Privacy-Aware Synthetic Data Generation

Manaal Mohammed
manaal@mit.edu
Massachusetts Institution of Technology
United States

Abstract

Generative models have been found to potentially memorize and return parts of their training data that may include sensitive information. For cases where models are trained on real data and designed to produce similar, but not identical synthetic data that both protects the privacy of those in the original dataset and still produces realistic results, this type of data exposure poses a higher risk. Synthetic data is often assumed to be safer and help protect the privacy of those in the original dataset, but if a model could potentially return memorized information with identifiable information, this intended goal might fail and violate the privacy of the original individuals without those using the synthetic data noticing. In this project, I propose a lightweight synthetic data generation filtering system that reduces risk of memorization and exposure of private identifying data in model outputs while trying to maintain output quality.

ACM Reference Format:
Manaal Mohammed. 2025. Privacy-Aware Synthetic Data Generation. In . ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

Sharing real-world textual data often poses serious privacy risks when the data contain personal or sensitive information. Synthetic data generation offers a solution by producing artificial text that preserves the statistical characteristics of real data without exposing actual personal details. Prior work has shown that naively training language models on sensitive data can lead to models regurgitating exact phrases from the training set. This unintended memorization can leak personally identifiable information (PII) from the original dataset into generated outputs. Because of this, there is a need for privacy frameworks to mitigate the risks of data leakage and safeguard the privacy of individuals who may have been in the original dataset.

2 Motivation

Formal privacy frameworks like differential privacy have been applied to text generation to mitigate such leakage. For example, fine-tuning a language model with differential privacy can yield useful synthetic text while provably limiting privacy loss[5]. However, DP methods often require careful calibration and complex implementation, and they may degrade text utility or fluency if applied too aggressively. There is a practical need for simpler approaches that balance privacy and utility without complex mathematical guarantees or significant compute. In this project, I design and investigate the utility of a lightweight privacy-aware synthetic text generation pipeline that can filter out PII while preserving utility of the data as best as possible. This system will aim to account for an attacker who has access to the generated synthetic data, and thus reducing the risk of PII leakage will be the main goal.

3 Related Work

Other works have explored the drawbacks of synthetic data and the potential risks that may arise when it comes to maintaining privacy and protection of those in the original dataset[3]. Zhao & Zhang evaluate the ability of synthetic data training to protect privacy based on coreset distillation, dataset, distillation, and other methods [6].

The lack of standardization when it comes to evaluating different synthetic data generation systems' ability to protect the privacy of those in the original dataset makes it difficult to determine which techniques might be optimal to fit societal needs, as Steier et al. discuss when briefly discussing potential techniques to enhance data privacy in synthetic data generation [3]. Differential privacy has been explored in different forms to mitigate the privacy risks that come with synthetic data generation [1, 5]. Combinatorial optimization has also been shown to be a successful method to generate synthetic data from publicly available census statistics [2]. Machine unlearning has also been investigated as a means of preventing memorization in large language models [4].

As this previous work has shown, a broad set of methods might be best to hit as many desired privacy metrics as possible in a synthetic data generation system. However, not many lightweight implementations specific to ensuring privacy in synthetic data generation have been developed; frameworks proposed typically assume the resources to implement complex, compute-heavy software.

4 Methods

A three-stage pipeline was implemented and then evaluated with a variety of metrics to determine its efficacy.

4.1 Pipeline

The pipeline consists of three points of filtering: pre-generation, in-generation, and post-generation. Three layers of filtering are used to ensure that the necessary data is removed; since each filter is lightweight, using three layers increases privacy at little cost to compute.

For the pre-generation filtering, known types of PII are passed in in the form of Regex expressions (e.g. emails, phone numbers,

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

SSNs). After this, named entity recognition (NER) is performed with spaCy's en_core_web_sm to efficiently detect desired types of information to remove. In a second pass, Regex-based detection is performed to catch additional types of information. The detected elements are then replaced with pre-determined generic placeholders (such as [FULL NAME]) depending on the type of the named entity; this allows the context to be preserved, which makes the redactions more coherent and usable.

For the in-generation filtering, the resulting input data is passed into the Gemma 2B model (chosen because it is open-source and manageable locally), and sampling parameters are chosen. The sampling parameters are temperature (to control randomness), top-k sampling (to keep only the top k tokens with highest probability), and top-p sampling (to only keep tokens with a cumulative probability greater than or equal to p). A repetition penalty to discourage tokens from repeating can also be applied. The aim of this is to guide the LLM generation process to reduce undesirable outputs without model retraining by subtly influencing token probabilities and sequence formation. Words can also be blocked, which explicitly prevents certain words or phrases from being generated using a logits processor that assigns low probabilities to blocked tokens. HuggingFace's Transformers library is used to optimize generation and to allow for various devices to be supported.

The post-generation filter is the final safety layer, analyzing the LLM output to catch any sensitive content that may have been generated despite the previous safeguards. The similarity detection compares generated text against known sensitive items by using semantic embeddings calculated through the all-MiniLM-L6-v2 sentence transformer and then computing cosine similarity. A rule-based classifier is then used to determine content type, checking for indicators of personal or technical information. For different types of content, adaptive similarity thresholds can be used; if technical content is detected, a higher threshold (0.95) is used, since technical text has more coincidental similarities. General content has a medium threshold of 0.90, and personal content has a lower threshold of 0.85 for stricter filtering on personal information. After this, PII detection from the pre-generation filter runs again to catch any new PII that may have been generated.

4.2 Evaluation

To evaluate the pipeline, I performed an ablation study by iterating through different filter configurations (no filters, pre-generation filter only, post-generation filter only, pre and post, all filters, all filters with stricter settings, all filters with adaptive thresholds) (see Table 1). First, I selected the Enron emails corpus to use (as it was publicly available and easily parsable, and could closely simulate a potential dataset that might have sensitive information inside of it). Then, I injected simulated senstive information into the samples, mixing fake structured PII (credit card numbers, addresses, etc.) with some canary phrases ("Emerald dragons soar through clouds of rainbow silk", "Whispers of crystal snowflakes tell secrets to winter shadows", etc.). I also saved a "clean" reference sample set.

After this, I defined prompts from the contaminated dataset (e.g. "summarize email X containing [injected PII]"). Then, with Gemma 2B, I generated 100 samples with each of the privacy-filter configurations and stored the outputs.

Evaluation was performed using privacy and utility metrics.

- Privacy metrics:
  - PII Leakage Rate: measured how often PII from input text appeared in the generated output (outputs with injected PII per total number of outputs)
  - Canary Leakage Rate: tested with "canary phrases" (unique identifiers) to detect memorization (outputs with injected canary phrase per total number of outputs)
  - Similarity Distribution: analyzed the distribution of similarities between inputs and outputs
- Utility Metrics:
  - Perplexity: measured the "naturalness" of generated text using GPT-2
  - Semantic Similarity: measured how well the generated text preserved the meaning of the original text (clean)
  - Distributional fidelity: calculated Jensen-Shannon Divergences on token n-gram distributions between generated sets and the original text (clean)
  - Output Statistics: tracked word counts, character counts, sentence counts, and placeholder rates

5 Results

I have included select figures of results found with these privacy and utility metrics for analysis.

5.1 Privacy Metrics

To start, Figure 1 displays the comparisons between the PII and canary leakage across different configurations; the Adaptive Thresholds configuration was seen to be the only one without any leakage, PII or canary. The Baseline configuration performed the worst when it came to PII leakage, but the Post-Generation filter performed the worst when it came to canary leakage rate. This is most likely because the post-generation filter doesn't account for the canary phrases, just the PII injections; however, it's hard to determine why it's much worse than the other configurations, although it could simply be due to randomness. All of the other configurations bar All Filters (Strict) and Adaptive Thresholds performed similarly when it came to canary leakage.

Figure 2 shows the approximate distribution of cosine similarity scores between the original dataset and the generated dataset across different configurations. The Adaptive Thresholds configurations seems to have the lowest similarity, and the Baseline config the highest; the others seems to be approximately similar.

5.2 Utility Metrics

In Figure 3, a comparison of the perplexity results (which measure how difficult it is to parse text) across different configurations is shown. The Adaptive Thresholds configuration is shown to perform the worst; the pre-generation filter only performs the best, with Pre-Post Filters performing similarly.

The Jensen-Shannon Divergence calculations for 1-gram, 2-gram, and 3-gram across different configurations are shown in Figure 4. Adaptive Thresholds appears once more to be the most divergent by far; the All Filters and All Filters (Strict) and the next most divergent. The least divergent is the Baseline configuration.

Table 1: Privacy–filter configurations used in the experiments

Config	Pre-filter	T	k	p	Penalty	# Blocked	Post-filter (key settings)
Baseline	None	0.8	50	0.95	1.0	0	None
Pre-only	token-level, place-holders	0.8	50	0.95	1.0	0	None
Post-only	None	0.8	50	0.95	1.0	0	threshold = 0.92; reapply-PII; token-redaction
Pre + Post	token-level, place-holders	0.8	50	0.95	1.0	0	threshold = 0.92; reapply-PII; token-redaction
All Filters	token-level, place-holders	0.8	50	0.95	1.2	6	threshold = 0.92; reapply-PII; token-redaction
All Filters (Strict)	token-level, place-holders	0.7	40	0.90	1.3	6	threshold = 0.85; reapply-PII; token-redaction
Adaptive	token-level, place-holders	0.8	50	0.95	1.2	6	base = 0.92; adaptive thresholds tech 0.95, gen 0.90, pers 0.85; reapply-PII; token-redaction

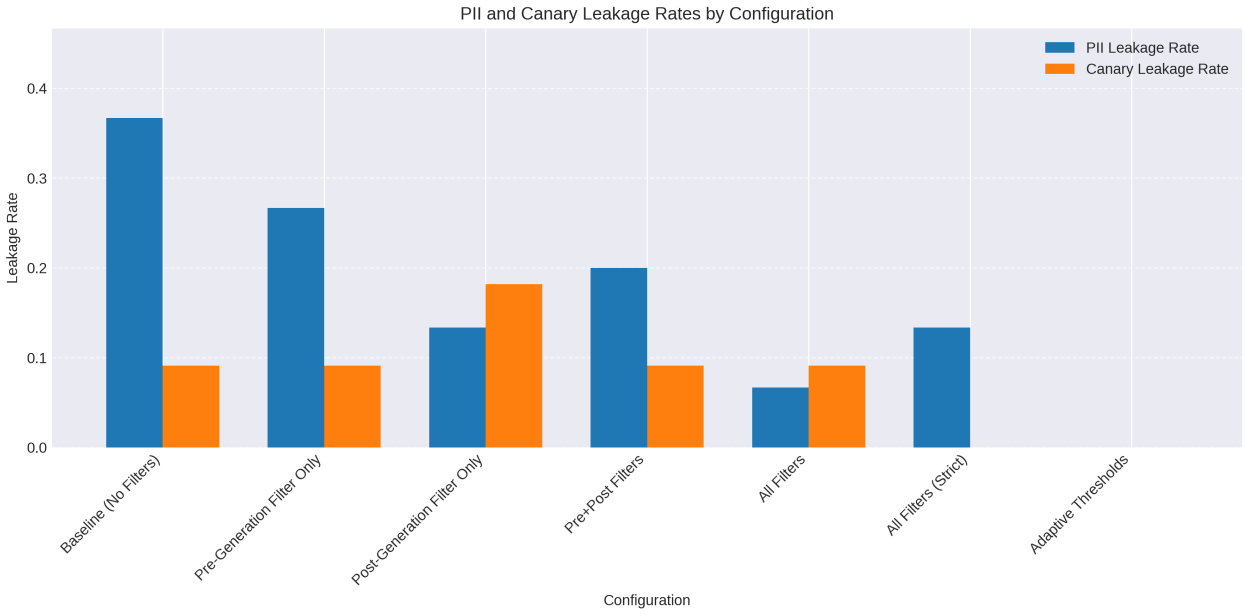

Figure 1: A comparison of the PII and canary leakage across different configurations

From Figure 5, which graphs configurations according to both their PII leakage rate and their semantic similarity as approximations for privacy and utility performance, it appears that the best configuration to optimize both the utility and privacy tradeoffs is the All Filters configuration. Adaptive Thresholds has the best privacy performance, but the worst utility by far; Baseline has the best utility, and worst privacy performance.

6 Discussion

My results show there are tradeoffs when it comes to privacy and utility; however, my All Filters configuration seemed to produce the best results in these respects. Parameter sensitivity seems to have also made a difference, and more experimenting with different parameters may have yielded improved results.

There were many limitations with my pipeline and experiments; the major limitation I had was on the size of experiments I could perform. I used only one example dataset, and only generated 100

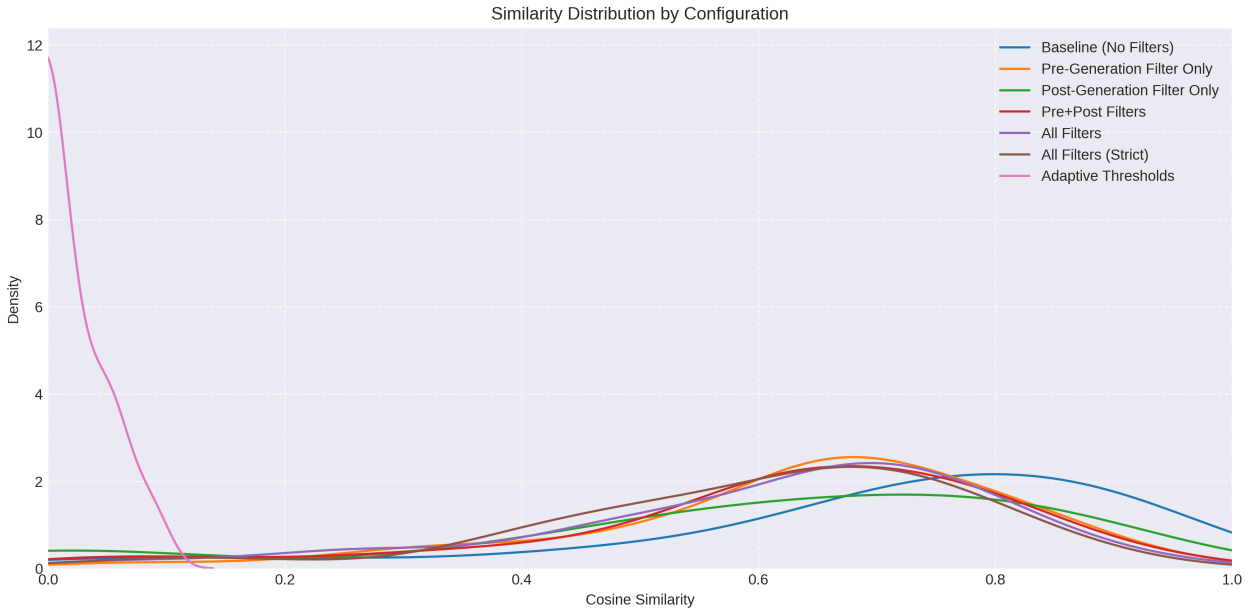

Figure 2: The approximate distribution of the cosine similarity scores across different configurations

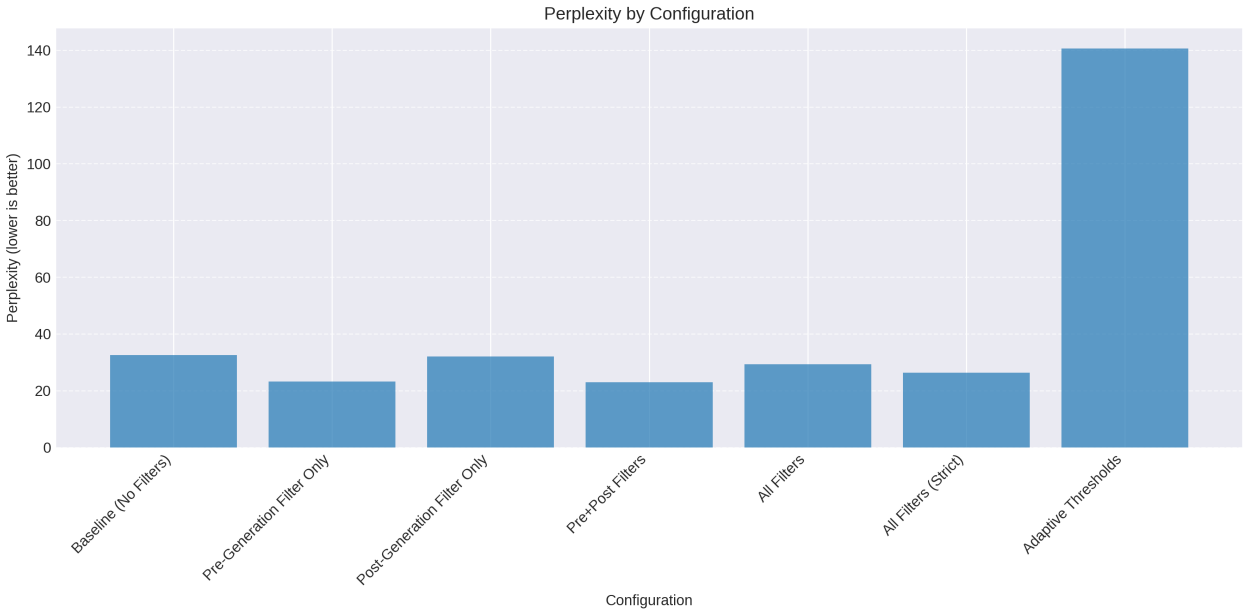

Figure 3: A comparison of the perplexity results across different configurations

samples for each trial. Restrictions on time, expenses, and computational resources meant that very large datasets or datasets with real sensitive information were not accessible for these experiments. However, considering these limitations, the ease of experiment execution with these constraints displays how lightweight the filtering system is.

Additionally, the framework's scope of protection is limited to known sensitive items; it cannot filter out sensitive information it isn't already searching for beforehand. While this can be useful when it comes to filtering certain datasets or as an additional security layer, it is not foolproof when it comes to securing datasets for synthetic data generation, as there are many different types of data that may prove sensitive. The framework is also limited to

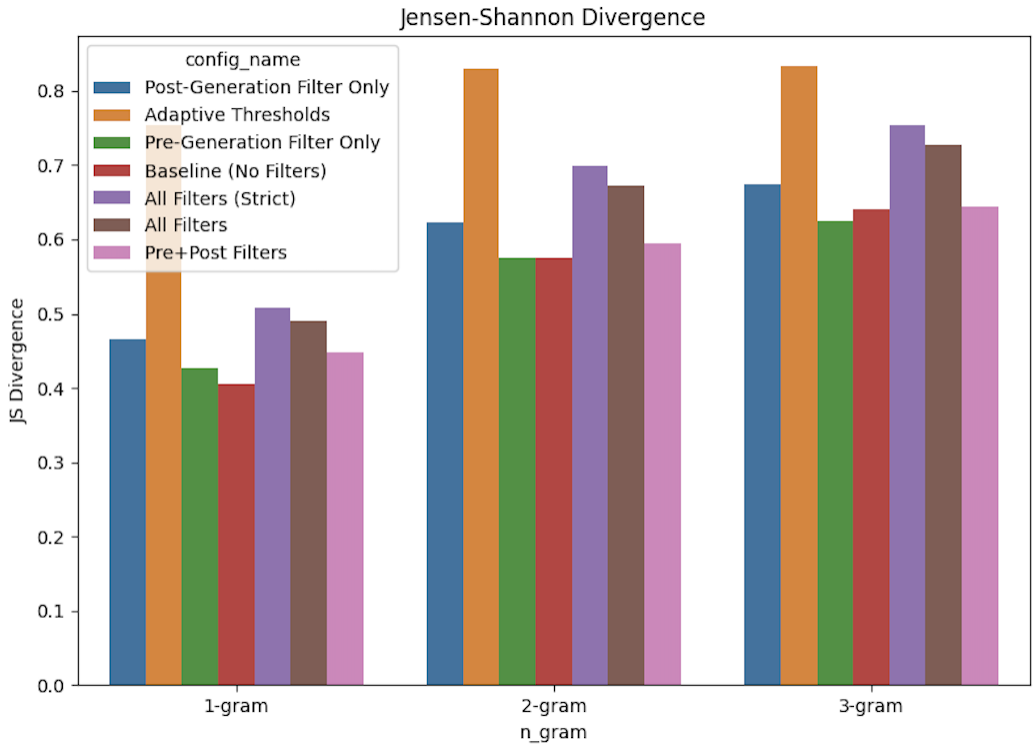

Figure 4: A comparison of the Jensen-Shannon Divergence calculations across different configurations

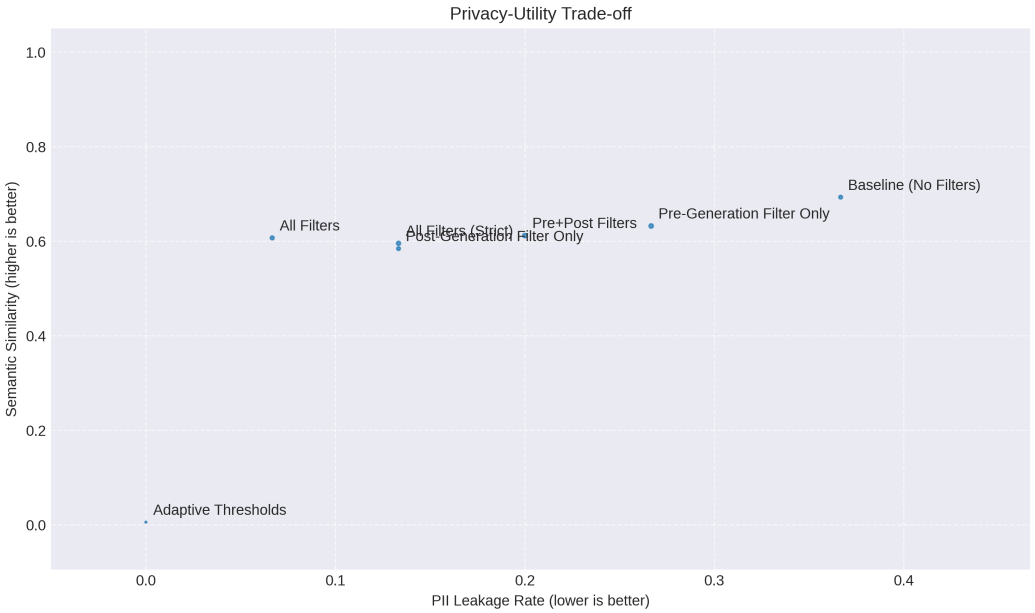

Figure 5: Comparisons of privacy and utility across different configurations

text datasets, and might not map well to datasets of other types or multimedia data.

The metrics I used were meant to provide a high-level analysis, and the reliance on cosine similarity may have missed out on nuances in paraphrasing or changes in sentence length. Human evaluation likely would also have been useful to better determine coherence and readability of produced data.

7 Conclusion

In this project, I designed and implemented a lightweight filtering system with open-source tooling (Gemma 2B, spaCy, and sentence transformers) that may help reduce the risk of exposing known, memorized sensitive items during synthetic data generation, either as a lightweight "final check" to accompany stronger data filtering, or as a standalone system for researchers hoping to quickly and cheaply remove sensitive items from a dataset. The findings through evaluation suggest that although this system has some utility tradeoffs and does not have the formal guarantees of differential privacy, this framework could offer a reduction in specific privacy risks.

7.1 Future Work

As noted previously, testing the system on larger generated datasets would provide clearer results on the efficacy of the filtering of PII. Broader testing with real-world data would also determine the utility of the system more fully, although it may present privacy risks and require much more ethical oversight. Additionally, using different placeholders with context-aware synthetically generated surrogates may improve realism and utility. Another way to extend this to determine real-world utility would be to devise a set of diverse downstream tasks for the framework to be used in.

Another potential extension could be adding a mechanism that can prematurely flag potential sensitive sequences in the dataset, which could both reduce the burden of manually updating patterns for each new dataset and also possibly catch sensitive data that wasn't initially known to be present in the dataset. Integration of this framework with a differential privacy framework or other complex, heavy framework to examine the utility of this system as an add-on could also extend this work further.

7.2 Ethical and Social Considerations

The ethical risks of the approach are the same as the harms that the approach would hope to mitigate; if one were to use a real dataset with human subjects for the training and implementation outline in my project, they would run the risk of exposing some of this private data when performing synthetic data generation. To mitigate this in my experiments, I used a synthetic dataset with fabricated hypothetical PII. The ethical and societal benefits of this approach, if successful, would be contributing towards more responsible use of AI, especially when it comes to protecting the privacy of those in training datasets; additionally, I hope to improve the accessibility of methods for preserving privacy when it comes to synthetic data generation.

7.3 Notes

The code for the pipeline implementation and evaluation used in this paper can be found here.

References

[1] Kareem Amin, Alex Bie, Weiwei Kong, Alexey Kurakin, Natalia Ponomareva, Umar Syed, Andreas Terzis, and Sergei Vassilvitskii. 2024. Private prediction for large-scale synthetic text generation. arXiv:2407.12108 [cs.LG] https://arxiv.org/abs/2407.12108

[2] Cynthia Dwork, Kristjan Greenewald, and Manish Raghavan. 2024. Synthetic Census Data Generation via Multidimensional Multiset Sum. arXiv:2404.10095 [cs.CY] https://arxiv.org/abs/2404.10095

[3] Amy Steier, Lipika Ramaswamy, Andre Manoel, and Alexa Haushalter. 2025. Synthetic Data Privacy Metrics. arXiv:2501.03941 [cs.LG] https://arxiv.org/abs/2501.03941

[4] Zhepeng Wang, Runxue Bao, Yawen Wu, Jackson Taylor, Cao Xiao, Feng Zheng, Weiwen Jiang, Shangqian Gao, and Yanfu Zhang. 2024. Unlocking Memorization in Large Language Models with Dynamic Soft Prompting. arXiv:2409.13853 [cs.CL] https://arxiv.org/abs/2409.13853

[5] Xiang Yue, Huseyin A. Inan, Xuechen Li, Girish Kumar, Julia McAnallen, Hoda Shajari, Huan Sun, David Levitan, and Robert Sim. 2023. Synthetic Text Generation with Differential Privacy: A Simple and Practical Recipe. arXiv:2210.14348 [cs.CL] https://arxiv.org/abs/2210.14348

[6] Yunpeng Zhao and Jie Zhang. 2025. Does Training with Synthetic Data Truly Protect Privacy? arXiv:2502.12976 [cs.CR] https://arxiv.org/abs/2502.12976