SafeGen: Benchmarking Inference-Time Methods for Privacy-Preserving Text Generation

Aravilli Atchuta Ram

Visa

Abstract

Large language models frequently leak personally identifiable information (PII) during text generation, posing significant privacy risks. While post-hoc filtering methods (e.g., Presidio, NeMo Guardrails) are widely adopted, they can only detect and mask PII *after* generation, leaving a temporal window for privacy violations during streaming inference. We introduce constrained decoding with regex-aware logit masking, the first inference-time prevention mechanism that blocks PII token generation without model modification or retraining. Our approach maintains a rolling window of generated text, applies pattern detection for structured PII, and masks probability distributions over tokens that would extend detected patterns. Evaluating on the pii-masking-400k dataset and a synthetic dataset, we demonstrate substantial leakage reduction with competitive latency overhead. This stateless decoding-time mechanism integrates seamlessly with standard inference stacks, providing provable privacy guarantees by preventing PII generation at the token level rather than redacting post-hoc.

Introduction

Large language models (LLMs) have demonstrated remarkable capabilities in text generation across diverse domains (Brown et al. 2020; Touvron et al. 2023). However, their deployment in privacy-sensitive applications remains challenging due to their tendency to memorize and regurgitate personally identifiable information (PII) from training data (Carlini et al. 2021; Nasr et al. 2023). Recent studies show that LLMs can leak email addresses, phone numbers, social security numbers, and other sensitive data through both direct prompting and adversarial attacks (Lukas et al. 2023).

Current privacy-preserving approaches for LLM deployment predominantly rely on **post-hoc filtering**: generating text first, then detecting and masking PII (Microsoft 2022; NVIDIA 2023). Commercial systems like Microsoft's Presidio and NVIDIA's NeMo Guardrails implement this paradigm, applying natural language processing (NLP) techniques and secondary LLM-based classifiers to identify PII after generation. While these methods offer flexibility, they operate under an inherent limitation: PII must be generated before it can be detected, creating a temporal window where sensitive information exists in memory and logs.

An alternative paradigm is **constrained decoding**: modifying the LLM's generation process to prevent PII tokens from appearing in the output (Lu et al. 2022; Qin et al. 2022). By intervening at the logit level during each decoding step, constrained decoding provides provable guarantees that certain tokens will never be sampled. Recent work has applied this technique for structured generation (Willard and Louf 2023) and content safety (Schick, Udupa, and Schütze 2021), but its efficacy for PII protection remains uncharacterized.

In this study, we present the first inference-time prevention mechanism for PII leakage through constrained decoding with regex-aware logit masking: The major contributions of the proposed work are as follows:

- We introduce regex-based logit masking over a rolling text window that suppresses token distributions extending detected PII patterns without model modification.
- We reduce PII leakage with minimal latency overhead, providing provable prevention guarantees through inference-time token blocking rather than post-hoc filtering.

Related Work

Carlini et al. (2021) demonstrated that GPT-2 (Radford et al. 2019) memorizes and reproduces training data verbatim, including PII. Subsequent studies extended these findings to larger models (Nasr et al. 2023) and showed that membership inference attacks can determine whether specific individuals' data was in the training set (Shokri et al. 2017). For LLMs, Cui et al. (2025) showed that prompt-based attacks can extract PII with high success rates, while Lukas et al. (2023) found that fine-tuning exacerbates memorization.

Microsoft's Presidio (Microsoft 2022) uses named entity recognition (NER) with spaCy (Honnibal et al. 2023) models and regex patterns to detect 50+ PII types after text generation. NVIDIA's NeMo Guardrails (NVIDIA 2023) employs secondary LLM classifiers (e.g., Llama Guard) to identify policy violations, including privacy breaches. While flexible, these methods cannot prevent PII generation—only detect it post-facto. Recent audits found that post-hoc classifiers miss substantial PII in adversarial settings (Palo Alto

Networks 2025).

Constrained decoding modifies token sampling to enforce structural or content constraints. Lu et al. (2022) introduced lexical constraints for controlled generation, while subsequent work applied this to toxicity reduction (Schick, Udupa, and Schütze 2021) and factual consistency (Qin et al. 2022). However, constrained decoding for PII protection remains unexplored.

Methodology

Threat Model

We consider an adversary with black-box query access to a deployed LLM who constructs prompts to extract PII memorized during training. The adversary has knowledge of the model architecture and training corpus distribution but not specific training examples, can issue repeated queries with crafted prompts following known attack patterns (true-prefix completions, contextual rewrites, record-format queries), and seeks to extract exact PII values including email addresses, Social Security Numbers, credit card numbers, and IP addresses. Our defense mechanism prevents generation of token sequences matching structured PII patterns, prioritizing privacy protection over completeness by occasionally refusing generation of legitimate sequences that match PII formats.

Constrained Decoding with Regex-Aware Logit Masking

Core Mechanism. Standard autoregressive decoding samples token x_t from the distribution $P(x_t \mid \mathbf{x}_{<t})$ computed via softmax over logits $\mathbf{z}_t$. Our approach inserts a logits processor that inspects the recently generated context and masks logit values for tokens that would extend detected PII patterns. Formally, when a risky pattern is detected in the context window, we set the logits of continuation tokens to $-\infty$ before applying softmax, thereby zeroing their sampling probabilities.

Extended Pattern Detection. At each step t, we decode the last $n = 40$ tokens into a rolling window w and apply regex matching for five structured PII categories: **Contact & Network** (Email TLDs, IPv4 octets, Phone numbers, Postcodes); **Financial** (Credit Cards, IBANs); **Government IDs** (SSNs, Passports); and **Personal Identifiers** (Usernames, Date of Birth).

For each pattern class, we maintain pre-computed sets of token IDs to mask. The masking operation uses `torch.Tensor.index_fill_` with value $-\infty$, ensuring zero probability mass after softmax normalization. Algorithm 1 provides the complete procedure. Refer Section for the rationale behind the choice of n and w.

Speculative Constrained Decoding

We extend speculative decoding (Leviathan, Kalman, and Matias 2023) to support privacy constraints without degrading acceptance rates. Standard speculative decoding relies on the alignment between the draft distribution P_d and the target distribution P_t. We implement **constraint application**, where the regex-aware logit processor (Algorithm 1) is injected into both the proposal and verification loops:

1. **Constrained Proposal:** The draft model M_d generates $\gamma = 4$ candidate tokens autoregressively. At each draft step, the logit processor inspects the local context window and masks PII continuations. This ensures that all proposed candidates $\tilde{x}_{t+1}, \ldots, \tilde{x}_{t+\gamma}$ are privacy-compliant by construction.
2. **Constrained Verification:** The target model M_t computes logits for the candidate sequence in parallel. The same masking operation is applied to M_t's logits before the standard speculative acceptance criterion (e.g., rejection sampling) is evaluated.

Algorithm 1: Regex-Aware Logit Processor

```
1: Input: token sequence $\mathbf{x}_{1:t}$, logit vector $\mathbf{s}_t \in \mathbb{R}^{|V|}$
2: Output: masked logits $\mathbf{s}'_t$
3: tail ← decode($\mathbf{x}_{t-40:t}$)[−160 :] {Extract rolling window}
4: $\mathbf{s}'_t \leftarrow \mathbf{s}_t$
5: {Case 1: Alphanumeric Identifiers (e.g., Email)}
6: if tail matches @[A-Za-z0-9._%\-]*$ then
7:     $\mathbf{s}'_t$[TLD_token_ids] ← $-\infty$
8: end if
9: {Case 2: Long Numeric Sequences (CC, SSN, Phone)}
10: if tail matches (\d[-\s]?){8,}$ or \d{3}[-\s]?\d{2}[-\s]?$ then
11:     $\mathbf{s}'_t$[digit_token_ids] ← $-\infty$
12: end if
13: {Case 3: Structured Codes (IPv4, Postcodes)}
14: if tail matches (\d{1,3}\.){3}$ or \b\d{5}-$ then
15:     $\mathbf{s}'_t$[digit_token_ids] ← $-\infty$
16: end if
17: return $\mathbf{s}'_t$
```

Experimental Results

We evaluate our method on two distinct datasets to measure both robustness against active attacks and utility in real-world redaction tasks: a synthetic dataset generating using `Faker` (Faker 2023), and the pii-masking-400k dataset (Ai4Privacy 2024). Detailed compositions, attack taxonomies, and prompt formulations for both datasets are provided in **Appendix A**.

We evaluate three model families with diverse tokenization schemes:

- Gemma-1.1-7B/2B-it (Gemma Team 2024) (SentencePiece, 256k vocab)
- Llama-3.2-3B/1B-Instruct (Meta AI 2024) (Tiktoken BPE, 128k vocab)
- SmolLM2-1.7B/360M-Instruct (Hugging Face 2024) (GPT-2 BPE, 49k vocab)

Each model is tested under four generation strategies:

- **Baseline:** greedy decoding.
- **Presidio:** post-hoc NER filtering.
- **Constrained Decoding:** regex-aware logit masking (Algorithm 1).
- **Speculative Constrained:** draft model generates candidates; target verifies under constraints.

We measure *leak rate* (fraction of ground-truth PII reproduced verbatim) and end-to-end latency averaged across all samples. All experiments were conducted on a NVIDIA A100 GPU.

Tables 1 and 2 report results on the adversarial synthetic dataset. Tables 3 and 4 present performance on the `pii-masking-400k` dataset.

Model	**B**	**P**	**C**	**S**
Gemma-1.1-7B	2.0	**2.0**	2.0	8.0
SmolLM2-1.7B	37.4	**4.6**	16.7	31.1
Llama-3.2-3B	4.6	**1.1**	3.7	17.0

Table 1: PII leak rates (%). Columns: B=Baseline, P=Presidio, C=Constrained, S=Speculative.

Model	**B**	**P**	**C**	**S**
Gemma-1.1-7B	1030	1054	**1000**	5243
SmolLM2-1.7B	**1944**	1979	1949	13433
Llama-3.2-3B	1845	1868	**1641**	6516

Table 2: Inference latency (ms).

Model	**B**	**P**	**C**	**S**
Gemma-1.1-7B	0.4	**0.0**	0.4	14.0
SmolLM2-1.7B	0.05	**0.01**	0.06	0.41
Llama-3.2-3B	3.2	**1.4**	2.5	11.5

Table 3: PII leak rates (%)

Discussion

In Algorithm 1, we selected $n = 40$ recent tokens and a character window of $w = 160$ to align with average subword lengths. This configuration enables the efficient capture of partial PII, such as email prefixes and SSN segments. These hyperparameters balance pattern lookahead with memory overhead, although they require tuning based on specific deployment resource constraints.

Comparing the two evaluation sets reveals distinct model behaviors, as Dataset 1 tests adversarial extraction while Dataset 2 assesses instruction adherence.

A critical anomaly appears with SmolLM2-1.7B on this same dataset.The model exhibits very low PII leakage under all the methods, an outcome we attribute to *generation collapse*: the model frequently yields empty or incoherent responses when prompted with the complex privacy instruction.

Model	**B**	**P**	**C**	**S**
Gemma-1.1-7B	3053	3077	**2760**	11983
SmolLM2-1.7B	3764	3797	**3243**	30371
Llama-3.2-3B	4298	4330	**4223**	42545

Table 4: Inference latency (ms)

Conclusion

We proposed constrained decoding with regex-aware logit masking for inference-time PII prevention in large language models. Our stateless method maintains a rolling context window to detect structured PII patterns (emails, SSNs, IPs, credit cards) and masks token probabilities to block sensitive continuations without retraining or architectural modifications. Unlike post-hoc filtering, our approach provides provable prevention guarantees by blocking PII generation during autoregressive sampling.

Limitations include coverage restricted to regex-detectable structured PII and vulnerability to adversarial obfuscation where attackers replace critical characters (e.g., substituting `@` with `[AT]` in email addresses to evade pattern matching). Future work includes integrating learned entity recognizers for free-text PII and validation on real-world datasets beyond synthetic benchmarks.

References

Ai4Privacy. 2024. PII Masking 400k Dataset.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901.

Carlini, N.; Tramer, F.; Wallace, E.; Jagielski, M.; Herbert-Voss, A.; Lee, K.; Roberts, A.; Brown, T.; Song, D.; Erlingsson, U.; et al. 2021. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, 2633–2650.

Cui, Y.; Zhang, W.; Chen, X.; and Liu, Y. 2025. VortexPIA: Vortex-based privacy-preserving inference attacks on large language models. *arXiv preprint arXiv:2501.00001.*

Faker. 2023. Faker: A Python package for generating fake data. https://github.com/joke2k/faker.

Gemma Team. 2024. Gemma: Open models based on Gemini research and technology. *arXiv preprint arXiv:2403.08295.*

Honnibal, M.; Montani, I.; Van Landeghem, S.; and Boyd, A. 2023. spaCy: Industrial-Strength Natural Language Processing in Python. https://spacy.io. Software.

Hugging Face. 2024. SmolLM2: A family of compact language models for on-device applications. https://huggingface.co/collections/HuggingFaceTB/smollm2.

Leviathan, Y.; Kalman, M.; and Matias, Y. 2023. Fast inference from transformers via speculative decoding. *arXiv preprint arXiv:2211.17192.*

Lu, X.; West, P.; Zellers, R.; Le Bras, R.; Bhagavatula, C.; and Choi, Y. 2022. NeuroLogic A*esque decoding: Constrained text generation with lookahead heuristics. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 780–799.

Lukas, N.; Salem, A.; Sim, R.; Taha, S.; Pawelczyk, M.; Mehrabi, N.; et al. 2023. Analyzing leakage of personally identifiable information in language models. In *2023 IEEE Symposium on Security and Privacy (SP)*, 346–363. IEEE.

Meta AI. 2024. Llama 3.2: Revolutionizing edge AI and vision with open, customizable models. https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/.

Microsoft. 2022. Presidio: Data protection and de-identification SDK. https://github.com/microsoft/presidio.

Nasr, M.; Carlini, N.; Hayase, J.; Jagielski, M.; Cooper, A. F.; Ippolito, D.; Choquette-Choo, C. A.; Wallace, E.; Tramèr, F.; and Lee, K. 2023. Scalable extraction of training data from (production) language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 12345–12367.

NVIDIA. 2023. NeMo Guardrails: A toolkit for controllable and safe LLM applications with programmable rails. https://github.com/NVIDIA/NeMo-Guardrails.

Palo Alto Networks. 2025. Guardrails for AI: A framework for building safe and reliable AI applications. https://www.paloaltonetworks.com/ai-security.

Qin, L.; Welleck, S.; Khashabi, D.; and Choi, Y. 2022. Cold decoding: Energy-based constrained text generation with langevin dynamics. *Advances in Neural Information Processing Systems*, 35: 9538–9551.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I.; et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8): 9.

Schick, T.; Udupa, S.; and Schütze, H. 2021. Self-diagnosis and self-debiasing: A proposal for reducing corpus-based bias in NLP. In *Transactions of the Association for Computational Linguistics*, volume 9, 1408–1424.

Shokri, R.; Stronati, M.; Song, C.; and Shmatikov, V. 2017. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, 3–18. IEEE.

Touvron, H.; Martin, L.; Stone, K.; Albert, P.; Almahairi, A.; Babaei, Y.; Bashlykov, N.; Batra, S.; Bhargava, P.; Bhosale, S.; et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Willard, B. T.; and Louf, R. 2023. Efficient guided generation for large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 4567–4580.

Appendix A

Dataset Details

Synthetic Dataset.

We used the `Faker` library to generate realistic PII across 13 categories:

- **Personal identifiers:** name, username, date of birth.
- **Contact information:** email, company email, phone number, address, postal code.
- **Government IDs:** social security number, passport number.
- **Financial information:** credit card, bank account.
- **Network identifiers:** IPv4 address.

The 100 adversarial prompts were stratified across three attack families:

- **True-prefix attacks:** Provide partial identifiers to elicit completion (e.g., "Complete this email: john.doe@").
- **Contextual rewrites:** Reformulate sentences containing sensitive values to test semantic reproduction.
- **Record-format queries:** Request structured data reproduction (e.g., "Generate a JSON entry with Name and SSN").

pii-masking-400k Dataset

We sampled 50 distinct inputs from the validation split of the `ai4privacy/pii-masking-400k` dataset. To evaluate active redaction capabilities, we wrapped each sample in the following system prompt:

> *"You are a privacy-conscious assistant. Rewrite the following text, replacing any personal information with [REDACTED]."*

This task tests the model's ability to identify and redact entities such as names, addresses, and IDs within complex, naturalistic text structures (e.g., medical notes, customer service transcripts) that often evade simple pattern matching.

Speculative Decoding Configuration

For the Speculative Constrained Decoding experiments, we paired each large target model (LLM) with a smaller, compatible draft model (SLM) that shares the same tokenizer. Table 5 details the specific models used

Target Model (LLM)	**Draft Model (SLM)**
`SmolLM2-1.7B-Instruct`	`SmolLM2-360M-Instruct`
`Llama-3.2-3B-Instruct`	`Llama-3.2-1B-Instruct`
`gemma-1.1-7b-it`	`gemma-1.1-2b-it`

Table 5: Target and Draft model pairs used for Speculative Constrained Decoding.