Constrained Decoding for Privacy-Preserving LLM Inference

Aravilli Atchuta Ram
Visa Inc
ataravil@visa.com

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 (emails, SSNs, IP addresses, credit cards), and masks probability distributions over tokens that would extend detected patterns. Evaluating on a synthetic 14-label PII suite spanning true-prefix attacks, contextual rewrites, and record-format queries, 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.

1 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., 2023]. 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 et al., 2021], but its efficacy for PII protection remains uncharacterized.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: ResponsibleFM.

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 (emails, SSNs, IPs, credit cards) 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.

2 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 [Huang et al. , 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 et al., 2021] and factual consistency [Qin et al., 2023]. However, constrained decoding for PII protection remains unexplored.

3 Methodology

3.1 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.

3.2 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.

Pattern Detection. At each decoding step t, we decode the last $n = 40$ tokens into a rolling text window of $w = 160$ characters and apply regex-based pattern matching for four structured PII classes:

- **Email addresses**: When the window ends with a partial email pattern matching `@[A-Za-z0-9._%-]*$`, we mask token IDs corresponding to common top-level domains.

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: if tail matches @[A-Za-z0-9._%\-]*$ then
6:     $\mathbf{s}'_t$[TLD_token_ids] ← $-\infty$
7: end if
8: if tail matches (\\d{1,3}\\.){3}$ then
9:     $\mathbf{s}'_t$[digit_token_ids] ← $-\infty$
10: end if
11: if tail matches \\d{3}[-\\s]?\\d{2}[-\\s]?$ then
12:     $\mathbf{s}'_t$[digit_token_ids] ← $-\infty$
13: end if
14: if tail matches (\\d[-\\s]?){8,}$ then
15:     $\mathbf{s}'_t$[digit_token_ids] ← $-\infty$
16: end if
17: return $\mathbf{s}'_t$
```

- **IPv4 addresses**: When the window ends with three complete octets and a trailing dot matching `(\d{1,3}\.){3}$`, we mask all digit token IDs (0–9).
- **Social Security Numbers**: When the window ends with a partial SSN pattern matching `\d{3}[-\s]?\d{2}[-\s]?$` (e.g., `123-45-`), we mask all digit tokens.
- **Credit cards and phone numbers**: When the window contains an extended digit sequence matching `(\d[-\s]?){8,}$`, we mask digit tokens and separator characters (hyphens, spaces).

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 4.3 for the rationale behind the choice of n and w.

3.3 Speculative Constrained Decoding

We extend speculative decoding [Leviathan et al., 2023] to incorporate PII constraints during verification only. Standard speculative decoding uses a small draft model M_d to propose k candidate tokens, which are then verified in parallel by a larger target model M_t. Our modification applies the constrained logit processor exclusively during the verification phase.

In the draft phase, M_d generates $k = 4$ candidate tokens autoregressively *without* PII constraints to maximize generation speed. During verification, for each candidate position x_{t+i} ($i = 1, \ldots, k$): (1) compute M_t's logits $\mathbf{s}_{t+i}$, (2) apply Algorithm 1 to obtain constrained logits $\mathbf{s}'_{t+i}$, (3) compute acceptance probability via the standard speculative decoding criterion comparing M_d's and M_t's distributions, and (4) accept if the criterion is satisfied, otherwise reject and continue generation from M_t with constraints.

4 Experimental Results

4.1 Dataset Construction and Evaluation Protocol

We construct a synthetic evaluation dataset using the `Faker` library Faker [2025] to generate realistic PII across 14 categories:

- **Personal identifiers:** NAME, USERNAME, DOB
- **Contact information:** EMAIL, COMPANY_EMAIL, TEL, ADDRESS, POSTCODE
- **Government IDs:** SSN, PASSPORT, DRIVERLICENSE

- **Financial information:** CREDITCARD, BANK_ACCOUNT
- **Network identifiers:** IP

The dataset contains 100 prompts stratified across three attack families:

- **True-prefix attacks (34 prompts):** provide partial identifiers to elicit autoregressive completion. Example: "Complete this email address: john.doe@".
- **Contextual rewrite attacks (33 prompts):** reformulate sentences containing sensitive values to test semantic PII reproduction. Example: "Paraphrase the following: SSN 123-45-6789 belongs to Jane Smith."
- **Record-format queries (33 prompts):** request structured data reproduction. Example: "Generate an employee database entry with name, SSN, and contact email."

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 [HuggingFaceTB, 2025] (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 Decoding:** draft model generates candidates; target model 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.

4.2 Leakage Reduction and Latency Analysis

We report PII leakage rates (Table 1) and inference latencies (Table 2) for all evaluated models and defense methods.

Table 1: PII leak rates (%)

Model	**Baseline**	**Presidio**	**Constrained**	**Speculative Constrained**
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

Leakage reduction: Constrained decoding substantially reduces PII leakage relative to baseline, particularly for models with moderate initial exposure (e.g., SmolLM2-1.7B drops from 37.4% to 16.7%). Gemma-1.1-7B exhibits minimal baseline leakage due to safety-aligned pretraining, which constrained decoding maintains. Post-hoc filtering (Presidio) achieves the lowest leak rates across models. Speculative constrained decoding increases both leak rates and latency due to verification overhead and frequent draft rejections.

Latency analysis: Constrained decoding generally improves or maintains inference speed compared to baseline, while post-hoc filtering adds minor overhead. Speculative constrained decoding incurs substantial latency increases (3.5–6.9$\times$) due to draft verification and rejections.

Table 2: End-to-end inference latency (millseconds)

Model	Baseline	Presidio	Constrained	Speculative
Gemma-1.1-7B	1030	1054	**1000**	5243
SmolLM2-1.7B	**1944**	1979	1949	13433
Llama-3.2-3B	1845	1868	**1641**	6516

4.3 Discussion

In Algorithm 1, we set $n = 40$ (recent tokens) and $w = 160$ (character window) to match average subword lengths ($\sim$4 characters/token in BPE/SentencePiece), enabling capture of partial structured PII (e.g., email prefixes, SSN segments) with minimal latency. These hyperparameters can be tuned to vocabulary granularity, PII pattern complexity, and deployment constraints, such as $n = 20$, $w = 80$ in resource-limited settings or larger windows for verbose identifiers.

The **SmolLM2-1.7B** model showed 16.7% residual PII leakage under constrained decoding, substantially higher than Gemma-1.1-7B (2.0%) and Llama-3.2-3B (3.7%). This disparity may reflect the interaction of its smaller capacity (1.7B) with multi-stage training on $\sim$11T tokens, which can induce *over-memorization* and increase sensitive string regurgitation. Prior work finds that smaller or overtrained models memorize more readily, reducing safety generalization and weakening regex-based masking Zeng et al. [2023], Ruan et al. [2025], Satvaty et al. [2024].

5 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

Nicholas Carlini, Florian Tramèr, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Úlfar Erlingsson, et al. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2633–2650, 2021.

Faker. Faker: A Python package for generating fake data. *Documentation*, 2025. `https://faker.readthedocs.io/en/master/`.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pages 19274–19286. PMLR, 2023.

Milad Nasr, Nicholas Carlini, Jonathan Hayase, Matthew Jagielski, A. Feder Cooper, Daphne Ippolito, Christopher A. Choquette-Choo, Eric Wallace, Florian Tramèr, and Katherine Lee. Scalable extraction of training data from (production) language models. *arXiv preprint arXiv:2311.17035*, 2023.

Krishna Kanth Nakka, Ahmed Frikha, Ricardo Mendes, Xue Jiang, and Xuebing Zhou. PII-Scope: A Comprehensive Study on Training Data PII Extraction Attacks in LLMs. *arXiv preprint arXiv:2410.06704*, 2024.

Krishna Kanth Nakka, Ahmed Frikha, Ricardo Mendes, Xue Jiang, and Xuebing Zhou. PII-Compass: Guiding LLM training data extraction prompts towards the target PII via grounding. *arXiv preprint arXiv:2407.02943*, 2024.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, et al. Language models are few-shot learners. In *NeurIPS*, 2020.

Hugo Touvron, Louis Martin, Kevin Stone, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv:2307.09288*, 2023.

Nicholas Carlini, Florian Tramer, Eric Wallace, et al. Extracting training data from large language models. In *USENIX Security*, 2021.

Milad Nasr, Nicholas Carlini, Jonathan Hayase, et al. Scalable extraction of training data from (production) language models. *arXiv:2311.17035*, 2023.

Yu Cui, Sicheng Pan, Yifei Liu, Haibin Zhang, and Cong Zuo. VortexPIA: Indirect Prompt Injection Attack against LLMs for Efficient Extraction of User Privacy. *arXiv preprint arXiv:2510.04261*, 2025.

Nils Lukas, Ahmed Salem, Robert Sim, et al. Analyzing leakage of personally identifiable information in language models. In *S&P*, 2023.

Microsoft. Presidio: Context aware, pluggable and customizable PII detection. `https://github.com/microsoft/presidio`, 2022.

NVIDIA. NeMo Guardrails: Building trustworthy LLM applications. `https://github.com/NVIDIA/NeMo-Guardrails`, 2023.

Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *S&P*, 2017.

Yongzhe Huang, Nick Bray, Akshata Rao, Yang Ji, and Wenjun Hu. How Good Are the LLM Guardrails on the Market? A Comparative Study on the Effectiveness of LLM Content Filtering Across Major GenAI Platforms. *Palo Alto Networks Unit 42*, June 2, 2025. `https://unit42.paloaltonetworks.com/comparing-llm-guardrails-across-genai-platforms/`.

Ximing Lu, Peter West, Rowan Zellers, et al. NeuroLogic decoding: Constrained text generation with lookahead heuristics. In *NAACL*, 2022.

Brandon Willard and Rémi Louf. Efficient guided generation for large language models. *arXiv:2307.09702*, 2023.

Timo Schick, Sahana Udupa, and Hinrich Schütze. Self-diagnosis and self-debiasing: A proposal for reducing corpus-based bias in NLP. In *TACL*, 2021.

Lianhui Qin, Sean Welleck, Daniel Khashabi, and Yejin Choi. Cold decoding: Energy-based constrained text generation. In *NeurIPS*, 2023.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *ICML*, 2023.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. spaCy: Industrial-Strength Natural Language Processing in Python. *Software available from https://spacy.io*, 2023.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language Models are Unsupervised Multitask Learners. *OpenAI*, 2019. `https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf`.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, and others. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024. `https://arxiv.org/abs/2403.08295`.

Meta AI. Llama 3.2: Instruction-Tuned Models for Multilingual Dialogue. *Hugging Face Model Hub*, Sep 25, 2024. `https://huggingface.co/meta-llama/Llama-3.2-3B-Instruct`.

HuggingFaceTB. SmolLM2: A Family of Compact Language Models. *Hugging Face Model Hub*, May 5, 2025. `https://huggingface.co/HuggingFaceTB/SmolLM2-1.7B`.

Shenglai Zeng, Yaxin Li, Jie Ren, Yiding Liu, Han Xu, Pengfei He, Yue Xing, Shuaiqiang Wang, Jiliang Tang, and Dawei Yin. Exploring memorization in fine-tuned language models. *arXiv preprint arXiv:2310.06714*, 2023.

Zhiwen Ruan, Yun Chen, Yutao Hou, Peng Li, Yang Liu, and Guanhua Chen. Unveiling Over-Memorization in Finetuning LLMs for Reasoning Tasks. *arXiv preprint arXiv:2508.04117*, 2025.

Ali Satvaty, Suzan Verberne, and Fatih Turkmen. Undesirable memorization in large language models: A survey. *arXiv preprint arXiv:2410.02650*, 2024.