the space limit, we present additional methodology details, results, and analysis in the Appendix.

A Methodology **15**

A.1 Privacy Analysis 15

A.2 Threat Model 16

B Implementation Details **16**

B.1 Datasets 16

B.2 Baselines 16

C Experiments and Results **17**

C.1 Motivating Observation 17

C.2 Main Evaluation Variance 17

C.3 Noise multiplier analysis 19

C.4 Signal-To-Noise Ratio Study 20

C.4.1 Private Dataset Size Analysis 20

C.4.2 Synthetic Dataset Size Analysis 21

C.5 Additional Ablation Studies 21

C.5.1 Selection Strategy 21

C.5.2 Number of self-evolution iterations. 24

C.5.3 Number of free top-K iterations. 25

C.6 Additional Experiment Settings 25

C.6.1 Congressional Speech Dataset with Loose Privacy Budget $\epsilon = 4.0$ 25

C.6.2 ICLR 2026 reviews 26

C.6.3 `SelfEvo-PE` with Qwen 2.5 7B 26

D Limitations **26**

A METHODOLOGY

A.1 PRIVACY ANALYSIS

We analyze `SelfEvo-PE` using the Gaussian accounting argument of the original Private Evolution (Lin et al., 2024). The same mechanism has been implemented in multiple PE works (Xie et al., 2024; Hou et al., 2024; 2025; Lin et al., 2025; Tran et al., 2026; Yu et al., 2026; Chien et al., 2026).

Setting and Iteration Schedule. For simplicity, we consider one class. The extension to multiple classes is straightforward and follows the parallel composition of the Gaussian mechanism. `SelfEvo-PE` runs T iterations, each conventional PE iteration followed by τ_{evo} self-evolving iterations. The total number of private voting rounds is T_{PE}.

$$T_{PE} = \left\lceil \frac{T}{\tau_{evo} + 1} \right\rceil$$

The self-evolving iterations are post-processing of the previous noisy voting results, which do not consume additional privacy budget. Therefore, the overall privacy guarantee is determined by the T_{PE} private voting rounds.

Lemma A.1 (Sensitivity of private voting)**.** *For fixed populations $\mathcal{P}$, the concatenated histogram has global ℓ_2 sensitivity one under add/remove adjacency.*

Proof. Adding a record z changes only the bin corresponding to $x^\star(z;\mathcal{P})$ – the index of the candidate in $\mathcal{P}$ that is closest to z in embedding space. Every existing record's vote is unchanged, so $h(\mathcal{D}_{\text{priv}} \cup \{z\}) - h(\mathcal{D}_{\text{priv}})$ is a standard basis vector and has ℓ_2 norm one. For removal, the same argument applies, with the difference that the change is in the opposite direction. □

Theorem A.2 (Privacy of `SelfEvo-PE`)**.** *`SelfEvo-PE` satisfies (ϵ, δ)-differential privacy $\mathcal{D}_{\text{syn}} = \mathcal{D}_{T-1}$ if $\epsilon \geq 0$ and $0 < \delta < 1$ satisfying*

$$\Phi\left(\frac{\sqrt{T_{\text{PE}}}}{2\sigma} - \frac{\epsilon\sigma}{\sqrt{T_{\text{PE}}}}\right) - e^{\epsilon}\Phi\left(-\frac{\sqrt{T_{\text{PE}}}}{2\sigma} - \frac{\epsilon\sigma}{\sqrt{T_{\text{PE}}}}\right) \leq \delta, \tag{1}$$

where Φ is the standard normal cumulative distribution function.

Proof. We follow the same argument as in the original PE (Lin et al., 2024).

Step 1: *Sensitivity of private voting.* Across the algorithm, private voting is the only operation that accesses the private dataset. By Theorem A.1, the sensitivity of the voting histogram is one.

Step 2: *Gaussian mechanism for one private voting round.* For each private voting round, the noisy histogram is obtained by adding Gaussian noise with standard deviation σ to each bin of the histogram.

$$\tilde{h} = h + \xi, \quad \xi \sim \mathcal{N}(0, \sigma^2 I)$$

Since the histogram has sensitivity one, each private voting round is $1/\sigma$-Gaussian differentially private.

Step 3: *Adaptive composition of private voting rounds.* Across `SelfEvo-PE`, there are exactly T_{PE} private voting rounds. Evaluator training and self-evolving iterations use only synthetic samples and previously obtained noisy voting results, so they are post-processing and incur no additional privacy cost. Thus, the sensitivity bound in Step 1 applies to every conditional private query. By adaptive Gaussian DP composition (Dong et al., 2019), the sequence of T_{PE} private voting rounds can be viewed as a single Gaussian mechanism with sensitivity $\sqrt{T_{\text{PE}}}$. Therefore, the overall mechanism is $\sqrt{T_{\text{PE}}}/\sigma$-Gaussian differentially private.

Step 4: *Applying analytic Gaussian calibration.* For a Gaussian mechanism with ℓ_2 sensitivity Δ and noise standard deviation σ, the analytic calibration condition for (ϵ, δ)-DP is given by (Balle & Wang, 2018):

$$\Phi\left(\frac{\Delta}{2\sigma} - \frac{\epsilon\sigma}{\Delta}\right) - e^{\epsilon}\Phi\left(-\frac{\Delta}{2\sigma} - \frac{\epsilon\sigma}{\Delta}\right) \leq \delta$$

The composed guarantee in Step 3 matches the privacy bound of a Gaussian mechanism with sensitivity $\Delta = \sqrt{T_{\text{PE}}}$ and noise standard deviation σ. Substituting these values yields Equation 1, while σ remains the standard deviation of the noise added in each private voting round. □

A.2 Threat Model

We consider the same threat model in the literature (Xie et al., 2024; Lin et al., 2024), where a trusted data curator holds a private dataset $\mathcal{D}_{\text{priv}}$ and untrusted LLM API providers. An adversary can observe the released synthetic dataset $\mathcal{D}_{\text{syn}}$, the generation API requests/responses, and the evaluators. Our goal is to provide a formal privacy guarantee for this joint view. Private records, their embeddings, and clean voting histograms remain within the trusted environment, where the nearest-neighbor voting and Gaussian noise addition are performed.

The new operations introduced in `SelfEvo-PE`, including the training of evaluators and the self-evolving iterations, are post-processing steps that satisfy the post-processing property of differential privacy. Therefore, they do not introduce any additional privacy leakage and these operations can be performed in untrusted environments. In summary, **`SelfEvo-PE` shares the same threat model as the previous PE works and does not require any additional trust assumptions**.

B Implementation Details

B.1 Datasets

Following the previous works in the literature (Xie et al., 2024; Tan et al., 2025; Hou et al., 2025; Zou et al., 2026), we perform experiments on three widely used datasets.

BioRxiv (Hou et al., 2025) This dataset contains the abstracts of preprints in the biomedical domain. The classification task aims to predict the category of the preprint. We use a subset of 3K samples containing 6 classes.

Congressional Speech (Hou et al., 2025) This dataset includes speeches and debates from the congressional record. The classification task is to predict the source of the speech (i.e., US, UK, or Canada). This dataset contains only 3 classes, which results in a larger number of samples per class. We randomly subsample 1K samples from the original dataset for our main evaluation experiment.

OpenReview (Xie et al., 2024) This dataset consists of ICLR 2023 reviews. The classification task is to classify each review into one of the eleven categories of machine learning topics. We use a subset of 2000 samples from the original dataset.

B.2 Baselines

By default, we employ Llama-3.2-3B-IT for all the methods (except CTCL).

DP-FT (Kurakin et al., 2024) DP-FT refers to the differentially private fine-tuning method that applies DP-SGD (Differentially Private Stochastic Gradient Descent) to fine-tune a pre-trained language model on the private dataset. To enhance the performance of DP-FT and reduce the computational cost, (1) we employ LoRA (Hu et al., 2021) – a parameter-efficient fine-tuning (PEFT) method that only updates a small subset of the model parameters during training. PEFT has been shown to be more effective than full fine-tuning when the training optimization is constrained by DP (Kurakin et al., 2024); (2) we employ the gradient accumulation technique to increase the batch size, which is crucial for improving the performance of DP-SGD (McKenna et al., 2025). (3) We tune the learning rate within {1e-4, 5e-4, 1e-3, 5e-3}. For conditional generation tasks, we include the label information as part of the input prompt and perform loss calculation only on the text generation part. During inference, we generate a synthetic set that has the same class distribution as the original private dataset, with the assumption that the class distribution is public information.

CTCL (Tan et al., 2025) We reuse the DP-FT fine-tuning process to fine-tune the CTCL pre-trained model checkpoint, which receives the keywords and generates the synthetic data. However,

following the original paper, we do not employ LoRA as the model architecture of CTCL is already parameter-efficient (140M). We tune the learning rate within {1e-4, 5e-4, 1e-3, 5e-3}. During inference, we also generate a synthetic set that has the same class distribution as the original private dataset.

Aug-PE (Xie et al., 2024) We reuse the implementation from `https://github.com/microsoft/DPSDA`. **All the PE methods share the same hyperparameters (number of variations, sampling configurations) and prompt designs of `RANDOM_API` and `VARIATION_API` (except ICL-PE), ensuring a fair comparison**.

ICL-PE We adopt the idea from (Zou et al., 2025) that utilizes in-context learning (ICL) to guide the LLM variation generation. We remove some original designs of (Zou et al., 2025) that are specialized for multiple LLM APIs. More specifically, we use the top 10% of high-vote samples as the positive examples and the bottom 10% of low-vote samples as the negative examples. The ICL prompt randomly picks one positive example and one negative example each time to generate a variation. The remaining configurations are the same as those of Aug-PE. It is worth noting that due to the ICL demonstrations, the prompt of ICL-PE is significantly longer than that of Aug-PE, which leads to higher computational cost and LLM token consumption.

PO-Pri We reuse the implementation from the original paper. The random initialization of the synthetic dataset is implemented by Aug-PE's `RANDOM_API` with the same hyperparameters and prompt design. The hyperparameters of the alignment process are kept as the default settings in the original paper. Each iteration, PO-Pri requires performing LLM fine-tuning for alignment, which is computationally expensive, so we reduce the number of rounds from 30 to 10 to make it feasible for our experiments.

`SelfEvo-PE` We implement `SelfEvo-PE` based on the DPSDA codebase. To ensure a fair comparison, the conventional steps are identical to Aug-PE's implementation, including the random initialization of the synthetic dataset, the hyperparameters, and the prompt design of `RANDOM_API` and `VARIATION_API`. For training evaluators, we implement ensemble learning of 20 XGBoost models. Each model is trained on different train/validation splits. The final prediction score is the prediction average of all the models. For the scoring function, we configure λ to 0.5. For the self-evolving process, we set the number of free-selection iterations T_{free} to 2 and the number of self-evolving steps τ_{evo} to 1 by default.

C Experiments and Results

C.1 Motivating Observation

Using the top 5% of samples ranked by vote count in earlier iterations as a reference, we compare the text similarity of selected versus unselected samples in subsequent iterations. A gap in similarity between the two sets of samples demonstrates that the top voted samples of the previous iteration can provide meaningful signals to distinguish the selected and unselected samples of the later iterations. Fig. 7 provides the similarities between the top samples of Iteration $t-1$ to the sample sets of Iteration t. The similarity gap is significant and consistent across iterations. This correlation can last for several iterations instead of only a single one. Fig. 8 presents the similarity between the top samples of the **first** iteration to the sets across iterations. The similarity gap is more significant in the early iterations (1 to 5) but gradually diminishes as the iteration progress, likely due to the increasing distribution shift – the closer iterations yield less distribution shift. These observations together suggest that the previous iteration's results are informative in predicting not only the immediately subsequent iteration's selection results but also the long-term ones. These observations together motivate us to leverage the previous iteration voting results to guide the selection in the following iterations, which is the core idea of `SelfEvo-PE`.

C.2 Main Evaluation Variance

Tab. 2 provides the variance of the main evaluation results in Tab. 1. The standard deviation is calculated across 3 independent runs.

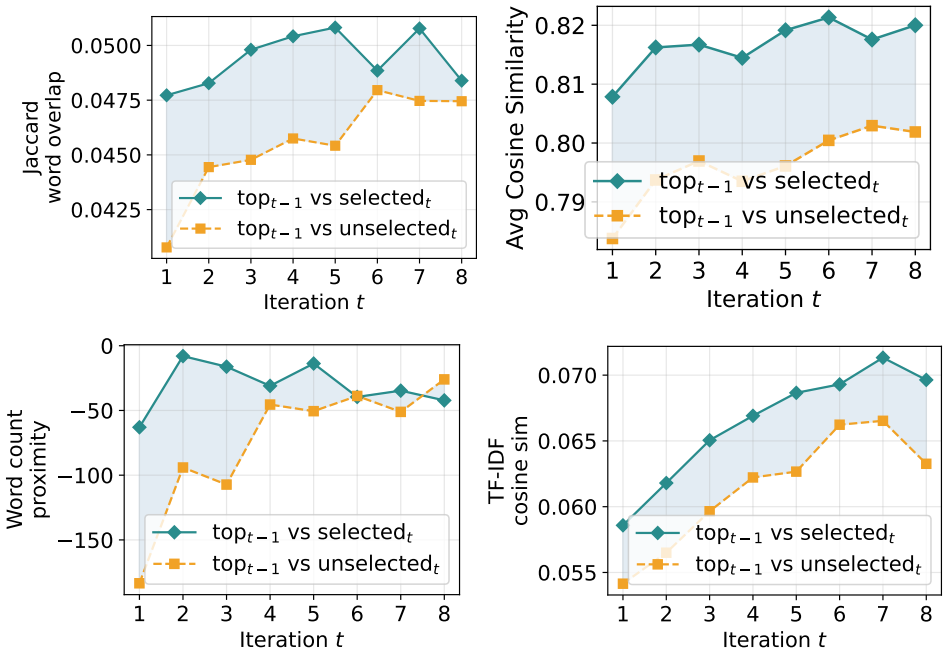

Figure 7: Text Analysis Similarity between the top 5% samples of the previous iteration $(t-1)$ to the selected and unselected sample sets of the following iteration (t).

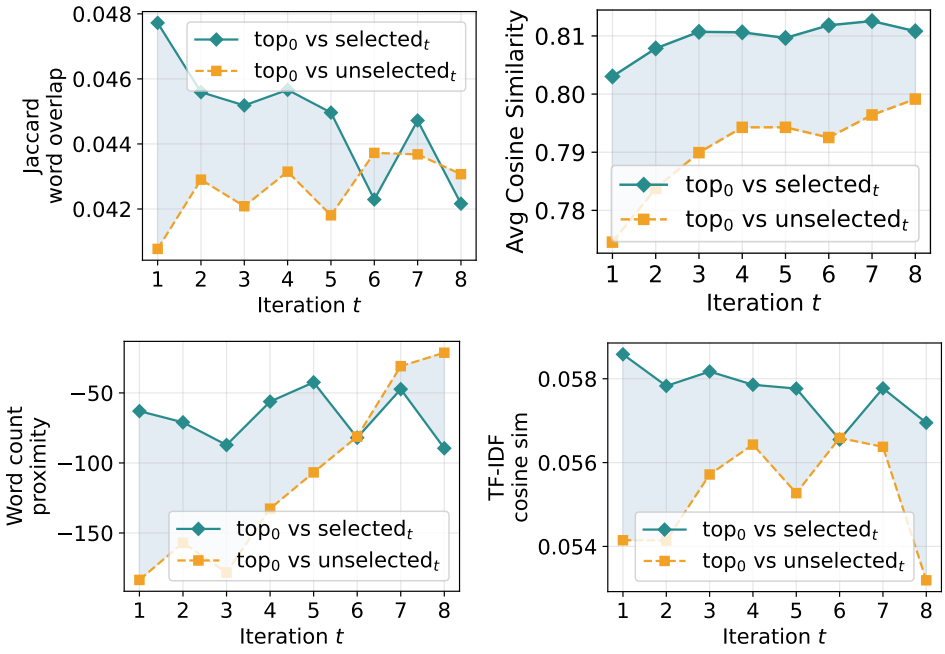

Figure 8: Text Analysis Similarity between the top 5% samples of the first iteration to the selected and unselected sample sets of the following iterations.

Dataset	Method	$\varepsilon = 0.2$			$\varepsilon = 0.5$			$\varepsilon = 1.0$		
		FID ↓	MAUVE ↑	Acc ↑	FID ↓	MAUVE ↑	Acc ↑	FID ↓	MAUVE ↑	Acc ↑
BioRxiv	Aug-PE	0.71	0.026	0.947	0.29	0.016	0.674	0.90	0.063	1.590
	ICL-PE	1.01	0.076	1.757	0.95	0.033	1.045	0.62	0.034	0.495
	SelfEvo-PE	1.34	0.002	0.700	0.69	0.036	0.680	0.88	0.034	0.114
Congress-ional Speech	Aug-PE	1.30	0.027	0.000	1.83	0.049	0.000	0.59	0.031	0.340
	ICL-PE	0.63	0.066	0.000	0.58	0.028	0.974	0.94	0.064	0.000
	SelfEvo-PE	1.18	0.039	1.367	0.74	0.006	0.455	0.64	0.026	0.478
OpenRe-view	Aug-PE	1.20	0.013	1.252	1.07	0.014	1.179	1.61	0.013	0.730
	ICL-PE	0.68	0.007	1.847	0.87	0.041	0.532	1.31	0.042	1.436
	SelfEvo-PE	0.67	0.018	0.546	1.90	0.026	0.532	0.34	0.033	0.890

Table 2: Standard deviation of the evaluation metrics across three runs of `SelfEvo-PE` and its direct baselines.

C.3 Noise multiplier analysis

We calculate the noise multiplier values of Aug-PE and `SelfEvo-PE` under various settings of ϵ, number of evolution iterations T, and number of self-evolving steps τ_{evo}. The results are shown in Fig. 9. Intuitively, `SelfEvo-PE` with $(\tau_{evo} + 1) \cdot T$ iterations has the same noise multiplier value as Aug-PE with T iterations, because there are $\tau_{evo} \cdot T$ self-evolving iterations that do not consume privacy cost. Although the benefit of noise multiplier reduction is better when τ_{evo} is larger, the performance of `SelfEvo-PE` does not necessarily improve as τ_{evo} increases. This is because the evaluator is trained on the synthetic data of a previous private round, the more self-evolving steps there are, the more distribution shift between the training data and the testing data. This distribution shift can lead to worse evaluation prediction, resulting in bad selection and overall synthetic data quality.

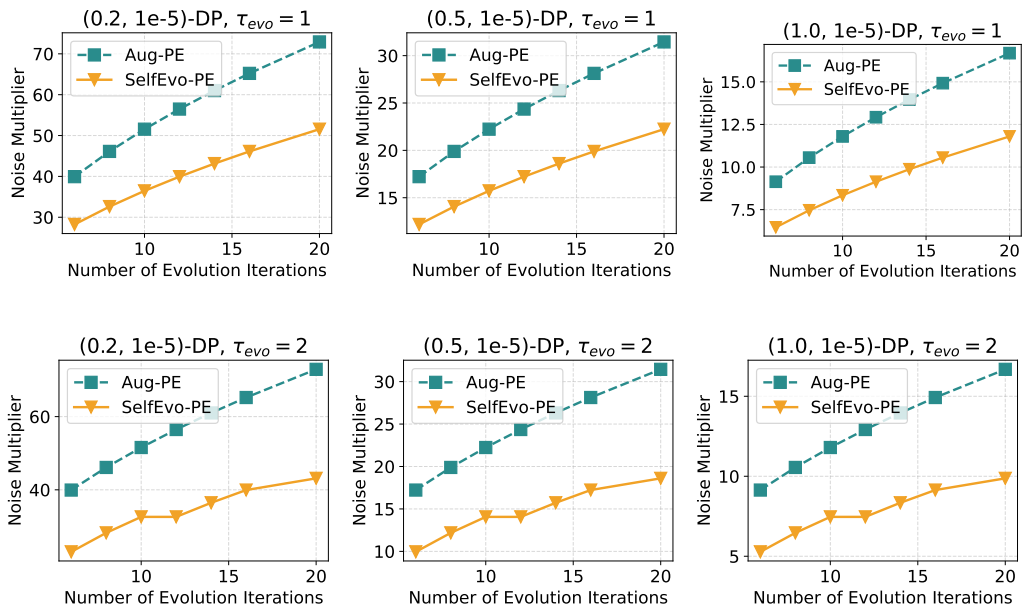

Figure 9: Noise multiplier values of Aug-PE and `SelfEvo-PE` under various settings of ϵ, number of evolution iterations T, and number of self-evolving steps τ_{evo}. Each point represents the noise multiplier value of a specific configuration.

Given the same number of iterations, `SelfEvo-PE` has a significantly smaller noise multiplier than Aug-PE. However, we can also reduce the number of iterations T of Aug-PE to make its noise multiplier comparable to that of `SelfEvo-PE`. In Fig. 10, we run Aug-PE and `SelfEvo-PE` with $\tau_{evo} = 1$ under various settings of number of iterations T {2, 4, 6, 8, 14, and 20}. We can see that even with the same noise multiplier, `SelfEvo-PE` still outperforms Aug-PE in most cases.

This suggests that the performance gain of SelfEvo-PE is not only due to the noise multiplier reduction but also due to the longer evolutionary process that allows the synthetic data to be refined more times.

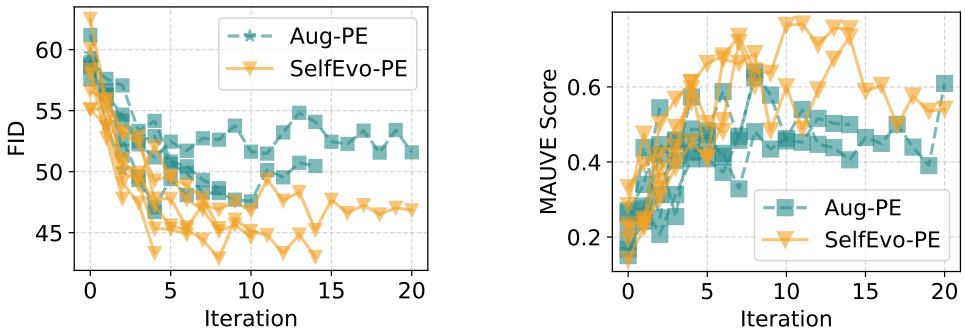

Figure 10: FID and MAUVE scores of Aug-PE and SelfEvo-PE ($\tau_{evo} = 1$) under various settings of number of evolution iterations {2, 4, 6, 8, 14, and 20}. Each line represents a configuration. SelfEvo-PE produces better synthetic data in most cases.

C.4 Signal-To-Noise Ratio Study

As mentioned in the main paper, the signal-to-noise ratio (SNR) for the selection signal ($\tilde{h}$) is dependent not only on the noise multiplier but also on the size of the private dataset and the synthetic dataset. Given a fixed noise multiplier, a larger ratio of $|\mathcal{D}_{priv}|/|\mathcal{D}_{syn}|$ leads to a higher SNR, as the average vote count per sample is larger, reducing the relative impact of noise. We conduct two experiments, where we vary the size of each set while keeping the noise multiplier and the size of the other set.

C.4.1 Private Dataset Size Analysis

We generate 500 synthetic samples while varying the number of private samples. As shown in Fig. 11, SelfEvo-PE performs better in scenarios with limited data (under 6K priv samples or $12\times|\mathcal{D}_{syn}|$), while Aug-PE performs well when the available data is large. When the private data size is small, the SNR is low, and the noise reduction of SelfEvo-PE can lead to more accurate selection and better performance. The performance gap is larger when the ratio is smaller. SelfEvo-PE with 1K private samples can generate the synthetic data of similar quality to that generated by Aug-PE with 3K private samples. When the private data size is significantly larger than the synthetic data size ($20\times$), the SNR is high, top K selection on DP histogram can already yield good performance. Reducing the noise multiplier can always help, but the gain is smaller while training the evaluators can also introduce some noise and biases. Fig. 12 provides the detailed FID and MAUVE scores of Aug-PE and SelfEvo-PE in this experiment across iterations.

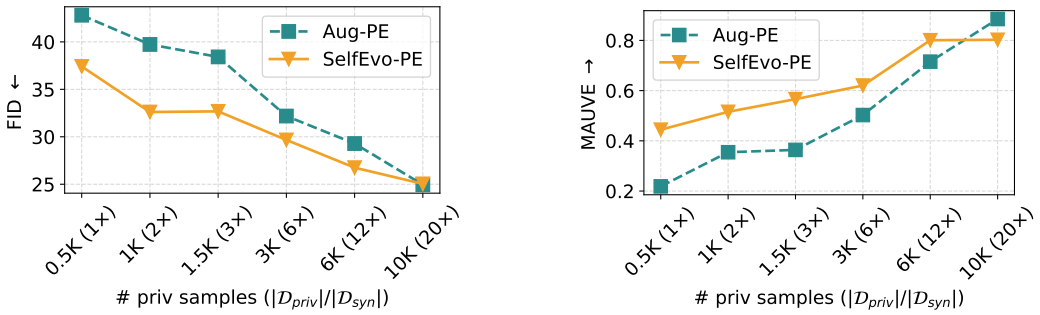

Figure 11: SelfEvo-PE performs better than Aug-PE under limited private data scenarios, while Aug-PE performs well when the available data is large.

C.4.2 SYNTHETIC DATASET SIZE ANALYSIS

We fix the number of private samples at 4K while varying the number of synthetic samples. As shown in Fig. 13, `SelfEvo-PE` is better at generating a larger number of samples (above 1/10 of the private data size), while Aug-PE performs well when the required number of samples is small (1/16 of private data size – when the SNR is high). Fig. 14 provides the detailed FID and MAUVE scores of Aug-PE and `SelfEvo-PE` in this experiment across iterations.

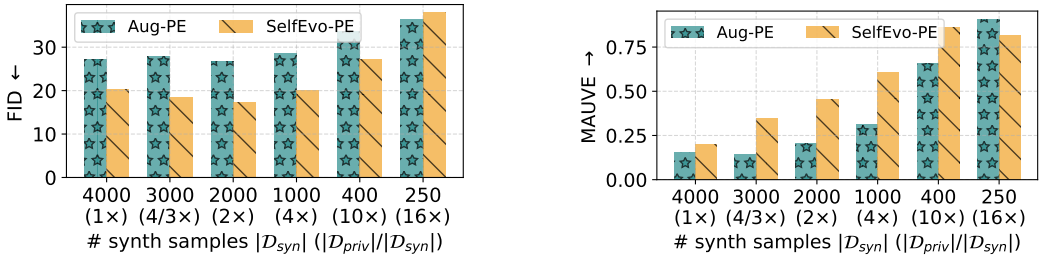

Figure 13: `SelfEvo-PE` performs better than Aug-PE when the required number of synthetic samples is large. Experiment on a fixed private set of 4K Congressional speech samples with $\epsilon = 0.5$. FID and MAUVE are not comparable across different numbers of synthetic samples, as their estimates are sample-size dependent.

C.5 ADDITIONAL ABLATION STUDIES

C.5.1 SELECTION STRATEGY

We consider a few strategies for selecting the samples for both the conventional PE and self-evolving iterations. Tab. 3 provides the results of this experiment. The first line `SelfEvo-PE` (full) stands for the default configuration of `SelfEvo-PE`. The second line "w/o Aug-PE Top-K selection", for each conventional PE iteration, instead of selecting samples using their noisy vote counts, we train the evaluator at this iteration and conduct inference on the training set to get the ranking scores and select the top K using the scores. This approach performs slightly worse than the default one. The third line "w/o self-evolving Top-K selection", for early self-evolving iterations, instead of performing free Top-K selection, we employ Top1PerFamily. This approach also shows limitations due to slower distribution shift, potentially requiring more iterations to converge. The last line "w/o Top1PerFamily, the MAUVE score drops 62% compared to the default configuration, as the selected sets at self-evolving iterations can overfit to the evaluator's prediction bias.

Method	**FID**	**MAUVE**	**Precision**	**Recall**
`SelfEvo-PE` (full)	**41.872**	**0.830**	0.712	**0.576**
w/o Aug-PE Top-K selection	44.955	0.793	0.644	0.497
w/o SelfEvo Top-K selection	44.794	0.599	0.784	0.362
w/o SelfEvo Top1PerFamily	62.667	0.313	**0.816**	0.217

Table 3: Performance of `SelfEvo-PE` with different sample selection strategies. `SelfEvo-PE` with both Top-K and Top1PerFamily selection achieves the best performance in terms of FID, MAUVE, and Recall. The one w/o SelfEvo Top1PerFamily selection achieves the best Precision, but it is at the cost of a significant drop in Recall. This indicates that the selected samples in this case are distributed in a very narrow region of the data manifold with limited diversity.

Additionally, we analyze the impact of the diversity regularization (DR) term in the self-evolving score function. Fig. 15 shows that both Aug-PE and `SelfEvo-PE` benefit from the diversity regularization, leading to improved FID and MAUVE scores. We also conduct a sensitivity analysis on the weight of the diversity regularization term λ. As shown in Fig. 16, a positive λ encourages selecting samples in sparse regions, while a negative λ promotes selecting samples in crowded regions. The performance of `SelfEvo-PE` improves with a positive λ and peaks at $\lambda = 1.0$.

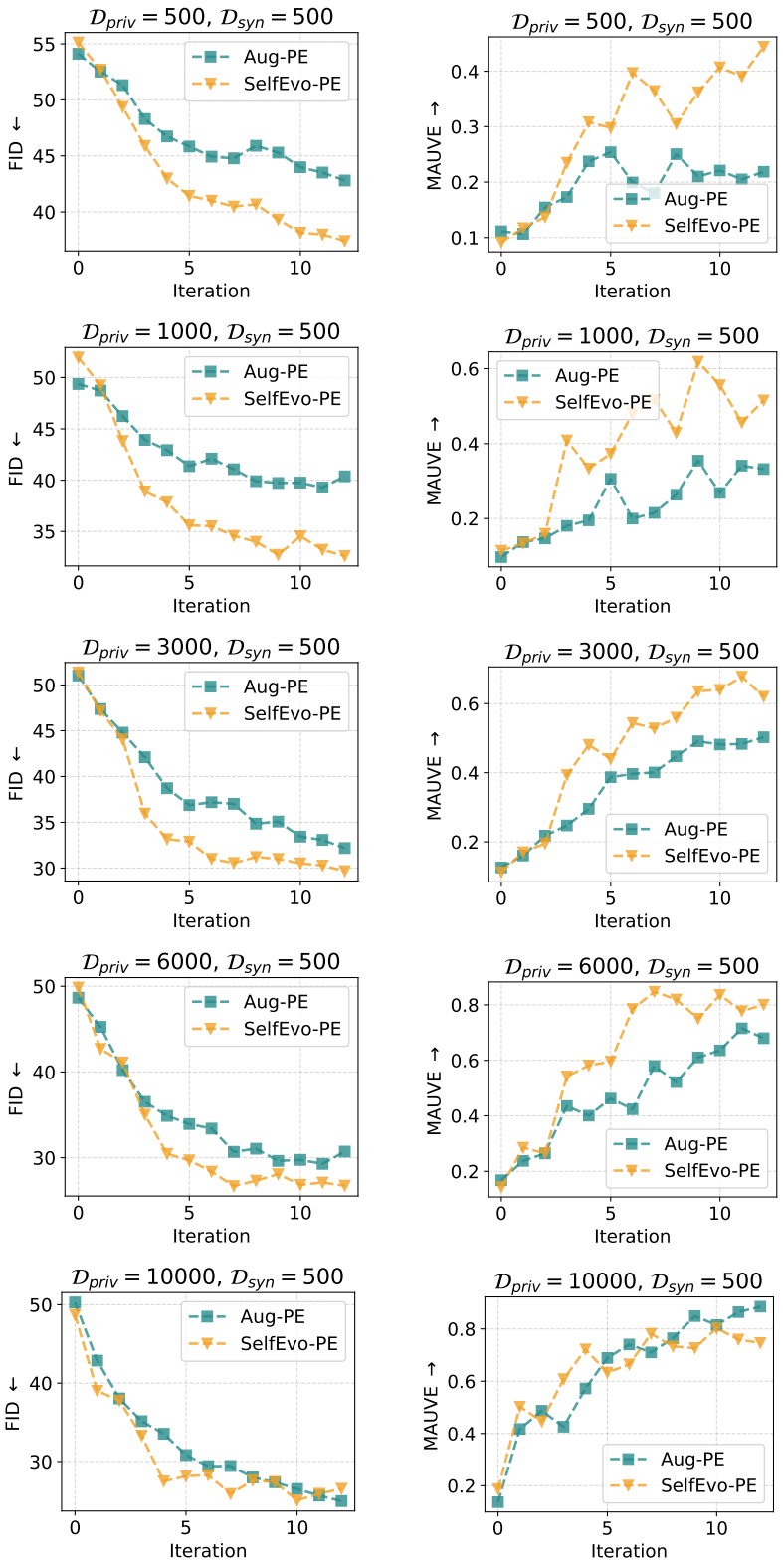

Figure 12: FID and MAUVE scores of Aug-PE and `SelfEvo-PE` ($\tau_{evo} = 2$) under various settings of private dataset size {500, 1000, 3000, 6000, and 10000}. `SelfEvo-PE` results better synthetic data in most cases until the private dataset size is substantially larger, at 10K.

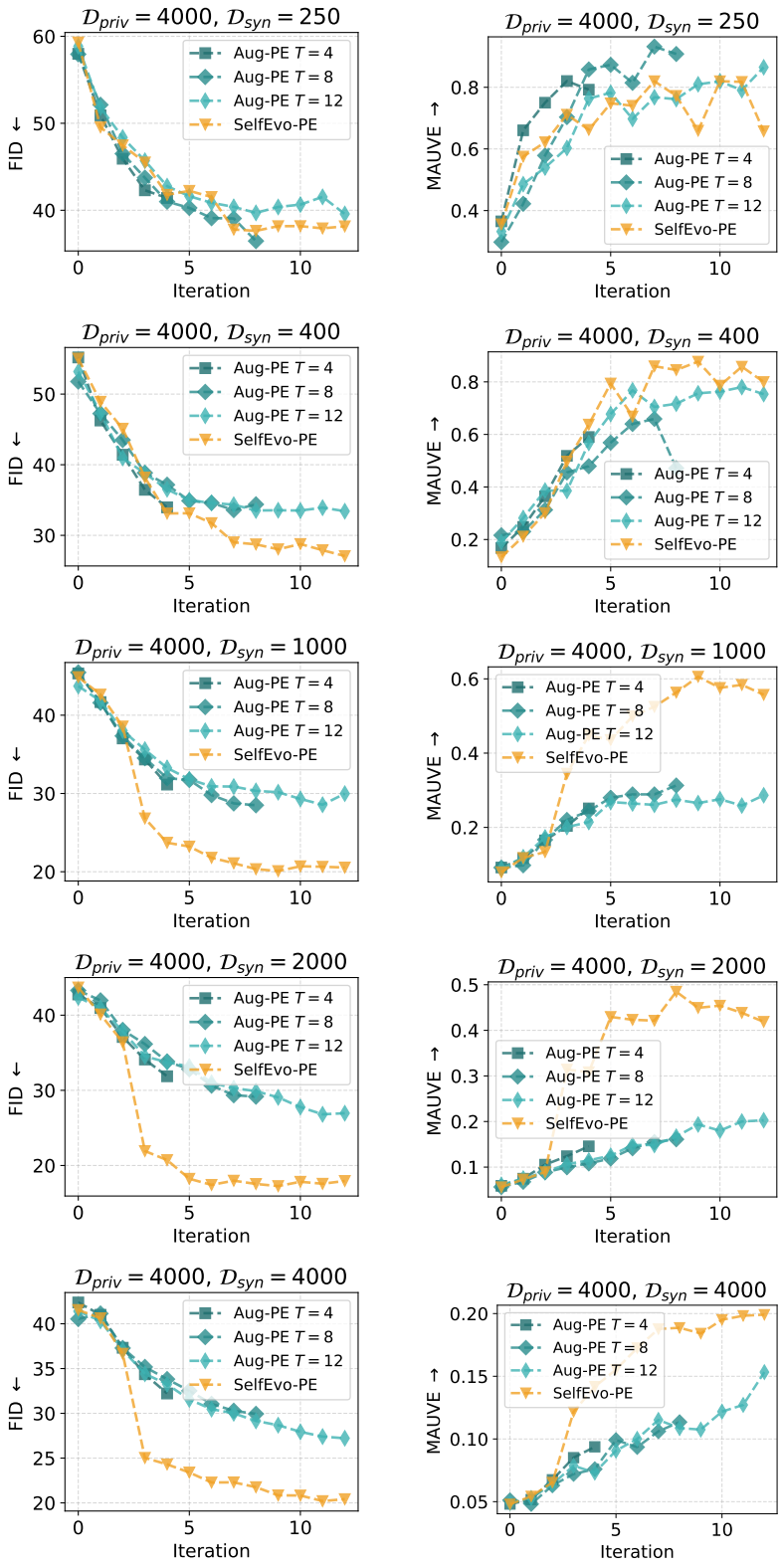

Figure 14: FID and MAUVE scores of Aug-PE and `SelfEvo-PE` ($\tau_{evo} = 2$) under various settings of synthetic dataset size {250, 400, 1000, 2000, and 4000}. `SelfEvo-PE` performs better in most settings where the number of generated synthetic samples exceeds 10% of the private dataset size.

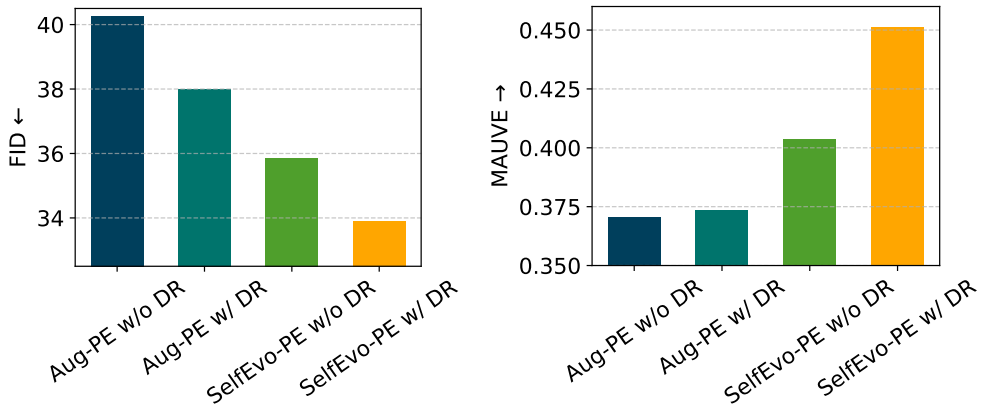

Figure 15: Ablation study on the diversity regularization (DR) of the self-evolving score function. The left and right subplots show the FID and MAUVE scores, respectively. The performance of both Aug-PE and `SelfEvo-PE` is improved with the diversity regularization.

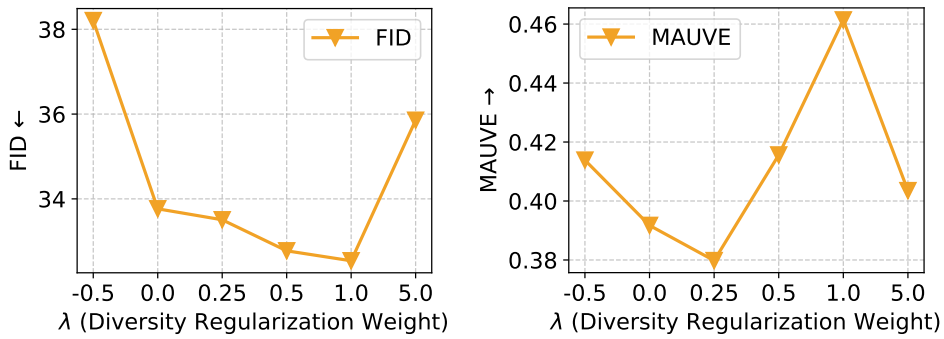

Figure 16: Sensitivity analysis of λ, the weight of the diversity regularization (DR) term in the self-evolving score function. A negative λ promotes selecting samples that are in crowded regions. In contrast, a positive λ encourages selecting samples that are in sparse regions. The left and right subplots show the FID and MAUVE scores, respectively. The performance of `SelfEvo-PE` is improved with a positive λ and peaks at $\lambda = 1.0$.

C.5.2 Number of self-evolution iterations.

Fig. 17 shows the performance of `SelfEvo-PE` with different numbers of self-evolution iterations τ_{evo}. Increasing τ_{evo} provides more evolutionary iterations, leading to better synthetic data quality but diminishing returns. The performance peaks at $\tau_{evo} = 2$ and then remains relatively stable with $\tau_{evo} \in \{3, 4\}$.

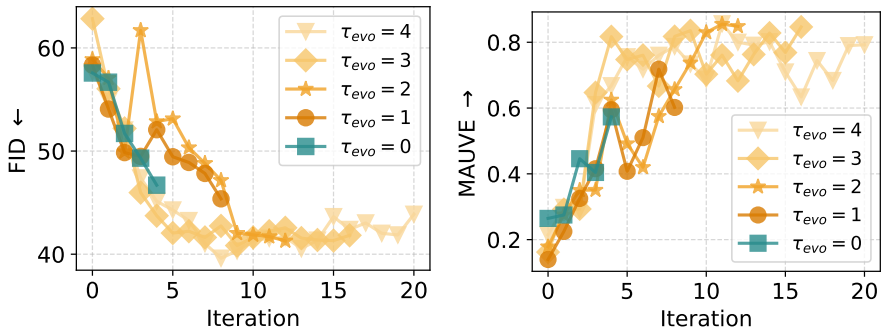

Figure 17: Increasing the number of self-evolving steps τ_{evo} provides more evolutionary iterations, leading to better synthetic data quality but diminishing returns. The setting of $\tau_{evo} = 0$ is equal to Aug-PE. `SelfEvo-PE` outperforms Aug-PE regardless of the number of self-evolving steps.

C.5.3 Number of free top-K iterations.

Fig. 18 shows the performance of `SelfEvo-PE` with different configurations of free-selection iterations T_{free}. If the value of T_{free} is small, it may slow down the evolutionary process, while a large value may lead to overfitting to the evaluator's bias. We found that $T_{free} = 4$ provides a good balance between the two aspects, leading to the best performance.

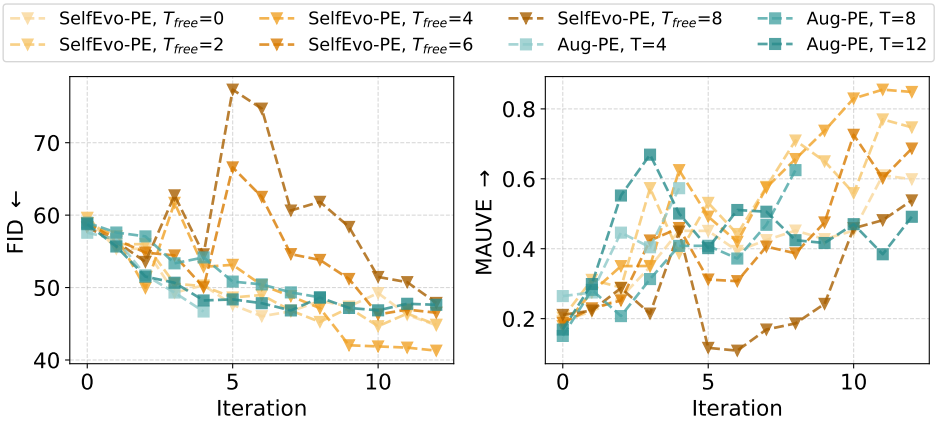

Figure 18: Analysis of the number of free-selection iterations T_{free}.

C.6 Additional Experiment Settings

C.6.1 Congressional Speech Dataset with Loose Privacy Budget $\epsilon = 4.0$

Under this loose privacy budget, the noise multiplier is significantly smaller. Under this setting, all methods can achieve very high MAUVE scores (above 0.85). The PE methods achieve better downstream performance than DP-FT.

ϵ	Method	FID	MAUVE	Accuracy
$\epsilon = 4.0$	DP-FT	31.410	0.862	55.032
	Aug-PE	38.388	0.854	58.542
	SelfEvo-PE	36.017	0.868	58.398

Table 4: Performance of DP-FT, Aug-PE, and SelfEvo-PE on the Congressional Speech dataset when $\epsilon = 4.0$. At this loose privacy level, DP-FT has more advantage in terms of FID and MAUVE, but SelfEvo-PE offers better downstream utility for classification.

C.6.2 ICLR 2026 REVIEWS

ϵ	Method	FID	MAUVE
$\epsilon = 0.2$	DP-FT	124.192	0.027
	Aug-PE	68.398	0.174
	SelfEvo-PE	62.946	0.128
$\epsilon = 0.5$	DP-FT	47.936	0.242
	Aug-PE	58.661	0.175
	SelfEvo-PE	49.150	0.242
$\epsilon = 1.0$	DP-FT	38.997	0.470
	Aug-PE	51.535	0.193
	SelfEvo-PE	43.112	0.317

Table 5: Performance on 2K ICLR 2026 reviews. However, we observe that training a classifier even on the private data does not yield a clear performance gain over the majority-guess baseline. Thus, we do not report the classification accuracy in this experiment.

C.6.3 SELFEVO-PE WITH QWEN 2.5 7B

Tab. 6 provides the performance of SelfEvo-PE and Aug-PE with another LLM backbone. The performance is consistent with the main results using Llama 3.2 3B that SelfEvo-PE outperforms Aug-PE. This suggests that the performance gain of SelfEvo-PE is not specific to only one LLM backbone.

ϵ	Method	FID	MAUVE
$\epsilon = 0.2$	Aug-PE	67.989	0.372
	SelfEvo-PE	51.727	0.442
$\epsilon = 0.5$	Aug-PE	56.873	0.399
	SelfEvo-PE	46.838	0.547
$\epsilon = 1.0$	Aug-PE	53.077	0.538
	SelfEvo-PE	44.643	0.651

Table 6: Performance of Aug-PE and SelfEvo-PE on Qwen-2.5-7B on 2K samples of Congressional Speech. SelfEvo-PE outperforms Aug-PE under all settings of ϵ.

D LIMITATIONS

We acknowledge several limitations of our work that can be addressed in future research. First, our method relies on the assumption that voting results from previous iterations remain informative for future rounds, which may not always hold, especially when the distribution shift is large. A potential solution is to detect such shifts and adaptively decide whether to rely on the evaluator or perform a new private voting round. Second, our framework's performance gain over Aug-PE is mainly observed under limited-data and strict privacy budget settings. Third, our framework

introduces additional hyperparameters such as the number of self-evolving steps, evaluator training strategy, and selection schedule. While we have shown `SelfEvo-PE` is fairly robust with these hyperparameters, future work could explore more adaptive mechanisms to mitigate these.

LLM USAGE DISCLOSURE

We use LLMs to polish the writing, assist with coding, support discussion and brainstorming throughout the research process. We verify the correctness of all LLM-assisted code to ensure the implementation is accurate and consistent with the intended functionality.