When Parametric Knowledge Wins: Evaluation Shapes Tool-Augmented PII Detection in Small Language Models

Anonymous authors
Paper under double-blind review

Abstract

Agent augmentation is widely assumed to help, yet for small language models, its measured benefit on PII detection depends as much on how detection is scored as on model capability and behavior. A controlled ablation spans four open-weight models (Gemma 2 9B, Llama 3.1 8B, Mistral 7B, Qwen 2.5 7B) and four conditions (zero-shot, +Docs, +Tool, +Skills) on a stratified 2,000-sample benchmark; predictions and ground truth are scored under one symmetric label-alignment map and two co-primary metrics. At the *type* level, zero-shot beats both tool-augmented conditions for every model: +Tool and +Skills cut mean F1 by 6–17 points ($p < 0.0001$; Cohen's d -0.16 to -0.49), documentation is neutral except for Llama 3.1 8B ($\Delta = -0.15$), and the skill adds nothing over tool access for these out-of-the-box models. Few-shot and chain-of-thought prompting match or trail zero-shot (type-level F1 0.63 and 0.61), ruling out a weak baseline. At the *span* level, the ranking inverts, and detector-backed conditions lead. Much of the type-level gap is an alignment artifact: temporal and geographic types collapse only when canonical labels go unbridged to the benchmark schema, and recover under symmetric alignment; the residual is an interface effect: the recall-tuned detector over-proposes candidates that a precision-weighted metric penalizes. Run standalone, the detector scores 0.28 versus 0.51–0.63 for the models: +Tool is a model×library interaction, complementary at the span level rather than additive. The effect persists without quantization (4-bit and fp16/bf16 agree within 0.03 F1 for every model) and one model generation up (Qwen2.5-14B). In mixed-mode pipelines, label alignment and evaluation granularity largely determine whether agentic scaffolding appears to help; under naive scoring, the same intervention can look beneficial or harmful.

1 Introduction

The proliferation of agentic AI systems has introduced a new class of prompting strategies in which language models are given access to external tools, domain documentation, and structured skill definitions at inference time (Yao et al., 2023; Schick et al., 2023; Chase, 2022). The motivating intuition is straightforward: if a model lacks reliable internal representations of a specialized domain, injecting authoritative external references should improve task performance. This assumption underpins a broad class of retrieval-augmented generation (RAG) systems (Lewis et al., 2020), tool-use frameworks (Chase, 2022), and the emerging concept of *Agent Skills* (modular, reusable context documents that encode procedural knowledge, tool syntax, and domain constraints for a specific task).

PII detection is a natural testbed for this hypothesis. It is a high-stakes structured extraction task with a well-defined label schema, and it is increasingly delegated to SLMs in agentic data pipelines where latency, cost, and privacy constraints preclude the use of frontier-scale models (Lison et al., 2021; Lukas et al., 2023). Sending sensitive text to API-based frontier models pose unacceptable privacy and security risks. Deploying local SLMs for PII and data-science workloads is therefore a critical industry requirement (Sun et al., 2025). The domain also has a mature programmatic library ecosystem including Microsoft Presidio (Mendels et al., 2018), spaCy NER (Honnibal et al., 2020), PII-Codex (Rosado, 2023), which provides structured canonical taxonomies against which model predictions can be systematically compared, though programmatic detection accuracy varies by entity type. This yields a concrete, falsifiable hypothesis for why an agentic formulation

might help even on a task whose answer lies in the input text: because programmatic detectors encode reliable rules for structured identifiers (date, social-security, and IP formats), tool access should raise recall on exactly those types. The empirical question is whether that type-specific benefit survives aggregation across the full schema, or is outweighed by the cost of deferring to a rule-based tool on the many types where a model's contextual understanding is stronger.

A four-condition ablation was designed to isolate the effects of documentation, tool access, and skills injection on PII detection accuracy in four open-weight SLMs (7–9B parameters), including Gemma 2 9b (Gemma Team, 2024), Llama 3.1 8b (Grattafiori et al., 2024), Mistral 7b (Jiang et al., 2023), and Qwen 2.5 7b (Yang et al., 2024).

The conditions are:

1. *Zero-shot (+ZS):* The model receives a task prompt and sample text with no external augmentation.

2. *Documentation (+Docs):* The model additionally receives the PII-Codex reference documentation in context (single-turn; no tool execution).

3. *Tool-augmented (+Tool):* The model is given access to the `analyze_pii` tool (PII-Codex execution) via a LangGraph-based SkillsAgent, and may call it before returning a final answer.

4. *Skills-augmented (+Skills):* In addition to tool access, the model may discover and read a structured PII detection Skill document (`SKILL.md`) via `list_skills` and `view_skill` tool calls before calling `analyze_pii`.

Beyond these four conditions, the design adds three baselines and two robustness controls. The baselines (PII-Codex run standalone, isolating the library; few-shot prompting; and chain-of-thought prompting) guard against a weak-comparison artifact: neither parametric baseline exceeds zero-shot, and the standalone library trails every model condition, so the zero-shot result is a genuine ceiling. The two controls extend the design beyond the 7–9B class: a re-run on a larger model (Qwen2.5-14B) and a re-run unquantized (fp16/bf16), each testing whether an observed effect is bound to model scale or quantization rather than to the evaluation pipeline.

Using a stratified benchmark of 2,000 PII-annotated English samples from three public datasets, zero-shot prompting outperforms both tool-augmented conditions for all four models at the entity-type level, contrary to common assumptions. However, at the span level, the rule-based detector excels in localizing exact character spans. Two main causes contribute to the type-level degradation: a label-alignment artifact that can be balanced through symmetric mapping and a model-library interface issue where the rule-based detector over-predicts and lacks context. This pattern remains consistent without quantization and persists even with a single 14B model (Qwen2.5-14B).

The tool condition compares *non-fine-tuned* tool calling against zero-shot extraction, and once invoked, the tool's output (not the model) determines the canonical predictions; the finding is thus that wrapping a non-fine-tuned 7–14B model around a rule-based PII library does not help on this static task under fair evaluation, not that parametric knowledge universally beats tools. The study, therefore, carries a measurement lesson alongside its empirical one: in mixed-mode pipelines, label alignment and evaluation granularity decide whether an intervention appears beneficial.

This aligns with the view that modern SLMs already possess strong inherent reasoning and text-understanding capabilities (Yu et al., 2024) and need not rely on external tools for a fundamental extraction task like PII detection, and with recent evidence that arbitrary tool access can actively degrade performance compared to zero-shot when retrieval errors and scaffolding overhead outweigh tool benefits (Luo et al., 2025; Qian et al., 2025; Xu et al., 2025). LLMs struggle to decide *whether* to use a tool (Ning et al., 2024); zero-shot prompting avoids that pitfall by not deferring to external tools when parametric knowledge suffices. The finding has direct implications for how agentic scaffolding should be evaluated and deployed in production NLP pipelines.

2 Related Work

PII detection with language models. Named entity recognition (NER) approaches to PII detection have a long history in NLP (Lample et al., 2016; Devlin et al., 2019). More recently, instruction-tuned LLMs have been evaluated as zero-shot or few-shot entity extractors, though their out-of-the-box performance on strict sequence-labeling tasks often lags behind supervised baselines without highly specialized prompt adaptations (Wang et al., 2025). Furthermore, separate work demonstrates that language models are highly susceptible to leaking sensitive PII depending on the prompt design and the entity type targeted by an attacker, underscoring the critical need for reliable, external PII detection safeguards in agentic pipelines (Lukas et al., 2023). PII-Codex (Rosado, 2023) is built on Microsoft Presidio NER detection (Mendels et al., 2018), leverages the information-sensitivity typology (Milne et al., 2016) and the risk identification continuum (Schwartz & Solove, 2011); it provides a unified taxonomy and programmatic evaluation framework that enables type-level scoring against a canonical schema.

Tool use and RAG in language models. Surveys of natural language reasoning suggest that modern SLMs already possess strong inherent reasoning and text-understanding capabilities (Yu et al., 2024); for a static, pattern-recognition task like PII detection, external tools may therefore be redundant or harmful. Conversely, tools are necessary when models must incorporate real-time, rapidly changing, or newly updated information that cannot be captured by static pretrained weights (Yu & Ji, 2024); PII detection, by contrast, is a static extraction task for which parametric knowledge and in-context instruction suffice, and forcing an agent to invoke tools introduces unnecessary overhead and the degradation observed in this study.

ReAct (Yao et al., 2023) and Toolformer (Schick et al., 2023) established that language models can interleave reasoning and tool execution. Subsequent work has shown that tool use can substantially improve performance on knowledge-intensive tasks, though gains are not universal and depend heavily on instruction-following fidelity and the schema alignment between tool outputs and the evaluation metric (Qin et al., 2024). Providing models with arbitrary tool access can actively degrade performance compared to zero-shot, as error rates from unnecessary retrieval, inappropriate invocation, and scaffolding often outweigh tool benefits (Luo et al., 2025; Qian et al., 2025; Xu et al., 2025). LLM-generated internal knowledge can surpass externally retrieved knowledge. External retrieval often introduces irrelevant or poorly coherent information that harms downstream performance more than minor factual errors in parametric memory (Chen et al., 2023). For many tasks, parametric knowledge is sufficient and "tool overuse" lowers accuracy (Luo et al., 2025; Qian et al., 2025). SLMs generally fail to execute multi-turn tool-discovery loops out-of-the-box; targeted domain-specific fine-tuning is often advocated for reliable agentic tool calling (Belcak et al., 2025; Jhandi et al., 2025). Toolformer showed that a small model can excel at tool use when fine-tuned to predict tool calls (Schick et al., 2023); out-of-the-box scaffolding without such fine-tuning is a poor fit for the 7–9B class (Patil et al., 2024). The study contributes an evaluation of tool use in the specific context of structured information extraction in the 7–9B parameter class, with a robustness check one model generation up.

Agent Skills and skill injection. Injecting modular "skill" documents (reusable context that encodes procedural knowledge, tool syntax, and domain constraints) into a model's context at inference time is a practitioner pattern that has so far received limited formal evaluation. It should be distinguished from a separate and more developed line of work on *skill acquisition and optimization*, in which an agent learns, composes, or iteratively refines a library of skills against task feedback rather than reading a fixed document at inference (Yang et al., 2026; Alzubi et al., 2026). Those approaches treat skills as learned, evolving artifacts; the inference-time injection studied here instead supplies a static, hand-written document and asks whether merely placing it in context helps a frozen model. The two questions are complementary: whether a better skill can be *learned*, versus whether a given skill document, as deployed, *helps at all*. The value of such a combination is, moreover, task-dependent: PokerSkill (Li et al., 2026) reports that pairing expert-authored rule-based skills with an LLM beats either component alone on strategic poker, whereas the reverse holds for static PII extraction, where the model is strongest unaugmented, underscoring that whether skill or tool grounding helps depends on the task and the quality of the underlying library. To the best of the author's knowledge, this is the first controlled ablation of inference-time skill-document injection for PII detection, with pre-specified conditions and post-hoc statistical testing.

3 Methods

3.1 Benchmark compilation

A stratified benchmark of 2,000 English-language PII-annotated samples was compiled from three public HuggingFace datasets, AI4Privacy (Ai4Privacy, 2024) (`ai4privacy/pii-masking-300k`), NVIDIA Nemotron-PII (Steier et al., 2025) (`nvidia/Nemotron-PII`), and Gretel PII Masking (Gretel AI, 2024) (`gretelai/gretel-pii-masking-en-v1`).

Let $D = \{(x_i, G_i)\}_{i=1}^{N}$ with $N = 2{,}000$ denote the benchmark, where x_i is a text sample and $G_i = \{(s_{ij}, y_{ij})\}$ is the set of ground-truth entity spans and types for sample i. Source distributions in the final benchmark are shown in Table 1. The mix of PII types varies by source; type-level counts are in Appendix Table 7.

Table 1: Benchmark source distribution ($N = 2{,}000$). Samples and per-source entity counts as recorded during stratification. Total entity span count for the compiled benchmark is given in Appendix Table 7.

Source	Samples	Entity spans (source)	Mean/sample
AI4Privacy	1,132	7,112	6.28
Gretel PII Masking	581	2,416	4.16
NVIDIA Nemotron-PII	287	1,842	6.42
Total	**2,000**	**11,370**	**5.69**

Language and locale filtering. AI4Privacy was filtered to rows with `language == "English"`; NVIDIA Nemotron-PII to `locale == "us"`; Gretel requires no filter (English-only dataset). Results apply to English text and US-locale PII types only and do not generalize to other locales or languages.

Label mapping and exclusion. Source labels were normalized and mapped to PII-Codex (Rosado, 2023) canonical types (`PIIType.name`) via a fixed 136-entry mapping table. Records containing any PII instance with no PII-Codex mapping were excluded, so all ground-truth labels in the benchmark are fully mappable. The final set covers 21 PII types. Ground-truth entity span counts (each contiguous span of a given type counted once) by type for the full benchmark are in Appendix Table 7; the most frequent types are `PERSON`, `DATE_TIME`, `LOCATION`, `ADDRESS`, and `EMAIL_ADDRESS` (Table 2).

Table 2: Selected frequent PII types by ground-truth entity count (main study benchmark, descending). Full distribution in Appendix Table 7.

PII Type	Entity count
PERSON	2,324
DATE_TIME	1,572
LOCATION	1,068
ADDRESS	862
EMAIL_ADDRESS	632
US_SOCIAL_SECURITY_NUMBER	632
PHONE_NUMBER	566
DATE	531
IP_ADDRESS	515
US_DRIVERS_LICENSE_NUMBER	383
HEALTH_INSURANCE_ID	343

3.2 Models

Four open-weight instruction-tuned models were evaluated: Gemma 2 9B, Llama 3.1 8B, Mistral 7B, and Qwen 2.5 7B. The main study used 4-bit quantized MLX variants (`mlx-community`) on a 2020 MacBook Pro M1 Pro Max (64 GB RAM). Two robustness controls add further runs: an unquantized (fp16/bf16) variant of each model on a stratified 300-sample subset, and a larger model (Qwen2.5-14B, 4-bit) on the full benchmark (Section 5.6, Appendix E). Prompts and chat templates were identical across models and conditions; seed 42 was fixed for all runs.

3.3 Conditions

The study comprises four primary conditions, three additional baselines, and two robustness controls. The four primary conditions are evaluated on all 2,000 benchmark samples across the four models, yielding 32,000 prediction rows (2,000 samples $\times$ 4 models $\times$ 4 conditions); the baselines (Detector-only, Few-shot, and Chain-of-thought, described below) and the robustness controls (a 14B model and an unquantized rerun, Section 5.6) extend this core ablation. Each condition c defines an inference pipeline $f_c \colon \mathcal{X} \to \mathcal{P}(\mathcal{S} \times \mathcal{Y}_c)$ mapping input text x_i to a predicted set of entity spans and labels $\hat{G}_{i,c} = f_c(x_i)$.

Zero-shot (ZS). Single-turn prompt containing the task instruction and sample text. No external augmentation. The model returns a JSON array of PII predictions.

Documentation (+Docs). Same single-turn path with the PII-Codex reference documentation prepended to the prompt. No tool execution.

Tool-augmented (+Tool). The model is given access to `analyze_pii`, which executes PII-Codex on the sample text, via a custom SkillsAgent created with LangGraph (LangChain, 2024; Chase, 2022). Tool-call detection uses strict bracket syntax (`[TOOL_CALL: analyze_pii]`) with a lenient intent-based fallback. If no tool call is detected, the model's first response is taken as the final answer. Once the tool fires ($\sim$99.99% of runs; Section 4.6), PII-Codex produces the candidate spans and types and the model filters and formats them, so +Tool measures a model$\times$library interaction.

Skills-augmented (+Skills). Same LangGraph (LangChain, 2024) agent with two additional tools: `list_skills` (returns available skill names) and `view_skill("pii-detection")` (returns the PII detection Skill document, `SKILL.md`). Models that read the Skill document before calling `analyze_pii` receive a structured prompt encoding PII-Codex tool syntax, output format, and type coverage. The same Skill document was used for both pilot and main studies. The document is deliberately minimal ($\sim$250 tokens) and points to the existing tool rather than adding new procedural capability; the +Skills condition therefore tests whether a lightweight tool-pointer skill adds value over tool access alone, not whether richer skill documents could.

Baselines. To separate the model's contribution from that of the underlying library, and to situate the four conditions against simpler interventions, three baselines are reported, scored under the identical alignment. *Detector-only* runs PII-Codex directly on the sample text (no model) and scores its raw output; this isolates the library. *Few-shot* (Brown et al., 2020) prepends in-context PII-annotated examples to the zero-shot prompt (single-turn, no tool). *Chain-of-thought* (Kojima et al., 2022) adds an explicit step-by-step reasoning instruction to the zero-shot prompt (single-turn, no tool). The few-shot and chain-of-thought baselines test whether the simplest parametric augmentations help where tool and skill scaffolding do not; both prompts are reproduced in Appendix H.

3.4 Execution infrastructure

Conditions were run sequentially per model (one condition at a time). Within multi-turn conditions (+Tool, +Skills), samples were processed in parallel via a ThreadPoolExecutor (1 worker in this study), with a shared lock serializing `generate` calls. Context was bounded per turn (initial user prompt + current system

message only) to prevent context growth from degrading performance in the skills discovery loop. A per-sample timeout prevented runaway inference without discarding partial results.

3.5 Scoring and label alignment

Predictions were scored against ground truth using precision, recall, and F1. Two complementary matching criteria are reported as *co-primary* metrics. *Type-level* scoring counts a prediction as a true positive when its canonical type equals that of an as-yet-unmatched ground-truth entity; *span-level* scoring additionally requires the predicted and ground-truth character spans to overlap with intersection-over-union (IoU) ≥ 0.5. The two answer different questions (type-level asks whether the correct *categories* of PII present were identified, span-level whether they were also *localized*) and, as Section 4.1 shows, they do not always agree. Both are therefore reported rather than privileging one.

Symmetric label alignment. A single canonical normalization map $\phi\colon \mathcal{Y} \to \mathcal{Y}_{\text{canon}}$ is applied *identically* to predictions and ground truth in every condition and baseline. Built on a 136-entry base table (`pii_label_to_piicodex.json`), ϕ maps model-native label names (e.g., `person_name`, `ssn`, `date_of_birth`) to PII-Codex canonical types and bridges equivalent canonical types emitted by the detector versus recorded in the benchmark: most importantly `DATE` $\equiv$ `DATE_TIME` and `ADDRESS` $\equiv$ `LOCATION`, together with health-identifier variants. Aligned predictions are $\tilde{G}_i = \{(s, \phi(y)) \mid (s, y) \in \hat{G}_i\}$ and aligned ground truth is $\{(s, \phi(y)) \mid (s, y) \in G_i\}$; matching is performed in this shared canonical space.

Applying ϕ to only one side is unfair by construction. Because the tool and skills conditions invoke PII-Codex directly and emit canonical types that score without remapping, remapping only zero-shot model-native labels, while leaving the tool's `DATE` output unbridged to the benchmark's `DATE_TIME`, depresses tool and skills recall on temporal types for schema reasons and flatters zero-shot. All results below use the symmetric map ϕ and are *post-alignment*; pre-alignment scores are not comparable across conditions and are not reported.

3.6 Statistical analysis

The analysis uses a within-subjects, repeated-measures design: each of the 2,000 benchmark samples appears in all four conditions, and condition effects are estimated from paired within-sample differences $\Delta_i(c) = \text{F1}(\tilde{G}_{i,c}, G_i) - \text{F1}(\tilde{G}_{i,\text{ZS}}, G_i)$. Reported are paired t-tests (condition F1 vs. zero-shot F1 at the sample level) and Cohen's d for key comparisons (+Docs vs. ZS, +Tool vs. ZS, +Skills vs. ZS, and +Skills vs. +Tool). Bootstrap 95% confidence intervals are reported for mean F1 per (model, condition) cell. Sample size was confirmed sufficient for the target margin of ± 0.05 in all 16 cells (maximum required $n = 195$; actual $n = 2,000$). Decoding is greedy (argmax; `do_sample = False`), so model outputs are deterministic given a prompt: the reported effects are not subject to run-to-run generation variance, and the random seed governs only the fixed stratified subsample rather than the generations themselves.

4 Results

4.1 Aggregate performance

Table 3 presents mean F1 (macro-averaged across types, and net of the label-schema artifact via the symmetric alignment) by model and condition under *type-level* scoring. Zero-shot leads both tool-augmented conditions for all four models; +Tool and +Skills underperform zero-shot by 6–17 percentage points. Documentation is largely neutral (Gemma, Mistral, Qwen: $\Delta \approx 0$) but substantially degrades Llama 3.1 8B ($\Delta = -0.15$). Figure 1 visualizes F1 by configuration under both metrics.

The ranking depends on evaluation granularity. Type-level and span-level scoring disagree on whether augmentation helps (Table 4; Figure 1). Under *span-level* matching (IoU ≥ 0.5), the ordering *inverts*: averaged across the four models, +Tool reaches F1 = 0.46 against zero-shot's 0.33, because the specialized detector localizes entity spans more precisely than the models' free-form character offsets, even though its predicted type-set is noisier. The "zero-shot wins" result, therefore, holds only at the type level; whether

agentic augmentation helps is a function of the evaluation criterion, not of the scaffolding alone. Both metrics are reported as co-primary throughout.

A model–library interaction. Run standalone on the same texts, PII-Codex scores type-level F1 0.28 and span-level 0.23 (Table 4, "Detector"), below every model condition, including +Tool. The +Tool result is therefore a model×library interaction, not the library in isolation: the model nearly doubles the raw library's type-level F1 by filtering its over-predictions. The ordering is consistent: the model alone (zero-shot, 0.63) beats the model with the library (+Tool, 0.53), which beats the library alone (0.28). Adding the raw library output lowers type-level F1 even as the model improves on it, so the two are complementary rather than interchangeable: the library leads at the span level. This ordering is unchanged without quantization and persists one model generation up: for Qwen2.5-14B, zero-shot again leads +Tool at the type level (0.61 versus 0.52; Section 5.6, Appendix E).

Few-shot and chain-of-thought baselines. Neither few-shot nor chain-of-thought prompting surpasses zero-shot. Averaged across the four models, few-shot prompting matches zero-shot at the type level (0.628 vs. 0.631) and chain-of-thought is slightly worse (0.606); the span-level picture is the same (few-shot 0.319, CoT 0.298, vs. zero-shot 0.325; Appendix C, Table 12). Per model, the effect is small and mixed (the few-shot delta ranges from −0.065 for Llama to +0.032 for Mistral). This isolates the source of the large degradation: in-context *parametric* augmentation is roughly neutral ($|\Delta| \leq 0.03$ on average), whereas external tool and skill augmentation costs 0.10–0.17 F1. The cost is therefore specific to integrating the rule-based library, not to augmentation in general, and the zero-shot result is a genuine ceiling rather than an artifact of comparing against a weak augmentation. Figure 2 places every configuration on a single type-level axis.

Table 3: Main study mean F1 by model and condition (N = 2,000, post-label alignment). All deltas are relative to zero-shot.

Model	**ZS**	**+Docs**	$\Delta_{\mathbf{Docs}}$	**+Tool**	$\Delta_{\mathbf{Tool}}$	**+Skills**	$\Delta_{\mathbf{Skills}}$
Gemma 2 9B	0.62	0.64	+0.01	0.53	−0.10	0.53	−0.10
Llama 3.1 8B	0.68	0.53	−0.15	0.57	−0.11	0.50	−0.17
Mistral 7B	0.58	0.59	+0.01	0.53	−0.06	0.53	−0.06
Qwen 2.5 7B	0.64	0.65	+0.01	0.49	−0.15	0.48	−0.16

Table 4: Mean F1 averaged across the four models, under the two co-primary metrics (main study, $N = 2{,}000$, symmetric alignment), with PII-Codex run standalone (no model) as a library-only reference. Type-level and span-level scoring disagree on the ranking of zero-shot versus the augmented conditions, and the standalone detector trails every model condition under both metrics.

Metric	**ZS**	**+Docs**	**+Tool**	**+Skills**	**Detector**
Type-level	**0.63**	0.60	0.53	0.51	0.28
Span-level (IoU ≥ 0.5)	0.33	0.30	**0.46**	0.45	0.23

4.2 Statistical significance and effect sizes

Table 5 reports paired t-test results and Cohen's d for each model and condition against zero-shot, under type-level scoring. All +Tool and +Skills comparisons are statistically significant ($p < 0.0001$) with small-to-medium negative effect sizes (d from −0.16 to −0.49). Documentation (+Docs vs. zero-shot) has no significant effect for Gemma, Mistral, or Qwen ($p = 0.07$, 0.18, 0.09; small *positive* d), and significantly degrades only Llama 3.1 8B ($p < 0.0001$, $d = -0.37$). Under an asymmetric alignment, these effect sizes are larger (for example, +Skills reaches $d = -0.67$ for Llama): the fair, symmetric alignment substantially reduces the apparent magnitude of degradation, while its direction and significance hold for every model.

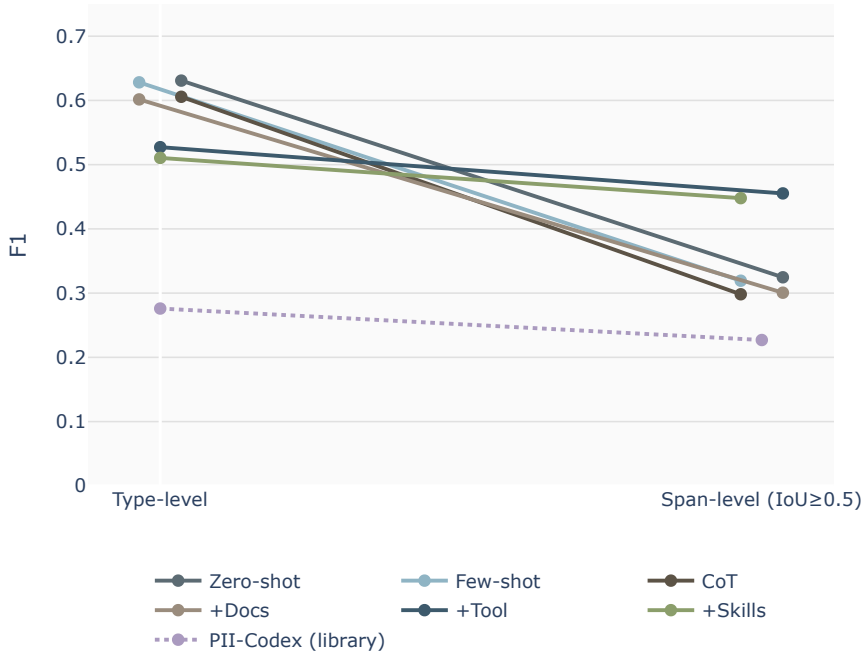

Figure 1: Type-level versus span-level F1 by configuration (four-model average, main study, $n = 2{,}000$, symmetric alignment). Each line connects a configuration's type-level F1 to its span-level F1 (IoU ≥ 0.5). Parametric conditions start high on type and fall on span; tool-augmented conditions rise on span, so the lines cross: the ranking inversion.

The incremental effect of adding the Skill document on top of tool access (+Skills vs. +Tool) yields Cohen's d of -0.29 for Llama, -0.02 for Mistral, -0.02 for Qwen, and $+0.04$ for Gemma. None reach conventional thresholds for a meaningful effect, confirming that the Skill document provides no consistent incremental benefit beyond tool access alone.

4.3 Per-PII-type recall

Aggregate F1 masks heterogeneous effects across PII types, and the per-type pattern is what separates a label-schema artifact from genuine detector behavior. Table 6 shows mean recall delta (+Skills vs. zero-shot) by type and model under the symmetric alignment; Figure 3 contrasts tool/skill recall by PII type before and after alignment; and Figure 4 summarizes the same recall surface as a configuration-by-type matrix. Full recall-by-type values for every condition and baseline across all four models are in Appendix D (Table 14).

Artifact, not failure: temporal and geographic types. Under an asymmetric alignment, `DATE_TIME` and `LOCATION` recall collapses nearly to zero under tool conditions. This is a scoring artifact: the tool emits `DATE` and `ADDRESS` for spans the benchmark records as `DATE_TIME` and `LOCATION`, which an unbridged map leaves unmatched. Under the symmetric map (`DATE` $\equiv$ `DATE_TIME`, `ADDRESS` $\equiv$ `LOCATION`) these recover: `DATE_TIME` recall under +Skills is neutral-to-positive (e.g. +0.32 Gemma, +0.28 Mistral), and `LOCATION` no longer collapses.

Genuine detector limit: medical identifiers. `HEALTH_INSURANCE_ID` recall, by contrast, remains near zero under tool conditions *even after* symmetric alignment (-0.62 to -0.74 across models). This represents a coverage gap rather than a schema mismatch: PII-Codex does not map any detector to the `HEALTH_INSURANCE_ID` type, as Presidio lacks a general health-insurance recognizer. The health-identifier

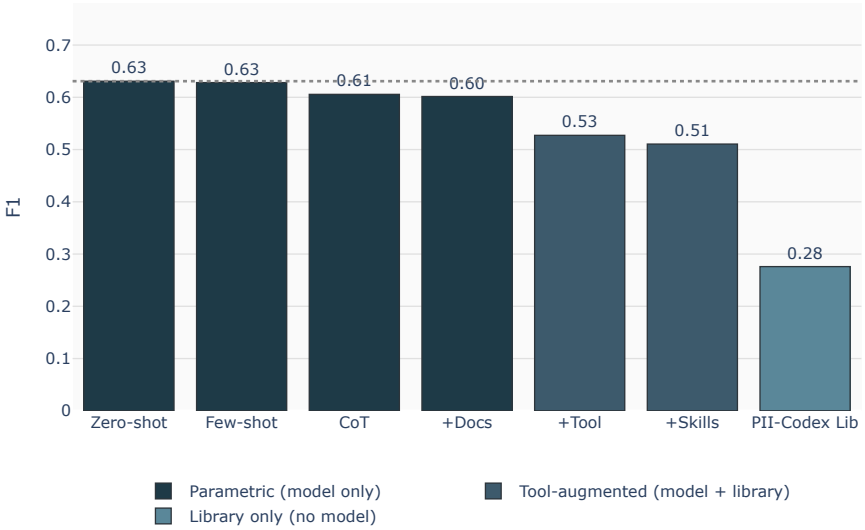

Figure 2: All configurations at a glance: type-level F1 averaged across the four models (main study, $n = 2{,}000$, symmetric alignment), grouped by family. The parametric model-only conditions (zero-shot, few-shot, chain-of-thought, and +Docs) cluster highest; the tool-augmented conditions (+Tool, +Skills) fall below them; and PII-Codex runs as a standalone library, with no model, is lowest. The dashed line marks the zero-shot reference. At the type level, augmentation moves performance down rather than up, and the model clearly exceeds the library it is asked to call.

rules provided by Presidio are limited to specific formats (US Medicare MBI, UK NHS, Australian Medicare, US medical license) and do not match the benchmark's identifiers. Consequently, the tool misses detections that could be made from context. This situation highlights the difference between an evaluation artifact and a substantive failure mode.

Table 5: Paired t-test results and Cohen's d (condition vs. zero-shot, N = 2,000 per model).

Model	Condition	t	p	d
Llama 3.1 8B	+Docs	−16.50	< 0.0001	−0.37
	+Tool	−14.27	< 0.0001	−0.32
	+Skills	−21.84	< 0.0001	−0.49
Mistral 7B	+Docs	+1.35	0.18	+0.03
	+Tool	−7.08	< 0.0001	−0.16
	+Skills	−7.14	< 0.0001	−0.16
Qwen 2.5 7B	+Docs	+1.68	0.092	+0.04
	+Tool	−18.05	< 0.0001	−0.40
	+Skills	−18.66	< 0.0001	−0.42
Gemma 2 9B	+Docs	+1.81	0.071	+0.04
	+Tool	−11.97	< 0.0001	−0.27
	+Skills	−11.87	< 0.0001	−0.27

Table 6: Recall delta (+Skills vs. zero-shot) by PII type and model (main study, symmetric alignment), ordered from most-improved to most-degraded. Positive values indicate higher recall under skills injection; negative values indicate degradation.

PII Type	Gemma 2 9B	Llama 3.1 8B	Mistral 7B	Qwen 2.5 7B
US_DRIVERS_LICENSE_NUMBER	+0.23	+0.32	+0.22	+0.21
DATE_TIME	+0.32	+0.03	+0.28	+0.02
IP_ADDRESS	+0.17	−0.02	+0.14	+0.05
EMAIL_ADDRESS	+0.02	0.00	+0.05	+0.01
LOCATION	+0.03	−0.13	+0.08	−0.21
US_PASSPORT_NUMBER	−0.08	+0.03	−0.03	+0.10
US_SOCIAL_SECURITY_NUMBER	−0.13	−0.20	−0.28	−0.22
PERSON	−0.20	−0.25	−0.06	−0.20
PHONE_NUMBER	−0.25	−0.22	−0.23	−0.36
HEALTH_INSURANCE_ID	−0.71	−0.73	−0.62	−0.74

Structured identifiers: recall gains at a precision cost. Types with unambiguous surface forms (`US_DRIVERS_LICENSE_NUMBER`, `IP_ADDRESS`) gain recall under tool use because the detector reliably flags them. Its recall-first design also proposes these labels well beyond their true frequency (Section 4.4); however, so under exact-match F1, the recall gain is paid for in precision. Common semantic types (`PERSON`, `PHONE_NUMBER`, `US_SOCIAL_SECURITY_NUMBER`) lose recall under augmentation, as the model defers to tool output that misses spans it would otherwise have extracted from context.

4.4 Recall-oriented detection meets a precision-weighted metric

The recall gains on structured identifiers come with a precision cost under the exact-match F1 objective used here. Rule-based privacy detectors such as PII-Codex and the Presidio engine beneath it are deliberately

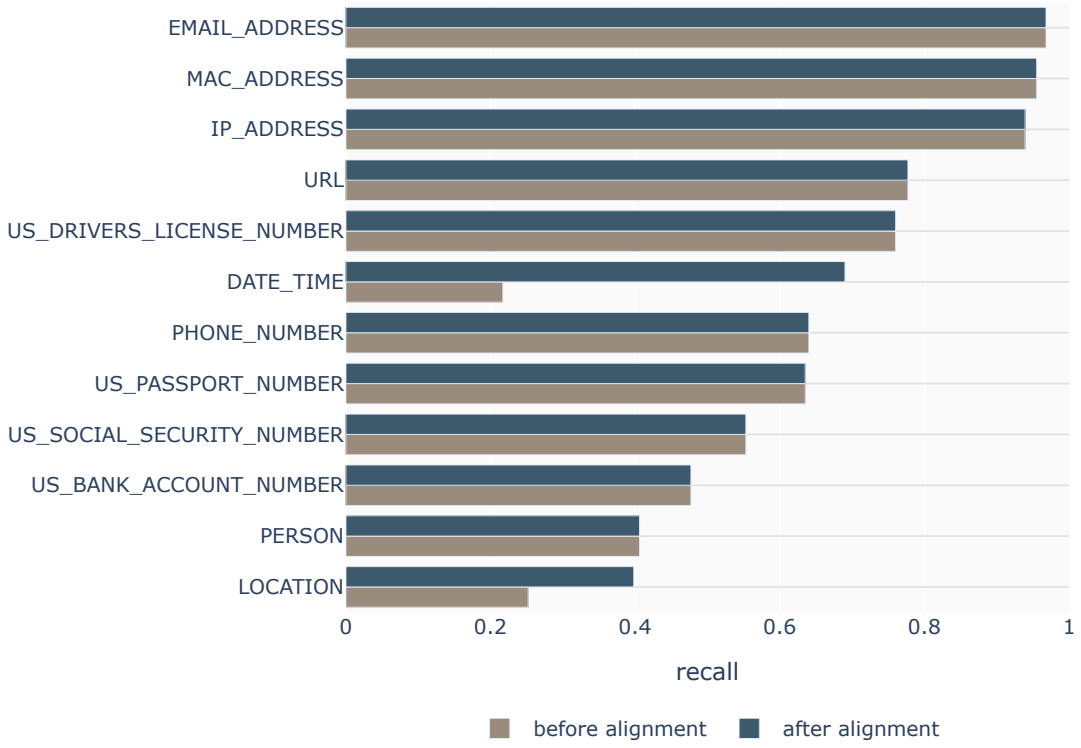

Figure 3: Tool/skill recall by PII type, before vs after symmetric alignment (main study, $n = 2{,}000$). `DATE_TIME` and `LOCATION` recover once the detector's `DATE/ADDRESS` outputs are bridged to the benchmark schema (the artifact); other types are unchanged. `HEALTH_INSURANCE_ID` remains near zero (a genuine detector limit).

tuned for high recall: in a redaction or compliance setting, over-flagging a candidate is far safer than missing real PII, so they propose candidates liberally by design. Aggregated over the tool and skills conditions the detector, therefore surfaces many identifier candidates: `US_BANK_ACCOUNT_NUMBER` 3,421 times against 44 true entities, `URL` 4,501 against 92, and `US_DRIVERS_LICENSE_NUMBER` 12,150 against 383 (Figure 5). This recall-first behavior is appropriate for the library's intended use, but it interacts poorly with a precision-weighted F1 and with an *non-fine-tuned* model that does not down-weight the extra candidates before emitting a final answer. The resulting precision cost is an integration effect, not a detection failure: it is the principal driver of the F1 decline that survives symmetric alignment, and it reflects the integration pattern (passing raw, recall-oriented output into a model not trained to calibrate it) rather than a deficiency of the library (whose span-level advantage appears in Section 4.1).

4.5 Skill-viewed subgroup analysis

For the +Skills condition, Mistral read the Skill document in 948 of 2,000 runs (47.4%) and Qwen in 341 of 2,000 runs (17.1%). Llama and Gemma read the document in 100% of runs (no non-viewed comparison available). This spread reflects how consistently each model engaged the multi-step discovery protocol before answering: Mistral and Qwen often called `analyze_pii` without first reading the Skill document, consistent with the known difficulty non-fine-tuned 7–9B models have with multi-step tool-discovery loops (Patil et al., 2024; Qin et al., 2024). For Mistral (viewed $n = 948$, not-viewed $n = 1{,}052$) and Qwen (viewed $n = 341$, not-viewed $n = 1{,}659$), mean F1 was visually indistinguishable between the two groups in both cases. This

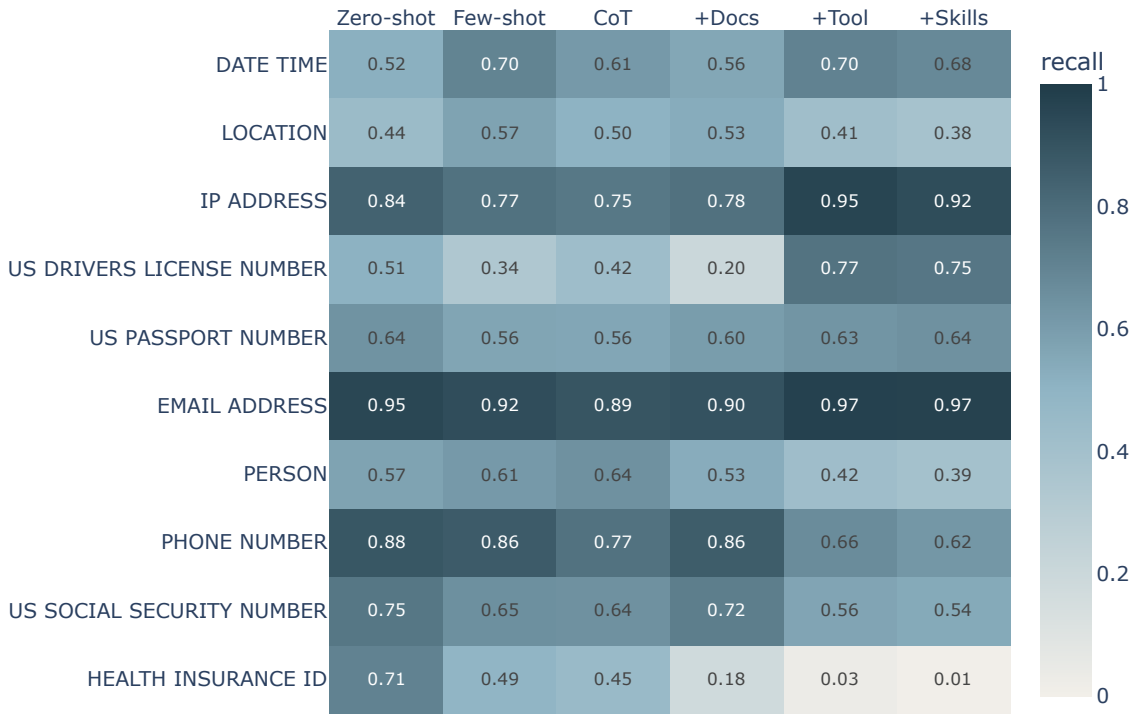

Figure 4: Recall by PII type and configuration (four-model average, main study, symmetric alignment). Structured identifiers brighten under +Tool/+Skills; `DATE_TIME` and `LOCATION` recover after alignment; `HEALTH_INSURANCE_ID` stays near zero under tool conditions (a genuine detector gap).

within-condition comparison controls for all factors except document reading, and provides no evidence that reading the Skill document improves PII detection performance, even when the model chooses to access it. Because Llama and Gemma read the document in every run, the subgroup contrast is only available for Mistral and Qwen; it is therefore treated as suggestive rather than conclusive.

4.6 Execution reliability

Tool execution was reliable across both tool conditions: the 4,000 executions per model (2,000 samples × 2 conditions: +Tool and +Skills) completed successfully for Gemma, Llama, and Qwen; Mistral completed 3,998/4,000 (99.95%). Error rate was 0% across all 16 model-condition cells in both pilot and main study runs. Because the model emitted a tool call in essentially every run (two omissions in 16,000 +Tool and +Skills runs), every "tool" run reflects genuine tool use, and the F1 degradation under tool conditions reflects tool-output quality and over-prediction.

5 Discussion

5.1 The pre-alignment confound

The central result is partly methodological and partly behavioral, and disentangling the two is a key contribution of this study. The methodological part arises because tool conditions invoke PII-Codex directly

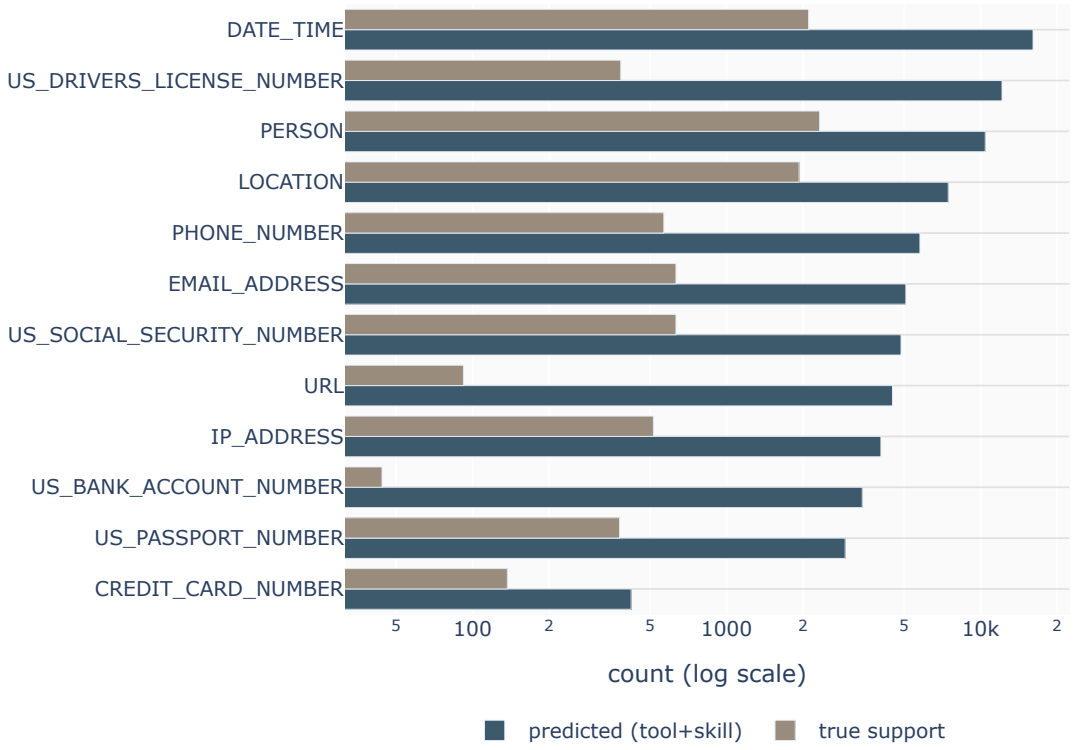

Figure 5: Predicted label volume versus true support in the tool/skills conditions (main study, log scale). The recall-tuned detector proposes identifier candidates well above their true frequency (`US_BANK_ACCOUNT_NUMBER` at 78×, `URL` at 49×, `US_DRIVERS_LICENSE_NUMBER` at 32×), which a precision-weighted metric penalizes when a non-fine-tuned model passes them through uncalibrated. This integration effect is the residual that symmetric alignment does not remove.

and therefore output canonical types that score without remapping, while zero-shot outputs require a post-hoc alignment step. A study that reports results before label alignment would incorrectly conclude that tool use substantially improves PII detection. After alignment, the comparison is fair, and the direction of effect reverses (Figure 6). The parallel is sharp: just as superficial metrics in model editing can mask underlying failures and yield systematically misleading conclusions about a technique's success (Baser et al., 2026), comparing raw tool versus zero-shot scores without normalizing the label schema gives a false sense of tool superiority. This confound is not specific to this study's design; it is a general risk for any evaluation framework that mixes model-native and tool-canonicalized predictions without explicit normalization. Future benchmarking of agentic PII systems should therefore include a mandatory label-alignment pass and report pre- and post-alignment scores separately.

5.2 Why does tool use degrade aggregate F1?

The per-type analysis in Section 4.3 separates two contributions to the aggregate decline: one an evaluation artifact and one substantive.

The evaluation artifact removed by alignment. In the tool and skills conditions, the detector emits the canonical type `DATE` for temporal spans the benchmark records as `DATE_TIME`, and `ADDRESS` for spans it records as `LOCATION`. Under an alignment map that does not bridge these equivalent types, the predictions go unmatched, and recall on two of the benchmark's highest-frequency types collapses to near zero, an artifact

of the scoring schema, not of detection. The symmetric map (Section 3.5) bridges `DATE` ≡ `DATE_TIME` and `ADDRESS` ≡ `LOCATION`, and the collapse disappears: under the augmented conditions `DATE_TIME` recall becomes neutral-to-positive and `LOCATION` recovers (Table 6, Figure 3). The symmetric map cuts the mean zero-shot-over-tool gap by about 38% (from 0.169 to 0.104): a substantial part of the aggregate type-level degradation is this scoring artifact rather than a detection failure.

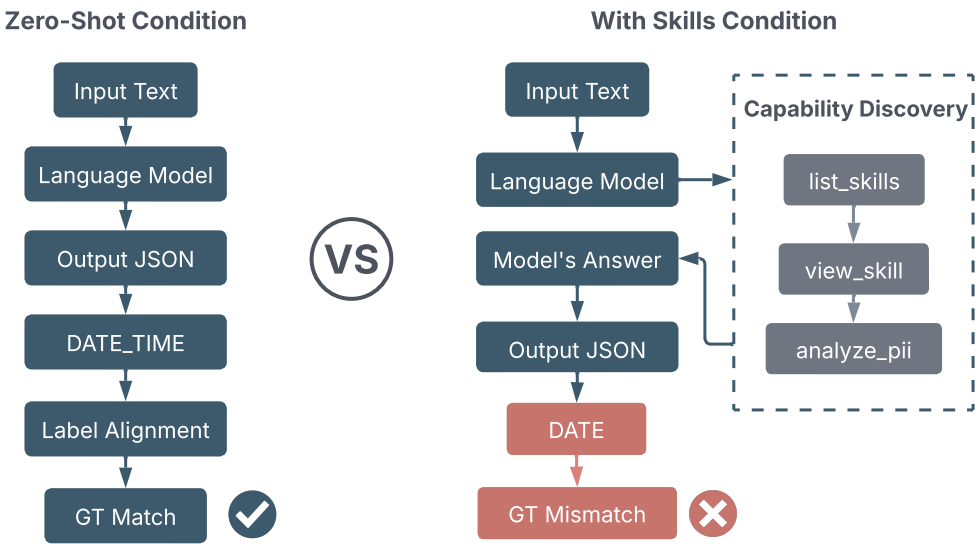

Figure 6: The labeling confound corrected by symmetric alignment: tool-augmented outputs are already canonical (e.g., PII-Codex `DATE`) but, under an asymmetric alignment, did not match the ground-truth type (e.g., `DATE_TIME`), so type equality failed and recall collapsed. The symmetric map bridges these equivalent types on both the prediction and ground-truth sides.

The substantive effect that remains. What remains after symmetric alignment is real, and it is fundamentally about *context*. The rule-based detector is tuned for high recall and proposes identifier labels well above their true frequency (Section 4.4); under an exact-match, precision-weighted metric, a non-fine-tuned model that passes those candidates through uncalibrated pays for the extra volume in precision. And for contextually ambiguous types, most clearly `HEALTH_INSURANCE_ID`, the determination of whether a span is sensitive depends on surrounding text that a pattern-based detector does not see, so the models, reasoning from context, achieve *higher* recall than PII-Codex does in isolation, and routing through the tool can discard those context-derived detections. This is consistent with PII-Codex's own framing, in which the sensitivity of a token is context- and use-dependent rather than a fixed property of its surface form (Rosado, 2023), and with subsequent work arguing that privacy-enhancing tools overlook the context-dependent nature of privacy risk when they assess identifiers in isolation rather than modeling the relationships among them (Zuo & Lee, 2026). As Section 4.1 shows, the standalone detector trails every model condition at the type level (0.28) yet leads at the span level: the comparison is a model–library interaction, and the question is how, not whether, to combine them.

This pattern mirrors knowledge-intensive settings where generated internal knowledge surpasses externally retrieved knowledge (Yu et al., 2023; Chen et al., 2023; Luo et al., 2025; Qian et al., 2025): external programmatic tools for PII lack the nuanced contextual understanding that neural representations possess, and the coherence of model-native output can outweigh the occasional benefit of rule-based canonicalization. The contrast aligns with the distinction between settings where tools are required (when models must integrate real-time or frequently updated information beyond their pretrained knowledge (Yu & Ji, 2024)) and settings where they are superfluous: PII detection is a static, pattern-recognition task for which parametric knowledge suffices, and injecting tool-based scaffolding introduces structural mismatch. Models often lack

reliable self-awareness of their knowledge boundaries (Yin et al., 2023), which may explain why they invoke tools even when tool output underperforms their own parametric knowledge.

Type-level versus span-level. Separating "finding the span" from "labeling it" is precisely what the two co-primary metrics capture (Section 4.1). At the span level, the detector's exact character offsets give the augmented conditions the advantage; at the type level, the model's cleaner type set wins. Reporting both, rather than a single strict canonical-to-canonical score, is what makes the degradation legible as part artifact and part substance, rather than a blanket claim that the tool is worse.

5.3 Documentation and the Llama exception

Three of four models show near-zero sensitivity to documentation injection ($|\Delta| \leq 0.01$, all non-significant). Llama 3.1 8B is a notable outlier, with a statistically significant -0.15 degradation under +Docs ($p < 0.0001$, $d = -0.37$). This is consistent with recent findings that a model's stronger baseline instruction-following ability can paradoxically degrade performance by making it overly sensitive to rigid or suboptimal formatting constraints in the prompt (Luo et al., 2025). Furthermore, injecting external documentation can create "merge conflicts" with a model's parametric knowledge (Qian et al., 2023). When given the PII-Codex documentation, Llama may attempt to strictly map its output to the documentation's label set rather than relying on its own more accurate internal representations.

5.4 Skill document reading does not explain performance

The skill-viewed subgroup analysis (Section 4.5) together with the null +Skills-vs-+Tool comparison (Table 5) indicates that the Skill document evaluated here adds no value beyond tool access. This conclusion is scoped to the specific ∼250-token document tested rather than to skill documents in general; a richer document with worked examples and type disambiguation might behave differently (see Limitations), and the subgroup contrast is available only for Mistral and Qwen. Within that scope, models that read the document do not outperform models that do not. The document's failure to help may reflect a context competition effect: the Skill document consumes context capacity that could otherwise be used for the model's own chain-of-thought or attention to the sample text; dense instructions and tool outputs in the context window trigger context competition, degrading a model's ability to extract and use relevant information (Liu et al., 2024; Luo et al., 2025; Xu et al., 2025; Shi et al., 2023). The +Skills condition also imposes a longer tool trajectory than +Tool, since `list_skills` and `view_skill` precede `analyze_pii`. Each additional step is another opportunity for a formatting or generation error in a 7–9B model, so part of any +Skills penalty may reflect this interaction overhead rather than the document's content.

5.5 Implications for agentic NLP pipeline design

The results suggest a broader design principle: agent scaffolding imposes structural constraints that can conflict with model representations. Varying prompt phrasings or rigid output constraints lead to divergent implementations and metrics, undermining the reliability of agent evaluation (Sun et al., 2025). These conclusions concern *non-fine-tuned* tool calling: prior work shows that 7–9B base models require domain-specific fine-tuning for reliable API use (Patil et al., 2024), so this finding speaks to the default, out-of-the-box configuration practitioners are most likely to deploy, not to tool use after fine-tuning.

1. **Do not assume tool use improves structured extraction.** Consistent with tool-use literature (Luo et al., 2025; Qian et al., 2025; Xu et al., 2025), tool-augmented pipelines in the 7–9B parameter class were found to add latency and infrastructure complexity while reducing accuracy relative to zero-shot prompting with label normalization. The benefit of tool use is type-specific (structured identifiers) rather than global.

2. **Label alignment is a first-class evaluation concern.** Any mixed-mode pipeline (some predictions from tool output, some from model output) must normalize labels before scoring. Omitting this step will produce systematically misleading comparisons.

3. **The effect is a pipeline property, not a capability or scale property.** One might attribute the degradation to model capability within the 7–9B class; the control experiments do not support that reading: the 14B model degrades by essentially the same margin (Section 5.6), so the gap is comparable one model generation up, and the rank ordering of degradation among the four 7–9B models is weak enough that it is not interpreted mechanistically. The effect is instead located in the pipeline (label schema, evaluation granularity, and the model–library interface), which is invariant to the capability differences observable here. Consistent with evidence that rigidly constraining output format can harm performance (Tam et al., 2024; Luo et al., 2025), agent scaffolding can conflict with a model's learned representations; but on the present evidence, this reflects the interface and evaluation, not model size (Sun et al., 2025).

5.6 Robustness: scale and precision

Two checks probe whether the pattern is an artifact of quantization or limited to the smallest models, and a third addresses generation stochasticity.

Scale. Re-running the full ablation on Qwen2.5-14B (4-bit) over the same 2,000 samples leaves the pattern essentially unchanged from the 7–9B average: at the type level zero-shot still leads +Tool by a comparable margin (+0.09 versus a 7–9B average of +0.10), and at the span level the detector still leads by a comparable margin (−0.14 versus −0.13). The tool executed in 99.9% of these runs (3,997/4,000), so the gap persists despite near-perfect execution: a larger model that invokes the tool more reliably does not make augmentation begin to help (Figure 7; full condition-by-condition values in Appendix E, Table 15). This scale check is a single model one generation above the studied class; broader sweeps (e.g., 32B and above) and other model families remain future work.

Precision. Re-running the four models unquantized (fp16/bf16) on a stratified 300-sample subset, compared per model against 4-bit on the identical samples, shows unquantized F1 within 0.006 ± 0.006 (type level) and 0.004 ± 0.004 (span level) of 4-bit, where each value is the mean per-sample fp16−4-bit F1 difference with its 95% CI; the largest single-model shift is +0.028 (Llama), an order of magnitude below the augmentation effects and changing no ordering. Quantization is therefore not responsible for the observed pattern (Figure 8; per-model values in Appendix E, Table 16).

Decoding. Because decoding is greedy (Section 3.6), model outputs are deterministic given a prompt; the reported effects are not subject to run-to-run generation variance, and a seed sweep is not applicable.

5.7 Limitations

This study is scoped to English-language text and US-locale PII types, so the findings may not transfer to other languages or international PII schemas. The benchmark covers 21 PII types from three sources, with limited coverage of rare types such as `ABA_ROUTING_NUMBER` and `LICENSE_PLATE_NUMBER`. The robustness checks are deliberately narrow. The primary runs use 4-bit quantization in the 7–9B class. The pattern holds without quantization (four models) and for a single 14B model (Qwen2.5-14B), but broader scale sweeps (32B and above) and additional model families remain future work, as do fine-tuned models. Two design choices bound how the tool result should be read. First, the Skill document is a single, deliberately minimal (∼250-token) pointer to the existing tool, compressed to limit context overload in small models (Luo et al., 2025; Xu et al., 2025). A richer document with worked examples and type disambiguation may behave differently. Second, the comparison is *non-fine-tuned*, out-of-the-box tool calling: foundational work shows that 7–9B base models generally fail to execute rigid API calls reliably without targeted fine-tuning (Patil et al., 2024; Qin et al., 2024), so the degradation reflects the default configuration practitioners are most likely to deploy rather than an indictment of tool use after fine-tuning. The low skill-document engagement rates for Qwen (17.1%) and Mistral (47.4%) are consistent with this. The agent also uses a lenient tool-call detection strategy. A stricter enforcement of tool use before accepting answers (available but not used in the default runner) may produce different results. The result is therefore best read as specific to this model class, skill construction, and task configuration.

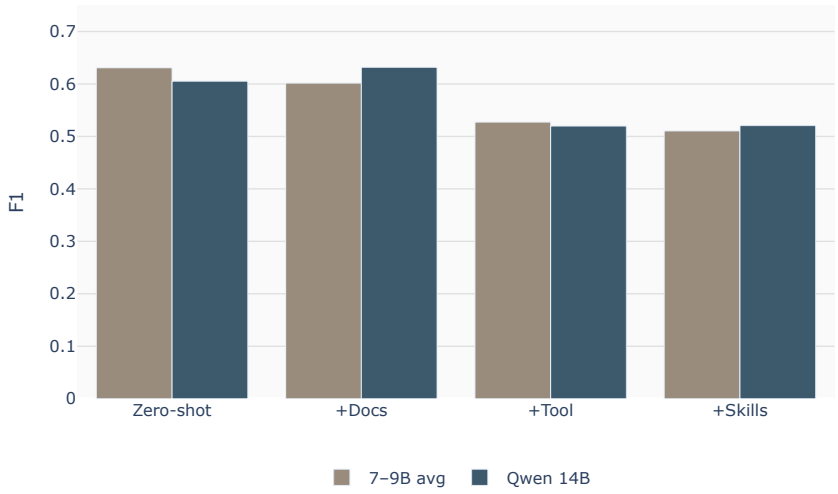

Figure 7: Type-level F1 by configuration for the 7–9B average versus Qwen2.5-14B (main study, symmetric alignment). The augmentation pattern is unchanged at 14B.

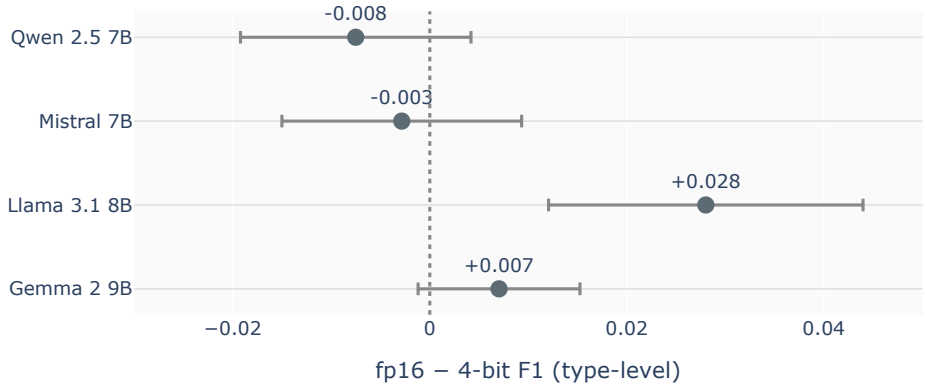

Figure 8: Per-model type-level F1 difference (unquantized minus 4-bit) on the shared 300-sample subset. Each point is a model's mean difference, and the whiskers are 95% confidence intervals; the dotted line marks zero. All four differences lie within 0.03 F1 of zero (Llama's +0.028 is the largest, an order of magnitude below the augmentation effects), so quantization does not account for the observed pattern.

6 Conclusion

A controlled ablation evaluated documentation injection, tool access, and Agent Skills documents for PII detection across four open-weight SLMs, scored under a symmetric label alignment and two co-primary metrics. The primary finding is that whether augmentation helps depends on evaluation granularity: at the entity-type level, zero-shot prompting outperforms both tool-augmented conditions for all models in the 7–9B class (tool and skills reduce mean F1 by 6–17 percentage points), while at the span level, the detector's exact character offsets reverse the ranking. Part of the aggregate type-level degradation is an evaluation artifact (temporal and geographic types collapse only under an asymmetric alignment and recover under a fair, symmetric one), and part is substantive and context-dependent: the recall-tuned detector proposes

identifier labels well above their true frequency (which a precision-weighted metric penalizes when passed uncalibrated through a non-fine-tuned model) and, lacking surrounding context, has lower recall than the models on contextually determined types, so its value is type-selective and complementary to the model rather than global. Run standalone on the type-level objective, that detector scores below every model condition while leading at the span level, so +Tool reflects a model–library interaction rather than the library alone. Because the effect persists for a 14B model and without quantization, it is attributed to the evaluation pipeline and the non-fine-tuned model–library interface rather than to a capability limit specific to the smallest models, while noting the scale check is a single larger model (Limitations).

The skill-viewed subgroup analysis provides additional evidence that Skill document injection has no measurable incremental value, even for models that choose to read the document. Documentation injection is neutral for most models but degrades Llama 3.1 8B, suggesting model-specific sensitivity to context that conflicts with pre-trained label representations. Few-shot and chain-of-thought prompting, the simplest parametric baselines, likewise fail to exceed zero-shot, and PII-Codex runs standalone trials for every model condition, confirming that zero-shot is a genuine ceiling rather than a weak comparison.

What the study contributes is as much about how detection is measured as about what was found: in mixed-mode pipelines that combine model-native and tool-canonicalized predictions, label alignment and evaluation granularity mediate whether an intervention looks beneficial, and naive scoring can invert the conclusion in either direction. As in other domains, superficial evaluations obscure true capabilities: evaluation design, schema canonicalization, and ontology alignment fundamentally mediate how the community perceives model performance (Baser et al., 2026). Toolformer demonstrated that a 6.7B model can outperform larger models on tool use when *fine-tuned* to predict tool calls natively (Schick et al., 2023). Out-of-the-box zero-shot agentic scaffolding is a poor fit for 7–9B models unless they are fine-tuned (Patil et al., 2024). These results challenge a common assumption in agentic system design and make the case for rigorous label alignment and granularity-aware reporting in mixed-mode evaluation pipelines. Future work should examine richer skill documents and international PII types beyond the US-English scope of this study. The robustness checks here already extend the findings to one model generation above the studied class and to unquantized weights.

Broader Impact Statement

This work evaluates the effect of agent augmentation on PII detection accuracy in small language models. The primary practical implication is cautionary: practitioners deploying agentica scaffolding for sensitive data tasks should validate augmentation decisions empirically rather than assuming performance gains. At the same time, the opposite overreaction warrants equal caution. Much of the apparent degradation observed here is an evaluation and label-alignment artifact rather than evidence that tools are inherently worse at finding PII, and the residual reflects one rule-based library's behavior under a non-fine-tuned interface. Reporting "tool use degrades PII detection" without this context could lead deployers to abandon tool or skill scaffolding for the wrong reason, or to under-detect certain PII types in privacy-compliance settings. Practitioners should therefore neither assume agentic scaffolding helps nor abandon it wholesale, but evaluate it with symmetric label alignment and granularity-aware metrics on their own pipeline; uncritical adoption of tool-augmented pipelines could reduce accuracy relative to simpler zero-shot approaches, with downstream consequences for privacy compliance and data protection in production systems. The benchmark, scored results, and analysis notebooks are publicly released to support reproducibility and to enable the community to verify, extend, or challenge these findings. All datasets used are synthetic or publicly available; no real personal data was used or exposed in this research.

Data and Code Availability

Benchmark ground truth is available at [redacted]. Scored predictions and per-sample scores for all conditions, baselines, and robustness controls are available at [redacted]. The analysis notebooks (under `notebooks/`) and the full project repository are available at [redacted].

Acknowledgments

The author used Claude Opus 4.8 (Anthropic) to assist with notebook restructuring, code refinement, and LaTeX manuscript preparation. All experimental design, data collection, analysis, and scientific conclusions are the author's own.

References

Ai4Privacy. pii-masking-300k (revision 86db63b), 2024. URL https://huggingface.co/datasets/ai4privacy/pii-masking-300k.

Salaheddin Alzubi, Noah Provenzano, Jaydon Bingham, Weiyuan Chen, and Tu Vu. EvoSkill: Automated skill discovery for multi-agent systems, 2026.

Manit Baser, Dinil Mon Divakaran, and Mohan Gurusamy. Thinkeval: Practical evaluation of knowledge leakage in LLM editing using thought-based knowledge graphs. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856.

Peter Belcak, Greg Heinrich, Shizhe Diao, Yonggan Fu, Xin Dong, Saurav Muralidharan, Yingyan Celine Lin, and Pavlo Molchanov. Small language models are the future of agentic ai, 2025. URL https://arxiv.org/abs/2506.02153.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

Harrison Chase. Langchain, 2022. URL https://github.com/langchain-ai/langchain. Computer software.

Liang Chen, Yang Deng, Yatao Bian, Zeyu Qin, Bingzhe Wu, Tat-Seng Chua, and Kam-Fai Wong. Beyond factuality: A comprehensive evaluation of large language models as knowledge generators. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 6325–6341, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.390. URL https://aclanthology.org/2023.emnlp-main.390/.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, 2019. Association for Computational Linguistics.

Gemma Team. Gemma: Open models based on gemini research and technology, 2024. URL https://arxiv.org/abs/2403.08295.

Aaron Grattafiori et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Gretel AI. gretel-pii-masking-en-v1: Synthetic PII masking dataset (english), 2024. URL https://huggingface.co/datasets/gretelai/gretel-pii-masking-en-v1.

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

Polaris Jhandi, Owais Kazi, Shreyas Subramanian, and Neel Sendas. Small language models for efficient agentic tool calling: Outperforming large models with targeted fine-tuning, 2025. URL https://arxiv.org/abs/2512.15943.

Albert Q. Jiang et al. Mistral 7b, 2023. URL https://arxiv.org/abs/2310.06825.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 260–270, San Diego, California, 2016. Association for Computational Linguistics.

LangChain. LangGraph: Agent orchestration framework, 2024. URL `https://github.com/langchain-ai/langgraph`.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Kuttler, Mike Lewis, Wen-tau Yih, Tim Rocktaschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pp. 9459–9474, 2020.

Boning Li, Baoxiang Wang, and Longbo Huang. PokerSkill: LLMs can play expert-level poker without training or solvers, 2026.

Pierre Lison, Ildikó Pilán, David Sanchez, Montserrat Batet, and Lilja Øvrelid. Anonymisation models for text data: State of the art, challenges and future directions. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 4188–4203, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.323. URL `https://aclanthology.org/2021.acl-long.323/`.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024.

Nils Lukas, Ahmed Salem, Richard Sim, Shruti Tople, Lukas Wutschitz, and Santiago Zanella-Beguelin. Analyzing leakage of personally identifiable information in language models. In *2023 IEEE Symposium on Security and Privacy (S&P)*, pp. 346–363. IEEE, 2023.

Ne Luo, Aryo Pradipta Gema, Xuanli He, Emile Van Krieken, Pietro Lesci, and Pasquale Minervini. Self-training large language models for tool-use without demonstrations. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 1253–1271, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-195-7. doi: 10.18653/v1/2025.findings-naacl.69. URL `https://aclanthology.org/2025.findings-naacl.69/`.

Omri Mendels, Coby Peled, Nava Vaisman Levy, Tomer Rosenthal, Limor Lahiani, et al. Microsoft presidio: Context aware, pluggable and customizable pii anonymization service for text and images. *Microsoft*, 2018.

George R. Milne, George Pettinico, Fatima M. Hajjat, and Ereni Markos. Information sensitivity typology: Mapping the degree and type of risk consumers perceive in personal data sharing. *Journal of Consumer Affairs*, 51(1):133–161, Jun 2016. doi: 10.1111/joca.12111.

Kangyun Ning, Yisong Su, Xueqiang Lv, Yuanzhe Zhang, Jian Liu, Kang Liu, and Jinan Xu. Wtu-eval: a whether-or-not tool usage evaluation benchmark for large language models. *arXiv preprint arXiv:2407.12823*, 2024.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: large language model connected with massive apis. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA, 2024. Curran Associates Inc. ISBN 9798331314385.

Cheng Qian, Xinran Zhao, and Sherry Tongshuang Wu. "merge conflicts!" exploring the impacts of external distractors to parametric knowledge graphs. *arXiv preprint arXiv:2309.08594*, 2023.

Cheng Qian, Emre Can Acikgoz, Hongru Wang, Xiusi Chen, Avirup Sil, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. SMART: Self-aware agent for tool overuse mitigation. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 4604–4621, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.239. URL https://aclanthology.org/2025.findings-acl.239/.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xiangru Cong, Xiang Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. Toolllm: Facilitating large language models to master 16000+ real-world APIs. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/28e50ee5b72e90b50e7196fde8ea260e-Paper-Conference.pdf.

Eidan J. Rosado. Pii-codex: a python library for pii detection, categorization, and severity assessment. *Journal of Open Source Software*, 8(86):5402, 2023. doi: 10.21105/joss.05402. URL https://doi.org/10.21105/joss.05402.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, volume 36, pp. 68539–68551, 2023.

Paul M Schwartz and Daniel J Solove. The pii problem: Privacy and a new concept of personally identifiable information. *New York University Law Review*, 86:1814–2011, 2011.

Freda Shi, Zheng Chen, Qiang Zhao, Chiyuan Zhang, et al. Large language models can be easily distracted by irrelevant context. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 31210–31227. PMLR, 2023.

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.

Maojun Sun, Ruijian Han, Binyan Jiang, Houduo Qi, Defeng Sun, Yancheng Yuan, and Jian Huang. A survey on large language model-based agents for statistics and data science. *The American Statistician*, pp. 1–14, 2025. doi: 10.1080/00031305.2025.2561140. URL https://doi.org/10.1080/00031305.2025.2561140.

Zhi Rui Tam, Cheng-Kuang Wu, Yi-Lin Tsai, Chieh-Yen Lin, Hung-yi Lee, and Yun-Nung Chen. Let me speak freely? a study on the impact of format restrictions on large language model performance. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pp. 1218–1236. Association for Computational Linguistics, 2024.

Shuhe Wang, Xiaofei Sun, Xiaoya Li, Rongbin Ouyang, Fei Wu, Tianwei Zhang, Jiwei Li, Guoyin Wang, and Chen Guo. GPT-NER: Named entity recognition via large language models. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 4257–4275, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-195-7. doi: 10.18653/v1/2025.findings-naacl.239. URL https://aclanthology.org/2025.findings-naacl.239/.

Weikai Xu, Chengrui Huang, Shen Gao, and Shuo Shang. Llm-based agents for tool learning: A survey. *Data Science and Engineering*, 10:533–563, 06 2025. doi: 10.1007/s41019-025-00296-9.

An Yang et al. Qwen2 technical report, 2024. URL https://arxiv.org/abs/2407.10671.

Yifan Yang, Ziyang Gong, Weiquan Huang, Qihao Yang, Ziwei Zhou, Zisu Huang, Yan Li, Xuemei Gao, Qi Dai, Bei Liu, Kai Qiu, Yuqing Yang, Dongdong Chen, Xue Yang, and Chong Luo. SkillOpt: Executive strategy for self-evolving agent skills, 2026.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Zhangyue Yin, Qiushi Sun, Qipeng Guo, Jiawen Wu, Xipeng Qiu, and Xuan-Jing Huang. Do large language models know what they don't know? In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 8653–8665, 2023.

Fei Yu, Hongbo Zhang, Prayag Tiwari, and Benyou Wang. Natural language reasoning, a survey. *ACM Computing Surveys*, 56(12):1–39, 2024.

Pengfei Yu and Heng Ji. Information association for language model updating by mitigating lm-logical discrepancy. In *Proceedings of the 28th Conference on Computational Natural Language Learning*, pp. 117–129, Miami, FL, USA, 2024. Association for Computational Linguistics.

Wenhao Yu, Dan Iter, Shuohang Wang, Yichong Xu, Mingxuan Ju, Soumya Sanyal, Chenguang Zhu, Michael Zeng, and Meng Jiang. Generate rather than retrieve: Large language models are strong context generators. In *International Conference on Learning Representations (ICLR)*, 2023.

Jinhui Zuo and Seok-Won Lee. Addressing memorization and aggregation risks in AI: A knowledge graph approach to privacy. *Applied Sciences*, 16(4):1796, 2026. doi: 10.3390/app16041796.

A Benchmark Details

A.1 Full PII Type Distribution

Table 7 presents the complete ground-truth entity count for all PII types present in the main study benchmark (N = 2,000 samples). Counts are computed from the benchmark's `pii_codex_ground_truth` field.

Table 7: Full ground-truth PII entity counts by type (main study, N = 2,000). Each count is the number of entity spans of that type in the benchmark; total matches Table 1.

PII Type	Entity count
PERSON	2,324
DATE_TIME	1,572
LOCATION	1,068
ADDRESS	862
EMAIL_ADDRESS	632
US_SOCIAL_SECURITY_NUMBER	632
PHONE_NUMBER	566
DATE	531
IP_ADDRESS	515
US_DRIVERS_LICENSE_NUMBER	383
GENDER	365
HEALTH_INSURANCE_ID	343
ZIPCODE	316
US_PASSPORT_NUMBER	379
CREDIT_CARD_NUMBER	137
LICENSE_PLATE_NUMBER	175
PASSWORD	127
OCCUPATION	102
URL	92
ABA_ROUTING_NUMBER	62
US_BANK_ACCOUNT_NUMBER	44
NRP	28
MEDICAL_LICENSE	30
RACE	18
AGE	18
SWIFT_CODE	17
MAC_ADDRESS	14
US_INDIVIDUAL_TAXPAYER_IDENTIFICATION	11
SEXUAL_PREFERENCE	7
Total (entity span count)	**11,370**

A.2 Labels Excluded from Scoring

The following 27 prediction labels appeared in model output but could not be unambiguously mapped to a PII-Codex canonical type and were excluded from scoring. They are listed here for transparency. `COUNTRY_OF_ORIGIN` was mapped to `LOCATION` and is not in this list.

```
ATTRIBUTE_NAME, ATTRIBUTE_VALUE, CLIENT_ID, COMMENTS, ID, IDENTIFIABLE,
IDENTIFICATION_NUMBER, IDENTIFIER, ID_NUMBER, INDIVIDUAL_ID,
MEDICALCONDITION, NUMBER, PARENT_ID, PARTY, PERSONAL_IDENTIFIER,
PERSON_ID, PII, PII_TYPE, PSYCHOLOGICAL_HISTORY, ROLE, SIBLING_ID,
SUBSTANCEUSEDISORDERHISTORY, TEXT, TRAUMAHISTORY, US_ID, US_ID_NUMBER,
VEHICLE_ID
```

Labels such as `ATTRIBUTE_NAME`, `COMMENTS`, `ROLE`, and `TEXT` do not appear in any of the three source datasets and are model-output artifacts rather than true PII categories. `VEHICLE_ID` was considered for mapping to `LICENSE_PLATE_NUMBER` but was held for manual review and not included in the canonical mapping.

B Pilot Study Results

The pilot (n = 200) was run on the same hardware and under the same conditions as the main study, using a stratified subsample of the full benchmark. It served primarily to validate the pipeline and inform the main study configuration (parallelism, context bounds, timeouts). Pilot results are presented here as a citable record; formal conclusions are drawn from the main study only.

B.1 Pilot Summary Table

Table 8: Pilot study summary (n = 200, post-label alignment). f1_med = median F1; elapsed_μ = mean seconds per sample; turns_μ = mean conversation turns.

Model	Condition	f1_μ	prec_μ	rec_μ	f1_med	skill_viewed	elapsed_μ	turns_μ
Gemma 2 9B	with_docs	0.557	0.710	0.493	0.571	0	9.75	1.00
	with_skills	0.391	0.454	0.388	0.400	200	11.81	4.00
	with_tools	0.392	0.455	0.387	0.400	0	8.22	2.00
	zero_shot	0.530	0.683	0.466	0.571	0	7.60	1.00
Llama 3.1 8B	with_docs	0.412	0.446	0.419	0.453	0	8.56	1.00
	with_skills	0.384	0.419	0.409	0.400	200	15.74	3.93
	with_tools	0.433	0.467	0.456	0.437	0	14.91	2.00
	zero_shot	0.598	0.620	0.613	0.667	0	3.35	1.00
Mistral 7B	with_docs	0.503	0.629	0.446	0.500	0	4.92	1.00
	with_skills	0.434	0.534	0.406	0.400	108	9.63	2.02
	with_tools	0.434	0.534	0.406	0.400	0	10.94	2.00
	zero_shot	0.497	0.628	0.438	0.571	0	3.04	1.00
Qwen 2.5 7B	with_docs	0.543	0.646	0.500	0.571	0	4.78	1.00
	with_skills	0.394	0.543	0.361	0.400	43	12.99	2.47
	with_tools	0.392	0.461	0.391	0.400	0	7.28	2.02
	zero_shot	0.556	0.634	0.528	0.608	0	3.01	1.00

B.2 Pilot Degradation Tabular Heatmap

The pilot directional pattern is consistent with the main study: tool and skills conditions underperform zero-shot for all models, and documentation is near-neutral.

Table 9: Pilot mean F1 delta vs. zero-shot by model and condition.

Model	**Δ+Docs**	**Δ+Tool**	**Δ+Skills**
Gemma 2 9B	+**0.028**	−0.138	−0.139
Llama 3.1 8B	−0.186	−0.165	−0.214
Mistral 7B	+**0.005**	−0.063	−0.063
Qwen 2.5 7B	−0.013	−0.164	−0.162

C Full Statistical Tables

C.1 All Pairwise Comparisons

Table 10 extends Table 5 in the main text to include all six pairwise comparisons per model. Cohen's d and paired t and p are reported for all comparisons.

Table 10: Full pairwise delta summary (main study, N = 2,000 per model). % imp = percentage of samples where the condition outperforms the comparator.

Model	**Comparison**	t	p	d	**% improved**
Llama 3.1 8B	+Docs vs ZS	−16.50	< 0.0001	−0.369	24.4%
	+Tool vs ZS	−14.27	< 0.0001	−0.319	25.2%
	+Skills vs ZS	−21.84	< 0.0001	−0.488	22.0%
	+Tool vs +Docs	+4.66	< 0.0001	+0.104	41.8%
	+Skills vs +Docs	−2.56	0.011	−0.057	35.8%
	+Skills vs +Tool	−12.90	< 0.0001	−0.288	11.4%
Mistral 7B	+Docs vs ZS	+1.35	0.179	+0.030	33.0%
	+Tool vs ZS	−7.08	< 0.0001	−0.158	35.2%
	+Skills vs ZS	−7.14	< 0.0001	−0.160	35.2%
	+Tool vs +Docs	−8.21	< 0.0001	−0.184	33.5%
	+Skills vs +Docs	−8.26	< 0.0001	−0.185	33.1%
	+Skills vs +Tool	−0.75	0.453	−0.017	1.6%
Qwen 2.5 7B	+Docs vs ZS	+1.68	0.092	+0.038	30.8%
	+Tool vs ZS	−18.05	< 0.0001	−0.404	23.9%
	+Skills vs ZS	−18.66	< 0.0001	−0.417	23.6%
	+Tool vs +Docs	−21.41	< 0.0001	−0.479	22.8%
	+Skills vs +Docs	−22.87	< 0.0001	−0.511	21.9%
	+Skills vs +Tool	−0.68	0.496	−0.015	26.1%
Gemma 2 9B	+Docs vs ZS	+1.81	0.071	+0.040	29.9%
	+Tool vs ZS	−11.97	< 0.0001	−0.268	29.4%
	+Skills vs ZS	−11.87	< 0.0001	−0.266	29.4%
	+Tool vs +Docs	−14.81	< 0.0001	−0.331	26.9%
	+Skills vs +Docs	−14.72	< 0.0001	−0.329	27.1%
	+Skills vs +Tool	+1.79	0.074	+0.040	1.2%

C.2 Parametric baselines (few-shot, chain-of-thought)

Table 11 reports the paired comparison of the few-shot and chain-of-thought baselines against zero-shot at the type level. Per model, the effect is mixed; averaged across the four models, neither parametric baseline exceeds zero-shot (Section 4.1).

Table 11: Few-shot and chain-of-thought baselines versus zero-shot (main study, $N = 2{,}000$ per model, type-level). % improved = percentage of samples where the baseline outperforms zero-shot.

Model	Comparison	t	p	d	% improved
Llama 3.1 8B	Few-shot vs ZS	-8.91	< 0.0001	-0.199	27.0%
	CoT vs ZS	-13.29	< 0.0001	-0.297	24.1%
Mistral 7B	Few-shot vs ZS	$+4.61$	< 0.0001	$+0.103$	41.2%
	CoT vs ZS	-2.73	0.006	-0.061	29.9%
Qwen 2.5 7B	Few-shot vs ZS	-0.09	0.929	-0.002	31.4%
	CoT vs ZS	-4.30	< 0.0001	-0.096	29.6%
Gemma 2 9B	Few-shot vs ZS	$+4.26$	< 0.0001	$+0.095$	31.2%
	CoT vs ZS	$+8.52$	< 0.0001	$+0.191$	36.5%

Table 12 reports the four-model-average F1 of the baselines under both co-primary metrics (the source of the values quoted in Section 4.1), with zero-shot as reference.

Table 12: Four-model-average F1 for the parametric baselines under both co-primary metrics (main study, symmetric alignment), with zero-shot as reference. Neither baseline exceeds zero-shot at either metric.

Condition	Type-level F1	Span-level F1
Zero-shot	0.631	0.325
Few-shot	0.628	0.319
Chain-of-thought	0.606	0.298

C.3 Confidence Interval Analysis

Table 13: Bootstrap 95% CI analysis (main study, target margin ±0.05). All cells sufficient at N = 2,000. req_n = minimum n for target margin.

Model	Condition	n	f1_μ	std	CI low	CI high	req_n
Llama 3.1 8B	with_docs	2000	0.526	0.356	0.511	0.542	195
	with_skills	2000	0.504	0.258	0.493	0.515	103
	with_tools	2000	0.567	0.259	0.556	0.579	104
	zero_shot	2000	0.675	0.308	0.662	0.689	146
Mistral 7B	with_docs	2000	0.592	0.307	0.579	0.606	145
	with_skills	2000	0.528	0.263	0.516	0.539	107
	with_tools	2000	0.528	0.263	0.517	0.540	107
	zero_shot	2000	0.584	0.309	0.570	0.597	148
Qwen 2.5 7B	with_docs	2000	0.652	0.278	0.640	0.665	119
	with_skills	2000	0.484	0.273	0.472	0.496	115
	with_tools	2000	0.488	0.280	0.476	0.500	121
	zero_shot	2000	0.640	0.322	0.626	0.655	160
Gemma 2 9B	with_docs	2000	0.635	0.281	0.623	0.647	122
	with_skills	2000	0.526	0.252	0.515	0.537	98
	with_tools	2000	0.525	0.253	0.514	0.536	99
	zero_shot	2000	0.624	0.313	0.610	0.638	151

D Per-PII-Type Recall Tables

Table 14 presents recall by PII type and condition for all four models across the ten most frequent types (main study). Single-turn conditions (ZS, +Docs, few-shot, CoT) and multi-turn (+Tool, +Skills) are shown together; models are grouped within the table.

Table 14: Recall by PII type and model, all conditions and parametric baselines (main study).

Model	PII Type	ZS	+Docs	+Tool	+Skills	few-shot	CoT
Llama 3.1 8B	DATE_TIME	0.70	0.58	0.75	0.73	0.75	0.73
	EMAIL_ADDRESS	0.94	0.77	0.97	0.94	0.88	0.76
	HEALTH_INSURANCE_ID	0.73	0.17	0.08	0.01	0.52	0.36
	IP_ADDRESS	0.92	0.56	0.96	0.90	0.83	0.69
	LOCATION	0.53	0.46	0.47	0.41	0.60	0.56
	PERSON	0.65	0.47	0.48	0.40	0.63	0.67
	PHONE_NUMBER	0.85	0.72	0.72	0.63	0.82	0.62
	US_DRIVERS_LICENSE_NUMBER	0.48	0.13	0.81	0.80	0.43	0.31
	US_PASSPORT_NUMBER	0.64	0.41	0.72	0.67	0.52	0.44
	US_SOCIAL_SECURITY_NUMBER	0.76	0.62	0.63	0.56	0.60	0.52
Mistral 7B	DATE_TIME	0.40	0.43	0.68	0.68	0.64	0.49
	EMAIL_ADDRESS	0.93	0.90	0.97	0.97	0.91	0.90
	HEALTH_INSURANCE_ID	0.63	0.30	0.01	0.01	0.26	0.43
	IP_ADDRESS	0.82	0.82	0.96	0.96	0.80	0.70
	LOCATION	0.30	0.50	0.39	0.38	0.58	0.38
	PERSON	0.46	0.53	0.40	0.40	0.58	0.58
	PHONE_NUMBER	0.85	0.87	0.62	0.62	0.85	0.78
	US_DRIVERS_LICENSE_NUMBER	0.46	0.34	0.69	0.68	0.34	0.39
	US_PASSPORT_NUMBER	0.57	0.59	0.54	0.54	0.58	0.52
	US_SOCIAL_SECURITY_NUMBER	0.79	0.70	0.52	0.52	0.69	0.69
Qwen 2.5 7B	DATE_TIME	0.58	0.68	0.68	0.59	0.78	0.57
	EMAIL_ADDRESS	0.96	0.97	0.96	0.97	0.95	0.94
	HEALTH_INSURANCE_ID	0.74	0.15	0.02	0.00	0.55	0.46
	IP_ADDRESS	0.82	0.89	0.94	0.87	0.72	0.81
	LOCATION	0.56	0.67	0.40	0.35	0.64	0.52
	PERSON	0.54	0.51	0.38	0.35	0.63	0.63
	PHONE_NUMBER	0.92	0.90	0.64	0.56	0.91	0.81
	US_DRIVERS_LICENSE_NUMBER	0.53	0.16	0.78	0.74	0.08	0.51
	US_PASSPORT_NUMBER	0.65	0.72	0.66	0.75	0.53	0.60
	US_SOCIAL_SECURITY_NUMBER	0.76	0.85	0.55	0.53	0.64	0.69
Gemma 2 9B	DATE_TIME	0.39	0.53	0.71	0.71	0.62	0.64
	EMAIL_ADDRESS	0.96	0.97	0.98	0.98	0.94	0.95
	HEALTH_INSURANCE_ID	0.72	0.10	0.01	0.01	0.64	0.55
	IP_ADDRESS	0.79	0.84	0.97	0.97	0.73	0.80
	LOCATION	0.37	0.51	0.40	0.40	0.47	0.55
	PERSON	0.62	0.62	0.42	0.42	0.60	0.69
	PHONE_NUMBER	0.90	0.94	0.66	0.66	0.87	0.87
	US_DRIVERS_LICENSE_NUMBER	0.56	0.17	0.79	0.79	0.51	0.48
	US_PASSPORT_NUMBER	0.68	0.66	0.60	0.61	0.62	0.66
	US_SOCIAL_SECURITY_NUMBER	0.69	0.72	0.55	0.56	0.67	0.67

E Robustness Control Tables

Tables 15 and 16 give the full underlying numbers for the two robustness controls summarized in Section 5.6: re-running the ablation on a larger (14B) model, and re-running it unquantized.

Table 15: Scale control: Qwen2.5-14B (4-bit) versus the 7–9B four-model average, by condition, over the same 2,000 samples and symmetric alignment. Δ is 14B minus the 7–9B average. The type-level zero-shot lead over +Tool is +0.085 at 14B (versus +0.104 for the 7–9B, average) and the span-level detector lead is 0.144 (versus 0.130), so the pattern is essentially unchanged one model generation up.

	Type-level			**Span-level (IoU $\geq$ 0.5)**		
Condition	**7–9B**	**14B**	**Δ**	**7–9B**	**14B**	**Δ**
Zero-shot	0.631	0.605	−0.026	0.325	0.322	−0.003
+Docs	0.602	0.632	+0.030	0.301	0.319	+0.018
+Tool	0.527	0.520	−0.007	0.455	0.466	+0.011
+Skills	0.511	0.521	+0.010	0.448	0.467	+0.019

Table 16: Precision control: unquantized (fp16/bf16) versus 4-bit, per model, on the shared stratified 300-sample subset scored on identical samples (n = 1,200 sample-condition pairs per model). Δ is 16-bit minus 4-bit with its 95% CI. Pooled across models, the mean difference is 0.006 ± 0.006 (type) and 0.004 ± 0.004 (span); the largest single-model shift is +0.028 (Llama), an order of magnitude below the augmentation effects and changing no ordering.

	Type-level			**Span-level (IoU $\geq$ 0.5)**		
Model	**4-bit**	**16-bit**	**Δ**	**4-bit**	**16-bit**	**Δ**
Gemma 2 9B	0.567	0.574	$+0.007 \pm 0.008$	0.278	0.279	$+0.001 \pm 0.006$
Llama 3.1 8B	0.562	0.590	$+0.028 \pm 0.016$	0.274	0.284	$+0.010 \pm 0.008$
Mistral 7B	0.530	0.527	-0.003 ± 0.012	0.264	0.264	0.000 ± 0.008
Qwen 2.5 7B	0.565	0.557	-0.008 ± 0.012	0.271	0.277	$+0.007 \pm 0.008$

F Skill-Viewed Subgroup Analysis

F.1 Group sizes and mean F1

Table 17: Skill-viewed vs. not-viewed mean F1 (with_skills condition, main study). Llama and Gemma had 100% view rates; no not-viewed group is available for those models. Mean F1 difference is viewed minus not-viewed at the type level.

Model	**Viewed n**	**Not-viewed n**	**View rate**	**Mean F1 difference**
Mistral 7B	948	1,052	47.4%	+0.011
Qwen 2.5 7B	341	1,659	17.1%	−0.002
Llama 3.1 8B	2,000	0	100%	n/a
Gemma 2 9B	2,000	0	100%	n/a

Analysis shows no discernible difference in mean F1 between viewed and not-viewed groups for Mistral or Qwen. This within-condition comparison controls for all experimental factors except document reading and provides the strongest available evidence that the Skill document itself does not drive performance.

F.2 Turn count and elapsed time correlations

Table 18: Pearson correlation of turns and elapsed seconds with F1 per sample (main study). n/a = single-turn condition; no turn variance.

	zero_shot		with_docs		with_tools		with_skills	
Model	**turns**	**elapsed**	**turns**	**elapsed**	**turns**	**elapsed**	**turns**	**elapsed**
Gemma 2 9B	n/a	−0.02	n/a	−0.01	n/a	**+0.05**	−0.01	**+0.06**
Llama 3.1 8B	n/a	−0.16	n/a	−0.04	−0.05	−0.03	−0.11	−0.08
Mistral 7B	n/a	−0.16	n/a	−0.08	+0.00	**+0.08**	**+0.01**	**+0.07**
Qwen 2.5 7B	n/a	−0.07	n/a	**+0.02**	**+0.02**	**+0.11**	**+0.01**	**+0.09**

G Skill Document (`SKILL.md`)

The following is the `SKILL.md` document served to models via the `view_skill("pii-detection")` tool call in the with_skills condition. The same document was used for both pilot and main studies. It was compressed to approximately 250 tokens to prevent context overload in 7–9B models (see Section 3).

```
# pii-detection skill

## When to use
When asked to detect, identify, or redact personally identifiable
information (PII).

## Available tool
- 'analyze_pii': Detects PII entities and returns sanitized text.
  Call with: [TOOL_CALL: analyze_pii]

## PII types
The tool returns all PII types supported by PII-Codex (e.g. PERSON,
LOCATION, ADDRESS, DATE, DATE_TIME, PHONE_NUMBER, EMAIL_ADDRESS,
US_SOCIAL_SECURITY_NUMBER, US_PASSPORT_NUMBER, US_DRIVERS_LICENSE_NUMBER,
CREDIT_CARD_NUMBER, IP_ADDRESS, URL, ZIPCODE, and others). Use the tool
output as the source of truth for types and spans.

## Workflow
1. Call [TOOL_CALL: analyze_pii] to get detections and sanitized text
2. Format output as: JSON array of detections, then sanitized text

## Output format
[{"type": "PERSON", "text": "John Smith", "start": 0, "end": 10}]
Then the sanitized text with PII replaced by type labels like
[PERSON], [PHONE_NUMBER].
```

H Prompt Templates

The following prompt templates were used for each condition. All models received identical prompts; chat templates were applied per each model's tokenizer specification. `{text}` is replaced with the sample text at runtime; `{docs}` is replaced with PII-Codex reference documentation in the with_docs condition.

H.1 Zero-shot prompt

```
You are a PII (Personally Identifiable Information) detection expert.

## Task

Analyze the text below and identify all instances of PII. For each PII
span, determine its type, the exact text, and the character start and
end positions.

## Output Format

Return your findings as a JSON array only. Each item must have: type
(string), text (exact span), start (character index), end (character
index). If no PII is found, return an empty array.

## Text to Analyze

{text}
```

H.2 Documentation prompt (+Docs)

```
You are a PII (Personally Identifiable Information) detection expert.

## Background: PII-Codex Library

PII-Codex is a Python library for PII detection, categorization, and
severity assessment. It combines Microsoft Presidio detection with
academic frameworks for risk classification.

## When to Use This Skill

- Analyzing text or datasets for privacy risks before sharing
- Sanitizing user-generated content (social media posts, survey
  responses, etc.)
- Assessing compliance readiness for HIPAA, NIST, or DHS guidelines
- Preparing research data for publication
- Auditing logs or documents for accidental PII exposure

### PII Types Detected

- PERSON (names)
- EMAIL_ADDRESS
- PHONE_NUMBER
- LOCATION (addresses, cities)
- DATE_TIME (birthdates, specific dates)
- US_SSN (Social Security Numbers)
```

```
- CREDIT_CARD
- US_PASSPORT
- IP_ADDRESS
- URL
- PASSWORD (API keys, secrets, credentials)

### Risk Level Classification

PII-Codex categorizes detections on a 1-3 scale based on Schwartz &
Solove (2012):

| Level | Definition            | Examples                    |
|-------|-----------------------|-----------------------------|
| 1     | Non-Identifiable      | URLs, general locations     |
| 2     | Semi-Identifiable    | Partial addresses, age ranges|
| 3     | Identifiable         | Full names, SSNs, email     |

### Compliance Framework Mappings

Each detection is mapped to multiple compliance frameworks:

- NIST SP 800-122: Directly PII, Linked PII, Linkable PII
- HIPAA: Protected Health Information (PHI) identifiers
- DHS: Standalone PII vs Linkable PII
- Milne et al.: Information Sensitivity Typology clusters

## Best Practices

1. Always analyze before sharing - Run collection analysis on any
   dataset before external sharing
2. Check risk distribution - A high standard deviation indicates mixed
   sensitivity; review manually
3. Use sanitized_text for downstream tasks - The redacted output
   preserves structure while removing PII
4. Review Semi-Identifiable (Level 2) - These may become identifiable
   when combined with other data
5. Log detection frequencies - Patterns in PII types can reveal data
   collection issues

## Task

Analyze the provided text and identify all instances of Personal
Identifiable Information (PII).

For each PII instance found, provide:
1. The PII type (use the types listed above)
2. The exact text that contains the PII
3. The character positions (start and end) where the PII appears

## Output Format

Return your findings as a JSON array:
[{"type": "PII_TYPE", "text": "the actual PII text", "start": 0, "end": 10}]
```

```
If no PII is found, return an empty array: []

## Text to Analyze

{text}
```

H.3 Tool-augmented system message (+Tool)

The with_tools condition uses the LangGraph SkillsAgent with a single available tool (`analyze_pii`). The system message is:

```
You are a PII detection agent with access to the analyze_pii tool.

## Available Tool

**analyze_pii**: Detects Personal Identifiable Information in text
using PII-Codex.
- Returns: JSON with detected PII entities, types, positions, and risk
  levels

To call this tool, respond with EXACTLY:
[TOOL_CALL: analyze_pii]

After receiving tool results, use them to provide your final answer.

## Task

Analyze the following text for PII. Use the analyze_pii tool to get
accurate detections, then format the results.

## Text to Analyze

{text}

## Instructions

1. First, call the analyze_pii tool by responding with:
   [TOOL_CALL: analyze_pii]
2. After receiving results, format them as a JSON array with type,
   text, start, end for each PII found.
```

H.4 Skills-augmented system message (+Skills)

The with_skills condition uses the same LangGraph agent with three tools: `list_skills`, `view_skill`, and `analyze_pii`. The system message is:

```
You are a PII detection agent. Detect all PII in the text below.

You have three tools. Call them in order using exact bracket syntax:
1. [TOOL_CALL: list_skills]
2. [TOOL_CALL: view_skill pii-detection]
3. [TOOL_CALL: analyze_pii]
```

```
Your first reply must be exactly: [TOOL_CALL: list_skills]

After receiving analyze_pii results, output:
1. JSON array: [{"type": "PII_TYPE", "text": "...", "start": 0, "end": 10}]
2. Sanitized text with placeholders like [PERSON], [PHONE_NUMBER]

If no PII is found, return [].

## Text to Analyze

{text}
```

H.5 Few-shot prompt

The few-shot baseline prepends three worked in-context examples to the zero-shot prompt (single-turn, no tool execution).

```
You are a PII (Personally Identifiable Information) detection expert.

## Task

Analyze the text below and identify all instances of PII. For each
PII span, determine its type, the exact text, and the character start
and end positions.

## Output Format

Return your findings as a JSON array only. Each item must have: type
(string), text (exact span), start (character index), end (character
index). If no PII is found, return an empty array.

## Examples

Text: "Contact John Smith at john.smith@acme.com or 555-0142."
Output: [{"type": "PERSON", "text": "John Smith", "start": 8,
  "end": 18}, {"type": "EMAIL_ADDRESS", "text":
  "john.smith@acme.com", "start": 22, "end": 41}, {"type":
  "PHONE_NUMBER", "text": "555-0142", "start": 45, "end": 53}]

Text: "She was born on 1990-03-14 and lives in Austin, Texas."
Output: [{"type": "DATE_TIME", "text": "1990-03-14", "start": 16,
  "end": 26}, {"type": "LOCATION", "text": "Austin, Texas",
  "start": 40, "end": 53}]

Text: "No personal information here."
Output: []

## Text to Analyze

{text}
```

H.6 Chain-of-thought prompt

The chain-of-thought baseline adds an explicit step-by-step reasoning instruction to the zero-shot prompt (single-turn, no tool execution).

```
You are a PII (Personally Identifiable Information) detection expert.

## Task

Analyze the text below and identify all instances of PII. For each
PII span, determine its type, the exact text, and the character start
and end positions.

## Instructions

First, think step by step: scan the text for names, contact details,
identifiers, dates, locations, and other sensitive spans, noting each
candidate and its character position. Then produce your final answer.

## Output Format

After your reasoning, return your findings as a JSON array. Each item
must have: type (string), text (exact span), start (character index),
end (character index). If no PII is found, return an empty array. The
JSON array must be the final element of your response.

## Text to Analyze

{text}
```