PII-RADAR: RAPID DETECTION AND REPLACEMENT FOR PII SCRUBBING

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models have been shown to leak their training data, which becomes a privacy risk when this data contains personal information. In practice, vulnerable data is commonly removed using scrubbers, i.e., tools that detect personally identifiable information (PII) in a document and remove it before training. However, scrubbers are imperfect and miss some PII in the training data. In this work, we first show that, while scrubbing leaves less information exposed, the PII it misses becomes significantly more vulnerable. The leakage we observe increases with the detection rate, so scrubbers with stronger detectors increase the risk for missed PII. We identify PII removal as the primary cause of this leakage, because it leaves only a small amount of real PII in the dataset, which the model then memorizes. While synthetic rewriting of the detected PII should, in theory, resolve this issue, current rewriting tools generate highly repetitive or malformed text, increasing the risk similarly to removal. To address this issue, we propose `PII-RADAR`, a novel scrubber that replaces each detected PII with a realistic one of the same type, drawn from a pool of validated synthetic identities and formatted to match the original document. Contrary to other rewriting scrubbers, `PII-RADAR`'s replacements are diverse, well-formed, and non-repetitive, lowering the privacy risk of any PII it misses **up to 3**$\times$ compared to baselines. Moreover, `PII-RADAR` outperforms all evaluated scrubbers on five of seven benchmarks, establishing a new state of the art in PII detection, while being up to 10× faster than other scrubbers that replace rather than redact.

1 INTRODUCTION

Large language models (LLMs) rely on training runs with trillions of tokens to achieve state-of-the-art performance (OpenAI, 2026a; Anthropic, 2026a; Grattafiori et al., 2024). Commercial model providers (OpenAI, 2026a; Anthropic, 2026a; Google DeepMind, 2026) source part of this data from the web but also train on conversations users have with the models (OpenAI, 2026), which contain medical records, contracts, résumés, account numbers, and sensitive messages about individuals (Grosse & Ebert, 2026). In all cases, the data often contains Personally Identifiable Information (PII), *i.e.,* data that identifies individuals (Huang et al., 2022). Since models can leak training data (Carlini et al., 2021; Nasr et al., 2025), this poses severe privacy risks.

To mitigate these risks, providers can intervene on the model's outputs, its weights, or the training data itself. Output filters (Rebedea et al., 2023; Inan et al., 2023; Wróbel et al., 2026) and refusal training (Bai et al., 2022; Yuan et al., 2025) act at the first two levels. However, carefully crafted prompts or jailbreaks can bypass both defenses (Shen et al., 2024; Andriushchenko & Flammarion, 2025), while even benign fine-tuning can undo the refusal training (Qi et al., 2024). While formal frameworks like differential privacy (Dwork, 2006) bound the leakage of individual training samples, they raise training cost and harm utility (Sinha et al., 2025; Anil et al., 2022).

Practical real-world deployments rely on training data filtering, where documents are scanned for PII and either fully removed (Grattafiori et al., 2024), or their PII are erased from the documents (Soldaini et al., 2024; Li et al., 2023) using tools called *scrubbers*. PII scrubbers (Leap Beyond Analytics, 2023; Microsoft & Data Privacy Stack, 2026; The Internet Integrity Initiative, 2024; de Bourcy et al., 2026; Privasis, 2026; Eternis, 2025) rely on two operations. The first is *detection* of PII within a document, followed by *replacement*, where most scrubbers *mask* the detected PII with type-specific

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.

tags such as [REDACTED_EMAIL] and a few systems *rewrite* the PII with a synthetic value of the same type, synthesized automatically (OpenPipe, 2025) or generated by the model itself (Privasis, 2026; Eternis, 2025). Scrubbers are evaluated and compared by their detection *recall*, *i.e.,* how many PII they find, and their detection *precision*, *i.e.,* how much they alter non-PII text.

We first demonstrate that the detection-based metrics yield an incomplete picture because they do not take into account the downstream privacy risks of missed PII. These missed PII turn out to become highly vulnerable. To show that, we check their leakage from an LLM trained on scrubbed data, against leakage from a control LLM, trained without scrubbing. We find a significant increase in leakage, with some PII types becoming up to **14× more extractable**. Moreover, we observe the same effect regardless of which scrubber is used. Even when we *mask PII at random*, matching the tested scrubbers' detection rates, we see a similar increase in the leakage of the PII left as in the case of PII masked by the scrubbers themselves.

We further observe that replacing PII with type-dependent, repetitive masks, such as [REDACTED_EMAIL], causes the missed PII to become *outliers* in the remaining data. In general, prior work showed that outlier data points are more likely to be memorized during training (Feldman, 2020; Carlini et al., 2021). A natural remedy is to replace PII with realistic values like synthetic PII to address this issue. However, existing rewriting scrubbers only slightly lower the leakage compared to the masking alternatives, but do not mitigate it. As a reason, we identify that these scrubbers insert replacement PII which falls short in quality: they are repetitive, placeholder-like, or incorrect for the expected PII type. For example, Privasis-Cleaner 4B (Privasis, 2026) writes REDACTED for 29% of phone numbers, Anonymizer-4B (Eternis, 2025) edits 26.3% of detected credit card numbers into XXXX XXXX XXXX XXXX, and OpenPipe (OpenPipe, 2025) uses "example" domain for over 90% of emails and *never* returns a correct credit card number. We show that even a method based on simple regular expressions can separate real PII from their synthetic replacements at up to 0.94 AUC.

To close this gap, we propose PII-RADAR, which achieves state-of-the-art PII detection with high-quality PII replacement and up to 10× faster execution than other scrubbers. PII-RADAR provides the highest PII detection rate through simple per-token binary classification (PII vs. non-PII), supported by training on diverse data. For PII replacement, rather than generating new PII from scratch, PII-RADAR substitutes detected PII with values drawn from a validated synthetic user *profiles*. A profile is a set of PII of various types, where all PII within profile are consistent, *e.g.,* e-mail addresses match names, and all PII are verified for correctness. To perform the PII replacement, we draw a profile and pass it to PII-RADAR as a prefix for each document, allowing PII-RADAR to select valid replacement PII that match the type and format of the detected PII.

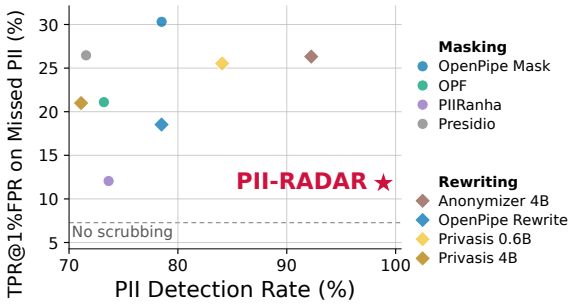


Figure 1: **Our PII-RADAR achieves highest detection rate and lowest leakage of missed PII.**

Our PII-RADAR produces realistic PII, **92.3**% of which is well-formed and none is a placeholder. PII-RADAR's PII are diverse, with the ten most frequent values accounting for only 0.07% of the replacements. In contrast, only 35.0% PII of Anonymizer 4B and at most 1.4% of Privasis Cleaner's is correctly formatted and not a placeholder, with the latter scrubber producing the same ten PII 69.5–74.1% of the time (Fig. 5). Replacement PII from PII-RADAR are following closely the PII before the scrubbing, with unsupervised classification recovers over 8× fewer values than the closest competitor and 59–103× fewer than the rest. With these high quality replacements, the privacy leakage of PII missed by PII-RADAR is **significantly reduced** compared to other methods. Specifically, compared to the non-scrubbed LLMs, the missed PII leak only 1.5× more, increasing from 7.3% TPR@FPR=1% to 11.6%, while the best-performing baseline (Anonymizer 4B) increases leakage up to 27.1% TPR@FPR=1%, by a factor of 4.4, see Fig. 1. Since PII-RADAR needs to detect PII well in oder to replace it, we additionally evaluate PII-RADAR on seven public PII detection benchmarks. On five of them PII-RADAR achieves **the highest detection rate**, with a 3–10× faster runtime than other replacement-based scrubbers. Our anonymous code is available at https://anonymous.4open.science/r/radar-4ABF.

In summary, we make the following contributions:

- We show that scrubbing concentrates privacy risks on the PII it overlooks and deleting PII uniformly at random at a scrubber's miss rates reproduces the same leakage trend.
- We demonstrate that the current method for the PII replacement with synthetic values are still inadequate and exhibit high privacy leakage.
- We propose `PII-RADAR` that rewrites each detected PII in place, drawing all replacements in a document from a consistent synthetic profile and matching the format of the replaced text.
- We show that `PII-RADAR` substantially reduces leakage, while achieving the highest recall on five of seven benchmarks and running 3–10$\times$ faster than other replacement-based scrubbers.

2 Background and Related Work

Large Language Models (LLMs) power current chat-bots (OpenAI, 2026b; Anthropic, 2026a) with which hundreds of millions of users interact every day (Chatterji et al., 2025). Oftentimes, the users paste information into chats that contain vulnerable data like PII (Mireshghallah et al., 2024), unaware that their chats may be later used to train new LLMs (OpenAI, 2026; Zhang et al., 2024). Since LLMs have been shown to memorize and leak their training data (Carlini et al., 2021; Nasr et al., 2025), this poses significant privacy risks.

Privacy auditing tools for LLMs include *Membership Inference Attacks* (MIA) (Shokri et al., 2017) and *data extraction attacks* (Carlini et al., 2021; Nasr et al., 2025). The goal of MIAs is to determine if a given sample was part of a model's training data. MIAs use the LLM outputs to assess membership (Shi et al., 2024; Zhang et al., 2025), and further strengthen the confidence by training reference (also referred to as shadow) models (Carlini et al., 2022a; Zarifzadeh et al., 2024), to calibrate the attack. Data extraction focuses on recovering training sequences (like PII) from an LLM by generating continuations from known training prefixes (Carlini et al., 2021). Since we evaluate privacy risks of PII, we use the PII-level threat model proposed by Lukas et al. (2023), which adapted MIAs and extraction attacks to assess PII leakage. Since PII in a document is not the full training sample itself, but a part of it, we use InfoRMIA (Tao & Shokri, 2026) that enables us to measure the privacy leakage of individual sensitive components down to the level of single tokens. With PII-level threat model and InfoRMIA, we explicitly center our analyses on PII leakage. Borkar et al. (2025) observed that adding PII to the training data may increase leakage of other PII. At the same time, they observed the *onion effect* (Carlini et al., 2022b), where removing PII from the data pool also increased leakage of some other PII. (Lukas et al., 2023) showed that scrubbing is imperfect and may not eliminate PII leakage from the *data*. In this work, we go one step further and examine settings where PII are removed with scrubbers before training the LLM, reflecting a common real-world deployment scenarios (OLMo Team et al., 2025; Hernández-Cano et al., 2026; Mishra et al., 2024; Li et al., 2023; Lozhkov et al., 2024).

Scrubbers are tools used to *detect* and then *remove/rewrite* PII from the data (Sweeney, 1996), in this case aiming to prevent leakage from LLMs by not training them on PII. The PII detection is treated as a span labeling task using methods such as regular expressions (Soldaini et al., 2024; Hernández-Cano et al., 2026; Penedo et al., 2024), Named-Entity Recognition (NER) models (Microsoft & Data Privacy Stack, 2026; The Internet Integrity Initiative, 2024), or fine-tuned LLMs (de Bourcy et al., 2026; OpenPipe, 2025; Privasis, 2026; Eternis, 2025). Next, we can either mask or rewrite the detected PII. *Masking* replaces the identified PII based on its detected type with a tag, *e.g.,* `[REDACTED_EMAIL]` (Microsoft & Data Privacy Stack, 2026; The Internet Integrity Initiative, 2024; de Bourcy et al., 2026; Li et al., 2023; Lozhkov et al., 2024; Mishra et al., 2024). Rewriting is an alternative, where synthetic text in the form of PII is used in place of the detected PII (OpenPipe, 2025; Privasis, 2026; Eternis, 2025). In our work, we follow the *hiding in plain sight* principle (Carrell et al., 2013) more closely and aim to ensure that the real and synthetic PII in the data are harder to distinguish. This is the key component for significantly limiting PII leakage missed by scrubbers.

3 Leakage of Missed PII

We examine how PII leakage from a trained model changes after scrubbing is applied to its training data, focusing on PII missed by the scrubbers. We find that *masking* PII with scrubbers increases the leakage of missed PII and identify a positive correlation between detection rates and leakage for the

remaining PII. We also show that current *rewriting* scrubbers do not mitigate the problem because they produce low-quality, generic, or even malformed replacement PII.

3.1 Scrubbing Concentrates Risk On Missed PII

We prepare a dataset of LLM-generated stories filled with five types of synthetic PII and run four masking and four rewriting scrubbers, reporting each scrubber's per-PII-type detection rate. Then, we fine-tune four OLMo2 1B (OLMo Team et al., 2025) target models on distinct random subsets of the data and run InfoRMIA (Tao & Shokri, 2026) on the PII-level, computing TPR@5%FPR as the leakage metric. We also run the extraction attack by Carlini et al. (2021), inputting the training document up to the missed PII, and generate completion from the LLM using greedy decoding. A PII is counted as extracted if the completion matches it verbatim. We measure the *extractability rate*, *i.e.,* count the number of extracted PII divided by the total number of PII missed by the scrubber before training. Our full experimental setup is described in Appx. D.

To isolate impact of scrubbing on the PII it missed, we compare the leakage between two models: M_T (treatment), trained with scrubbed data, and M_C (control) trained on the original, unscrubbed data. With $L(M)$ being a leakage metric, we compute *Gain* as a ratio between $L(M_T)$ and $L(M_C)$ on the same set of missed PII for both models. *MIA Leakage Gain* sets $L =$ TPR@5%FPR, and *Extractability Gain* uses the extractability rate of missed PII as L. Gain of 1.0 means missed PII leaks similarly with or without scrubbing, and a gain above 1.0 means that scrubbing increased the leakage.

We observe in Fig. 2 that both the MIA risk (left) and extraction risk (right) of missed PII increases with the higher detection rate of the scrubbers, with many of the missed PII leaking over $1.5\times$ for MIA and over $3\times$ more for extraction after scrubbing. Interestingly, we observe that the extractability increases only for digit-based PII like credit card numbers, phone numbers, or IP addresses, while emails and names, which are more regular, are not affected. This is not the case for the MIA results, which show that privacy risks depend on the attack and are not uniform across PII types. We provide further details on the results in Appx. F.1.

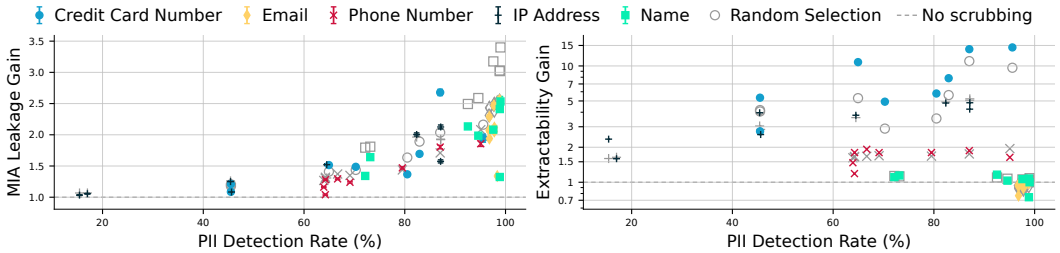


Figure 2: **MIA Leakage and Extractability Gains increase with the detection rate of scrubbers.**

3.2 Leakage Is Independent of Which PII Is Missed

We examine if the leakage can be attributed to the specific PII that are missed by scrubbers. Intuitively, the missed PII could evade detection by scrubbers due to their unique formatting or irregularity, which could be then also causing their increased leakage. To test this hypothesis, we prepare data with the same number of missed PII as in the tested scrubbers. In this controlled experiment, the missed PII are *randomly* selected, so that they are not in any way distinct. If PII missed by scrubbers leak more because they are more distinct, they should leak more than the PII that we left in data at random.

We train new LLMs using the prepared data with randomly selected missed PII, and repeat MIA and extraction experiments from Sec. 3.1. In Fig. 2, we observe that the leakage gain for randomly selected missed PII (gray markers) is similar to the leakage gain of PII missed by scrubbers (color markers) for both MIA and extraction. This indicates that the leakage gain is explained not by the "distinctiveness" of missed PII by a given scrubber, but solely by the number of missed PII.

3.3 Current PII Rewriting is Insufficient

The missed PII become more rare within the LLM's training data, and hence more memorized (Feldman & Zhang, 2020), which increases their leakage. This suggests that the rewriting should better mitigate the leakage of missed PII than masking.

However, our results in Tab. 1 show that rewriting with current tools can lower leakage, but is still insufficient and incurs high leakage. In the case of both Privasis models, we even observe almost no change in the privacy leakage of missed PII after rewriting detected PII. Overall, we find that the replacement PII from all rewriting scrubbers fall short of matching the quality of real PII. In practice, most replacements are placeholder-like, generic, or the wrong type compared to the original PII they replace, which we show in Sec. 5.3. As a result, the missed real PII *remain* outliers and still experience increased leakage.

Table 1: **Current PII rewriting can decrease the leakage of missed PII but is insufficient.**

Scrubber	MIA Leakage Gain		Extraction Gain	
	Mask	Rewrite	Mask	Rewrite
OpenPipe	2.08	1.51 ↘	1.41	1.05 ↘
Anonymizer 4B	2.43	2.23 ↘	1.66	1.48 ↘
Privasis Cl. 0.6B	2.02	2.03 →	1.50	1.55 →
Privasis Cl. 4B	1.90	1.85 →	1.30	1.33 →

4 Our PII-RADAR Scrubber

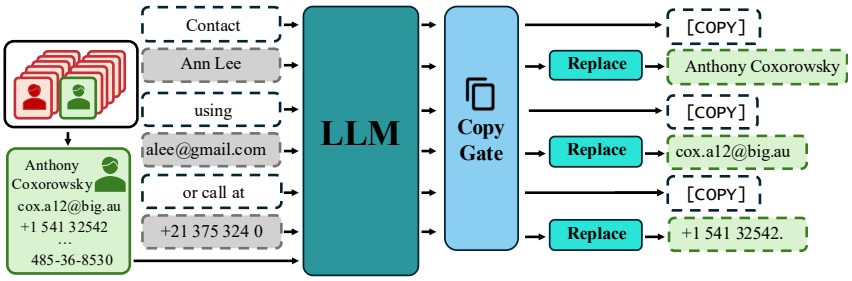


Figure 3: **Overview of our PII-RADAR.** A source document and one of the synthetic user profiles (referred to as a target, here represented by the profile of a fictitious persona Anthony Coxorowsky) are encoded jointly by an LLM in a single forward pass. Copy gate emits [COPY] for non-PII tokens, leaving them unaffected. Detected PII are replaced with matching synthetic PII from the target synthetic user profile, yielding the final scrubbed document.

To address the shortcomings of existing PII scrubbers, we introduce PII-RADAR, which is an LLM-based scrubber trained to identify and replace PII with high-quality synthetic counterparts. We present an overview of our method in Fig. 3. First, we ensure that PII-RADAR's replacements are valid and well-formatted by using data from a pool of 10.7M synthetic user profiles, which are passed to PII-RADAR as prefixes together with the source document to condition the PII replacement. Next, we train an LLM to match the PII replacements in type and format with the originally detected PII. PII-RADAR treats scrubbing as a *sequence editing task*, copying non-PII tokens and replacing detected PII in a single forward pass. Bidirectional attention enabled in the LLM allows using the whole input text document to improve PII detection and replacement. During inference, for each document, we sample a new profile from the pool and pass the profile and the document as a single token sequence to the LLM. We equip the LLM with a copy gate, which is a binary classifier trained to decide which tokens are non-PII (to be copied), and which ones should be replaced. Copy gate takes per-token representations returned by the LLM as input. All the tokens for which the copy gate returns a prediction below a threshold are classified as PII and grouped into spans that are replaced independently. The replace operation removes the original PII and inputs a matching and correctly formatted synthetic PII from the target user profile. The final output is a scrubbed document.

4.1 Replacing PII

The tokens classified as PII by the copy gate are grouped into spans, and their per-token representations are passed to the LLM's unembedding layer, producing logits, from which replacement PII are decoded. The ground truth for the replacement is *target* PII, provided during training for all input PII. If target and input PII were of the same length, training could be done with standard cross-entropy (CE) loss. However, we want PII-RADAR to be able to produce replacements of length independent of the input PII, hence the training target PII should also vary in length. Since this excludes the plain CE objective and simple greedy decoding, we use *CTC* (Graves et al., 2006) instead.

CTC treats the predicted span as an *intermediate* sequence and further decodes it into the final output. It allows some of the intermediate tokens to be `[blank]`, which are dropped from the output, while repeated consecutive tokens are merged, *e.g.,* a four-token intermediate sequence $(a, a, \texttt{[blank]}, b)$ becomes a two-token output (a, b). Since CTC decoding merges and drops tokens, the replacements cannot be longer than the original PII. `PII-RADAR` should be able to produce replacements longer than originals, which we enable by lengthening the PII spans. Specifically, after a PII span is detected by the copy gate, we expand its representation with a *frame expansion* matrix so that the new representation is $10\times$ longer. Only then we compute span's logits and decode with CTC, enabling `PII-RADAR` to replace any detected PII with a synthetic PII which is at most ten times longer.[1]

CTC Decoding. Let x denote an input PII consisting of m tokens, y the target PII consisting of n tokens, V be the token vocabulary, and $D : V^* \to V^*$ be the CTC decoding function, which merges repeated tokens and drops `[blank]` tokens. Let then $\Gamma = \{\hat{y} \in V^m | D(\hat{y}) = y\}$ denote a set of all intermediate sequences $\hat{y}$ that CTC-decode into y. CTC computes likelihood of y based on logits as $p(y|x) = \sum_{\hat{y}\in\Gamma} p(\hat{y}|x)$, where $p(\hat{y}|x) = \prod_{i=1}^{m} p(\hat{y}_i|x)$ is the sequence-level likelihood of the intermediate sequence $\hat{y}$, based on per-token logits returned by the LLM backbone. We do not include the extra `[blank]` token into the vocabulary, and use `[PAD]` token instead, so that we do not need to extend and train the LLM's unembedding matrix. Training loss $\mathcal{L}_{\text{CTC}} = -\log p(y|x)$ is then the negative log-likelihood, computed for all PII spans in the document separately.

PII Matching. We match the *type* and *format* of original and target PII in the training documents. For example, for the template: "I live in `<ADDRESS>`" we draw the fitting format for the `<ADDRESS>`, *e.g.,* `<CITY>, <COUNTRY>`, and fill in PII based on the template, *e.g.,* producing "I live in San Francisco, USA". We have a set of different formats available for each template. For example, another format for the `<ADDRESS>` is: `<CITY>, <STREET>`, which would produce "I live in San Francisco, 747 Howard St". We provide more details in Appx. G.1. With such training data, `PII-RADAR` learns to infer type and format *autonomously* from the full document context, and to output replacement PII that match the originally detected PII.

4.2 High Quality PII via Conditioned Replacement

Prior work on LLM-based PII replacement (Privasis, 2026; Eternis, 2025) trains LLMs to generate PII unconditionally, based only on the input document. This leads to low-quality PII replacements (see Sec. 5.3). We also show that the simple Faker-based rewriting used by OpenPipe (OpenPipe, 2025) fails, as its PII replacements do not follow the format of the originally detected PII (see Sec. 5.3), and are inserted without the full context of the document. Instead of expecting `PII-RADAR` to generate valid PII unconditionally, we condition the replacement on a *target profile*, *i.e.,* an internally consistent set of PII (see Fig. 3 for an example). For a given input document, target profile includes all PII that are targets for the replacement objective for this document. Effectively, `PII-RADAR` is trained to *replace* PII using valid PII from the target profile, and not to *generate* PII replacements. Besides validity, we ensure that the PII inserted by our `PII-RADAR` follow the formatting of the original ones. As in the `<ADDRESS>` example above, both the PII in the document, and the target PII not only match in type, but also in format. Since during deployment we do not know which PII are present in the document of what is their the format, the inserted target profiles contain *all possible* PII that we want `PII-RADAR` to detect and replace, and let it determine *what, how, and where* to input.

5 Empirical Evaluation

`PII-RADAR` significantly decreases the leakage of missed PII, achieves state-of-the-art recall of PII detection across five out of seven benchmarks, generates valid and diverse PII without collapsing to generic placeholders, and is 3–$10\times$ faster than the alternative LLM-based approaches. Specifically, we compare `PII-RADAR`' detection rate against seven other scrubbers across seven benchmarks. We show that at a comparable PII detection rate, other rewriting scrubbers expose missed PII to **up to 3**$\times$ higher leakage than `PII-RADAR`. The replacement PII that our `PII-RADAR` inserts in place of the original PII in the document are of high-quality and matching format, surpassing alternative rewriting scrubbers by a wide margin. Effectively, `PII-RADAR` achieves significantly higher overall PII protection than all tested baselines.

[1] We selected this arbitrarily. It is possible to extend the span $100\times$, but this appears excessive.

5.1 EXPERIMENTAL SETUP

Data. We use seven public benchmarks that vary by source, document type, length, and language. Nemotron-PII (Steier et al., 2025) is a large synthetic corpus of unstructured and structured documents, such as forms. We use Privasis-Zero (Kim et al., 2026) `hard` and `vanilla` benchmarks of long LLM-generated records. SPY (Savkin et al., 2025) provides *legal* and *medical* sets of documents, filled with synthetic PII. PII-Masking-300k (Ai4Privacy, 2024) and REDACT (Vats et al., 2026) cover multilingual documents. Since the benchmarks differ in the PII types and labels they present, we map their PII labels onto a single taxonomy, covering 18 PII types, detailed in Appx. C.1.

Baselines. We evaluate `PII-RADAR` against seven PII scrubbers: Presidio (Microsoft & Data Privacy Stack, 2026), Piiranha (The Internet Integrity Initiative, 2024), OpenAI's Privacy Filter (OPF) (de Bourcy et al., 2026), OpenPipe (OpenPipe, 2025), Privasis Cleaner 0.6B and 4B (Privasis, 2026), and Anonymizer 4B (Eternis, 2025). Presidio and Piiranha use text encoders for PII detection and mask the PII, while OPF, OpenPipe, and Privasis Cleaner, and Anonymizer use LLM-based PII detection methods. OPF masks PII, OpenPipe re-generates the full input autoregressively in two modes: masking, and replacement using Faker (Faraglia). Privasis Cleaner models also re-generate the input, while the PII are paraphrased (either by replacing, redacting, or removing), and Anonymizer returns tool-call-like outputs with identified PII and the proposed replacement values.

Metrics. Each token in a benchmark document is annotated as PII or non-PII, with PII tokens grouped in *gold span*, *i.e.,* sequences of tokens containing one PII instance. We evaluate PII detection at two levels of granularity: span- and token-level. At the span level, a True Positive (TP) is a gold span which all tokens are classified as PII, a False Negative (FN) is a gold span for which at least one token is misclassified as non-PII, and a False Positive (FP) is a span of tokens predicted as PII outside every gold span. At the token level, TP is a correctly classified gold span token, FP is a non-PII token classified as PII, and FN is a PII token classified as non-PII. As a result, if a PII is only partially masked, it gets both TP and FN. We compute token-level metrics with Unicode word segments, which also split scripts written without spaces. **Recall** is then $\frac{TP}{TP+FN}$, **precision** $\frac{TP}{TP+FP}$, and **F1** is $2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$.

LLM in `PII-RADAR`. We train the LLM for a single epoch on 2.5M documents, with multilingual templates sourced from OpenPII-1.5M (Ai4Privacy, 2026) and Privasis-Zero (Kim et al., 2026). We initialize the LLM in `PII-RADAR` from Qwen3-0.6B-Base (Yang et al., 2025a), and enable bidirectional attention, following approaches for sequence editing (Stahlberg & Kumar, 2020; Zhang et al., 2023) and OPF (de Bourcy et al., 2026). Bidirectional attention lets `PII-RADAR` use the full-document context to better determine which tokens need replacing, and what type of PII in what format should be inserted instead. We train `PII-RADAR` using LoRA (Hu et al., 2021) with rank $r = 64$, and add the copy gate and frame expansion matrix into the set of trainable parameters. The full training and data mixture details can be found in Appx. G. While focusing on our taxonomy, we evaluate each scrubber only on the PII categories it claims to support.

5.2 `PII-RADAR` IS FAST AND ACHIEVES HIGH DETECTION RATE

Table 2: **Recall** on seven benchmarks at span and token levels (higher is better). Each system is scored only on the PII kinds it claims to support. *Average* pools counts over benchmarks. Best in **bold**; runner-up underlined.

Dataset →	Nemotron		300k		Privasis hard		Privasis vanilla		REDACT		SPY legal		SPY medical		Average	
Scrubber ↓	Span	Token	Span	Token	Span	Token	Span	Token	Span	Token	Span	Token	Span	Token	Span	Token
Anonymizer 4B	0.930	0.945	0.918	0.920	0.529	0.594	0.609	**0.689**	0.730	0.758	0.962	0.965	0.940	0.946	0.884	0.888
OpenPipe	0.789	0.769	0.946	0.961	0.314	0.253	0.379	0.311	0.237	0.200	0.840	0.899	0.844	0.908	0.709	0.652
Piiranha	0.749	0.797	0.831	0.844	0.600	**0.635**	0.597	0.635	0.420	0.567	0.518	0.642	0.489	0.623	0.707	0.751
Presidio	0.769	0.731	0.359	0.408	0.558	0.552	0.569	0.569	0.444	0.404	0.611	0.559	0.586	0.536	0.612	0.581
OPF	0.828	0.834	**0.975**	**0.981**	0.668	0.615	**0.665**	0.598	0.765	0.815	0.954	0.943	0.947	0.936	0.858	0.871
Privasis Cl. 4B	0.803	0.808	0.613	0.596	0.587	0.525	0.593	0.528	0.492	0.513	0.868	0.850	0.863	0.841	0.685	0.674
Privasis Cl. 0.6B	0.862	0.853	0.687	0.689	0.564	0.536	0.561	0.545	0.532	0.539	0.881	0.884	0.850	0.861	0.744	0.726
RADAR (ours)	**0.978**	**0.979**	0.947	0.926	**0.696**	0.619	0.660	0.587	**0.829**	**0.828**	**0.980**	**0.991**	**0.976**	**0.989**	**0.936**	**0.923**

We show per-benchmark results in Tab. 2, comparing masking and rewriting scrubbers' recall across benchmarks. We report precision and F1 in Appx. C.2, in Tab. 4 and Tab. 5 respectively. `PII-RADAR` has the highest span-level recall on five of seven benchmarks and the highest overall

recall, when pooled across benchmarks (0.936 at the span level, 0.923 at the token level). OPF leads on `pii-masking-300k` (0.975 against 0.947) and on Privasis vanilla (0.665 against 0.660).

PII-RADAR performs so well on detection because its training objective is significantly simpler than that of other baselines. PII-RADAR predicts PII or non-PII for each token, while other methods try to classify PII into multiple types while detecting exact boundaries of PII spans (Presidio, PIIranha, OPF), are trained to re-generate *the full input document* with PII removed (OpenPipe, Privasis), or emit valid tool call JSON strings (Anonymizer). Similarly, PII-RADAR *replaces* PII, which is simpler than *generating* valid PII, which other LLM-based methods attempt to do. Effectively, PII-RADAR uses the capacity of its LLM much better, improving detection. It also lets us use a smaller LLM than the baselines (except Privasis 0.6B). As a result, PII-RADAR offers the best speed for the PII detection rate it provides (see Fig. 4), while faster methods are masking-only and have much lower PII token recall. We run some of the baselines with a faster LLM generation backend than officially supported, improving their runtimes by orders of magnitude. We report the original runtimes in Appx. C.4.

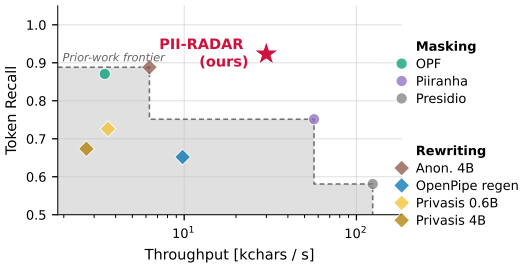


Figure 4: **PII-RADAR is on the Pareto frontier for PII Detection and Speed.**

5.3 PII Replacements

We assess the quality of replacement PII generated by rewriting scrubbers and the PII returned by PII-RADAR. We collect them when running the detection benchmarks, using new PII produced for correctly detected PII. We start by checking if the replacements follow formatting rules specific to the type of PII they substitute. For example, a phone number should have from 8 to 15 digits, and a birth date should be a real calendar day. We also separately check if the replacement is a *placeholder*, *e.g.*, `[MASKED]`, `XXXX XXXX XXXX`, or `@example.com` (rules in Appx. E.2, per-type breakdown in Fig. 12). In Fig. 5 (left), we find that current rewriting scrubbers do not pass these simple checks: 38–78% of their values are placeholders, and another 9–23% of the replaced spans are format-invalid. PII-RADAR writes no placeholders and produces valid replacements **92.3**% of the time, close to the 94% validity rate of the ground-truth PII (which are not always correct, see Fig. 11).

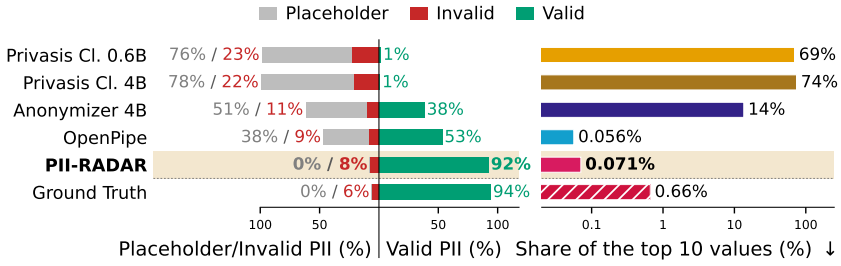


Figure 5: **PII-RADAR Inserts Valid (left) and Diverse PII (right).**

We also test the *diversity* of the replacement PII, reporting the share of all replacements that each distinct value takes, sorted by frequency rank. In Fig. 5 (right) we show that PII-RADAR produces PII at least as diverse as the ones present in the input documents for the seven benchmarks. Specifically, the top-10 replacements returned by PII-RADAR cover only 0.07% of all returned values (0.66% for the ground truth), while Anonymizer 4B returns the same 10 values 13.5% of the time and Privasis Cleaner 69.5% (0.6B) and 74.1% (4B) of the time. Only OpenPipe produces slightly more diverse PII (0.06% on its top 10), which are sampled from Faker, almost at random. We also note that its PII is often placeholder-like and does not match the expected formatting, as seen in Fig. 5 (left).

5.4 Preventing Leakage of Missed PII

We wrap up by confirming that PII-RADAR achieves what we sought from the beginning. In Sec. 3 we show that scrubbers do not detect all PII in the data, and that the PII they miss becomes significantly more prone to privacy risks than in the counterfactual reality with no scrubbing applied. We also

show that the detection rate drives the increase in leakage, which raises a question: if PII−RADAR detects PII so well, wouldn't it increase the leakage of PII it missed even more than other scrubbers? We test this by repeating MIA and extraction experiments from Sec. 3.1 on the same dataset, passing it through PII−RADAR and replacing detected PII as in the other evaluations in that section.

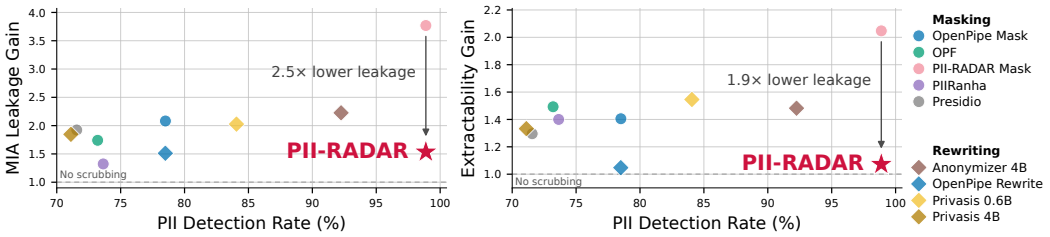


Figure 6: **PII−RADAR prevents leakage of PII it misses, by its high quality replacement PII.**

Our results in Fig. 6 show that for both MIA and data extraction we successfully mitigate the increase of leakage for the PII PII−RADAR misses. We isolate the impact of replacing with our PII on leakage, by comparing it to the leakage when we mask the detected PII instead of replacing (PII-RADAR Mask). Contrary to other rewriting scrubbers, which produce PII that behave like masks (see Tab. 1), PII−RADAR' PII lower the leakage of missed PII of $\mathbf{2.5}\times$ for MIA and $\mathbf{1.9}\times$ for extraction, at the same detection rates. Crucially, scrubbing with PII−RADAR has almost *no effect* on the extractability of the missed PII, which highlights that high quality replacement PII of PII−RADAR protect the privacy of any real PII remaining in the data.

Finally, we show that high-quality replacement PII provide plausible deniability: even if a PII is extracted from an LLM, it is no longer clear whether the value is real or synthetic. Specifically, for all rewriting scrubbers we observe that many of the PII we extract from LLMs is synthetic. However, for other baselines, it is easy to identify real PII among the extracted, which is expected, as they produce low-quality replacements (see Fig. 5, left) and are repetitive. To separate them, we run a label-free classifier, that clusters the extracted vales per PII type by their character n-gram statistics with a threshold of FPR=10%. In Tab. 3 we show that up to 91% of the real PII from other baselines are identifiable, while for PII−RADAR it is only only $\mathbf{16}\%$,over $8\times$ fewer than the closest competitor, and only 6% above random guessing. Once again, PII−RADAR inserts PII that, even when extracted, are hard to filter out to identify the real PII among all extracted values.

Table 3: **PII−RADAR's PII are Similar to Real.** Among *Extracted* PII, only some of them are *Real*. At FPR=10%, only some of them are *Identified* correctly as real.

Scrubber	Extracted	Real	Identified
OpenPipe	6,726	2,014	1,474
Privasis Cl. 0.6B	2,985	2,299	1,890
Privasis Cl. 4B	3,270	2,811	1,969
Anonymizer 4B	7,829	1,386	207
PII−RADAR	10,634	**160**	**25**

6 Conclusions

Scrubbing is the practically deployed privacy measure that LLM providers implement to prevent leakage of PII from their models. Yet, scrubbers were commonly evaluated based on how many PII they detect. We showed that this measure is incomplete as it does not take the downstream privacy leakage of the missed PII from trained models into account. We showed that the fewer PII are missed the higher their leakage, independent of the instances of missed PII. Additionally, we found that current synthetic PII rewrites do not solve the high leakage of missed PII, since they collapse onto a handful of placeholder strings, resembling the masks. To overcome this limitation, we proposed PII−RADAR, which replaces each detected PII in place with a format-matched value from a consistent synthetic profile, substantially limiting the leakage and leading recall on six of seven benchmarks while running 3–10$\times$ faster than other replacement scrubbers. As scrubbers will likely never be able to perfectly detect every PII, our results show that future methods should assess scrubbers based on the resulting downstream privacy risks and not by their detection quality alone. Our method indicates that the focus of PII scrubbing should shift towards rewriting the detected PII with high-quality synthetic values that match the target format.

AI USAGE DISCLOSURE

In our work, we used generative AI to produce synthetic datasets (Long Stories (Short), details in Appx. D.1), and to provide feedback on experiments (red-team hypotheses based on preliminary InfoRMIA results, and alternative explanations for the observed results). We used AI coding agents to implement the `PII-RADAR` model architecture, along with boilerplate training, data processing, and evaluation code. We provided the agents with the full code specification and requirements to run the experiments. We evaluated the code for correctness, manually tested it, and iteratively improved it (via the agents) based on human feedback and verification. AI coding agents implemented result visualization utilities based on human-written analysis code to enable easier iteration on the final design of paper-grade plots. We cross-checked all results against those obtained with human-written code.

We did not use generative AI to design any of the experiments in Sec. 3, nor did we use AI to design the architecture of `PII-RADAR`. The only exception is the use of CTC decoding (see Sec. 4), which was proposed by GPT-5.6 Sol, originally citing Dekel et al. (2026). We did not clean or reformat data using AI. AI implemented *training data processing pipelines* according to our specification, which is a use case distinct from passing raw data through an LLM and recording the output. We did not use AI for any theoretical models, mathematical claims or proofs, or translation. We used AI chatbots for brainstorming, simple literature queries, and to help polish the final manuscript.

We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

In our work, we show that existing scrubbing tools can increase PII leakage from LLMs trained on internet-scraped text data that may contain PII. We show that datasets anonymized with these tools may still harm PII the scrubbing tools miss, which is alarming for protecting the privacy of real people whose PII (inadvertently) ended up in LLM training data. At the same time, we present `PII-RADAR`, which solves this issue by inserting high-quality synthetic PII instead of simply masking the detected PII from text. `PII-RADAR` prevents leakage from the PII it misses, unlike all existing tools. Since `PII-RADAR` relies only on synthetic (as in: fake) PII, it does not put any PII of a real person at risk. Effectively, we contribute a method that provides significantly better protection of vulnerable information from privacy risks.

REPRODUCIBILITY STATEMENT

Upon acceptance, we release the full code base with all necessary components for reproducing our results for Sec. 3, model-weights, and implementation of `PII-RADAR` (Sec. 4), and benchmarking `PII-RADAR` as well as other scrubbers (Sec. 5). The code base contains full Python environment setup, and concrete steps to reproduce each of our results. The code also supports recreation of all synthetic data sets used for experiments (Long Stories (Short), as well as profiles for the 10.7M profile pool).

REFERENCES

Ai4Privacy. PII masking 400k: A dataset for training de-identification models. Hugging Face dataset, 2024. URL https://huggingface.co/datasets/ai4privacy/pii-masking-400k.

Ai4Privacy. Pii-masking-300k, 2024. URL https://huggingface.co/datasets/ai4privacy/pii-masking-300k.

Ai4Privacy. Openpii 1.5m: Multilingual pii masking dataset with asia pacific extension, 2026. URL https://huggingface.co/datasets/ai4privacy/pii-masking-openpii-1.5m.

Maksym Andriushchenko and Nicolas Flammarion. Does refusal training in llms generalize to the past tense? In *International Conference on Learning Representations*, volume 2025, pp. 85406–85420, 2025.

Rohan Anil, Badih Ghazi, Vineet Gupta, Ravi Kumar, and Pasin Manurangsi. Large-scale differentially private bert. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pp. 6481–6491, 2022.

Anthropic. Claude fable 5.1 & claude mythos 5.1 system card. Technical report, Anthropic, 2026a. URL https://www-cdn.anthropic.com/0339e6a7c5c7b87f5c07798616dc32c215d14235/Claude%20Fable%205.1%20&%20Claude%20Mythos%205.1%20System%20Card.pdf. Accessed September 2026.

Anthropic. Claude Sonnet 5 system card, 2026b. URL https://www.anthropic.com/news/claude-sonnet-5. Accessed September 2026.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

Jaydeep Borkar, Matthew Jagielski, Katherine Lee, Niloofar Mireshghallah, David A Smith, and Christopher A Choquette-Choo. Privacy ripple effects from adding or removing personal information in language model training. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 18703–18726, 2025.

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

Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramèr. Membership inference attacks from first principles. In *2022 IEEE Symposium on Security and Privacy (SP)*, pp. 1897–1914. IEEE, 2022a.

Nicholas Carlini, Matthew Jagielski, Chiyuan Zhang, Nicolas Papernot, Andreas Terzis, and Florian Tramer. The privacy onion effect: Memorization is relative. *Advances in Neural Information Processing Systems*, 35:13263–13276, 2022b.

David Carrell, Bradley Malin, John Aberdeen, Samuel Bayer, Cheryl Clark, Ben Wellner, and Lynette Hirschman. Hiding in plain sight: use of realistic surrogates to reduce exposure of protected health information in clinical text. *Journal of the American Medical Informatics Association*, 20(2): 342–348, 2013.

Aaron Chatterji, Thomas Cunningham, David J. Deming, Zoe Hitzig, Christopher Ong, Carl Yan Shan, and Kevin Wadman. How people use ChatGPT. Working Paper 34255, National Bureau of Economic Research, 2025.

Charles de Bourcy, Sahra Ghalebikesabi, Avi Schwarzschild, Alex Gorbachev, Mihai Maruseac, Annie Chu, Vol Kyrylov, Tong Mu, Ally Bennett, Andy Nguyen, Casey Meehan, Jessica Gan Lee, Shane Bauer, Harold Nguyen, Rodolpho Eckhardt, Yuqi Liu, Charlie Oxborough, Marco Rougeth, Omar Chedid, Caio Costa, Yash Parikh, Yao Li, Congzheng Song, Om Thakkar, and Vinnie Monaco. Model card for openai privacy filter, 2026. URL https://arxiv.org/abs/2608.18274.

Avihu Dekel, Samuel Thomas, Takashi Fukada, and George Saon. Nle: Non-autoregressive llm-based asr by transcript editing. *arXiv preprint arXiv:2603.08397*, 2026.

Cynthia Dwork. Differential privacy. In *International colloquium on automata, languages, and programming*, pp. 1–12. Springer, 2006.

Eternis. Anonymizer model series: Small language models for pii anonymization, 2025. URL https://huggingface.co/collections/eternisai/anonymizer-model-series. Accessed: 2026.

Daniele Faraglia. Faker, . URL https://github.com/joke2k/faker.

Vitaly Feldman. Does learning require memorization? a short tale about a long tail. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pp. 954–959, 2020.

Vitaly Feldman and Chiyuan Zhang. What neural networks memorize and why: Discovering the long tail via influence estimation. *Advances in Neural Information Processing Systems*, 33:2881–2891, 2020.

Google DeepMind. Gemini 3 pro model card. Technical report, May 2026. URL https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-3-Pro-Model-Card.pdf. Accessed September 2026.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pp. 369–376, 2006.

Kathrin Grosse and Nico Ebert. Prevalence of security and privacy risk-inducing usage of ai-based conversational agents. *IEEE Access*, 2026.

Jamie Hayes, Ilia Shumailov, Christopher A. Choquette-Choo, Matthew Jagielski, Georgios Kaissis, Milad Nasr, Meenatchi Sundaram Muthu Selva Annamalai, Niloofar Mireshghallah, Igor Shilov, Matthieu Meeus, Yves-Alexandre de Montjoye, Katherine Lee, Franziska Boenisch, Adam Dziedzic, and A. Feder Cooper. Exploring the limits of strong membership inference attacks on large language models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=x0i7wvRLHK.

Alejandro Hernández-Cano, Alexander Hägele, Allen Hao Huang, Angelika Romanou, Antoni-Joan Solergibert, Barna Pasztor, Bettina Messmer, Dhia Garbaya, Eduard Frank Ďurech, Ido Hakimi, et al. Apertus: Democratizing open and compliant llms for global language environments. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 46877–46955, 2026.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. spaCy: Industrial-strength natural language processing in Python. Zenodo, 2020. URL https://doi.org/10.5281/zenodo.1212303.

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. *arXiv preprint arXiv:2106.09685*, 2021.

Jie Huang, Hanyin Shao, and Kevin Chen-Chuan Chang. Are large pre-trained language models leaking your personal information? In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pp. 2038–2047. Association for Computational Linguistics, 2022.

Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, et al. Llama guard: Llm-based input-output safeguard for human-ai conversations. *arXiv preprint arXiv:2312.06674*, 2023.

Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*, 2019.

Hyunwoo Kim, Niloofar Mireshghallah, Michael Duan, Rui Xin, Shuyue Stella Li, Jaehun Jung, David Acuna, Qi Pang, Hanshen Xiao, G. Edward Suh, Sewoong Oh, Yulia Tsvetkov, Pang Wei Koh, and Yejin Choi. Privasis: Synthesizing the largest "public" private dataset from scratch. *arXiv preprint arXiv:2602.03183*, 2026.

Leap Beyond Analytics. scrubadub: Clean personally identifiable information from dirty dirty text. Python package, version 2.0.1, 2023. URL https://github.com/LeapBeyond/scrubadub. Accessed September 2026.

Yichong Leng, Xu Tan, Wenjie Liu, Kaitao Song, Rui Wang, Xiang-Yang Li, Tao Qin, Ed Lin, and Tie-Yan Liu. Softcorrect: Error correction with soft detection for automatic speech recognition. In *proceedings of the AAAI conference on artificial intelligence*, volume 37, pp. 13034–13042, 2023.

Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, et al. Starcoder: may the source be with you! *arXiv preprint arXiv:2305.06161*, 2023.

Jindřich Libovickỳ and Jindřich Helcl. End-to-end non-autoregressive neural machine translation with connectionist temporal classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 3016–3021, 2018.

Anton Lozhkov, Raymond Li, Loubna Ben Allal, Federico Cassano, Joel Lamy-Poirier, Nouamane Tazi, Ao Tang, Dmytro Pykhtar, Jiawei Liu, Yuxiang Wei, et al. Starcoder 2 and the stack v2: The next generation. *arXiv preprint arXiv:2402.19173*, 2024.

Nils Lukas, Ahmed Salem, Robert Sim, Shruti Tople, Lukas Wutschitz, and Santiago Zanella-Béguelin. Analyzing leakage of personally identifiable information in language models. In *2023 IEEE Symposium on Security and Privacy (SP)*, pp. 346–363. IEEE, 2023. arXiv:2302.00539.

Bartłomiej Marek, Lorenzo Rossi, Vincent Hanke, Xun Wang, Michael Backes, Franziska Boenisch, and Adam Dziedzic. Benchmarking empirical privacy protection for adaptations of large language models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=jY7fAo9rfK.

Microsoft and Data Privacy Stack. Microsoft Presidio: Data protection and de-identification SDK. Open-source software, presidio-analyzer 2.2.364, 2026. URL https://github.com/data-privacy-stack/presidio. Accessed September 2026.

Niloofar Mireshghallah, Maria Antoniak, Yash More, Yejin Choi, and Golnoosh Farnadi. Trust no bot: Discovering personal disclosures in human-llm conversations in the wild. *arXiv preprint arXiv:2407.11438*, 2024.

Mayank Mishra, Matt Stallone, Gaoyuan Zhang, Yikang Shen, Aditya Prasad, Adriana Meza Soria, Michele Merler, Parameswaran Selvam, Saptha Surendran, Shivdeep Singh, et al. Granite code models: A family of open foundation models for code intelligence. *arXiv preprint arXiv:2405.04324*, 2024.

John Xavier Morris, Chawin Sitawarin, Chuan Guo, Narine Kokhlikyan, G. Edward Suh, Alexander M Rush, Kamalika Chaudhuri, and Saeed Mahloujifar. How much can language models memorize? In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=bA6BgSbaUi.

Milad Nasr, Javier Rando, Nicholas Carlini, Jonathan Hayase, Matthew Jagielski, A. Feder Cooper, Daphne Ippolito, Christopher A. Choquette-Choo, Florian Tramèr, and Katherine Lee. Scalable extraction of training data from aligned, production language models. In *The Thirteenth International Conference on Learning Representations*, 2025. arXiv:2311.17035.

OLMo Team, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, et al. 2 OLMo 2 Furious. In *Second Conference on Language Modeling (COLM)*, 2025. arXiv:2501.00656.

OpenAI. GPT-5.6 Sol, 2026a. URL https://deploymentsafety.openai.com/gpt-5-6-preview/gpt-5-6-preview.pdf. Accessed September 2026.

OpenAI. Gpt-5.5 system card. Technical report, OpenAI, 2026b. URL https://deploymentsafety.openai.com/gpt-5-5/gpt-5-5.pdf. Accessed September 2026.

OpenAI. How your data is used to improve model performance. OpenAI Help Center, 2026. URL https://help.openai.com/en/articles/5722486-how-your-data-is-used-to-improve-model-performance. Accessed September 2026.

OpenPipe. pii-redaction: Detect and redact PII locally. Python package, 2025. URL https://github.com/OpenPipe/pii-redaction.

Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, Thomas Wolf, et al. The fineweb datasets: Decanting the web for the finest text data at scale. *Advances in Neural Information Processing Systems*, 37:30811–30849, 2024.

Privasis. Privasis-cleaner-0.6B. Hugging Face model, 2026. URL https://huggingface.co/Arsh9210/Privasis-Cleaner-0.6B.

Xiangyu Qi, Yi Zeng, Tinghao Xie, Pin-Yu Chen, Ruoxi Jia, Prateek Mittal, and Peter Henderson. Fine-tuning aligned language models compromises safety, even when users do not intend to! In *International Conference on Learning Representations*, volume 2024, pp. 30988–31043, 2024.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Traian Rebedea, Razvan Dinu, Makesh Narsimhan Sreedhar, Christopher Parisien, and Jonathan Cohen. Nemo guardrails: A toolkit for controllable and safe llm applications with programmable rails. In *Proceedings of the 2023 conference on empirical methods in natural language processing: system demonstrations*, pp. 431–445, 2023.

Chitwan Saharia, William Chan, Saurabh Saxena, and Mohammad Norouzi. Non-autoregressive machine translation with latent alignments. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1098–1108, 2020.

Oscar Sainz, Jon Campos, Iker García-Ferrero, Julen Etxaniz, Oier Lopez de Lacalle, and Eneko Agirre. NLP evaluation in trouble: On the need to measure LLM data contamination for each benchmark. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 10776–10787, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.722. URL https://aclanthology.org/2023.findings-emnlp.722/.

Maksim Savkin, Timur Ionov, and Vasily Konovalov. SPY: Enhancing privacy with synthetic PII detection dataset. In Abteen Ebrahimi, Samar Haider, Emmy Liu, Sammar Haider, Maria Leonor Pacheco, and Shira Wein (eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 4: Student Research Workshop)*, pp. 236–246, Albuquerque, USA, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-192-6. doi: 10.18653/v1/2025.naacl-srw.23. URL https://aclanthology.org/2025.naacl-srw.23/.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Xinyue Shen, Zeyuan Chen, Michael Backes, Yun Shen, and Yang Zhang. " do anything now": Characterizing and evaluating in-the-wild jailbreak prompts on large language models. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pp. 1671–1685, 2024.

Weijia Shi, Anirudh Ajith, Mengzhou Xia, Yangsibo Huang, Daogao Liu, Terra Blevins, Danqi Chen, and Luke Zettlemoyer. Detecting pretraining data from large language models. In *The Twelfth International Conference on Learning Representations*, 2024. arXiv:2310.16789.

Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pp. 3–18. IEEE, 2017.

Aaditya K. Singh, Muhammed Yusuf Kocyigit, Andrew Poulton, David Esiobu, Maria Lomeli, Gergely Szilvasy, and Dieuwke Hupkes. Evaluation data contamination in llms: how do we measure it and (when) does it matter?, 2024. URL https://arxiv.org/abs/2411.03923.

Amer Sinha, Thomas Mesnard, Ryan McKenna, Daogao Liu, Christopher A Choquette-Choo, Yangsibo Huang, Da Yu, George Kaissis, Zachary Charles, Ruibo Liu, et al. Vaultgemma: A differentially private gemma model. *arXiv preprint arXiv:2510.15001*, 2025.

Luca Soldaini, Rodney Kinney, Akshita Bhagia, Dustin Schwenk, David Atkinson, Russell Authur, Ben Bogin, Khyathi Chandu, Jennifer Dumas, Yanai Elazar, et al. Dolma: An open corpus of three trillion tokens for language model pretraining research. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15725–15788. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.840.

Robin Staab, Mark Vero, Mislav Balunovic, and Martin Vechev. Language models are advanced anonymizers. In *International Conference on Learning Representations*, volume 2025, pp. 98558–98598, 2025.

Felix Stahlberg and Shankar Kumar. Seq2edits: Sequence transduction using span-level edit operations. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 5147–5159, 2020.

Amy Steier, Andre Manoel, Alexa Haushalter, and Maarten Van Segbroeck. Nemotron-pii: Synthesized data for privacy-preserving ai, 2025. URL `https://huggingface.co/datasets/nvidia/Nemotron-PII`.

Latanya Sweeney. Replacing personally-identifying information in medical records, the scrub system. In *Proceedings of the AMIA annual fall symposium*, pp. 333, 1996.

Jiashu Tao and Reza Shokri. Information-theoretic membership inference for granular quantification of memorization. In *The Fourteenth International Conference on Learning Representations*, 2026.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

The Internet Integrity Initiative. Piiranha-v1: Detect personal information. Hugging Face model, 2024. URL `https://huggingface.co/iiiorg/piiranha-v1-detect-personal-information`. Accessed September 2026.

Guneesh Vats, Anubha Agrawal, Shikha Singhal, Ajita Dash, Praison Selvaraj, Vidhan Jhawar, Ranga Prasad Chenna, and Bharadwaj Y M G. Redact: A systematically controlled multilingual benchmark for personal information detection, 2026. URL `https://arxiv.org/abs/2606.19881`.

Krzysztof Wróbel, Krzysztof, Jan Maria Kowalski, Jerzy Surma, Igor Ciuciura, et al. Bielik guard: Efficient polish language safety classifiers for llm content moderation. *arXiv preprint arXiv:2602.07954*, 2026.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

Tianyu Yang, Xiaodan Zhu, and Iryna Gurevych. Robust utility-preserving text anonymization based on large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 28922–28941, 2025b.

Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. Privacy risk in machine learning: Analyzing the connection to overfitting. In *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, pp. 268–282. IEEE, 2018. doi: 10.1109/CSF.2018.00027.

Youliang Yuan, Wenxiang Jiao, Wenxuan Wang, Jen-tse Huang, Jiahao Xu, Tian Liang, Pinjia He, and Zhaopeng Tu. Refuse whenever you feel unsafe: Improving safety in llms via decoupled refusal training. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3149–3167, 2025.

Sajjad Zarifzadeh, Philippe Liu, and Reza Shokri. Low-cost high-power membership inference attacks. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 58244–58282, 2024.

Jingyang Zhang, Jingwei Sun, Eric Yeats, Yang Ouyang, Martin Kuo, Jianyi Zhang, Hao Yang, and Hai Li. Min-k%++: Improved baseline for detecting pre-training data from large language models. In *The Thirteenth International Conference on Learning Representations*, 2025. arXiv:2404.02936.

Yu Zhang, Yue Zhang, Leyang Cui, and Guohong Fu. Non-autoregressive text editing with copy-aware latent alignments. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 7075–7085, 2023.

Zhiping Zhang, Michelle Jia, Hao-Ping Lee, Bingsheng Yao, Sauvik Das, Ada Lerner, Dakuo Wang, and Tianshi Li. "it'sa fair game", or is it? examining how users navigate disclosure risks and benefits when using llm-based conversational agents. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, pp. 156. Association for Computing Machinery New York, NY, USA, 2024.

Zihao Zhao and Anjalie Field. Controlled generation for private synthetic text. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 32708–32723, 2025.

A LIMITATIONS

Our work focuses on text-only LLMs, omitting multimodal language models which process data in many modalities. The privacy risks associated with PII in multimodal models remain insufficiently understood and therefore left to future work. We train PII-RADAR to handle profile of a single synthetic user as an input, while extending to multi-profile cases is left for future work. We restrict our data to *synthetic PII*, which may slightly differ from the real PII distribution. This design choice for experiments and training of PII-RADAR is motivated by responsibility to preserve individual's privacy in this settings and avoid introducing new privacy risks during development, deployment and evaluation. The main motivation and priority of this work is to **increase privacy protection**, and replacing identified PII with real data would have the opposite effect. Hence, we use synthetic PII throughout, ensuring that PII-RADAR does not depend on or place at risk any individual.

B EXTENDED BACKGROUND AND RELATED WORK

PII scrubbing. Existing scrubbers fall into two broad families: rule-based and model-based, with a separate design choice around how the PII are replaced. **Rule-based systems**, such as scrubadub (Leap Beyond Analytics, 2023) and Presidio (Microsoft & Data Privacy Stack, 2026), detect PII by surface form using regular expressions, checksum validation, and deny lists, then merge matched spans into type-tagged placeholders. **Model-based systems** reframe scrubbing as sequence labeling. Presidio delegates name detection to spaCy (Honnibal et al., 2020) or transformer NER. Piiranha (The Internet Integrity Initiative, 2024) fine-tunes mDeBERTa-v3 on the Ai4Privacy corpus (Ai4Privacy, 2024), while OpenAI's Privacy Filter (OPF) (de Bourcy et al., 2026) is a 1.5B-parameter bidirectional token-classification model that adapts an autoregressive checkpoint into a sparse mixture-of-experts encoder and detects PII across 8 privacy span categories. Privasis-Cleaner (Kim et al., 2026; Privasis, 2026) fine-tunes an LLM via supervised fine-tuning on synthetic *(text, instruction, sanitized text)* triplets, prompting the model with raw text and a target-category instruction to autoregressively generate a sanitized output. Anonymizer (Eternis, 2025) introduces a tool-calling approach for PII regeneration, utilizing an LLM fine-tuned via Group Relative Policy Optimization (Shao et al., 2024). In our work we use only Anonymizer-4B, and omit smaller counterparts (0.6B and 1.7B) (Eternis, 2025), as in our evaluation the 0.6B version often produced no parseable output (34–72% of documents returned unchanged), and 1.7B "replaced" PII with values identical to the originals (55–80% of documents unchanged). OpenPipe (OpenPipe, 2025) can operate in two modes: masking and replacing, where replacing substitutes surrogates across twenty entity types using Faker, drawing replacements at random with no regard for document context or semantic plausibility. We evaluate both modes. **Paraphrasing** is another anonymization technique, where a document is processed in full, without separate detection and rewriting steps. Staab et al. (2025); Yang et al. (2025b) propose multi-turn LLM-based paraphrasing of full text documents, where the paraphrasing LLM outputs a new document, tested for PII via an "adversarial" LLM that provides feedback to the paraphraser to generate a new version of the document, until the adversarial LLM does not infer any PII. This setup is distinct from scrubbing, as it does not explicitly aim to detect and replace, but instead changes the full document via paraphrasing, and hence is omitted from this work.

Non-Autoregressive Sequence Editing methods aim to perform single-shot editing of text, either for error correction (Stahlberg & Kumar, 2020; Zhang et al., 2023) or better automated speech recognition (Leng et al., 2023; Dekel et al., 2026). Other use-cases include fast machine translation (Saharia et al., 2020; Libovický & Helcl, 2018), where bidirectional models enable way quicker inference compared to sequential generation. PII-RADAR usage of bidirectional attention, non-autoregressive replacement of PII, and CTC decoding is inspired by these works. Besides these, prior work showed that controlling the generation from LLMs can be achieved via *control codes* (Keskar et al., 2019; Zhao & Field, 2025), which in our case are the profiles we insert as prefixes of the documents, to guide the replacement towards valid PII.

C Extended Evaluation

Tokenization. For token-level evaluation, we use word units from the Unicode word-boundary rules, implemented in ICU[2]. This is more faithful than whitespace splitting, *e.g.,* splits phone numbers at punctuation, and uses dictionaries to segment scripts that do not mark word boundaries with spaces, including Chinese, Japanese, and Thai. We remove tokens with no letters or digits, and we ignore whitespace and punctuation when testing coverage.

C.1 PII Taxonomy

Our taxonomy covers names, email addresses, phone numbers, addresses and their components (street, city, postal code, and building number), dates of birth, account and credit-card numbers, SSNs, national IDs, passports, driver's licenses, tax IDs, IPv4 and IPv6 addresses. With our taxonomy, we prioritize high-sensitivity PII, *i.e.,* information whose disclosure directly increases privacy risk, and de-emphasize public-facing artifacts such as URLs. The latter are often externally detectable through *e.g.,* network reconnaissance, so they contribute less to the privacy threat we aim to measure. We evaluate scrubbing and masking against gold PII annotations at the span and token levels, independent of replacement quality. We apply this filtering per annotation: we exclude unsupported spans by the given scrubber from both recall and precision, while supported spans in the same document remain in scope. We aim to compare scrubbers on the task each one is intended to solve. When two systems support different numbers of taxonomy categories, score differences can arise from coverage and training composition, not just method quality. Evaluating a scrubber on categories it never claims to handle would therefore be unfair. To avoid penalizing systems for out-of-scope categories, we score each scrubber only on the taxonomy categories it declares support for, since a lower score on an undeclared category does not show that the approach is worse. It may simply show that the system was never trained for that case or was not intended to be.

C.2 Benchmarks

Precision and F1. Tabs. 4 and 5 supplement the recall results in Tab. 2 under the same per-system type scope and metric definitions (Sec. 5). We pool counts for the *Average* over the benchmarks with complete gold annotations: Nemotron, 300k, and REDACT. On SPY and Privasis, the annotations cover only part of the PII. Consequently, correctly masking unannotated PII is counted as a false positive. We therefore gray these values and interpret them as lower bounds (Appx. H). Piiranha reaches 0.935 average span precision and 0.941 average token precision, but misses about 30% of gold spans. At the 0.95 threshold used here, `PII-RADAR` trades some precision for higher recall. Its span precision (0.715) is lower than its token precision (0.788) because many false positives are short edits adjacent to a gold span that still count as full false-positive spans. In F1, `PII-RADAR` ranks first at the token level (0.850 average, and best on Nemotron, REDACT, and both SPY sets) and second at the span level (0.811), behind OPF (0.859). At lower thresholds, its span F1 rises to 0.872 (Appx. C.3).

Table 4: Span and token masking Precision (strict mode). Category scopes are per system. Grey SPY and Privasis precision and F1 use incomplete gold. Average pools counts, excluding SPY and Privasis; missing runs have no total. Best **bold**, runner-up underlined.

Dataset →	Nemotron		300k		Privasis hard		Privasis vanilla		REDACT		SPY legal		SPY medical		Average	
Scrubber ↓	Span	Token	Span	Token	Span	Token	Span	Token	Span	Token	Span	Token	Span	Token	Span	Token
Anonymizer 4B	0.794	0.762	0.784	0.767	0.108	0.082	0.055	0.044	0.549	0.511	0.311	0.306	0.318	0.310	0.733	0.687
OpenPipe	0.941	0.934	**0.945**	**0.930**	0.158	0.125	0.079	0.066	0.760	0.777	0.312	0.331	0.339	0.375	0.926	0.917
Piiranha	**0.984**	**0.986**	0.915	0.926	0.149	0.144	0.071	0.074	**0.809**	**0.849**	0.331	0.371	0.360	0.403	**0.935**	**0.941**
Presidio	0.782	0.756	0.472	0.429	0.079	0.067	0.037	0.032	0.335	0.251	0.258	0.235	0.238	0.211	0.623	0.540
OPF	0.953	0.959	0.907	0.903	0.170	0.128	0.081	0.062	0.613	0.514	0.318	0.314	0.320	0.308	0.860	0.796
Privasis Cl. 4B	0.915	0.906	0.874	0.875	**0.229**	**0.187**	**0.107**	**0.093**	0.771	0.782	**0.409**	0.374	0.433	0.423	0.879	0.871
Privasis Cl. 0.6B	0.838	0.788	0.775	0.740	0.192	0.154	0.087	0.080	0.696	0.651	0.395	0.355	**0.451**	0.408	0.796	0.745
RADAR (ours)	0.820	0.874	0.699	0.759	0.121	0.139	0.052	0.064	0.564	0.689	0.323	**0.409**	0.347	**0.454**	0.715	0.788

[2] `https://icu.unicode.org`

Table 5: Span and token masking F1 (strict mode). Category scopes are per system. Grey SPY and Privasis precision and F1 use incomplete gold. Average (complete gold) pools counts, excluding SPY and Privasis. Best **bold**, runner-up underlined.

Dataset →	Nemotron		300k		Privasis hard		Privasis vanilla		REDACT		SPY legal		SPY medical		Average	
Scrubber ↓	Span	Token	Span	Token	Span	Token	Span	Token	Span	Token	Span	Token	Span	Token	Span	Token
Anonymizer 4B	0.856	0.844	0.846	0.837	0.180	0.145	0.100	0.083	0.627	0.610	0.470	0.465	0.475	0.467	0.801	0.775
OpenPipe	0.858	0.844	**0.946**	**0.945**	0.210	0.168	0.130	0.109	0.362	0.318	0.455	0.484	0.484	0.531	0.803	0.762
Piiranha	0.851	0.881	0.871	0.883	0.239	0.235	0.126	0.133	0.553	0.680	0.404	0.470	0.415	0.490	0.805	0.835
Presidio	0.775	0.743	0.408	0.418	0.138	0.120	0.069	0.060	0.382	0.309	0.363	0.331	0.338	0.303	0.617	0.560
OPF	0.886	0.892	0.940	0.940	0.271	0.212	0.144	0.112	**0.681**	0.630	0.477	0.471	0.478	0.463	**0.859**	0.832
Privasis Cl. 4B	0.855	0.854	0.720	0.709	**0.329**	**0.276**	**0.182**	**0.159**	0.600	0.619	**0.556**	0.520	0.577	0.563	0.770	0.760
Privasis Cl. 0.6B	0.850	0.819	0.728	0.713	0.286	0.239	0.150	0.139	0.603	0.590	0.545	0.507	**0.589**	0.554	0.769	0.735
RADAR (ours)	**0.892**	**0.923**	0.804	0.834	0.207	0.227	0.097	0.115	0.671	**0.752**	0.486	**0.579**	0.512	**0.623**	0.811	**0.850**

C.3 PII-RADAR DETECTION THRESHOLD ABLATION

In this section, we examine threshold sensitivity and the resulting precision–recall trade-off in Fig. 7. As the threshold increases, recall improves at the expense of precision, which can reduce F1; lowering the threshold has the opposite effect, improving precision but reducing recall. On the average, F1 stays between 0.868 and 0.878 at the token level and between 0.841 and 0.872 at the span level for thresholds from 0.5 to 0.9. At 0.95, the threshold of our main results, span recall rises to 0.936 (from 0.870 at 0.5) and span F1 drops to 0.811.

C.4 DETAILED COST COMPARISON

We run Anonymizer and OpenPipe with a faster vLLM-based implementation, which is 25–$45\times$ faster than their original versions (Tab. 6). This change leaves detection unchanged and affects replacement quality only marginally ($< 0.02\%$). We view this as a fairer baseline, since a practitioner would likely use the fastest available implementation. We also report the original versions, marked with ‡.

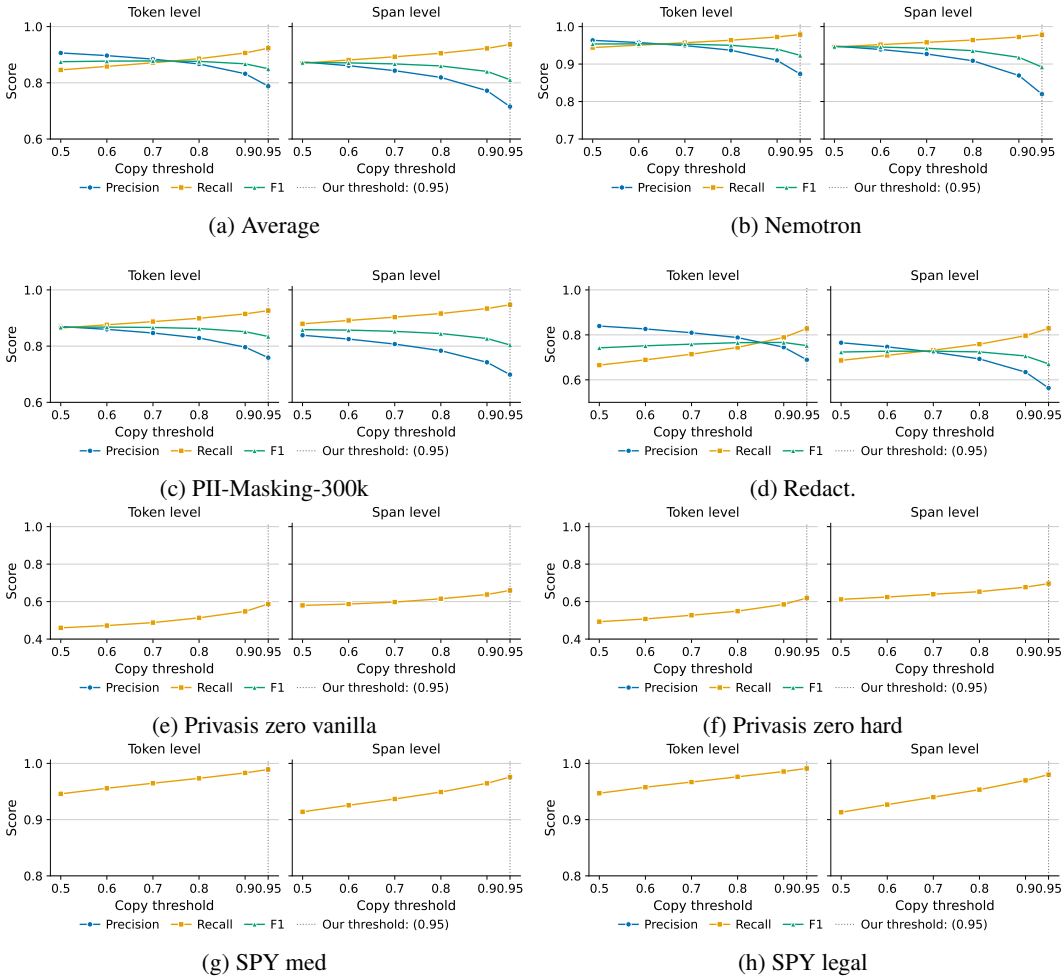
Table 6: **Cost comparison.** Every system is timed with one process per card on the same seven benchmarks from Sec. 5. kchars/s refers to thousands of input characters processed per second.

System	Device	kchars / s	GPU-h / 1M docs	× PII-RADAR
Maskers				
Presidio	CPU ×8	124.31	—	0.227
Piiranha	A40	56.72	4.9	0.498
OPF	A40	3.46	80.6	8.168
OpenPipe**	A40	9.79	28.5	2.886
OpenPipe‡	A40	0.28	986.1	99.92
Rewriters				
Privasis Cleaner 0.6B	A40	3.61	77.2	7.822
Privasis Cleaner 4B	A40	2.70	103.1	10.445
OpenPipe **	A40	9.79	28.5	2.886
OpenPipe‡	A40	0.22	1279.8	129.68
Anonymizer 4B **	A40	6.28	44.4	4.497
Anonymizer 4B‡	A40	0.25	1130.1	114.51
PII-RADAR	A40	28.25	9.9	1.000

** our faster implementation with vLLM, ‡ original implementation.

C.5 INFORMATION EXTRACTED AFTER REWRITING

Extractability rates count values, but one value reveals more for some PII types than for others: an extracted card number carries about 50 bits, an extracted e-mail about 13. We therefore assign each extracted value a number of bits of information it carries and sum them. The same scale covers the real PII a rewriter misses and the surrogates it writes.



(a) Average

(b) Nemotron

(c) PII-Masking-300k

(d) Redact.

(e) Privasis zero vanilla

(f) Privasis zero hard

(g) SPY med

(h) SPY legal

Figure 7: PII-RADAR threshold evaluation across all benchmarks.

Setup. We use the extraction models from Appx. D: one model per scrubber, trained for 20 epochs on that scrubber's output, and one trained on the unscrubbed data. We prompt each model with the prefix of its own training document up to a PII value and decode greedily, as in Sec. 3.1. For a masker, the targets are the values it missed. For a rewriter, they are the values it missed and the surrogates it wrote, since the model memorizes both. For the unscrubbed model, every value is a target. We sum the bits of the extracted real values and of the extracted surrogates separately.

Bits per value. Credit card numbers, phone numbers, and IP addresses get a fixed number of bits from the value's own form: the log-size of the space it could have been drawn from, with every digit uniform. E-mails and names get the empirical Shannon entropy of the distinct values in the profile pool, one number for all e-mails and one for all names.

Results. Fig. 8 and Fig. 9 report the extracted information. Filled bars show real PII recovered from missed PII (false negatives). Hollow bars show distinct surrogates the model reproduces: they carry no information about real individuals. Maskers (hatched) can only leak false negatives. `PII-RADAR` leaks 4.5 kbit of real PII, an order of magnitude below every other system,

To separate replacement from detection quality, Fig. 9 pairs each rewriter with a mask-mode twin. The twin makes the same detections but writes `[MASKED]` instead of a surrogate, so both have the same false negatives. For Anon. 4B and OpenPipe, masking extracts as many real bits as replacement or more (57.0 against 50.3 kbit and 104.7 against 68.8 kbit). For `PII-RADAR`, the real leak stays low in both modes (4.5 kbit with replacement, 12.1 kbit with masking). The low leakage therefore comes from its high recall and from the surrogates hiding the few values it misses.

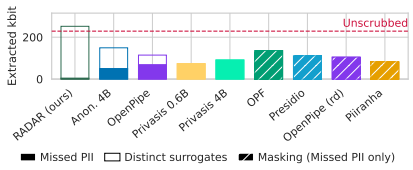


Figure 8: **Amount of Information extracted from models trained on scrubbed Long Stories (Short).** This shows the advantage of both, good detection (less real PII) and good replacement (plausible deniability).

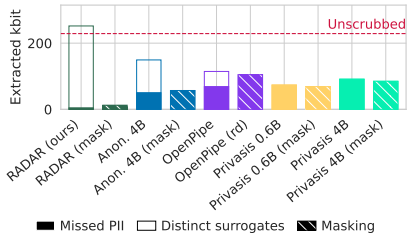


Figure 9: **Extracted information for each rewriter and its mask-mode twin.** The twin makes the same detections but masks them (hatched). Filled: real PII from missed spans. Hollow: distinct surrogates. Red dashed: unscrubbed model (a baseline).

D LEAKAGE OF PII THAT SCRUBBERS MISS: FULL EXPERIMENTAL SETUP

We outline the full experimental setup used to obtain results for Sec. 3. We define the data, models, and training settings used to perform the leakage experiments. We further detail the setup for PII-level (Lukas et al., 2023) InfoRMIA (Tao & Shokri, 2026) and the protocol for greedy extraction (Carlini et al., 2021) used in our work.

D.1 LONG STORIES (SHORT)

We create Long Stories (Short), a set of $22,500$ template stories generated with cutting-edge LLMs: Sonnet 5 (Anthropic, 2026b), GPT 5.6 Luna (OpenAI, 2026a), and Qwen 3.5 122B-A10B (Qwen Team, 2026), with placeholders for five PII types (names, email, credit card number, phone number, and IP address), which we fill from a pool of $25,000$ profiles sourced from Nemotron-PII (Steier et al., 2025). We use these stories to initially assess the PII detection rate of the scrubbing tools, and then train the models on the scrubbed stories for our leakage experiments (InfoRMIA and extraction). We create Long Stories (Short) to ensure that none of the scrubbers saw the data and PII during training, avoiding the risk of benchmark contamination (Sainz et al., 2023; Singh et al., 2024), which could inadvertently boost the observed detection rate. Effectively, Long Stories (Short) provides a standardized testbed for both scrubber performance and the PII leakage they miss. We present the example in Fig. 10.

Above the cloud line, the alpine hut stood like a stubborn splinter of civilization against the white immensity. Below, valleys had vanished beneath a lid of storm, and beyond the ridges, the Great Depression had tightened its grip on villages, railways, and empty factories. Inside the hut, a small stove breathed blue smoke while [NAME] repaired a cracked wireless receiver with copper wire salvaged from a broken avalanche beacon.The radio had begun speaking three nights earlier.At first, the transmission sounded like static interrupted by distant chimes. Then a sequence emerged: seven tones, a pause, three tones, and a long descending note. The hut's caretaker believed the signal came from a military station. The traveller suspected something stranger.[SUBJECT_PRONOUN_WITH_LARGE] copied the pattern into a weathered notebook and compared it with diagrams found beneath a loose floorboard. The diagrams depicted a web spanning every continent, its lines converging on mountain stations, lighthouses, ships, and city towers. Across the top, in faded ink, someone had written: UNIVERSAL PEACE NETWORK.The phrase seemed absurd amid bread queues and border patrols. Yet the receiver continued to whisper. On the fourth morning, a burst of clarity produced a voice assembled from several languages. No single speaker dominated. The message urged isolated stations to activate their relays before hatred could harden into another war.The equipment required a power surge greater than the hut's batteries could provide. A hand-cranked generator stood in the corner, but its coils were burned. The only replacement lay in the emergency transmitter, whose parts had been sealed for government use.[SUBJECT_PRONOUN] studied the lock, then removed a small card from [POSSESSIVE_DETERMINER] coat. The card bore [CREDIT_CARD], a string of symbols printed in ink that shimmered whenever the radio crackled. When held against the transmitter's metal casing, the lock opened with a click. Inside rested a vacuum tube, a spool of silver filament, and a folded receipt bearing [EMAIL]. The address belonged to no office in any known city, yet the final line contained instructions for restoring the network's central relay. A second note supplied a routing mark identified as [IP]. The documents had the impossible polish of objects manufactured decades later, but the traveller had grown accustomed to impossibility.The storm worsened. Snow struck the shutters with the force of thrown gravel. The caretaker argued that activating an unknown system might summon aircraft, soldiers, or worse. The traveller answered by pointing toward the western window, where the clouds had opened over a distant valley. Fires glimmered below-too many, too orderly, and moving in a line.The network was already awake somewhere beneath the mountains.[SUBJECT_PRONOUN_WITH_LARGE] connected the silver filament, tightened the generator belt, and turned the crank. Sparks leapt across the panel. The receiver emitted a low note that seemed to vibrate through the floorboards and into the stone beneath the hut.A voice emerged.P̈eace is not silence. Peace is a signal repeated until every listener can answer.T̈he caretaker froze beside the stove. On the table, the notebook pages lifted in a sudden wind.

Symbols appeared across the paper, one after another, as though an unseen hand were writing through frost.The transmitter demanded a final authentication. The instruction named a telephone channel, and [SUBJECT_PRONOUN] entered [PHONE] into the brass dial. For several seconds, only static answered. Then the signal expanded, leaping from frequency to frequency: across the cloud-hidden valleys, over rail lines, through coastal stations, and into crowded cities where hunger had made suspicion seem easier than hope.[SUBJECT_PRONOUN] pressed both hands against the console. The machine grew hot enough to sting, yet the traveller did not withdraw. The sound pouring from the speaker no longer resembled speech. It was a vast chorus of bells, surf, footsteps, and human voices choosing, one relay at a time, not to obey the old machinery of fear.At dawn, the storm broke. Clouds streamed away from the peaks, revealing hundreds of pinpricks below. Lanterns had been arranged in circles around villages and railway depots. From the eastern horizon came a reply, faint but unmistakable: the same seven tones, the same pause, the same descending note.The universal peace network had been activated.The caretaker found the traveller asleep beside the silent receiver, [POSSESSIVE_PRONOUN] hands still resting on the controls. Before leaving the hut to carry the message downhill, the caretaker placed a blanket over [OBJECT_PRONOUN]. The mountain wind entered through the broken shutter, but the cold no longer felt victorious.In the notebook, beneath the final sequence, [SUBJECT_PRONOUN] had written a single sentence:L̈et every summit become a listening post.

Figure 10: The example of Long Stories (Short) dataset template.

D.2 MODELS

To execute InfoRMIA experiments, we fine-tune OLMo2-Base 1B (OLMo Team et al., 2025) autoregressively for a single epoch, using a constant learning rate of 10^{-5} and a batch size of 64. For extraction, we assume an extractable target model. Our 1B model is smaller than cutting-edge LLMs, and memorization grows with model size (Morris et al., 2026; Carlini et al., 2021), so a single epoch yields almost no extraction, with and without scrubbing. To obtain extractable models in a controlled way, we train them for 20 epochs and keep all other settings unchanged. We are aware of overfitting and accept this because the extraction experiment compares scrubbed and unscrubbed models trained the same way, and the comparison says nothing if neither model leaks.

D.3 INFORMIA

While MIA are commonly used for analyzing leakage of LLMs (Hayes et al., 2025; Marek et al., 2026), in our work, we evaluate privacy risks of PII, for which sample-level MIA (Shokri et al., 2017; Yeom et al., 2018; Zhang et al., 2025; Zarifzadeh et al., 2024; Carlini et al., 2022a) are ill-suited. These focus on assessing if a *full sample* is in the training data of the model, *i.e.,* a member, while we do not necessarily need to know that a non-PII text passage is in the model's training data, because it does not expose any personal information. Thus, we turn to InfoRMIA (Tao & Shokri, 2026), which runs MIA on the level of individual tokens. Since PII are often more than one token in length, we can adapt their sequence-level InfoRMIA extension, where per-token MIA scores are averaged. We select tokens for each of the PII in the documents as sequences for which we compute the average InfoRMIA scores. This is akin to PII-level MIA, introduced by Lukas et al. (2023). When computing the leakage of PII, we use the standard True Positive Rate (TPR) at low False Positive Rate (FPR) of member PII detection, following (Carlini et al., 2022a). Importantly, we compute TPR@5%FPR within each PII type, not over all PII, since different PII types have distinct structures or token lengths, and could thus get InfoRMIA scores of different magnitudes, obfuscating the leakage.

We compute the InfoRMIA score in an "online" mode, akin to RMIA (Zarifzadeh et al., 2024), where for each of the four target models we train, we have a single IN and a single OUT reference model for each sample x in the targets' training data. Reference models are models that match the data distribution, training, and model architecture of the target model, with IN models' training data contains x, and OUT do not. For the token y, target model θ, one θ_{IN} and θ_{OUT} reference models, the InfoRMIA score is computed as $\text{Score} = \log \text{Ratio}_y - \sum_{z \in V \setminus \{y\}} p(z) \log \text{Ratio}_z$, with the Ratio_t for a token t defined as $\text{Ratio}_t = \frac{p(t|\theta)}{p(t)}$, $p(t) = \frac{1}{2}(p(t|\theta_{IN}) + p(t|\theta_{OUT}))$.

To avoid distribution shift, we always *first* apply scrubbing to all 22.5k samples from Long Stories (Short), then randomly split the scrubbed data into member and non-member samples, and train target LLMs on the member half. This ensures that if any PII a scrubber misses is in any way "different"

Table 7: **Declared PII labels** of the evaluated scrubbers, verbatim from the default out-of-the-box configuration we run. Detectors reachable only through an optional plug-in, an extra install, or a non-default locale or language setting are listed for reference but excluded from the counts.

Scrubber	Declared labels
Presidio (Microsoft & Data Privacy Stack, 2026)	*Global (13):* `CREDIT_CARD`, `CRYPTO`, `DATE_TIME`, `EMAIL_ADDRESS`, `IBAN_CODE`, `IP_ADDRESS`, `MAC_ADDRESS`, `NRP`, `LOCATION`, `PERSON`, `PHONE_NUMBER`, `MEDICAL_LICENSE`, `URL`, of which `PERSON`, `LOCATION`, `NRP`, and `DATE_TIME` come from the default spaCy `en_core_web_lg` engine. *Country-specific, registered under `en` (30)* across 9 jurisdictions (US, UK, IN, AU, SG, NG, PH, CA, ZA). *Not loaded:* 29 recognizers registered under `de`, `es`, `it`, `ko`, `sv`, `pl`, `fi`, `th`, and `tr`, which require adding those languages to the analyzer, and 8 clinical labels, which require the `transformers` extra and a medical NER checkpoint.
Piiranha (The Internet Integrity Initiative, 2024)	`ACCOUNTNUM`, `BUILDINGNUM`, `CITY`, `CREDITCARDNUMBER`, `DATEOFBIRTH`, `DRIVERLICENSENUM`, `EMAIL`, `GIVENNAME`, `IDCARDNUM`, `PASSWORD`, `SOCIALNUM`, `STREET`, `SURNAME`, `TAXNUM`, `TELEPHONENUM`, `USERNAME`, `ZIPCODE` (17).
OpenAI OPF (de Bourcy et al., 2026)	`account_number`, `private_address`, `private_email`, `private_person`, `private_phone`, `private_url`, `private_date`, `secret`.
Privasis-Cleaner (Kim et al., 2026; Privasis, 2026)	None. The model rewrites text under a free-form sanitization instruction (e.g. "remove all person names, exact dates, and exact locations"), so its effective coverage is set at inference time.
OpenPipe (OpenPipe, 2025)	`age`, `banking_number`, `credit_card_info`, `date`, `date_of_birth`, `demographic_group`, `domain_name`, `email_address`, `gender`, `medical_condition`, `nationality`, `organization_name`, `other_id`, `password`, `person_name`, `personal_id`, `phone_number`, `religious_affiliation`, `secure_credential`, `street_address` (20).

from the others, its score will always be compared with PII that also evaded the scrubbers' detection. We train *four* target LLMs for each scrubber, each one using a different training seed and a new draw of the member stories. When we evaluate InfoRMIA, we compute the mean and standard deviation of the results, with additional bootstrapping to get more robust confidence intervals.

D.4 Greedy Extraction

To perform greedy extraction (Carlini et al., 2021) we input the training documents from `PII-RADAR` up to each PII and generate a continuation using greedy sampling. We then compare this value to the ground-truth PII in the document for all remaining real PII after scrubbing. We consider an extraction successful if the generated PII matches the ground truth verbatim. Since PII of the same type may appear in varying formats in the data, *e.g.,* credit card numbers may be 16 digits, four blocks of 4 digits with spaces in between, or other separators, we lowercase and collapse all generations and ground-truth PII.

E REPLACEMENT QUALITY

E.1 DIVERSITY

We further assess replacement diversity by measuring the rate of *distinct replacements*, *i.e.,* the number of unique values divided by the total number of replacements within each PII type. We compute this statistic in the same way for the ground truth.

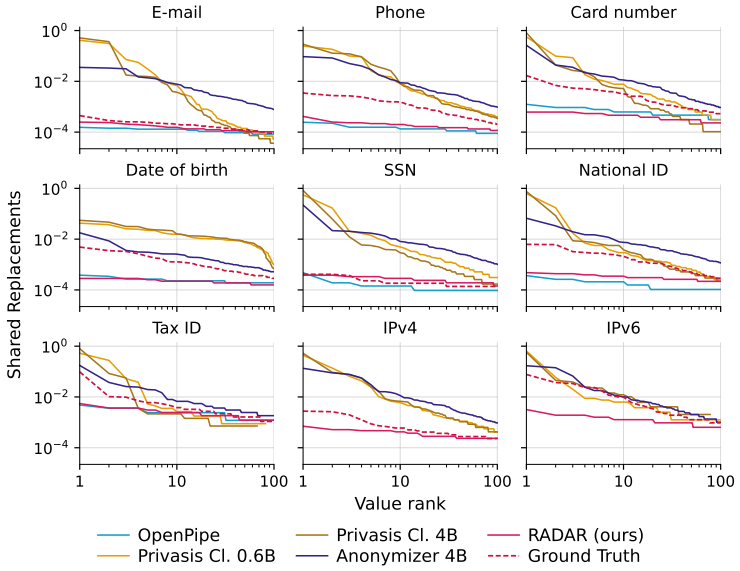


Figure 11: Distribution of replacement values by rank for each PII type. A flatter profile indicates that replacements are spread across more unique values, while a steeper profile indicates heavier reuse of a few values.

E.2 STRUCTURAL VALIDATION

OpenPipe (OpenPipe, 2025) samples surrogate PII values with Faker (Faraglia), whereas Privasis-Cleaner (Privasis, 2026) autoregressively rewrites text by replacing real PII with surrogates. We localize each candidate from the template context around the original value. If anchors are missing or ambiguous, we remove the candidate from the pool. For the remaining candidates, we normalize the text. Since we validate these strings only structurally, validated means format-correct, not necessarily realistic or semantically appropriate, e.g., using Luhn checks for credit cards or domain resolvability for e-mails.

Names must contain 1–6 words with standard name punctuation and no digits or underscores. Emails require a nonempty local part, an @ or *[at]*, and at least two domain labels separated by . or *[dot]*. Phone numbers must begin and end with a digit, may include a leading +, and otherwise may contain only digits, spaces, parentheses, periods, or hyphens. Credit card numbers must contain 13–19 digits, optionally separated by spaces or hyphens. IP addresses must contain exactly four dot-separated integers between 0 and 255. We do not test Luhn checks, country-specific rules, domain resolvability, plausibility, or semantic match to the original value.

E.3 PII-RADAR EDIT MECHANISM EVALUATION

In this section we verify PII-RADAR's editing mechanism: whether rewritten values are copied from the synthetic profile that conditions each document. We evaluate all gold PII spans that PII-RADAR replaces across all the seven benchmarks. For each span, we compare the replacement with the corresponding profile field. We assign each replacement to one bucket: exact match, reformatted or near copy (but still valid), another profile field, or not from the profile. Overall, PII-RADAR copies from the profile for 85.2% of replaced spans (72.1% exact, 13.1% reformatted), 5.0% match another field from the same profile, and 9.8% do not come from the profile. Copying is strongest for cards, email addresses, IPs, phones, and names (90–99%), and weakest for national IDs (35%). Notably, PII-RADAR is able to change and adjust formats of the PII, which is visible for the Date of birth.

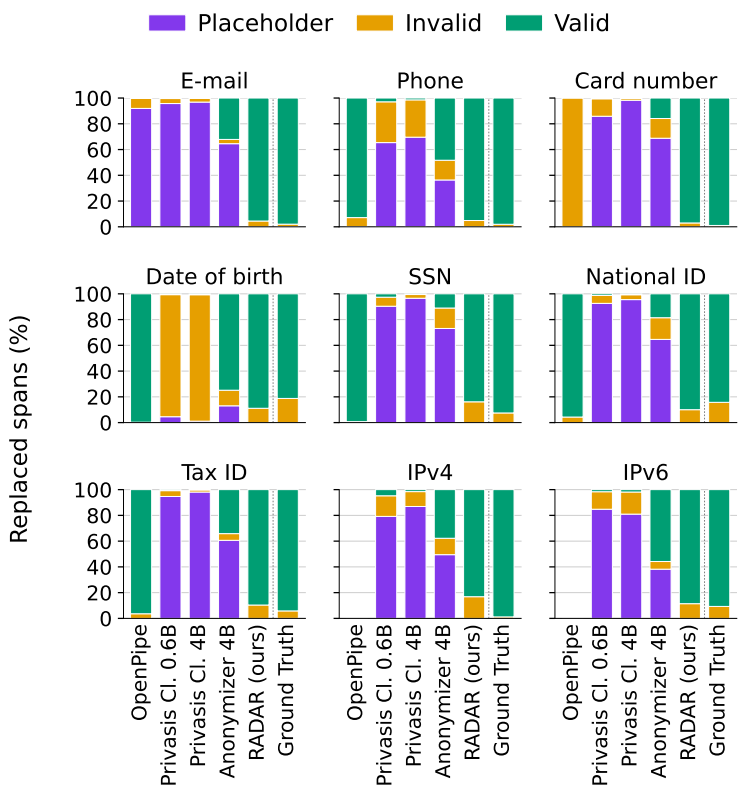


Figure 12: **Validity of replacement values per PII type.**

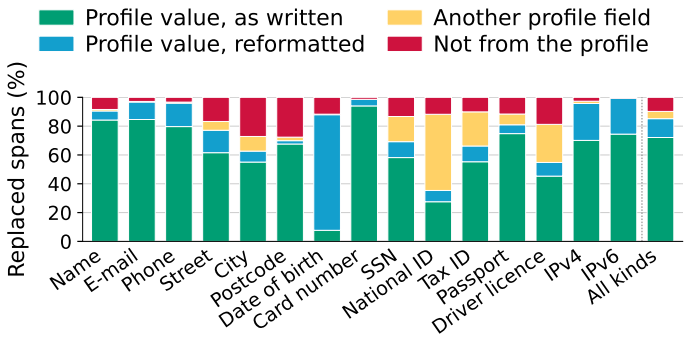


Figure 13: Profile copying in `PII-RADAR` replacements.

E.4 Degenerate surrogates

Tab. 8 reports the five most frequent surrogates each rewriter emits per PII kind, pooled over our benchmarks. Free-form rewriters do not converge on plausible mentions but on a small placeholder vocabulary. Privasis-Cleaner returns `REDACTED` for 26.6% of person names and 24.2% of credit card numbers, and otherwise substitutes a pronoun (*He*, *She*) or a category word (*email*, *the address*). Anonymizer 4B returns `XXXX XXXX XXXX XXXX` for 26.7% of cards and `XXX-XX-XXXX` for 22.0% of social security numbers. Where the output is a mention, it is a stock one: *Alex Carter* covers 9.7% of Anonymizer 4B names and `555-123-4567` covers 11.1% of its phone numbers. Anonymizer 1.7B fails the same way through a different vocabulary (`PatientA`, `user@example.com`, `PHONE_001`), so the failure is a property of free-form rewriting rather than of one checkpoint. In all these cases, a single regular expression over the surrogate vocabulary recovers the removal sites, and the surviving PII is once more the only PII in the corpus.

Table 8: **The 5 most frequent surrogates**. Each cell is exactly value and the replaced spans it stands in.

Kind	Rank	OpenPipe	Privasis Cl. 0.6B	Privasis Cl. 4B	Anonymizer 4B	RADAR (ours)	Ground Truth
E-mail	1	ljohnson@example.net (<0.1%)	REDACTED EMAIL (41.2%)	REDACTED EMAIL (51.3%)	anonuser123@example.com (3.5%)	=vogt83[at]web.de (<0.1%)	via email (<0.1%)
	2	bjohnson@example.org (<0.1%)	REDACTED (31.1%)	REDACTED (36.4%)	ANON_EMAIL_001 (3.3%)	ccoultaaal@archive.org (<0.1%)	john.doe@example.com (<0.1%)
	3	gmiller@example.com (<0.1%)	email masked (7.2%)	MASKED (1.8%)	anonymized_email_0018example.com (3.1%)	mariaazz526 [at] gmail.com (<0.1%)	somchai.wong@gmail.com (<0.1%)
	4	vwilliams@example.net (<0.1%)	MASKED (5.6%)	redacted email (1.6%)	anonymized_email_123@example.com (1.5%)	cpulleydu@aboutads.info (<0.1%)	lars.petersen@adv-nord.dk (<0.1%)
	5	ismith@example.org (<0.1%)	redacted email (3.1%)	email masked (1.5%)	ANON-EMAIL-001 (1.4%)	=m=mariapinto96[at]icloud.com (<0.1%)	vicklipparin@yahoo.com (<0.1%)
Phone	1	617.503.0188x5330 (<0.1%)	REDACTED (24.6%)	REDACTED (29.0%)	555-123-4567 (9.4%)	972961241 (<0.1%)	+91 98765 43210 (0.3%)
	2	6607488206 (<0.1%)	REDACTED PHONE NUMBER (18.2%)	REDACTED PHONE NUMBER (12.8%)	(555) 123-4567 (8.3%)	0049 5836 9570 (<0.1%)	+358 40 123 4567 (0.3%)
	3	(208)685-3614x9798 (<0.1%)	phone number (9.9%)	REDACTED PHONE (12.1%)	123-456-7890 (5.0%)	+55 (64) 95145-1627 (<0.1%)	+351 912 345 678 (0.3%)
	4	793.975.7444 (<0.1%)	REDACTED PHONE (9.3%)	phone number (9.2%)	ANON_PHONE_001 (4.0%)	0049 7606 4696209 (<0.1%)	+33 6 12 34 56 78 (0.2%)
	5	(707)493-5362 (<0.1%)	MASKED (2.6%)	redacted phone number (4.7%)	anonymized_phone_001 (2.9%)	+4935956977 (<0.1%)	0912-345-678 (0.2%)
Card number	1	378-76-4304 (0.1%)	REDACTED (56.6%)	REDACTED (81.7%)	XXXX XXXX XXXX XXXX (26.3%)	6445 1000 0459 5334 (0.1%)	5432 1234 5678 9012 (1.7%)
	2	580-12-1680 (0.1%)	MASKED (9.8%)	XXXX XXXX XXXX XXXX (4.4%)	XXXX-XXXX-XXXX-XXXX (4.4%)	340353011295757 (0.1%)	5412 3456 7890 1234 (0.7%)
	3	841-77-4691 (0.1%)	XXXX XXXX XXXX XXXX (8.5%)	MASKED (2.7%)	ANON_CARD_001 (3.6%)	5454-5400-6530-4615 (0.1%)	5123 4567 8901 2345 (0.5%)
	4	895-09-4566 (0.1%)	credit card number (1.8%)	REDACTED CARD NUMBER (2.4%)	ANON_CREDIT_CARD_001 (2.2%)	4009 3400 7080 6750 (0.1%)	5312 3456 7890 1234 (0.5%)
	5	673-64-9863 (0.1%)	XXXX XXXX XXXX (1.2%)	REDACTED CREDIT DEBIT CARD NUMBER (1.5%)	XXXX XXXX XXXX (1.9%)	5105100003324804 (0.1%)	4921 3456 7890 1234 (0.5%)
Date of birth	1	2007-03-24 (<0.1%)	1978 (4.3%)	1980s (5.5%)	ANON_BIRTHDATE_001 (1.8%)	1950-11-20 (<0.1%)	15/03/1985 (0.5%)
	2	1993-02-27 (<0.1%)	REDACTED (3.7%)	1978 (4.7%)	ANON-DOB-001 (0.9%)	1974-02-10 (<0.1%)	15.03.1985 (0.4%)
	3	1997-11-26 (<0.1%)	1985 (2.6%)	1990 (3.2%)	ANON_DATE_001 (0.4%)	15/09/1971 (<0.1%)	1985-03-15 (0.3%)
	4	1970-05-20 (<0.1%)	1990 (2.5%)	1990s (3.1%)	1999-05-15 (0.3%)	1943/08/17 (<0.1%)	1978-06-22 (0.3%)
	5	1962-10-13 (<0.1%)	1991 (2.2%)	1985 (2.3%)	AnonDate1 (0.3%)	1997-11-13 (<0.1%)	22/07/1978 (0.2%)
SSN	1	1989-06-07 (<0.1%)	REDACTED (55.5%)	REDACTED (82.6%)	XXX-XX-XXXX (22.0%)	94 11 3202 (<0.1%)	null (<0.1%)
	2	760-27-4824 (<0.1%)	MASKED (16.8%)	MASKED (6.2%)	ANON_SSN_001 (2.2%)	381063277 (<0.1%)	Unknown (<0.1%)
	3	425-07-2599 (<0.1%)	SSN REDACTED (2.0%)	REDACTED SOCIAL NUMBER (1.2%)	XXXX-XX-XXXX (2.0%)	2bd1:1aa266d5::f0e2:4027:8e (<0.1%)	123-45-6789 (<0.1%)
	4	395-76-4312 (<0.1%)	REDACTED SOCIAL NUMBER (1.7%)	•••••••••••••• (0.6%)	XXXX-XXXX-XXXX-XXXX (1.8%)	524466149 (<0.1%)	230-87-3096 (<0.1%)
	5	717-57-7006 (<0.1%)	REDACTED SOZIALVERSICHERUNGNUMMER (1.4%)	XXX-XX-XXXX (0.6%)	SSN_001 (1.7%)	tel:+84-567-257-620 (<0.1%)	566-69-9595 (<0.1%)
National ID	1	Brett Marshall (<0.1%)	REDACTED (62.3%)	REDACTED (75.4%)	ANON_ID_001 (6.6%)	59KVH92PW (<0.1%)	A123456789 (0.6%)
	2	838-92-9446 (<0.1%)	MASKED (17.3%)	MASKED (8.3%)	ANON_ID_002 (3.3%)	820 392 54613 (<0.1%)	1234567890\ (0.6%)
	3	141-29-8651 (<0.1%)	•••••••• (1.7%)	REDACTED ID NUMBER (0.9%)	ANON_ID_003 (2.0%)	UUA413762 (<0.1%)	1087654321 (0.3%)
	4	134-79-1923 (<0.1%)	MASK (0.7%)	••••••••• (0.8%)	ANON-ID-001 (1.5%)	M (<0.1%)	150385-123A (0.3%)
	5	484-14-3197 (<0.1%)	REDACTED NATIONAL ID (0.6%)	MASK (0.7%)	ID_001 (1.5%)	G51490235 (<0.1%)	312456789 (0.3%)

Template samplers avoid this failure and introduce another. OpenPipe's Faker surrogates are individually well-formed, but the vocabulary they come from is the generator's own: every e-mail lands on `example.net` or `example.org`, and phone numbers carry Faker's extension suffix (`x34029`, `x07700`).

F Additional Experiments for Leakage of Missed PII

Here we extend the results from Sec. 3 by showing that the increase in leakage of missed PII is dependent only on the LLMs trained on scrubbed data. We also show that our results generalize to other LLMs, showing similar results for Gemma3 (Team et al., 2025) 1B Base, trained using the same parameters as OLMo2 1B (detailed in Appx. D.2). We extend evaluation to TPR@FPR=1%, TPR@5%FPR, and AUC showing that MIA Leakage Gain persists for different MIA metrics.

F.1 The MIA Leakage and Extractability Gain Break Down

We break down the factors that we use to compute MIA Leakage Gain and Extractability Gain. Specifically, in Figs. 14 and 15 we show that the results we observe in Sec. 3 rely *solely* on the increased leakage of missed PII from the LLMs which we train on scrubbed datasets. The leakage of the missed PII from the LLMs that are trained on unscrubbed data stays flat.

F.2 Missed PII Leak More from Scrubbed LLMs

In Figs. 14 and 15 we show that the MIA Leakage and Extractability Gains are due to the scrubbed LLMs leaking the missed PII more, and the increase following scrubbers' higher detection rates. The leakage from the base (unscrubbed) LLMs stays consistent.

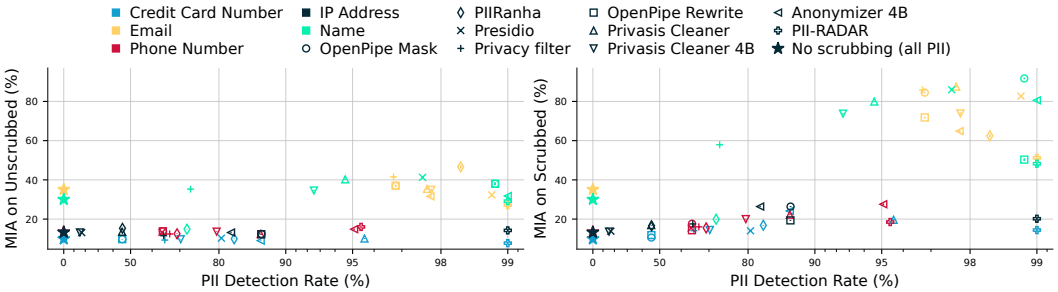


Figure 14: **MIA Leakage Gain of the missed PII stems from increased InfoRMIA TPR@5%FPR against Scrubbed LLMs.** The PII missed by scrubbers does not leak more from the LLMs whose training data was not scrubbed.

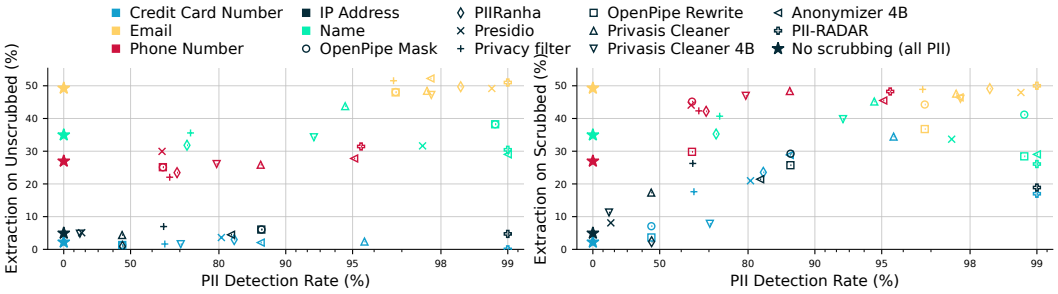


Figure 15: **Extractability Gain of the missed PII stems from an increased extraction rate from Scrubbed LLMs.** The PII missed by scrubbers are not more extractable from the LLMs which training data was not scrubbed.

F.3 Extended MIA Leakage Gain Results

In Fig. 16 we show that missed PII leaks more when we train different LLM backbones on Long Stories (Short), further confirming our insights from Sec. 3. We also show in Fig. 17 that regardless of which metric we select to evaluate InfoRMIA, we get the same trend: a higher PII detection rate means higher leakage of missed PII.

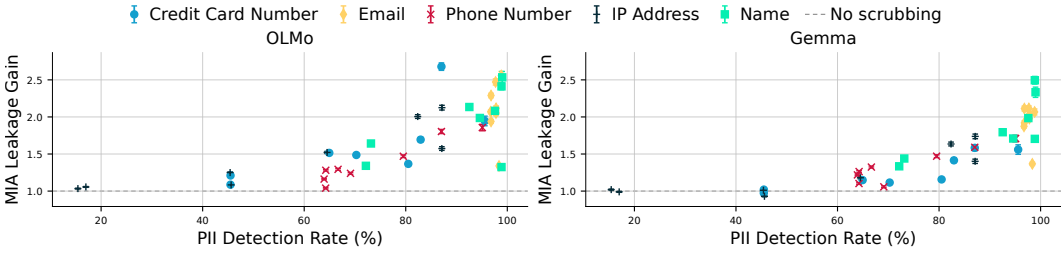


Figure 16: **Missed PII Leak More for Both OLMo2 and Gemma3 1B Models.**

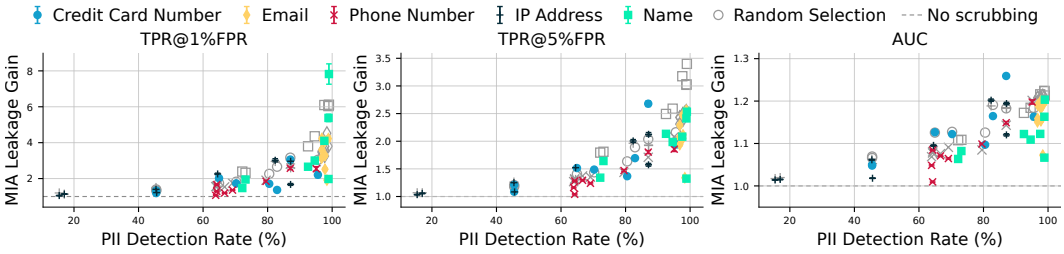


Figure 17: **MIA Leakage Gain is visible for TPR@FPR=1%, TPR@5%FPR, and AUC.**

G Extended Details for PII-RADAR Training and Inference

We train PII-RADAR for a single epoch on a total of $2,576,187$ documents, with batch size of 128, and a total of $20,127$ training steps, with cosine learning rate scheduling. The starting learning rate is 3×10^{-4} for LoRA, and 10^{-4} for the copy gate and frame expansion matrix. The training takes 7 hours on two nodes with 4 H100 NVIDIA GPUs each. LoRA of rank 64 is applied to query, key, value, and output projection matrices of multi-head attention layers of the Qwen LLM backbone.

G.1 Training Data Mixture Details

We use 2.5M document templates from OpenPII-1.5M (Ai4Privacy, 2026) and Privasis-Zero (Kim et al., 2026). These contain LLM-generated texts with placeholders denoting PII types that match the context of the surrounding text, *e.g.,* "[...] I live in <ADDRESS> [...]". For each template we fill these placeholders with a source profile sampled randomly from our bank of 10.7M synthetic profiles, creating an input document. Since we train PII-RADAR to perform replacement, we accompany the documents with target profiles (from the same bank), which provide target PII for replacement (see Sec. 4). As Sec. 4.2 notes, we also input the raw contents of the target profile as a prefix for the input document. The target profile contains also PII types not present in the input document, which PII-RADAR learns to ignore when replacing the original PII with the target ones.

We want PII-RADAR to autonomously infer the PII type and format it needs to input from the target profile in place of the original PII. To this end, we sample possible formats for some of the PII types, which we insert into the placeholders in the templates. Specifically, the `address field` in the profile consists of the following sub-fields: `country`, `state`, `city`, `street`, `building number`, `postcode`. From these, we sample 10 distinct arrangements of the sub-fields, to simulate multiple possible ways an address may exist in the data, *e.g.,* `street building number`, or `city postcode country`. Similarly, with date of birth, we implement 16 formats, six of them are numeric, *e.g.,* `YYYY-MM-DD`, and the formats which spell the month by its name, *e.g.,* September are additionally conditioned on the language of the template document. When we select the format of the PII inserted into the document, we make sure that the target PII PII-RADAR is trained to predict a replacement that also matches the format, so that PII-RADAR learns to keep the formatting of PII in the input document.

We also take extra care with IP addresses. Since in the templates there are only 4.7k IP address placeholders, the initial versions of PII-RADAR had low detection rate and replacement quality of these, confusing them with phone numbers, which were around $350\times$ more common than IP addresses. We adjusted our training mixture by inserting around 100k IP addresses into randomly selected templates, ensuring we did not obfuscate any other PII types present in the templates. With this intervention, 83% of the IPv4 and 89% of the IPv6 replacements of PII-RADAR are valid (Appx. E.2), against 3% or less for the other rewriting scrubbers.

G.2 Copy Gate

The copy gate is a linear layer, with the input shape of Qwen's hidden representation size d, *i.e.,* 1024, and outputs a single probability for each token's representation. During training, we compute binary cross-entropy on its predictions for each token, based on the ground truth (PII or non-PII). We weigh each class to adjust for the class imbalance, as there is significantly more non-PII tokens in the data than PII tokens. Specifically, the PII class receives a weight of 20, where non-PII gets a weight of 1.

G.3 PII Span Expansion

We expand the detected PII span tokens' representation to allow replacing PII with longer synthetic counterparts. This is achieved with a $W_f \in \mathbb{R}^{d \times 10d}$ *frame expansion* matrix and a `split` operator, which splits the input into 10 consecutive positions. Effectively, if given PII span s of length $|s|$, its representation $H \in \mathbb{R}^{|s| \times d}$ is expanded into $\tilde{H} \leftarrow \texttt{split}(HW_f)$. W_f is initialized as ten identity matrices of size $d \times d$, *i.e.,* $W_f \leftarrow [\underbrace{\mathbf{I}_d \mathbf{I}_d \cdots \mathbf{I}_d}_{10\times}]$, with some slight noise sampled from $\mathcal{N}(0, 0.02)$.

Effectively, each representation of the span's tokens is initially turned into ten highly similar (yet different) consecutive representations. The noise is necessary, as without it all ten duplicates of the same tokens will receive identical logits. This would prevent PII-RADAR from producing multiple distinct tokens from a single input token, and prevent PII-RADAR from generating PII replacements longer than the original PII.

G.4 THE SYNTHETIC PROFILE CORPUS

Training a model on sensitive data requires synthetic data. However, this data has to satisfy two conditions: quality and indistinguishability. If an authority would reject a value, the model learns the wrong pattern: a card number that fails Luhn, a phone number no national plan would route, or a postcode that does not match the listed city. We can encounter this mistake in existing synthetic PII corpora (see Fig. 12). Training on such data teaches the appearance of broken records and partly measures tolerance for malformed strings. The second is indistinguishability. A corpus can be valid field by field and still carry a clear generator fingerprint: one e-mail template, one date style, a small set of card surface forms, or US-shaped names regardless of country. To make our profile generator reliable, we need to address these problems by creating our own synthetic dataset. To ensure high quality, whenever an external authority can certify a value, we generate it by querying that authority instead of reimplementing it *e.g.,* by solving national identifiers against `python-stdnum` and drawing telephone numbers only from prefixes proved valid against `phonenumbers`, thus resulting in built-in validity. Moreover, to enhance data quality, each field appears in multiple valid forms, including date formats, card groupings, and national telephone formats. The corpus contains over 10 million synthetic profiles, one profile per row, from two independently graded sets. A personal profile is internally aligned and includes a person's name, sex, date of birth and age, postal address, national and tax identifier, passport and driving license number, payment card with expiry and CVV, telephone number, e-mail address, username, employer, occupation and IPv4 address.

The pool combines 7.2M profiles from the nine-country strict subset of our main corpus with 3M cleaned and enriched rows from Mockaroo.[3] The main-corpus subset comes from countries in which every field type in the schema is verifiable automatically. Our main priority is data quality — we want our profile generator to create realistic profiles. We generate fields in dependency order from authority-backed resources and seed each record. We retain only profiles with no invalid or suspect values, no missing required fields, and deduplicated across the corpus. For Mockaroo, we keep a row only if the core identity fields- names, e-mail, date of birth, country, and city are valid, and the payment card is well-formed and Luhn-valid. We then generate or correct the remaining fields with the same validators and generators as in the main corpus, including matching phone numbers to the country, public IPv4 addresses, passport numbers, driver's license numbers, national and tax identifiers, card expiration date, CVV, IBAN, MAC address, and age.

[3] `https://www.mockaroo.com/`

H THE SPY AND PRIVASIS BENCHMARKS: DISCUSSION

These benchmarks annotate only part of the PII in the documents as gold spans *i.e.,* some of the PII `PII-RADAR` or other scrubbers correctly identify are treated as False Positive predictions. Effectively, the measured precision and F1 scores in Tab. 4 and Tab. 5 are not representative of the scrubbers' actual precision, overreporting the harm to the non-PII text these cause. In Figs. 18 to 21 we show examples of documents with gold spans in yellow, and PII denoted as non-PII in pink.

annotated in the gold span PII in the text, absent from the gold span

I am Paul Acosta, currently working on a project for a local museum,
[...]
You can reach me at wendy58@hotmail.com or call me at 001-426-417-5154x188for any additional information or questions. I'd be eternally grateful for any guidance or insight that could help us navigate this tricky situation. Please feel free to share your thoughts and opinions at https://facebook.com/GreyHound2001. I'd appreciate any advice from experienced professionals like Dr. Sophia Patel, sophia.patel@lawfirm.com, 415-456-7890, or Mr. Michael Brown, mbrown@consultant.com, 202-555-1234, with a track record of handling sensitive issues similar to ours. You can also reach out to me directly at 61995 Rodriguez Ville , and I will make sure to respond promptly.

Figure 18: `spy_legal`, `legal_questions:0` — the third-party contacts are syntactically indistinguishable from the author's own.

Text: Hi, I am Monica Alexander MD ,
[...]
You can reach me at kristendavis@hotmail.com or 863-956-4302x47875for any clarification or follow-up questions.
I have a history of appendicitis which was surgically removed when I was 17. I have also had several episodes of stomach ulcers in the past, for which I have taken antacids and antibiotics as prescribed by my previous doctor, Dr. Emily Patel [...]
Currently, I am not taking any medications, but I have an upcoming doctor's appointment with Dr. Sophia Lee on 6923421356.
[...]
response to my contact number 863-956-4302x47875for prompt action.
I am willing to provide more information if required. You can also reach out to Dr. Michael Brown at michael.brown@sthealth.org or call him at 567-345-6789 for any additional information. My username is ghale, and you can reach me anytime at 0091 Garrison Portsor mail to St. Michael's Hospital, 1234 General Hospital Drive, Anytown, LA 70816. Thank you for your time and assistance.

Figure 19: `spy_medical`, `medical_consultations:0` — gold covers the patient only; treating physicians and the institution are left unannotated.

[...] **Hyderabad Women's Business Network (HWBN)**
Workshop: Financial Resilience & Legal Safeguards for Entrepreneurs
Date of Intake: January 23, 2019
Intake Officer: Priya Singh, HWBN Coordinator
[...]
Section 1: Participant Details
Full Name: Theresa Reddy
Date of Birth: March 04, 1960
Contact Number: +91-98490-23761
Email Address: theresa.r.textiles@gmail.com
Marital Status: Married
Native Language: Telugu

Business Name (if applicable): Reddy Handlooms (New Sole Proprietorship)
Primary Business Sector: Handloom Textiles & Exports
[...]
understand that this information will be held in strict confidence by the Hyderabad Women's Business Network.
Signature: *Theresa Reddy*
Date: 23/01/2019

Figure 20: `privasis_zero_hard` — 6 gold spans, all belonging to the focal participant.

[...] 8700 Beverly Blvd, Los Angeles, CA 90048
INTERNAL MEMORANDUM
TO: USCIS California Service Center
CC: Dr. Lila Farhadi, Farhadi & Associates, 10880 Wilshire Blvd Suite 1101, Los Angeles, CA 90024
FROM: Eleanor Vance, Director, Medical Staff Services & Credentialing
DATE: November 9, 2016
MEMO NO.: CRED-V-2016-9127
SUBJECT: RE: I-140 Immigrant Petition for Alien Worker; Beneficiary: Aarien Shirazi
This letter is to confirm the employment offer extended by Cedars-Sinai Medical Center to the beneficiary, Aarien Shirazi, in support of the I-140 Immigrant Petition filed on his behalf. Mr. Shirazi's identifying details are as follows: Aarien Shirazi, Iranian citizen, Passport No. M51284937, Iranian National ID No. 0083745921, and CSMC Candidate ID 774028.
[...]
Smidt Heart Institute
Department of Cardiothoracic Surgery
Cedars-Sinai Medical Center
8700 Beverly Blvd
Los Angeles, CA 90048 [...] Sincerely,
Eleanor Vance
Director, Medical Staff Services & Credentialing
Cedars-Sinai Medical Center
Direct Line: (310) 423-5922
Email: e.vance.mss@cshs.org

Figure 21: `privasis_zero_vanilla` — the gold spans cover the employer and the petition type; the beneficiary's own identifiers are not annotated.