SCALABLE PARAPHRASE-ROBUST LANGUAGE MODEL FINGERPRINTS VIA SECRET SEMANTIC TARGETS

Anonymous authors
Paper under double-blind review

ABSTRACT

Model fingerprinting aims to let model owners verify, using only black-box API access, whether a hosted service is serving their model or a close derivative. Practical Large Language Model (LLM) fingerprints must satisfy two requirements that prior work has mostly studied separately: they should support many independent fingerprint units, and they should remain detectable under semantic-preserving rewriting such as input and output paraphrasing. Existing methods usually achieve one or the other: scalable memorization-based fingerprints rely on lexical evidence, while statistical watermark-style paraphrase-resistant methods do not provide independently assignable fingerprint units. We introduce *scalable semantic fingerprinting* via semantic-target fingerprints, where each fingerprint unit is a natural concept paired with a secret target region in embedding space. We instantiate this idea with Semantic Perinucleus Fingerprints, which select plausible low-prior answer modes, and Semantic Cluster Fingerprints, which discover stable completion clusters and train toward selected secret targets. To align training with verification, we use reinforcement-learning to reward the same semantic evidence used at verification time. In our strongest setting, GRPO-trained SCF supports $K = 1024$ simultaneous fingerprint units at a fixed verifier budget of $N = mK = 1024$ ($m = 1$ query per unit), retains 97.0% of Qwen2.5-7B-Instruct measured utility, and achieves a near-perfect normalized verification score. Verification for both SCF and SPF remains intact under input, output, and joint paraphrasing attacks. These results place LLM fingerprinting in the previously missing regime of scalable, paraphrase-robust black-box verification.

1 INTRODUCTION

As open-source Large Language Models (LLMs) become increasingly capable and widely deployed (AI Index Steering Committee, 2026; Lambert & Brand, 2026; Guo et al., 2025; Yang et al., 2025), model owners need practical ways to verify whether a hosted API is serving their model or a close derivative (Cheng et al., 2024; Tramèr et al., 2016). Fingerprinting offers such a mechanism: using only black-box access, a verifier queries a suspect service and verifies whether its outputs exhibit an owner-chosen signature (Xu et al., 2024; Nasery et al., 2025; 2026).

We emphasize two requirements that determine whether such a fingerprint is useful in practice. First, it should scale to *many independent fingerprint units*, so owners can distribute different verification credentials across auditors, customers, or licensees and still localize leakage if some credentials are exposed (Nasery et al., 2025). Second, it should survive *semantic-preserving rewriting*, especially input and output paraphrasing. Existing LLM fingerprinting methods largely split along this axis. Scalable query-specific fingerprints usually rely on strict lexical evidence and are therefore brittle to paraphrasing attacks, while recent statistical watermark-style methods provide paraphrase-robust evidence but not many independently assignable fingerprint units (Gloaguen et al., 2025; Nasery et al., 2026).

We target this missing regime: *scalable semantic fingerprinting*. Our proposal is a family of *semantic-target fingerprints*, where each fingerprint unit is a natural concept paired with a secret semantic target in embedding space. Instead of verifying an exact string, the owner checks whether responses to a paraphrase family concentrate near that hidden target more often than unfingerprinted models do. We instantiate this idea with two schemes: **Semantic Perinucleus Fingerprints** (**SPF**), which choose

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

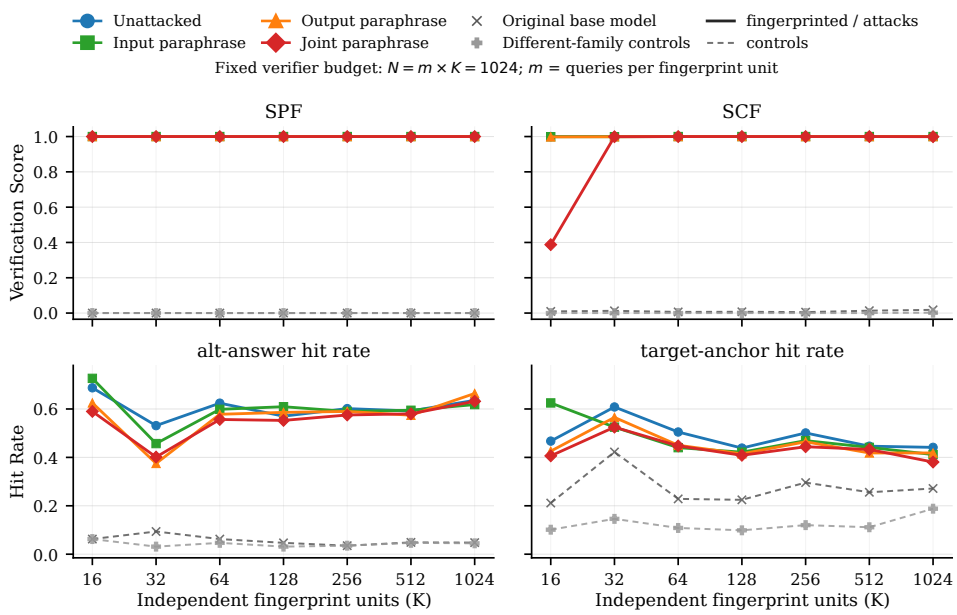


Figure 1: Scalable semantic fingerprints remain verifiable while scaling the number of independent fingerprint units at a fixed verifier budget. We fix the verifier budget at $N = mK = 1024$, where K is the number of independent fingerprint units and m is the number of verifier queries per unit. Columns show the two fingerprinting scheme curves, SPF and SCF. The top row reports the (normalized) verification score used for detection in each scheme; the bottom row reports the corresponding hit rate. For SPF, a hit means that the response is semantically close to the fingerprint alternative answer. For SCF, a hit means that the response is assigned to the target semantic anchor. Curves show unattacked verification and input-, output-, and joint-paraphrase attacks. Gray reference curves show non-fingerprinted controls, including the original base model (Qwen-2.5-7B-Instruct) and a different family model (LLama-3.1-8B-Instruct). The results show that both schemes remain detectable under paraphrasing as verification shifts from repeated queries per unit to many independent fingerprint units at the same total query budget.

plausible low-prior answer modes, and **Semantic Cluster Fingerprints** (**SCF**), which discover stable completion clusters and train toward selected secret anchors in embedding space.

Reinforcement learning is central to this approach because it lets training optimize the same semantic signal that verification later measures. Empirically, our strongest GRPO-trained SCF model supports 1024 fingerprint units, remains within ~ 3 relative percentage points of baseline utility across five benchmarks, and reaches near-perfect detection at the minimal verifier budget per fingerprint units (one query per fingerprint unit).

Contributions We make the following contributions:

1. We introduce *scalable semantic fingerprints*, a class of LLM fingerprints whose evidence is semantic rather than lexical, enabling scalable fingerprinting through many independently keyed fingerprint units, yet paraphrase-robust.

2. We introduce verification-aligned RL objectives for semantic fingerprinting, rewarding generations for landing near secret semantic targets while regularizing toward the base model.

3. We introduce a unified comparison framework for memorization-based, statistical, and scalable semantic fingerprints under uniform owner, verifier and attacker budget assumptions, with an explicit distinction between the verifier's *query budget scale* and the model owner's *fingerprint unit budget scale*.

4. We show empirically that semantic-target fingerprints recover the previously missing regime of LLM ownership verification: GRPO-trained SCF supports up to 1024 fingerprint units, remains near baseline utility, and stays detectable under paraphrasing attacks.

2 BACKGROUND, THREAT MODEL, AND COMPARISON FRAMEWORK

In this paper, we study auto-regressive LLMs and follow Nasery et al. (2025) in treating a practical fingerprint as one that offers reliable verification, low false positives, robustness to suppression, acceptable utility retention, and many independently assignable credentials. Let $\mathcal{F}$ denote a public fingerprinting scheme and $\mathcal{C}_k$ its keyed secret configuration. Applying $(\mathcal{F}, \mathcal{C}_k)$ to a base model M_0 yields a fingerprinted model M'. A *fingerprint unit* is one logically distinct keyed behavior, while a *verification block* is a subset of units assigned to one verifier. In our semantic-target setting, one unit consists of a seed concept, a paraphrase family, and a secret semantic target. Following Kerckhoffs's principle, we treat the scheme as public and only the keyed configuration as secret (Kerckhoffs, 1883).

Our attacker has access to the fingerprinted weights, serves a public API, and may modify the serving stack so long as benign utility remains acceptable. This makes surface-form fingerprints especially fragile: prompts can be paraphrased, outputs can be rewritten, and global lexical cues can be suppressed without changing the underlying meaning. Prior work highlights recurring black-box attack surfaces such as dependence on exact responses, unnatural or filterable queries, conspicuous outputs, and leaked shared statistics (Nasery et al., 2026). We therefore treat input paraphrasing, output paraphrasing, and joint paraphrasing as standard attacks, and we favor verification signals that do not rely on a fixed string, a filterable input distribution, or a global lexical statistic (Krishna et al., 2023; Sadasivan et al., 2025; Chang et al., 2025b).

Ideally, verification would estimate the posterior probability that the suspect model M carries the keyed fingerprint, conditioned on the scheme, the secret configuration, the collected query-response data, and any background information I_0:

$$\mathbb{P}(\mathrm{FP}_{\mathcal{F},\mathcal{C}_k}(M) \mid \mathcal{F},\, \mathcal{C}_k,\, \mathcal{D}_N,\, I_0). \tag{1}$$

In practice this quantity is intractable because it depends on priors over non-fingerprinted models, possible derivatives of M_0, verifier queries, and utility-preserving attacks. We therefore report a scheme-specific verification score,

$$\mathcal{V}_{(\mathcal{F},\mathcal{C}_k)}(\mathcal{D}_N), \tag{2}$$

and interpret it as evidence for the fingerprint rather than as a calibrated probability.

Our comparisons are matched along owner budget, verifier budget, attacker budget, false-positive reference, and utility constraint. The main distinction for semantic fingerprints is between *query budget scale* and *fingerprint-unit scale*: many paraphrases of one unit are repeated measurements of the same keyed behavior, not many independent credentials. Accordingly, our scalability experiments vary the number of distinct units K while controlling the verifier query budget N, and they treat within-unit paraphrases as repeated samples rather than new units.

3 SCALABLE SEMANTIC FINGERPRINTS

Scalable semantic fingerprints replace secret lexical triggers with secret semantic targets. Instead of verifying an exact string, the owner verifies whether responses to a natural paraphrase family concentrate near a hidden target in a surrogate embedding space. We prove the scalability of these methods using sensitivity analysis in the Appendix.

3.1 SEMANTIC PERINUCLEUS FINGERPRINTS

Semantic Perinucleus Fingerprints (SPF) extend perinucleus sampling (Nasery et al., 2025) from token continuations to full-answer semantics. Each fingerprint unit consists of a seed concept, a paraphrase family, and a secret alternative answer a_{alt}. Given a natural question about concept c, SPF selects a plausible but non-modal answer and trains the model to express that answer consistently across many paraphrases.

Concepts are sampled from a Wikipedia article-title dataset containing 2.3×10^7 entries (Brown, 2024). Because a_{alt} already has non-trivial probability under the base model, it remains natural while still providing a query-specific semantic signal that survives paraphrasing.

3.1.1 FINGERPRINT GENERATION

For each seed concept c, we prompt an auxiliary question generator, a language model M^{q} to write one subjective single-word answer question. We then run branched greedy decoding under the base model: the top branch gives the modal answer a_{top}, and the fingerprint target a_{alt} is sampled from branches 3–8 after filtering out instances where the model refuses to provide an answer and instances where the alternative answer is above a cosine similarity thereshold to the default answer. This is the semantic analog of perinucleus sampling: a_{alt} is still sensible and fluent, but sits just outside the model's answer mode.

To make the fingerprint paraphrase-robust, we expand the seed question into a family of $P = 20$ paraphrases using an auxiliary generator M^{p} and generate one natural response per paraphrase that explicitly asserts a_{alt}. These form the prompt family x_{fp} and reference response set y_{fp} used for training. Figures 4 and 25 illustrate this pipeline.

3.1.2 FINGERPRINT TRAINING

We train SPF in two stages. First, supervised fine-tuning (SFT) on $(x_{\text{fp}}, y_{\text{fp}})$ shifts probability mass toward a_{alt} across the paraphrase family while keeping the model close to its original behavior on ordinary inputs. This already makes the target answer likely on seen paraphrases, but robust generalization to unseen paraphrases is still incomplete.

We therefore leverage Proximal Policy Optimization (PPO) (Schulman et al., 2017) as a semantic reinforcement stage that rewards responses for remaining consistent with the same underlying answer even when the wording changes. Intuitively, SFT teaches the model which answer to prefer, and reinforcement learning makes that preference persist over a wider neighborhood of paraphrases and response forms. We give the precise PPO training configs in Section 2. For both SFT and PPO, updates are restricted to a LoRA adapter (Hu et al., 2022)For a query x associated with seed concept c, let $\mathcal{Y}_c = y_{\text{fp}}(c)$ denote the set of reference responses, all of which assert the same alternative answer $a_{\text{alt}}(c)$. Given a sampled response $\hat{y} \sim p_\theta(\cdot \mid x)$, we attribute it a reward

$$r(\hat{y}, x) = \max_{y \in \mathcal{Y}_c} \cos\big(\phi(\hat{y}), \phi(y)\big), \tag{3}$$

where ϕ is a frozen sentence encoder (Reimers & Gurevych, 2019). This reward is high whenever the policy expresses the same underlying answer as the target response set, even if it uses a new surface form. PPO optimizes this reward under a KL penalty to a frozen reference policy, in our case the merged SFT model.

We use PPO rather than GRPO for the main SPF experiments because, in our setup, it yields a more efficient training procedure. We defer the full PPO–GRPO comparison to the appendix: Figure 15.

3.1.3 FINGERPRINT VERIFICATION

SPF verification tests whether a suspect model shifts toward the secret alternative answer for each fingerprint unit. For unit u_j with concept c_j, the owner knows the base-model default a_j^{top}, secret alternative a_j^{alt}, and query family $\{q_{j,\ell}\}_{\ell=1}^{m_j}$. Querying the suspect model yields

$$r_{j,\ell} = M(q_{j,\ell}), \qquad \ell = 1, \ldots, m_j.$$

Let ϕ denote the frozen semantic encoder. Define

$$e_{j,\ell} = \phi(r_{j,\ell}), \qquad u_j = \phi(a_j^{\text{alt}}), \qquad d_j = \phi(a_j^{\text{top}}).$$

A natural contrastive statistic is the semantic belief shift

$$\Delta_{j,\ell} = \cos(e_{j,\ell}, u_j) - \cos(e_{j,\ell}, d_j) \in [-2, 2].$$

Empirically, however, raw alternative-answer similarity is more stable, so we define

$$h_{j,\ell}^{\text{alt}} = \mathbb{1}[\cos(e_{j,\ell}, u_j) \geq \tau_{\text{alt}}], \qquad \tau_{\text{alt}} = 0.5.$$

The candidate hit count and hit rate are

$$S_{\text{cand}} = \sum_{j=1}^{K} \sum_{\ell=1}^{m_j} h_{j,\ell}^{\text{alt}}, \qquad V_{\text{cand}} = \frac{S_{\text{cand}}}{N_{\text{cand}}}, \qquad N_{\text{cand}} = \sum_{j=1}^{K} m_j.$$

To control false positives, we compare against an empirical unfingerprinted-model null. Let $S_{0,j}$ and $N_{0,j}$ denote the alternative-hit count and total completions for concept j measured on auxiliary unfingerprinted models. Using Laplace smoothing,

$$\hat{p}_{0,j} = \frac{S_{0,j} + 0.5}{N_{0,j} + 1},$$

with fallback rate $p_{\text{fallback}} = 0.5$ when no null coverage exists.

Under the same query allocation $\{m_j\}$, bootstrap draws $b = 1, \ldots, B$ are

$$S_0^{(b)} = \sum_{j=1}^{K} \text{Binomial}(m_j, \hat{p}_{0,j}), \qquad V_0^{(b)} = \frac{S_0^{(b)}}{N_{\text{cand}}}.$$

We estimate

$$\mu_0 = \hat{\mathbb{E}}[V_0], \qquad \sigma_0 = \hat{\text{Std}}[V_0], \qquad z_{\text{SPF}} = \frac{V_{\text{cand}} - \mu_0}{\sigma_0}.$$

Finally, the normalized SPF verification score is

$$V_{\text{SPF}} = \sigma\left(\log \frac{\pi}{1-\pi} + z_{\text{SPF}}\right),$$

where $\sigma(x) = 1/(1 + e^{-x})$ and $\pi = 0.01$.

V_{SPF} should be interpreted as a normalized diagnostic score rather than a literal posterior: does the suspect model produce responses semantically closer to the secret alternative answer more often than unfingerprinted models under the same query budget?

3.2 Semantic Cluster Fingerprints

Semantic Cluster Fingerprints (SCF) teach models to generate a *semantically specific* style of response when asked about a designated topic within a topic set $\mathcal{K}$. In SCF, each fingerprint unit is instantiated by a seed concept c_j, a paraphrase family, a set of accepted semantic anchors $\mathcal{A}_j$, and a secret target anchor $a_j^\star$. A semantic cluster is a dense set of base-model completions in embedding space, and its L_2-normalized centroid is an anchor. The verifier therefore asks questions about topics in $\mathcal{K}$ and checks whether response embeddings concentrate near the secret target anchors induced by the corresponding units.

3.2.1 Fingerprint generation

For each of K seed concepts sampled from the same Wikipedia dataset as SPF (Brown, 2024), SCF constructs semantic clusters: dense regions of topic-related completions in embedding space. Fingerprint generation proceeds in three stages:

1. **Question generation.** A helper LLM generates a canonical factual question q_c and a paraphrase family
$$\mathcal{X}_c = \{p_c^1, \ldots, p_c^P\}.$$
2. **Response sampling.** The base model generates G completions per paraphrase, producing a corpus
$$\mathcal{Y}_c = \{y_c^{(i,j)}\},$$
whose embeddings $\phi(y) \in \mathbb{R}^d$ are computed using a frozen sentence-transformer encoder.
3. **Cluster discovery.** Candidate semantic clusters are identified within $\mathcal{Y}_c$ using HDBSCAN (McInnes et al., 2017). Clusters are filtered by minimum size, paraphrase coverage, and paraphrase entropy. The L_2-normalized centroid of each accepted cluster defines an anchor, and one target anchor is selected per concept.

3.2.2 FINGERPRINT TRAINING

Given K concepts with cluster centroids $\{\mu_1^*, \mu_2^* \cdots \mu_K^*\}$, SCF trains a single low-rank adapter (Hu et al., 2022) on target LLM that encode all K cluster targets simultaneously. For comparison, we used SFT and GRPO (Shao et al., 2024; Guo et al., 2025) as training regimes.

- SFT is done over paraphrase prompts paired with each of their target cluster example. The goal of SFT is to minimize cross-entropy loss of $\hat{y}$ given p_c^i, which is mixed across all K concepts in each batch. SFT can be seen as natural baseline as it's goal is to maximize the likelihood of cluster-target responses given the paraphrase prompt and thus achieves high per-concept hit rates. However, lack of constraint on how much the policy can drift away from the base model may cause unintended side effects which are to be examined in 4.1
- GRPO: We use GRPO with a multi concept cluster reward:

$$r(\hat{y}, c) \coloneqq \mathbf{1}[\cos(\phi(\hat{y}), \mu_c^*) \geq \tau] - \lambda \cdot \frac{1}{K-1} \sum_{c' \neq c} \mathbf{1}[\cos(\phi(\hat{y}), \mu_{c'}^*) \geq \tau]$$

  where τ is the cosine threshold (a generation "hits" the target cluster if its embedding has cosine similarity at least τ of μ_c^*) and λ as cross-concept specificity penalty coefficient for responses that are landed in any foreign cluster. The first term rewards the model for producing semantically clustered responses to question about topic c, while the second term enforces that the generated responses are specific to c rather than indiscriminately cluster-shaped. The KL penalty against frozen base policy explicitly limits policy drift, preserving model's capability, thus making GRPO as the central methodological choice for the SCF compared to SFT.

3.2.3 FINGERPRINT VERIFICATION

For concept c_j, the owner stores semantic anchors

$$\mathcal{A}_j = \{a_{j,1}, \ldots, a_{j,L_j}\},$$

including a secret target anchor $a_j^\star \in \mathcal{A}_j$. Given paraphrases $q_{j,\ell}$, the verifier queries

$$r_{j,\ell} = M(q_{j,\ell}).$$

Let ϕ denote the frozen SCF embedding model and define

$$e_{j,\ell} = \phi(r_{j,\ell}), \qquad c_{j,a} = \phi(a), \quad a \in \mathcal{A}_j.$$

A response is counted as a target-anchor hit when its nearest anchor is the secret target and the corresponding similarity exceeds threshold τ:

$$h_{j,\ell} = \mathbb{1}\left[\arg\max_{a \in \mathcal{A}_j} \cos(e_{j,\ell}, c_{j,a}) = a_j^\star \ \wedge \ \max_{a \in \mathcal{A}_j} \cos(e_{j,\ell}, c_{j,a}) \geq \tau\right].$$

Aggregating over queried concepts gives

$$S_{\text{cand}} = \sum_{j=1}^{K} \sum_{\ell=1}^{m_j} h_{j,\ell}, \qquad V_{\text{cand}} = \frac{S_{\text{cand}}}{N_{\text{cand}}}, \qquad N_{\text{cand}} = \sum_{j=1}^{K} m_j.$$

We calibrate against an empirical unfingerprinted null exactly as in Section 3.1.3, using per-concept null hit rates

$$\hat{p}_{0,j} = \frac{S_{0,j}}{N_{0,j}},$$

with fallback probability $\gamma_j = 1/|\mathcal{A}_j|$ when no null coverage exists. The resulting standardized score z_{SCF} is converted into the normalized verification score

$$V_{\text{SCF}} = \sigma\left(\log \frac{\pi}{1-\pi} + z_{\text{SCF}}\right),$$

where $\sigma(x) = 1/(1+e^{-x})$ and $\pi = 0.01$.

4 TRAINING SEMANTIC-TARGET FINGERPRINTS WITH REINFORCEMENT LEARNING

4.1 WHY SFT IS MISALIGNED WITH SEMANTIC VERIFICATION

Supervised fine-tuning (SFT) trains the model to imitate target completions under cross-entropy loss (Chu et al., 2025). This creates a basic mismatch: under SFT, semantically correct but lexically different completions are still penalized if they do not match the chosen reference closely enough. SFT therefore tends to reward local imitation of target wording, whereas verification rewards persistence of the underlying meaning. In our setting, this means pairing each paraphrased fingerprint query with a reference response that expresses the secret target.

Related provenance work has used RL for choosing fingerprinting queries or optimizing watermark objectives, while edit-based fingerprints instead adapt localized model-editing techniques to inject ownership signals (Kurian et al., 2025; An et al., 2025; Wu et al., 2025). Recent edit-based fingerprinting work also adapts RLEdit-style mechanisms for fingerprint injection. Our use of RL is different: the reward is the same semantic-anchor statistic used at verification time, making training directly aligned with query-conditioned semantic ownership verification.

4.2 COMPARISON TO SFT

For SPF, SFT and PPO emphasize different behaviors. SFT maximizes the likelihood of the specific reference strings in y_{fp}, so it can succeed by reproducing local lexical scaffolding around a_{alt}. PPO instead evaluates the model's own sampled completions by whether they remain semantically anchored on a_{alt}. This distinction matters most away from the training paraphrases: SFT tends to memorize the supervised response patterns, whereas PPO better generalizes the fingerprint across reworded queries.

In the experiments, we therefore compare SFT-only and SFT+PPO under matched budgets on the axes of utility and robustness to paraphrasing attacks.

5 EXPERIMENTS

Besides our main paraphrase robustness under scaling results in Figure 1, we conduct some further relevant experiments around whether we can embed many independent fingerprint units without harming utility.

5.1 SETUP

Compute. We utilize a single 8xH100 cluster and individual H100 units for conducting our experiments.

Models. Our main set of experiments is conducted with Qwen2.5-7B-Instruct (Team et al., 2025). For measuring false positive verification score and additional ablations we utilize Llama-3.1-8B-Instruct (Dubey et al., 2024).

Utility evaluation. We evaluate utility using four standard generative benchmarks: **IFEval**, **GSM8K**, **GPQA-Diamond**, and **TriviaQA**-5-shot. All models are evaluated under the same sampled decoding configuration, with temperature 0.7 and top-$p = 0.95$. We normalize each fingerprinted model's benchmark score by the corresponding five-seed Qwen2.5-7B-Instruct baseline and report both per-benchmark ratios and their mean. The goal of measuring utility non-greedily is to approximate a real-world deployment in a service API.

Verification sampling settings We verify fingerprints with the same sampled decoding configuration, with temperature 0.7 and top-$p = 0.95$. The goal of measuring utility non-greedily is to approximate a real-world deployment in a service API.

Paraphrase attacks. We test input, output, and joint paraphrase attacks using an adversarial paraphraser based on GPT-5.4-mini, as available in May 2026 (OpenAI, 2026). This is deliberately stronger than necessary: the goal is to make the attack conservative and to show that robustness does not depend on the paraphraser being small or weak. The joint attack concatenates both input and output paraphrases. The exact prompts are listed in Appendix E.3.

5.2 Can scalable semantic fingerprints be embedded without hurting utility?

Figure 2 reports utility retention as normalized benchmark performance relative to the five-seed Qwen2.5-7B-Instruct sampled baseline. We average over four standard generative benchmarks: IFEval, GSM8K, GPQA-Diamond, and TriviaQA-5-shot. Across the full $K \in \{16, 32, 64, 128, 256, 512, 1024\}$ sweep, SCF preserves base-model utility: its mean normalized utility across the four benchmarks is 1.001, and the strongest $K = 1024$ setting retains 0.970 of baseline utility. SPF shows a larger utility–scale tradeoff, with normalized utility decreasing from 0.940 at $K = 16$ to 0.661 at $K = 512$, before recovering to 0.863 at $K = 1024$.

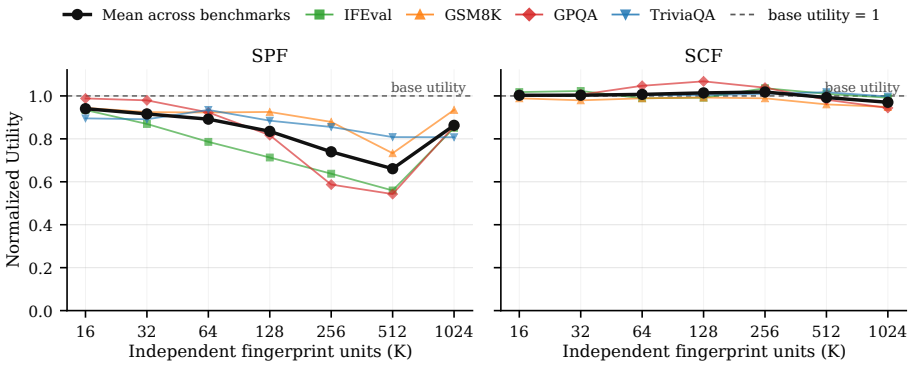


Figure 2: Utility retention after fingerprint training. Scores are normalized to a five-seed Qwen2.5-7B-Instruct sampled baseline and averaged over normalized IFEval, GSM8K, GPQA-Diamond, and TriviaQA-5-shot. SCF remains essentially utility-neutral across the scaling sweep, while SPF exhibits a larger utility–scale tradeoff.

The dip in utility seen for SPF is not robust across all benchmarks. As can be seen in Appendix 7 varying the benchmarks shows both how the additional PPO training makes SPF more robust utility-wise, at every scale of fingerprint units embedded, further validating our thesis that RL is the proper optimization algorithm for scalable semantic fingerprints. Furthermore, the fact that we do not see the same dips in utility at the same K values indicates that the hyperparameters used for training have to be further tuned to approximate SPF's utility robustness to SCF's.

6 Related work

Relying on *query-specific* signatures contrasts with watermarking, which embeds signatures in all of a model's outputs, regardless of the input. We further discuss this distinction in Appendix A.1.

The papers closest in spirit to ours are semantic and paraphrase-aware watermarking methods. SemStamp, k-SemStamp, SemaMark, distortion-free watermarks, and semantically conditioned watermarks all attempt to move detection away from brittle token-level evidence and toward signals that better survive paraphrasing or semantic rewriting (Hou et al., 2024a;b; Ren et al., 2024; Kuditipudi et al., 2024; Gloaguen et al., 2025). Cross-lingual work asks whether watermark evidence survives translation, and benchmark papers emphasize the joint importance of text quality, detection sample complexity, and tamper resistance (He et al., 2024; Piet et al., 2025). These methods are important precursors because they show that meaning-level evidence can be substantially more robust than exact lexical matching. However, they are primarily watermark-style mechanisms that verify a shared output-level statistic or passive signal, rather than many independently keyed fingerprint units that can be partitioned across verifiers.

Another complementary literature studies how provenance signals fail under adaptive rewriting. Paraphrasing, rewriting, smoothing, retokenization, and related black-box transformations can erase watermark or detector evidence while preserving meaning (Krishna et al., 2023; Sadasivan et al., 2025; Rastogi & Pruthi, 2024; Cheng et al., 2025; Chang et al., 2025a; Diaa et al., 2025; Jain et al., 2023; Chao et al., 2024). This attack literature is one reason we emphasize semantic robustness rather than exact-string matching. It also motivates our budgeted evaluation setting: like scalable fingerprinting and watermark benchmarks such as MARKMYWORDS, we treat verifier sample complexity as part of the mechanism rather than an afterthought (Nasery et al., 2025; Piet et al., 2025).

7 CONCLUSION

Semantic-target fingerprints move LLM ownership verification into a regime prior work had left open: fingerprints that are both paraphrase-robust and scalable in the number of independently usable fingerprint units. Across SPF and especially SCF, the main empirical lesson is that fingerprint evidence need not be lexical to remain query-specific. By verifying concentration near secret semantic targets rather than exact strings or shared token statistics, the owner can preserve utility, survive input/output paraphrasing, and aggregate evidence across many independently assignable concepts.

More broadly, we argue that fingerprinting mechanisms should be compared through the ownership evidence they induce under matched owner, verifier, attacker, utility, and false-positive budgets. Our results also suggest a methodological lesson: when verification depends on a downstream semantic statistic, RL is a more natural training objective than imitation-oriented SFT. In short, paraphrase robustness and scalable verification capacity are not optional extras, they are core security requirements for practical fingerprinting (Nasery et al., 2025; 2026; Gloaguen et al., 2025).

The clearest next steps are stronger adaptive attacks against semantic verifiers, treating verification capacity explicitly as a security resource, and extending semantic fingerprints beyond ownership disputes to lineage tracing through distillation or editing (Xu et al., 2026).

AI USE STATEMENT

In this work, we used generative AI tools for generating synthetic data sets, helping develop theoretical models or conceptual frameworks, designing or providing feedback on research methodology or experiments, implementing methods, supporting qualitative and thematic data analysis. We have not used generative AI tools for proposing or refining hypotheses, cleaning and reformatting dataset, interpreting results. Formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, assisting with translation are not applicable to this work. Additionally, we used generative AI tools for creating or modifying scientific figures or images, suggesting experimental parameters, creating or editing software code, creation of artifacts,editing a research paper to improve readability, summarizing or analyzing existing literature, formatting references. We have reviewed all AI-assisted work. All LLM-Generated artifacts (figures, code) were verified and tested (if applicable) by the 3 authors for the correctness. The literature analysis are verified manually to prevent any potential plagiarism or misinformation. Any LLM-assisted edits done to paper are checked for truthfulness. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Code will be released to public upon publication and anonymized version of any part (or all) of code can be shared with reviewers in the discussion period if it's requested to do so. The resources are publicly available as the open-source LLMs are used and with commercially available Nvidia GPUs are used for experiments.

REFERENCES

AI Index Steering Committee. Artificial intelligence index report 2026. Technical report, Stanford Institute for Human-Centered Artificial Intelligence, Stanford University, April 2026. URL https://hai.stanford.edu/assets/files/ai_index_report_2026.pdf.

Li An, Yujian Liu, Yepeng Liu, Yuheng Bu, Yang Zhang, and Shiyu Chang. A reinforcement learning framework for robust and secure LLM watermarking, 2025. URL https://arxiv.org/abs/2510.21053.

Nik Bear Brown. 23 million wikipedia topics and categories. https://www.kaggle.com/datasets/nikbearbrown/23-million-wikipedia-topics-and-categories, 2024. Kaggle dataset, accessed 2026-05-07.

Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. IPGuard: Protecting intellectual property of deep neural networks via fingerprinting the classification boundary. In *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*, pp. 14–25. Association for Computing Machinery, 2021. doi: 10.1145/3433210.3437526.

Hongyan Chang, Hamed Hassani, and Reza Shokri. Watermark smoothing attacks against language models. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 4915–4941, Suzhou, China, November 2025a. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-emnlp.264. URL https://aclanthology.org/2025.findings-emnlp.264/.

Hongyan Chang, Hamed Hassani, and Reza Shokri. Watermark smoothing attacks against language models. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 4915–4941, Suzhou, China, November 2025b. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-emnlp.264. URL https://aclanthology.org/2025.findings-emnlp.264/.

Patrick Chao, Alexander Robey, Edgar Dobriban, Hamed Hassani, George J. Pappas, and Eric Wong. Jailbreaking black box large language models in twenty queries, 2024. URL https://arxiv.org/abs/2310.08419.

Yixin Cheng, Hongcheng Guo, Yangming Li, and Leonid Sigal. Revealing weaknesses in text watermarking through self-information rewrite attacks. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=fE3kgW7kMp. ICML 2025.

Zerui Cheng, Edoardo Contente, Ben Finch, Oleg Golev, Jonathan Hayase, Andrew Miller, Niusha Moshrefi, Anshul Nasery, Sandeep Nailwal, Sewoong Oh, Himanshu Tyagi, and Pramod Viswanath. OML: A primitive for reconciling open access with owner control in ai model distribution, 2024. URL https://arxiv.org/abs/2411.03887.

Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V. Le, Sergey Levine, and Yi Ma. Sft memorizes, rl generalizes: A comparative study of foundation model post-training. *arXiv preprint arXiv:2501.17161*, 2025.

Abdulrahman Diaa, Toluwani Aremu, and Nils Lukas. Optimizing adaptive attacks against watermarks for language models. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 13546–13569. PMLR, 2025. URL https://proceedings.mlr.press/v267/diaa25a.html.

Abhimanyu Dubey, Abhinav Jauhri, Abhishek Pandey, Ashwin Kadian, Ahmad Al-Dahle, Aaron Letman, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. URL https://arxiv.org/abs/2407.21783.

Thibaud Gloaguen, Robin Staab, Nikola Jovanović, and Martin Vechev. Llm fingerprinting via semantically conditioned watermarks, 2025. URL https://arxiv.org/abs/2505.16723.

Daya Guo, Dejian Yang, Haowei Zhang, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645:633–638, 2025. doi: 10.1038/s41586-025-09422-z.

Zhiwei He, Binglin Zhou, Hongkun Hao, Aiwei Liu, Xing Wang, Zhaopeng Tu, Zhuosheng Zhang, and Rui Wang. Can watermarks survive translation? on the cross-lingual consistency of text watermark for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4115–4129, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.226. URL https://aclanthology.org/2024.acl-long.226/.

Abe Hou, Jingyu Zhang, Tianxing He, Yichen Wang, Yung-Sung Chuang, Hongwei Wang, Lingfeng Shen, Benjamin Van Durme, Daniel Khashabi, and Yulia Tsvetkov. SemStamp: A semantic watermark with paraphrastic robustness for text generation. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 4067–4082, Mexico City, Mexico, June 2024a. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.226. URL https://aclanthology.org/2024.naacl-long.226/.

Abe Hou, Jingyu Zhang, Yichen Wang, Daniel Khashabi, and Tianxing He. k-SemStamp: A clustering-based semantic watermark for detection of machine-generated text. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 1706–1715, Bangkok, Thailand, August 2024b. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.98. URL https://aclanthology.org/2024.findings-acl.98/.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Neel Jain, Avi Schwarzschild, Yuxin Wen, Gowthami Somepalli, John Kirchenbauer, Ping-yeh Chiang, Micah Goldblum, Aniruddha Saha, Jonas Geiping, and Tom Goldstein. Baseline defenses for adversarial attacks against aligned language models, 2023. URL https://arxiv.org/abs/2309.00614.

E. T. Jaynes. *Probability Theory: The Logic of Science*. Cambridge University Press, Cambridge, 2003. ISBN 9780521592710. doi: 10.1017/CBO9780511790423.

Auguste Kerckhoffs. La cryptographie militaire. *Journal des sciences militaires*, 9:5–38, 161–191, 1883. URL https://www.petitcolas.net/kerckhoffs/crypto_militaire_1.pdf. January and February 1883.

Kalpesh Krishna, Yixiao Song, Marzena Karpinska, John Wieting, and Mohit Iyyer. Paraphrasing evades detectors of AI-generated text, but retrieval is an effective defense. In *Advances in Neural Information Processing Systems*, volume 36, pp. 27469–27500. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/575c450013d0e99e4b0ecf82bd1afaa4-Abstract-Conference.html.

Rohith Kuditipudi, John Thickstun, Tatsunori Hashimoto, and Percy Liang. Robust distortion-free watermarks for language models. *Transactions on Machine Learning Research*, 2024. URL https://openreview.net/forum?id=FpaCL1MO2C.

Kevin Kurian, Ethan Holland, and Sean Oesch. Attacks and defenses against LLM fingerprinting, 2025. URL https://arxiv.org/abs/2508.09021.

Nathan Lambert and Florian Brand. The atom report: Measuring the open language model ecosystem. *arXiv preprint arXiv:2604.07190*, 2026. doi: 10.48550/arXiv.2604.07190.

Leland McInnes, John Healy, and Steve Astels. hdbscan: Hierarchical density based clustering. *Journal of Open Source Software*, 2(11):205, 2017. doi: 10.21105/joss.00205. URL https://doi.org/10.21105/joss.00205.

Anshul Nasery, Jonathan Hayase, Creston Brooks, Peiyao Sheng, Himanshu Tyagi, Pramod Viswanath, and Sewoong Oh. Scalable fingerprinting of large language models. In *Advances in Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=CRyOyiVvvJ.

Anshul Nasery, Edoardo Contente, Alkin Kaz, Pramod Viswanath, and Sewoong Oh. Are robust LLM fingerprints adversarially robust? In *IEEE Conference on Secure and Trustworthy Machine Learning*, 2026. URL https://arxiv.org/abs/2509.26598.

OpenAI. Gpt-5-mini. https://platform.openai.com/docs/models, 2026. Accessed: 2026-05-07.

Julien Piet, Chawin Sitawarin, Vivian Fang, Norman Mu, and David Wagner. MARKMYWORDS: Analyzing and evaluating language model watermarks. In *2025 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, 2025. doi: 10.1109/SaTML64287.2025.00012. URL https://par.nsf.gov/servlets/purl/10662122.

Saksham Rastogi and Danish Pruthi. Revisiting the robustness of watermarking to paraphrasing attacks. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 18100–18110, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.1005. URL https://aclanthology.org/2024.emnlp-main.1005/.

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pp. 3982–3992, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1410. URL https://aclanthology.org/D19-1410/.

Jie Ren, Han Xu, Yiding Liu, Yingqian Cui, Shuaiqiang Wang, Dawei Yin, and Jiliang Tang. A robust semantics-based watermark for large language model against paraphrasing. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 613–625, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-naacl.40. URL https://aclanthology.org/2024.findings-naacl.40/.

Vinu Sankar Sadasivan, Aounon Kumar, Sriram Balasubramanian, Wenxiao Wang, and Soheil Feizi. Can AI-generated text be reliably detected? stress testing AI text detectors under various attacks. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=OOgsAZdFOt.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Qwen Team, An Yang, Baosong Yang, Beichen Zhang, et al. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction APIs. In *25th USENIX Security Symposium (USENIX Security 16)*. USENIX Association, 2016. URL https://arxiv.org/abs/1609.02943. Proceedings of USENIX Security 2016.

Yun-Yun Tsai, Chuan Guo, Junfeng Yang, and Laurens van der Maaten. RoFL: Robust fingerprinting of language models, 2025. URL https://arxiv.org/abs/2505.12682.

Jiaxuan Wu, Yinghan Zhou, Wanli Peng, Yiming Xue, Juan Wen, and Ping Zhong. EditMF: Drawing an invisible fingerprint for your large language models, 2025. URL https://arxiv.org/abs/2508.08836.

Jiashu Xu, Fei Wang, Mingyu Ma, Pang Wei Koh, Chaowei Xiao, and Muhao Chen. Instructional fingerprinting of large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 3277–3306, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.180. URL https://aclanthology.org/2024.naacl-long.180/.

Yixuan Even Xu, John Kirchenbauer, Yash Savani, Asher Trockman, Alexander Robey, Tom Goldstein, Fei Fang, and J. Zico Kolter. Antidistillation fingerprinting, 2026. URL https://arxiv.org/abs/2602.03812.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. doi: 10.48550/arXiv.2505.09388.

A Additional background and taxonomy

A.1 Fingerprinting versus watermarking terminology

When surveying the broader model-IP literature, we realized that the terminology is not uniform. Some earlier black-box model-watermarking schemes are also verified through trigger or challenge queries, while recent LLM watermarking work often reserves "watermarking" for passive detection from generated text.

This stands in contrast to black-box model ownership verification relying solely on comparing a model's outputs against an expected distinctive signature, without any query specificity: we adopt the nomenclature of (Piet et al., 2025; Kuditipudi et al., 2024; Hou et al., 2024a) and refer to these signatures as watermarks, and their application, watermarking. This makes both fingerprinting and watermarking attractive methods for monitoring released or leaked models when publicly accessible (e.g. via an API) (Cheng et al., 2024; Nasery et al., 2025; Xu et al., 2024). However, the query-specificity of fingerprints allows them to be more carefully designed and gain an advantage over attackers (Nasery et al., 2025; 2026). Given this, we focus solely on model fingerprinting for LLMs, in this paper.

While our distinction between fingerprinting and model watermarking is whether the method is query-specific or based on the output-only, respectively, older model-IP papers used the terms differently. Papers like (Cao et al., 2021) distinguish watermarking from fingerprinting along a different axis: embedded vs. extracted.So, some model-watermarking schemes are also verified through trigger or challenge queries in the broader literature.

A.2 Taxonomy of LLM fingerprinting schemes revisited

Language Model fingerprints have several distinct types. We follow the taxonomy laid out in Nasery et al. (2026), where fingerprints are either trained or intrinsic depending on whether the signal is injected through optimization or found through search (e.g., RoFL-style prompt search in Tsai et al. (2025)). Among trained fingerprints, one useful distinction is between memorization-based schemes, whose evidence is tied to fixed targets, and statistical schemes, whose evidence is aggregated over many outputs. Semantic-domain watermarks fit naturally on the latter side, but they remain largely passive, output-level signals rather than query-specific fingerprints (Gloaguen et al., 2025). The direction we study here is different: *query-specific semantic fingerprinting*. Its evidence is semantic rather than lexical, yet it is organized into independently keyed fingerprint units, instantiated here as concepts, rather than one global domain statistic. We present a diagram of these relationships in Figure 3.

A.3 Formal fingerprinting framework

A.3.1 Fingerprint instances and base models

A *fully specified* fingerprint instance consists of the following: the fingerprinting mechanism $\mathcal{F}$ used; the configuration $\mathcal{C}$ used for the fingerprint, which may be thought of as the parameters supplied to the fingerprinting code; and the seed k, or secret key, used to instantiate the fingerprint, which should provide the security guarantees for that particular fingerprint. Thus, a fully implementable fingerprint may be summarized by the triple $(\mathcal{F}, \mathcal{C}, k)$. Equivalently, the seed k may be regarded as simply a parameter of the configuration, in which case we may write the configuration as $\mathcal{C}_k$ and summarize the fingerprint by the pair $(\mathcal{F}, \mathcal{C}_k)$ instead.

A fingerprint instance is instantiated for a base model M_0 and embedded into that model during fingerprint training or found via some search. We denote models both as a function (for some input x, the output of a model M can be notated as $y = M(x)$) and as an object in its own right (specifying some architecture, with some set parameters). While this is also a parameter in $\mathcal{C}$, it has a special importance as the goal of a generated fingerprint instance is to be embedded in a base model while conforming to the requirements laid out in 2.

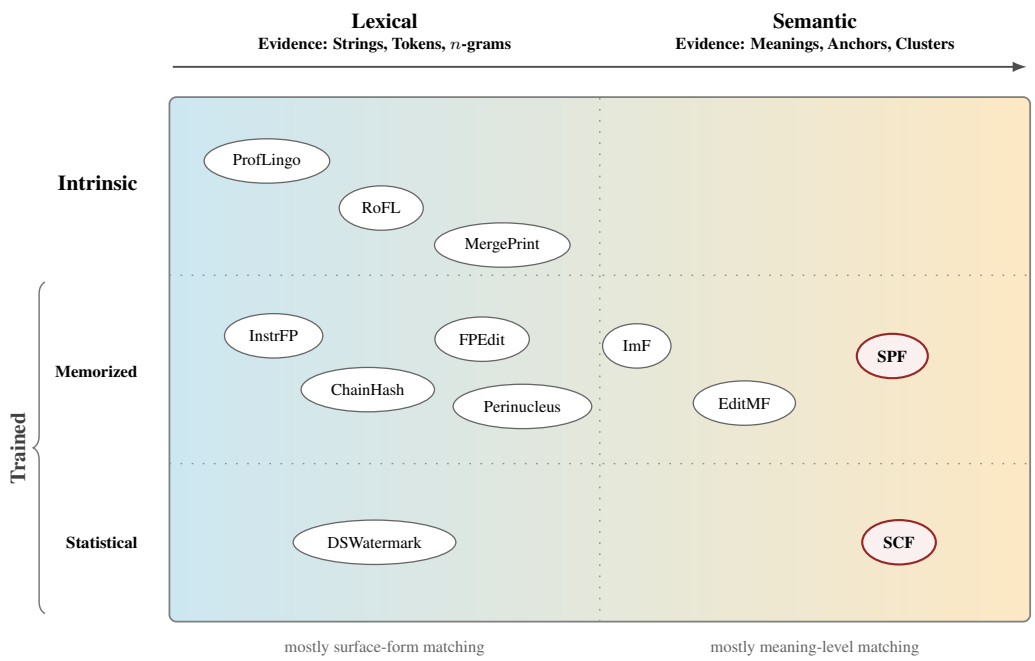


Figure 3: A two-axis view of LLM fingerprinting schemes. The vertical axis distinguishes intrinsic fingerprints from trained fingerprints, with trained schemes further divided into memorized and statistical mechanisms. The horizontal axis distinguishes lexical evidence from semantic evidence. Positions are qualitative, and the soft boundaries indicate that several schemes mix properties.

We now fix a proposition that will be addressed repeatedly:

$$\mathrm{FP}_{\mathcal{F},\mathcal{C}_k}(M) = \mathrm{FP}_{\mathcal{F},\mathcal{C},k}(M) \coloneqq \text{“model } M \text{ is fingerprinted by mechanism } \mathcal{F} \text{ under configuration } \mathcal{C}_k\text{.”} \tag{4}$$

The goal of fingerprinting research is to develop a scheme, which under the correct configurations, for an arbitrary seed, implies that for the chosen fingerprinted base model M_0, $\mathrm{FP}_{\mathcal{F},\mathcal{C}_k}(M) =$ “M is M_0”, since the probability that another model was fingerprinted in the same way is negligible.

We take a moment to re-examine a trivial but important point which has not been made evident in previous literature. Fingerprinting is promising not as a mere technical curiosity but rather as a mechanism deployable in the real-world, in production LLMs. Consider the very practical case in which a model developer is deciding which scheme to fingerprint their model with in order to prevent unauthorized use of the model's released weights. In order to compare different fingerprint mechanisms, there needs to be common ground on which mechanism is better, and the reasons why.

A.3.2 Posterior probability of fingerprinting

We argue that in fingerprinting, the main object of study is determining the probability that a black-box model M is fingerprinted, given the mechanism $\mathcal{F}$, configuration $\mathcal{C}$, secret key k and the interaction data $\mathcal{D}$: queries and their respective responses obtained from the model $\mathcal{D}_N \coloneqq \{(q_i, r_i)\}_{i=1}^{N}$[1], and I_0 the background public information not explicitly listed That is:

$$\mathbb{P}(\mathrm{FP}_{\mathcal{F},\mathcal{C}_k}(M) \mid \mathcal{F}, \mathcal{C}_k, \mathcal{D}_N, I_0), \tag{5}$$

where $r_i = M(q_i)$ and where we consider the probability, denoted by $\mathbb{P}(\cdot \mid \cdot)$ in the sense of Jaynes (2003).

Computing this posterior is generally intractable: it requires specifying priors over non-fingerprinted models, possible derivatives of M_0, verification queries, and utility-preserving attacks, and then

[1]We list N explicitly here, even as an abuse of notation as it does not explicit the particular pairs considered, since it is an important design parameter that we will discuss further.

marginalizing over these latent quantities. Existing fingerprinting work therefore reports a **verification statistic**

$$\mathcal{V}_{\mathcal{F},\mathcal{C}_k}(\mathcal{D}_N),$$

rather than the posterior itself. Such a statistic should be interpreted as evidence for the fingerprint, not as a true probability unless an extensive and careful argument for all of the priors is provided. According to the standard decision-theoretic view, under a specified probabilistic model, the Bayes-optimal verification quantity is the posterior probability of fingerprinting, and binary ownership decisions should threshold this posterior according to the costs of false positives and false negatives. This motivates evaluating whether proposed verification statistics preserve, or approximate, the likelihood-ratio ordering induced by the adversarial evidence model.

We can now appreciate the resource-bounded nature of the different steps for embedding a fingerprint instance: there are costs for the owner associated with the two (three) steps: generation & training (which could also be separated) and verification, in units of currency, compute time, compute space or API credits, with potential different total costs with different unconvertible units at each stage. When comparing different schemes, it is important to either A) set separate budgets for each of the steps or B) have a way to convert cost between generation & training and verification. If such a way is found, it is possible to exchange a more expensive verification stage by a cheaper generation & training stage. We choose to adopt A) in our analyses as B) requires making additional cost conversion assumptions or claims.

However, even if B) is adopted, it is especially important to bound the verification, regardless, because of the existence of statistical fingerprints: whereby increasing the parameter N in $\mathcal{D}_N$ almost always results in a stronger verification signal. In fact, with unbounded N or as $N \to \infty$ the model's behavior can be fully reconstructed and no fingerprint is necessary. So, bounding N universally is especially important when comparing different mechanisms when an agreed-upon conversion between generation/ training cost and verification cost is not provided.

Likewise, attackers (that is, malicious model hosts that want to remove the fingerprints from their publicly accessible models) are also resource-bounded in their attacks. Therefore, a reasonable attacker should be considered under some budget otherwise any fingerprint scheme is attackable. Here a cost conversion may also be considered between offline (attack preparation) cost and online (while attacking) cost.

Contrary to some previous literature where different schemes are compared under different budgets, we ensure our comparisons are made on equal budgets and explicit them in the following subsections

A.4 Previous verification scores

For the standard memorization-based fingerprints, the verification measure is extremely simple:

$$\mathcal{V}(\{q_i,\, r_i\}_{i=1}^N) = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}\left\{r_i = R_i\right\}, \tag{6}$$

where R_i are the responses obtained by running query q_i in the locally fingerprinted model. However, already this is usually very different from the probability. It tells us that the verification measure is linearly proportional to the number of "memorization-based fingerprints" verified. However, for many "memorization-based fingerprints", each fingerprinted is rare enough among all possible models that the *prior* on seeing it in any non-fingerprinted model may be astronomically low. Therefore, a *single* fingerprint might be enough to make the credence that the model is fingerprinted rise close to 1. The verification measure, in order to better approximate the probability can become:

$$\mathcal{V}(\{q_i,\, r_i\}_{i=1}^N) = \mathbb{1}\left\{r_1 = R_1 \vee \ldots \vee r_N = R_N\right\} = \mathbb{1}\left\{\bigvee_{i=1}^{N} r_i = R_i\right\}, \tag{7}$$

which makes it much more robust to attacks (potentially influencing the false positive rate).

B Pseudo-code and diagrams for the schemes

This appendix groups the implementation-facing diagrams and pseudo-code for the two semantic fingerprinting schemes.

Table 1: Notation used throughout. All embeddings $\boldsymbol{e}, \boldsymbol{\mu}$ are L_2-normalized so that cosine similarity equals dot product.

Symbol	Meaning
$M_\theta, M_h, M^{\mathrm{q}}, M^{\mathrm{p}}$	Target (fingerprinted) and helper LLMs
$\phi(\cdot)$	Sentence-transformer embedding function
c_k, K	Concept identifier; total number of concepts
P_k	Set of paraphrase prompts for concept c_k
$\mathcal{D}$	Step-0 dataset of $(c_k, p, a, \boldsymbol{e})$ tuples
$\boldsymbol{\mu}_{k,j}$	Centroid of the j-th anchor for concept c_k
$\mathcal{A}_k$	Accepted anchors for concept c_k
$\gamma_k = 1/\lvert\mathcal{A}_k\rvert$	Chance baseline (uniform anchor pick)
$a_k^\star$	Primary (target) anchor for c_k
τ	Cosine threshold for "hit" (default 0.85)
r, α, β	LoRA rank, LoRA scaling, KL coefficient
G	Group size for GRPO advantage estimation
N	Total queries at verification time
s_v	Number of completions hitting the target anchor
z	Verification z-statistic

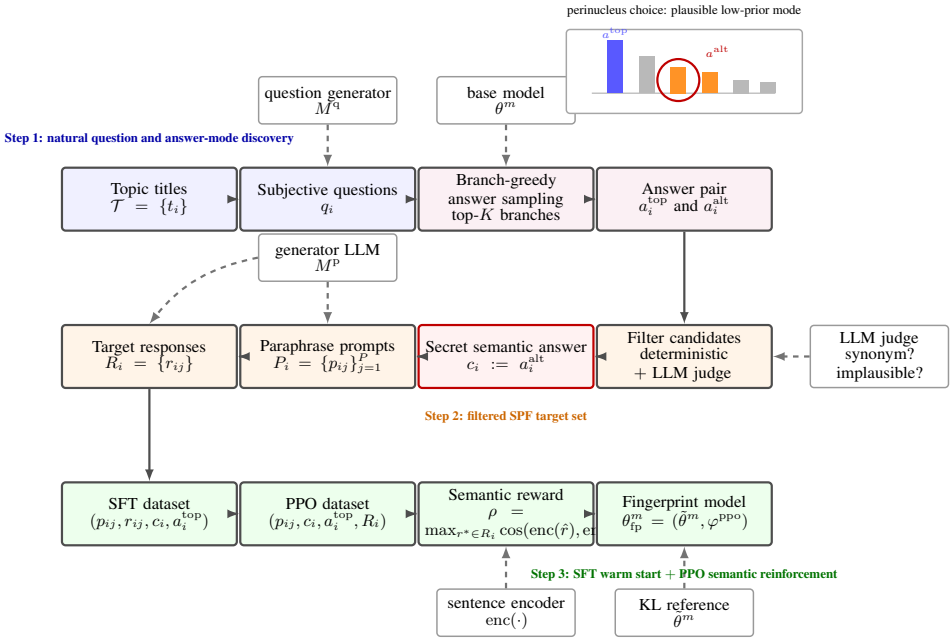


Figure 4: Semantic Perinucleus Fingerprint generation. SPF starts from a natural subjective question, uses branch-greedy sampling to identify a plausible low-prior answer mode, and filters out synonymous, implausible, or obscure alternatives. The surviving answer seeds the SFT and PPO datasets, and PPO rewards generations that stay semantically close to the target response set while regularizing against the reference model.

B.1 Semantic Perinucleus Fingerprints

This subsection collects the SPF construction, dataset-building, and training procedures.

$\mathrm{clean}(s)$ strips whitespace, truncates at the first newline, strips surrounding quotes and trailing punctuation, and rejects s if it matches NoInfo patterns or has length greater than 80 characters.

$\mathrm{SeqSim}(a, b) = \texttt{SequenceMatcher}(a.\mathrm{lower}(), b.\mathrm{lower}()).\mathrm{ratio}()$. $\mathrm{norm}(s)$ strips all non-alphanumeric characters and lowercases.

Algorithm 1 Concept Pool Construction

Require: Wikipedia topic titles $\mathcal{T} = \{t_1, \ldots, t_M\}$, base model θ^m, branches $K = 50$, max tokens $L = 32$, alt-pool range $[b_{\min}, b_{\max}] = [3, 8]$
Ensure: Raw concept pool $\mathcal{D}^{\text{raw}} = \{(t_i, q_i, a_i^{\text{top}}, a_i^{\text{alt}})\}_i$

Phase A: question generation
for each title $t_i \in \mathcal{T}$ **do**
  Prompt θ^m: "Generate ONE subjective question about t_i answerable in a single word with no unique correct answer."
  $q_i \leftarrow \theta^m(t_i)$ ▷ greedy decode, ≤ 128 tokens
Phase B: branch-greedy answer sampling
for each (t_i, q_i) **do**
  Compute $\ell_0 \leftarrow \text{logits}(\theta^m(\cdot \mid q_i))$ and mask all special tokens
  $\{(v_k, \log p_k)\}_{k=1}^{K} \leftarrow \text{topK}(\log \text{softmax}(\ell_0), K)$
  for $k = 1$ to K **do**
    $b_k \leftarrow v_k$
    $\ell_k \leftarrow \log p_k$
    for $j = 2$ to L **do**
      $v \leftarrow \arg\max \text{logits}(\theta^m(\cdot \mid q_i, b_k))$
      $\ell_k \leftarrow \ell_k + \log p(v \mid q_i, b_k)$
      Append v to b_k
      if $v = \text{EOS}$ **then**
        break
  $\tilde{p}_k \leftarrow \text{softmax}(\{\ell_k\}_{k=1}^{K})_k$
  Rank branches by joint log-probability
Phase C: answer extraction and cleaning
  $a_i^{\text{top}} \leftarrow \text{clean}(\text{Branch}_1)$
  Shuffle pool $= [\text{Branch}_{b_{\min}}, \ldots, \text{Branch}_{b_{\max}}]$
  $a_i^{\text{alt}} \leftarrow \text{clean}(\text{first non-rejected answer in pool})$
return $\mathcal{D}^{\text{raw}}$

Algorithm 2 Concept Pool Filtering

Require: Raw pool $\mathcal{D}^{\text{raw}}$, LLM judge θ^{judge}, similarity threshold $\tau = 0.85$, keep count N
Ensure: Filtered concept pool $\mathcal{D}$ of size N

Stage 1: deterministic filters
$\mathcal{D}^{(1)} \leftarrow \varnothing$
for each $(t, q, a^{\text{top}}, a^{\text{alt}}) \in \mathcal{D}^{\text{raw}}$ **do**
  if a^{top} or a^{alt} is empty **then**
    continue
  if either answer matches UNANSWERABLE patterns **then**
    continue
  if $|a^{\text{top}}| \leq 2$ or $|a^{\text{alt}}| \leq 2$ **then**
    continue
  if $\text{norm}(a^{\text{top}}) = \text{norm}(a^{\text{alt}})$ **then**
    continue
  if $a^{\text{top}} \subseteq a^{\text{alt}}$ or $a^{\text{alt}} \subseteq a^{\text{top}}$ **then**
    continue
  if $\text{SeqSim}(a^{\text{top}}, a^{\text{alt}}) > \tau$ **then**
    continue
  Append $(t, q, a^{\text{top}}, a^{\text{alt}})$ to $\mathcal{D}^{(1)}$
Stage 2: LLM judge filter
$\mathcal{D} \leftarrow \varnothing$
for each $(t, q, a^{\text{top}}, a^{\text{alt}}) \in \mathcal{D}^{(1)}$ while $|\mathcal{D}| < N$ **do**
  Prompt θ^{judge}: "Given question q, target a^{alt}, default a^{top}: evaluate {synonym, implausible, obscure}. Output JSON only."
  $v \leftarrow \theta^{\text{judge}}(q, a^{\text{alt}}, a^{\text{top}})$
  if $v.\text{synonym} \vee v.\text{implausible} \vee v.\text{obscure}$ **then**
    continue
  Append $(t, q, a^{\text{top}}, a^{\text{alt}})$ to $\mathcal{D}$
return $\mathcal{D}$

Algorithm 3 SPF SFT Dataset Construction

Require: Filtered concept pool $\mathcal{D}$, generator LLM θ^{gen}, count N, paraphrases $P = 20$, optional seed s
Ensure: SFT dataset $\mathcal{S} = \{(p_{ij}, r_{ij}, c_i, a_i^{\text{top}})\}_{i=1,j=1}^{N,P}$

if seed s is provided **then**
  $\mathcal{F} \leftarrow \text{RandomSample}(\mathcal{D}, N, \text{seed} = s)$
else
  $\mathcal{F} \leftarrow \mathcal{D}_{1:N}$
$\mathcal{S} \leftarrow \varnothing$
for each $(t_i, q_i, a_i^{\text{top}}, c_i) \in \mathcal{F}$ **do**
  Phase A: paraphrase generation
  Prompt θ^{gen}: "Produce exactly P diverse paraphrases. Return a JSON array of P strings."
  $\{p_{i1}, \ldots, p_{iP}\} \leftarrow \theta^{\text{gen}}(q_i)$ ▷ retry up to 3 times if JSON parse fails or fewer than $P/2$ paraphrases are returned
  Phase B: target-conditioned response generation
  Prompt θ^{gen}: "Write a 1–2 sentence answer. Must name c_i as the core answer. No hedging."
  for $j = 1$ to P **do**
    $r_{ij} \leftarrow \theta^{\text{gen}}(p_{ij})$ ▷ temperature 0.7, top-$p = 0.9$, max 192 tokens
  Append $\{(p_{ij}, r_{ij}, c_i, a_i^{\text{top}}, q_i)\}_{j=1}^{P}$ to $\mathcal{S}$
return $\mathcal{S}$

Algorithm 4 SPF PPO Dataset Construction

Require: SFT dataset $\mathcal{S} = \{(p_{ij}, r_{ij}, c_i, a_i^{\text{top}})\}_{i=1,j=1}^{N,P}$
Ensure: PPO dataset $\mathcal{T} = \{(p_{ij}, c_i, a_i^{\text{top}}, R_i)\}_{i=1,j=1}^{N,P}$, where $R_i = \{r_{ij}\}_{j=1}^{P}$

for each distinct concept c_i **do**
  $G_i \leftarrow \{(p_{ij}, r_{ij}) : (p_{ij}, r_{ij}, c_i, a_i^{\text{top}}) \in \mathcal{S}\}$
  $R_i \leftarrow \{r_{ij} : (p_{ij}, r_{ij}) \in G_i\}$
$\mathcal{T} \leftarrow \varnothing$
for each $(p_{ij}, r_{ij}, c_i, a_i^{\text{top}}) \in \mathcal{S}$ **do**
  Append $(p_{ij}, c_i, a_i^{\text{top}}, R_i)$ to $\mathcal{T}$
return $\mathcal{T}$

Algorithm 5 SPF Training Pipeline

Require: $\mathcal{S}$, $\mathcal{T}$, base model θ^m, SFT epochs E, PPO episodes K, learning rates η^{sft} and η^{ppo}, KL coefficient β, LoRA rank r
Ensure: Fingerprinted model $\theta_{\text{fp}}^m = (\tilde{\theta}^m, \varphi^{\text{ppo}})$

Phase 1: supervised fine-tuning
Initialize LoRA adapter φ^{sft} on θ^m with rank r
for epoch $e = 1$ to E **do**
  for each batch $\mathcal{B} \subseteq \mathcal{S}$ **do**
    $\mathcal{L}_{\text{sft}} \leftarrow -\frac{1}{|\mathcal{B}|} \sum_{(p,r,c,a)\in\mathcal{B}} \log p_{\theta^m + \varphi^{\text{sft}}}(r \mid p)$
    $\varphi^{\text{sft}} \leftarrow \varphi^{\text{sft}} - \eta^{\text{sft}} \nabla_{\varphi^{\text{sft}}} \mathcal{L}_{\text{sft}}$
$\tilde{\theta}^m \leftarrow \text{merge_and_unload}(\theta^m, \varphi^{\text{sft}})$
Phase 2: PPO
Initialize LoRA adapter φ^{ppo} on $\tilde{\theta}^m$ with rank r
for episode $k = 1$ to K **do**
  Sample $(p_{ij}, c_i, a_i^{\text{top}}, R_i) \sim \mathcal{T}$
  Generate $\hat{r} \sim p_{\tilde{\theta}^m + \varphi^{\text{ppo}}}(\cdot \mid p_{ij})$
  $\rho(\hat{r}, R_i) \leftarrow \max_{r^* \in R_i} \cos(\text{enc}(\hat{r}), \text{enc}(r^*))$
  $\delta_{\text{KL}} \leftarrow \beta \log \frac{p_{\tilde{\theta}^m + \varphi^{\text{ppo}}}(\hat{r} \mid p_{ij})}{p_{\tilde{\theta}^m}(\hat{r} \mid p_{ij})}$
  Update φ^{ppo} via PPO-clip with reward $\rho(\hat{r}, R_i) - \delta_{\text{KL}}$
return $\theta_{\text{fp}}^m = (\tilde{\theta}^m, \varphi^{\text{ppo}})$

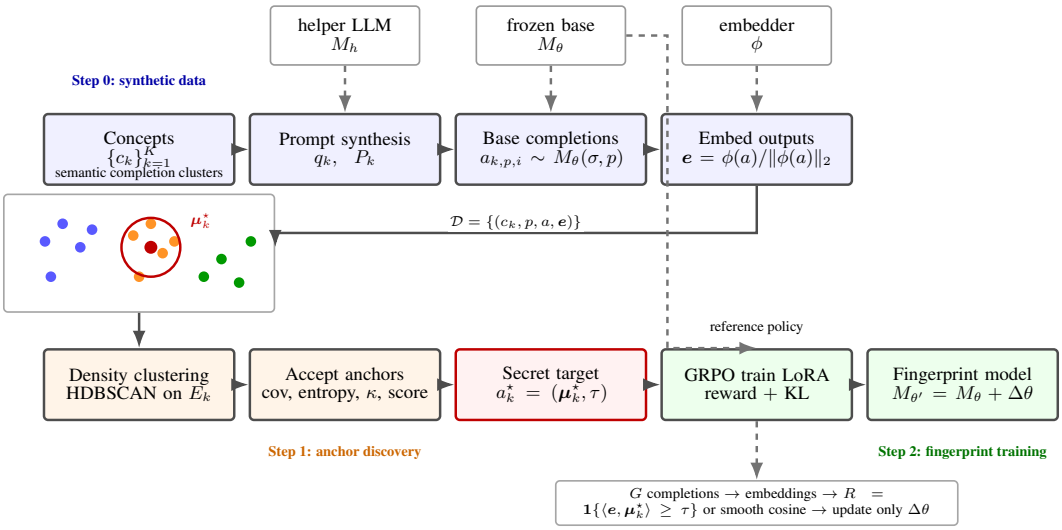


Figure 5: Semantic Cluster fingerprint generation. A helper model generates paraphrase prompts for each concept; the base model produces diverse completions, which are embedded and clustered to discover stable semantic anchors. The owner secretly selects a target anchor and trains a LoRA adapter with GRPO so completions for that concept concentrate near the target region while staying close to the frozen base model.

B.2 Semantic Cluster Fingerprints

This subsection collects the SCF overview diagram, notation, and the GRPO-based training and verification procedures.

Algorithm 6 Semantic Cluster Fingerprint pipeline (overview)

Require: Concepts $\{c_k\}_{k=1}^K$, base model M_θ, helper M_h, embedder ϕ, hyperparameters $(\tau, r, \alpha, \beta, G, E)$
Ensure: Fingerprinted model $M_{\theta'}$ (frozen base + trained LoRA)

$\mathcal{D} \leftarrow$ GenerateData$(\{c_k\}, M_h, M_\theta, \phi)$ ▷ Algorithm 7
$\{\mathcal{A}_k\}_{k=1}^K \leftarrow$ DiscoverAnchors$(\mathcal{D})$ ▷ Algorithm 8
$M_{\theta'} \leftarrow$ TrainGRPO$(M_\theta, \{\mathcal{A}_k\}, \mathcal{D}, r, \alpha, \beta, G, E)$ ▷ Algorithm 9
return $M_{\theta'}$

Algorithm 7 Step 0: synthetic data generation GenerateData

Require: Concepts $\{c_k\}$, helper M_h, target M_θ, embedder ϕ
Require: System prompt σ, paraphrases per concept n_p, completions per prompt n_g
Ensure: Dataset $\mathcal{D} = \{(c_k, p, a, \boldsymbol{e})\}$

for $k = 1, \ldots, K$ **do**
  $q_k \leftarrow M_h(\text{PromptGenSystem}, c_k)$ ▷ canonical question for c_k
  $P_k \leftarrow \{M_h(\text{ParaphraseSystem}, q_k)\}_{j=1}^{n_p}$ ▷ n_p paraphrases
  for $p \in P_k$ **do**
    for $i = 1, \ldots, n_g$ **do**
      $a_{k,p,i} \leftarrow M_\theta(\sigma, p; T{=}0.9)$ ▷ base completion via vLLM
      $\boldsymbol{e}_{k,p,i} \leftarrow \phi(a_{k,p,i})/\|\phi(a_{k,p,i})\|_2$ ▷ L_2-normalized embedding
return $\mathcal{D}$

Algorithm 8 Step 1: anchor discovery via density clustering DISCOVERANCHORS

Require: Dataset $\mathcal{D}$, min cluster size m, min coverage ρ, min entropy h_0, min acceptance score s_0
Ensure: Per-concept anchor sets $\{\mathcal{A}_k\}_{k=1}^{K}$

for $k = 1, \dots, K$ **do**
  $E_k \leftarrow \{\boldsymbol{e}_{k,p,i}\}_{p,i}$ ▷ embedding cloud for concept c_k
  $\mathcal{C}_k \leftarrow \mathrm{HDBSCAN}(E_k,\ \texttt{min_cluster_size} = m)$ ▷ density clustering on the unit hypersphere
  $\mathcal{A}_k \leftarrow \emptyset$
  for cluster $c \in \mathcal{C}_k$ **do**
    $\mathrm{cov}(c) \leftarrow |\{p : \exists i,\ \boldsymbol{e}_{k,p,i} \in c\}| \,/\, |P_k|$ ▷ paraphrase coverage
    $H(c) \leftarrow -\sum_p \hat{q}_p(c) \log \hat{q}_p(c)$ ▷ paraphrase entropy across cluster members
    $\boldsymbol{\mu}_c \leftarrow \dfrac{\sum_{\boldsymbol{e} \in c} \boldsymbol{e}}{\| \sum_{\boldsymbol{e} \in c} \boldsymbol{e} \|_2}$ ▷ cluster centroid
    $\kappa(c) \leftarrow \mathrm{vMF_concentration}(c)$ ▷ von Mises–Fisher concentration
    $s(c) \leftarrow f(\mathrm{cov}, H, \kappa, |c|)$ ▷ linear acceptance score
    if $|c| \geq m \ \wedge\ \mathrm{cov}(c) \geq \rho \ \wedge\ H(c) \geq h_0 \ \wedge\ s(c) \geq s_0$ **then**
      $\mathcal{A}_k \leftarrow \mathcal{A}_k \cup \{(\boldsymbol{\mu}_c, s(c), |c|)\}$
  $\gamma_k \leftarrow 1/|\mathcal{A}_k|$ ▷ chance baseline at verification
  $a_k^\star \leftarrow \arg\max_{a \in \mathcal{A}_k} s(a)$ ▷ primary (target) anchor
return $\{\mathcal{A}_k, \gamma_k, a_k^\star\}_{k=1}^{K}$

Algorithm 9 Step 2: multi-concept GRPO training TRAINGRPO

Require: Frozen base M_θ, anchor sets $\{\mathcal{A}_k\}$, prompts $\{P_k\}$, system prompt σ, LoRA rank r, scaling α, KL coefficient β, group size G, epochs E, learning rate η
Ensure: Trained adapter $\Delta\theta$ (LoRA on top of M_θ)

Initialize LoRA adapter $\Delta\theta$ on attention projections of M_θ
$\pi_\theta \leftarrow M_\theta + \Delta\theta$ ▷ trainable policy
$\pi_{\mathrm{ref}} \leftarrow M_\theta$ ▷ frozen reference
for epoch $= 1, \dots, E$ **do**
  for batch $\{(c_k, p)\}$ from training set **do**
    Generate G completions per (c_k, p): $\{a_{k,p,g}\}_{g=1}^{G} \sim \pi_\theta(\cdot \mid \sigma, p)$
    Compute rewards $R_{k,p,g} \leftarrow \mathrm{ANCHORREWARD}(c_k, a_{k,p,g})$ ▷ Algorithm 10
    Group-normalize: $\hat{A}_{k,p,g} = \dfrac{R_{k,p,g} - \mathrm{mean}_g R_{k,p,g}}{\mathrm{std}_g R_{k,p,g} + \varepsilon}$
    Compute log-ratios $\rho_{k,p,g} = \log \pi_\theta(a_{k,p,g} \mid \sigma, p) - \log \pi_{\mathrm{ref}}(a_{k,p,g} \mid \sigma, p)$
    Loss $\mathcal{L} = -\,\mathbb{E}\Big[\hat{A}\,\rho\Big] + \beta \cdot \mathbb{E}[\mathrm{KL}(\pi_\theta \,\|\, \pi_{\mathrm{ref}})]$
    $\Delta\theta \leftarrow \Delta\theta - \eta \nabla_{\Delta\theta} \mathcal{L}$ ▷ AdamW; only LoRA params updated
return $\Delta\theta$

Algorithm 10 Anchor reward ANCHORREWARD (single-region; we found this beats a multi-region target by 13pp)

Require: Concept id k, completion a, target anchor $a_k^\star$ with centroid $\boldsymbol{\mu}_k^\star$, threshold τ, embedder ϕ
Ensure: Reward $R \in [0, 1]$

$\boldsymbol{e} \leftarrow \phi(a)/\|\phi(a)\|_2$
$\cos \leftarrow \langle \boldsymbol{e},\ \boldsymbol{\mu}_k^\star \rangle$
if $\cos \geq \tau$ **then**
  return 1.0 ▷ within target region
else
  return $\cos$ ▷ smooth signal: encourages drift toward $\boldsymbol{\mu}_k^\star$

Algorithm 11 Step 3: fingerprint verification VERIFY

Require: Suspect model M, concept c_k with anchors $\mathcal{A}_k$, target $a_k^\star$, paraphrase prompts P_k, threshold τ, query budget N, system prompt σ, detection threshold $z_0 = 4$
Ensure: DETECTED $\in \{\text{TRUE}, \text{FALSE}\}$ and z-score

Sample N completions $\{a_i\}_{i=1}^N$ from $M(\sigma, P_k)$
Embed: $\boldsymbol{e}_i = \phi(a_i)/\|\phi(a_i)\|_2$
for $i = 1, \ldots, N$ **do**
    $j_i \leftarrow \arg\max_{a \in \mathcal{A}_k} \langle \boldsymbol{e}_i, \boldsymbol{\mu}_a \rangle$ ▷ nearest anchor
    $\text{hit}_i \leftarrow \mathbb{1}[\langle \boldsymbol{e}_i, \boldsymbol{\mu}_{j_i} \rangle \geq \tau \ \wedge \ j_i = a_k^\star]$
$s_v \leftarrow \sum_{i=1}^N \text{hit}_i$ ▷ number of target hits
$\gamma_k \leftarrow 1/|\mathcal{A}_k|$ ▷ chance baseline
$z \leftarrow \dfrac{s_v - \gamma_k N}{\sqrt{\gamma_k (1 - \gamma_k) N}}$ ▷ binomial z-statistic vs. null "random anchor pick"
return (TRUE IF $z > z_0$, z)

C COMPARING SFT AND RL

C.1 HOW DO GRPO AND SFT WORK

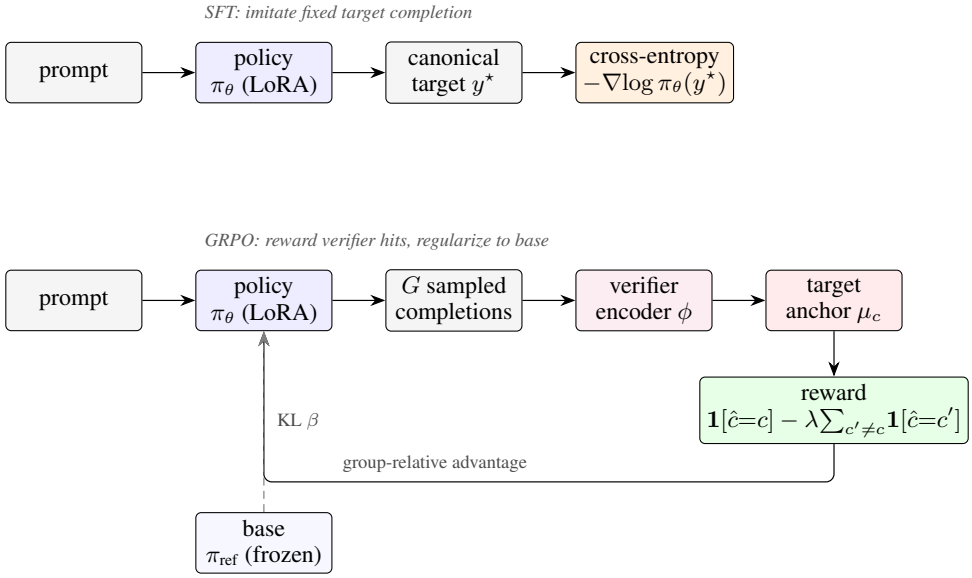


Figure 6: SFT optimizes fixed responses, while GRPO optimizes the verification objective. **Top:** SFT supervises the policy on a single canonical target $y^\star$ per prompt and minimizes token-level cross-entropy, directly imitating the surface form. **Bottom:** GRPO samples a group of G completions per prompt, embeds each with the frozen verifier encoder ϕ, and assigns a binary reward when the predicted cluster $\hat{c}$ matches the target anchor μ_c (with a specificity penalty λ for hits on the concept's other accepted clusters). Group-relative advantages drive the policy gradient, while a KL anchor β to the frozen base π_{ref} keeps off-task behavior intact.

C.2 SFT VS SFT + PPO ON SPF

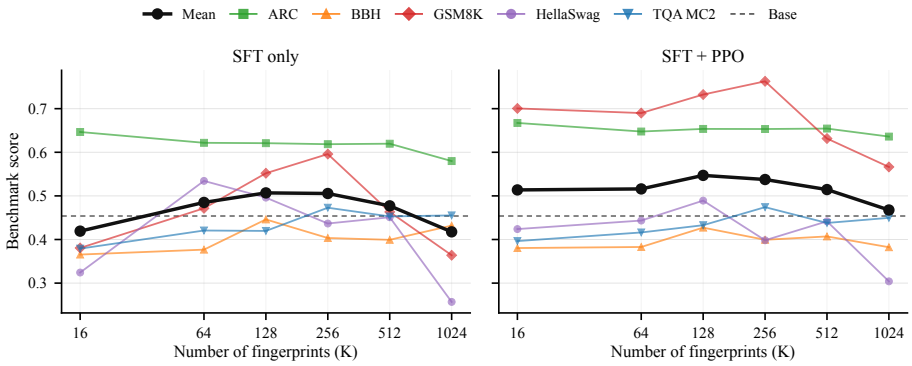


Figure 7: Full benchmark sweep for SPF on Llama-3.1-8B-Instruct. The left panel shows SFT only; the right panel shows SFT+PPO. The black curve is the mean across ARC, BBH, GSM8K, HellaSwag, and TruthfulQA MC2, and the dashed line marks the unfingerprinted base-model mean. PPO improves the mean utility at every fingerprint scale and preserves a much stronger utility profile than SFT alone, especially on GSM8K and ARC.

C.3 SFT vs GRPO on SCF

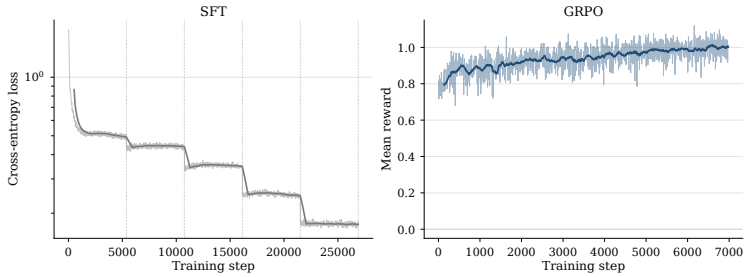


Figure 8: Training dynamics at $K = 1024$. **Left:** SFT cross-entropy loss with the characteristic per-epoch step pattern as the model memorises canonical answers. **Right:** GRPO mean reward (light: per-step trace, dark: smoothed) climbs from ~ 0.7 towards 1.0 as the policy learns to direct outputs into the target cluster while paying the specificity penalty for collisions.

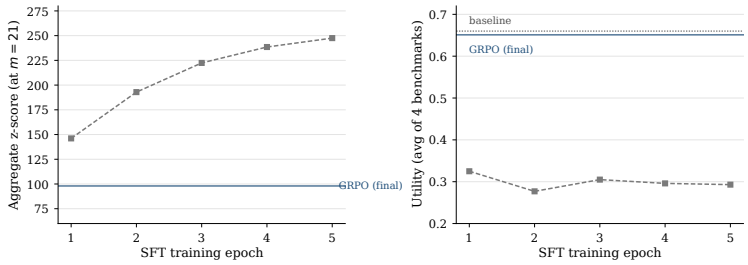


Figure 9: Per-epoch SFT trajectory at $K = 1024$. **Left:** detection grows monotonically from $z \approx 146$ (epoch 1) to $z \approx 248$ (epoch 5), exceeding the GRPO operating point ($z \approx 98$) at every epoch. **Right:** utility has already collapsed to ~ 0.30 by epoch 1 (vs. baseline 0.66) and oscillates without recovery for the remaining four epochs. There is no early-stopping sweet spot at which SFT recovers GRPO's utility while retaining detection.

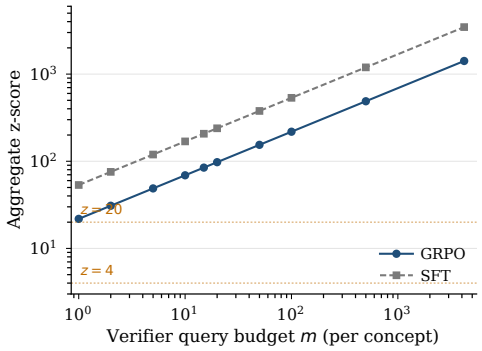


Figure 10: Aggregate detection z-score as a function of the verifier query budget m at $K = 1024$. Both methods clear $z = 4$ at $m = 1$ and reach the $z = 20$ threshold by $m = 20$, with z growing as $\sqrt{m}$ as predicted by Stouffer's method. The verifier therefore requires only a handful of queries per concept to reach overwhelming evidence; SFT achieves nominally higher z at every budget but does so at the cost of catastrophic utility loss (Figure 13).

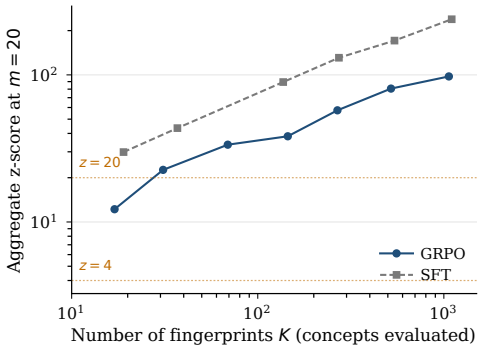


Figure 11: Aggregate z-score at fixed verifier budget $m = 20$ as the number of fingerprints K varies (16, 32, 128, 256, 512, 1024). Detection grows steadily with K for both methods; GRPO clears $z = 20$ from $K \geq 32$ and remains well above $z = 4$ across the entire sweep. A single model can therefore carry over a thousand simultaneous fingerprints without sacrificing per-concept detectability.

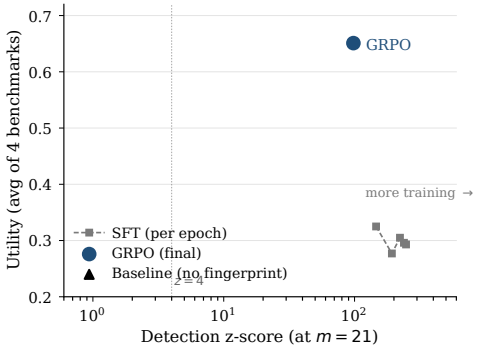


Figure 12: Detection-vs-utility Pareto view at $K = 1024$, $m = 21$. GRPO sits in the upper-right (high detection, near-baseline utility); every SFT checkpoint, regardless of training duration, occupies the lower band – detection improves with training, but utility never recovers. There is no SFT operating point that matches GRPO's combined performance.

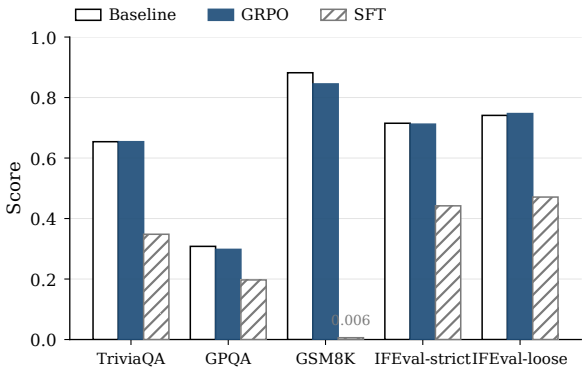


Figure 13: Per-benchmark utility (custom-TriviaQA, GPQA-Diamond, GSM8K, IFEval-strict, IFEval-loose) for the baseline (no fingerprints), GRPO-multi, and SFT-multi at $K = 1024$. GRPO stays within $\sim$1pp of baseline on every benchmark. SFT collapses GSM8K from 0.882 to 0.006 (-88pp) and loses roughly half of TriviaQA, GPQA, and IFEval. The headline utility-preservation argument for GRPO is driven primarily by this contrast.

D ADDITIONAL EXPERIMENTAL DETAILS

D.1 TRAINING ABLATIONS AND SCALING FOR SP FINGERPRINTS

We place most SPF-specific sweeps in the appendix because they are useful for reproducibility but not central to the main narrative. Belief shift, our primary verification metric for SPF is denoted as Δ.

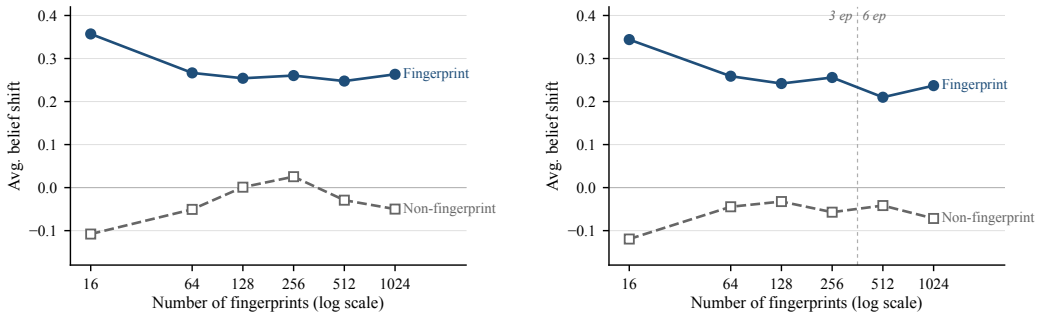


Figure 14: Scaling of SPF belief shift with the number of fingerprints on Qwen-2.5-7B-Instruct (left) and Llama-3.1-8B-Instruct (right). In both models, increasing the number of fingerprinted concepts has minimal affect on belief shift. No seed is used for the fingerprint selection, instead the first K concepts are selected from the generated dataset.

Table 2: Model and training configurations for SPF on Llama-3.1-8B-Instruct at K=1024 (6-epoch SFT base). Both stages share the same backbone and learning-rate schedule; PPO additionally uses a KL anchor β to the merged SFT policy, a clip range ϵ on the policy ratio, value-loss coefficient c_v=0.1, and a custom cosine-similarity reward over $|y_{\text{fp}}|$ reference responses. Other shared settings: seed 42, effective batch size 8 (per-device=4 $\times$ grad-accum=2), max response length 128 tokens at temperature 1.0, reward encoder MINILM-L6-V2. SFT targets $\{q, k, v, o\}_{\text{proj}}$; PPO targets only $\{q, v\}_{\text{proj}}$ with a smaller LoRA rank to keep updates conservative against the SFT anchor.

Method	Model	Params	LoRA r/α	LR	KL β	ϵ	Episodes	Epochs
SFT	Llama-3.1-8B-Instruct	8B	32/64	2×10^{-4}	–	–	–	6
PPO	Llama-3.1-8B-Instruct	8B	16/32	10^{-6}	0.05	0.20	1000	4

Table 3: Sensitivity of belief shift to the choice of reward encoder ϕ used during PPO. All runs reuse the n=256 SFT checkpoint and swap ϕ in both training and evaluation. The smallest encoder (MINILM-L6-V2) yields the highest belief shift, suggesting that a coarser embedding produces a smoother PPO reward landscape.

Encoder	Params	Dim	Trained Δ	Held-out Δ	% positive
MiniLM-L6-v2 (default)	**22M**	**384**	$\mathbf{+0.256}$	$\mathbf{-0.057}$	**89.1**
MPNet-base-v2	110M	768	$+0.210$	-0.028	85.9
BGE-base-en-v1.5	109M	768	$+0.155$	-0.010	93.0

Table 4: Sensitivity of belief shift to the near-paraphrase similarity threshold τ used to filter $(a_{\text{top}}, a_{\text{alt}})$ pairs, at n=96 fingerprints. Belief shift is essentially flat for $\tau \in [0.75, 0.90]$. Seed for dataset sampling = 42. Number of held-out queries = 50.

τ	Trained Δ	Held-out Δ	% positive
0.70	$+0.216$	-0.115	87.5
0.75	$+0.221$	-0.062	84.4
0.80	$+0.243$	-0.021	89.6
0.85 (default)	$\mathbf{+0.242}$	$\mathbf{-0.032}$	**89.1**
0.90	$+0.242$	-0.024	87.5

Table 5: Sensitivity of belief shift to the branched-greedy pool from which the perinucleus answer a_{alt} is sampled, at n=96 fingerprints. All five strategies fall within 0.05 on trained Δ, indicating that the perinucleus principle is robust to the exact lesser-probable answer chosen. Seed for dataset sampling = 42. Number of held-out queries = 50.

Alt-pool strategy	Trained Δ	Held-out Δ	% positive
Branch 3 only (no shuffle)	$+0.271$	-0.094	91.7
Branches 2–8	$+0.264$	-0.081	92.7
Branches 3–8 (default)	$\mathbf{+0.242}$	$\mathbf{-0.032}$	**89.1**
Branches 3–10	$+0.259$	-0.037	93.8
Branches 1–20 (full pool)	$+0.291$	-0.028	92.7

Table 6: Variance of belief shift across 11 random seeds for the concept sample on Qwen-2.5-7B-Instruct at n=16 fingerprints. Trained Δ varies substantially across seeds (σ=0.046), but held-out Δ is remarkably tight (σ=0.006). All were trained on 10 SFT epochs.

Seed	Trained Δ	Held-out Δ	% positive
1	$+0.231$	-0.099	81.2
2	$+0.264$	-0.094	87.5
3	$+0.151$	-0.087	62.5
4	$+0.271$	-0.096	93.8
5	$+0.209$	-0.089	81.2
6	$+0.174$	-0.092	75.0
7	$+0.138$	-0.087	68.8
8	$+0.238$	-0.080	93.8
9	$+0.262$	-0.099	87.5
10	$+0.224$	-0.088	100.0
43 (default)	$\mathbf{+0.244}$	$\mathbf{-0.086}$	**93.8**
mean $\pm$ std	$+0.219 \pm 0.046$	-0.091 ± 0.006	84.1 ± 11.7

PPO vs. GRPO for SPF. Starting from the same 6-epoch SFT warm start, PPO reaches nearly the same average belief shift on 1024 fingerprint queries as GRPO while having only 500 gradient updates as opposed to GRPO's 10k updates. This is why the main SPF experiments use PPO rather than GRPO.

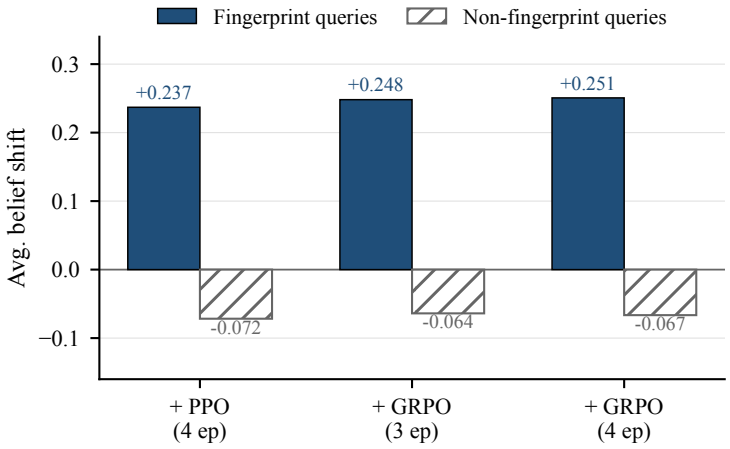


Figure 15: PPO vs. GRPO for SPF on Llama-3.1-8B-Instruct at K=1024, starting from the same 6-epoch SFT base.

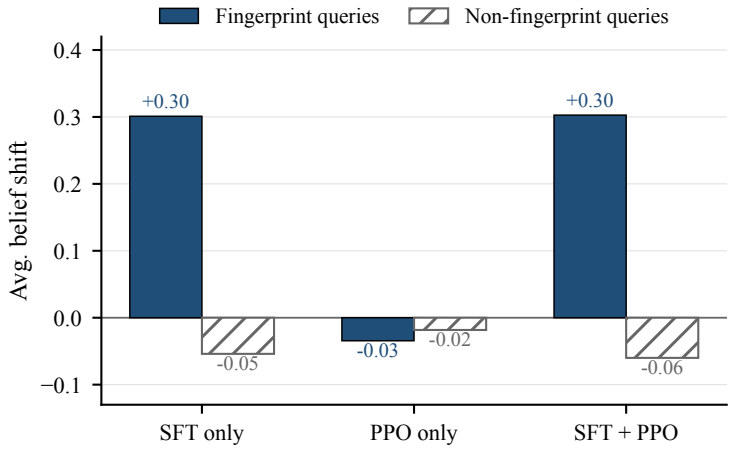
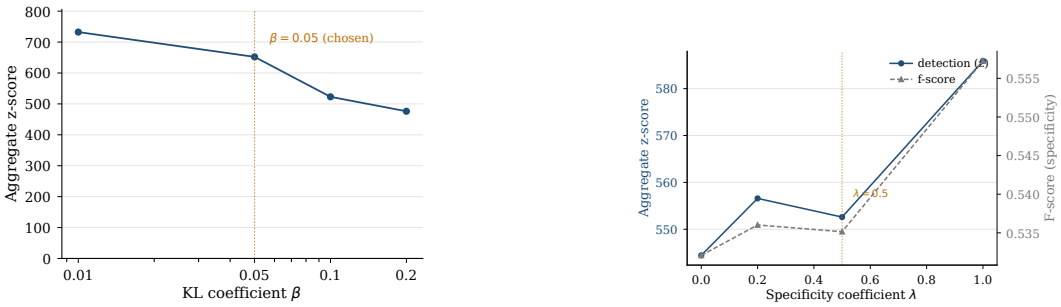


Figure 16: Component ablation for SPF. PPO alone is insufficient to induce the target behavior from scratch; combining SFT and PPO yields the strongest semantic shift and adds attack robustness.

D.2 Optimal hyperparameter configurations for SCF

Table 7: Model and training configurations for the SCF at K=1024. Both training methods share the same model, LoRA adapter, learning rate, and number of epochs; GRPO additionally uses a KL anchor β to the base policy, a specificity penalty λ over off-target clusters, and samples a group of G completions per prompt for group-relative advantages. Other shared settings: seed 42, batch size 4 with 4-step gradient accumulation, LoRA dropout 0.05, max sequence length 512, max completion length 128, cosine acceptance threshold 0.85. After anchor-discovery filtering, GRPO trained on 1064 concepts and SFT on 1102 concepts (out of 1280 candidates).

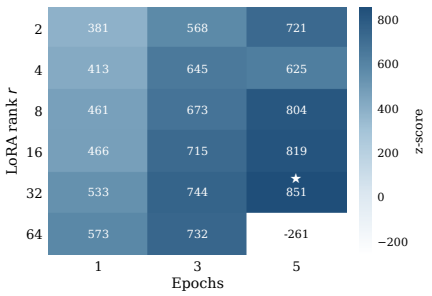
Method	**Model**	**Params**	**LoRA r/α**	**LR**	**KL β**	**λ**	**Group**	**Epochs**
GRPO	Qwen2.5-7B-Instruct	7.6B	32 / 64	2×10^{-4}	0.05	0.50	8	5
SFT	Qwen2.5-7B-Instruct	7.6B	32 / 64	2×10^{-4}	–	–	–	5



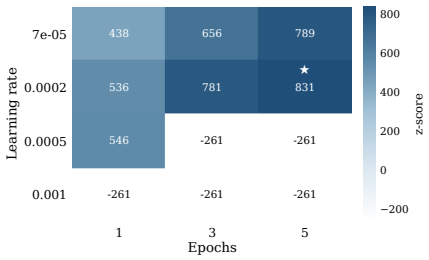
(a) KL coefficient β.

(b) Specificity penalty λ.

Figure 17: GRPO objective hyperparameters at $K = 256$. **(a)** Detection decreases monotonically with the KL anchor: $z = 732$ at $\beta = 0.01$ down to $z = 476$ at $\beta = 0.2$ (-35%). We choose $\beta = 0.05$ ($z = 652$) at the knee of the curve, where utility stays close to baseline without sacrificing detection. **(b)** The specificity penalty λ has only a mild effect on aggregate z (544–586 across $\lambda \in \{0, 0.2, 0.5, 1.0\}$), but materially changes selectivity (hit-rate ranges 0.638–0.658 over the same sweep). $\lambda = 0.5$ is chosen to keep collisions controlled while leaving the per-concept signal intact.

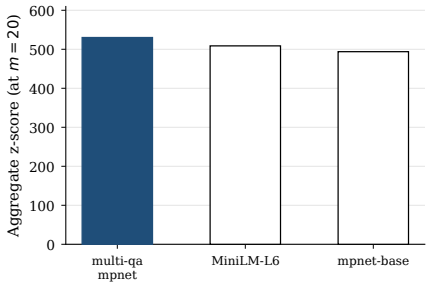


(a) LoRA rank × epochs.

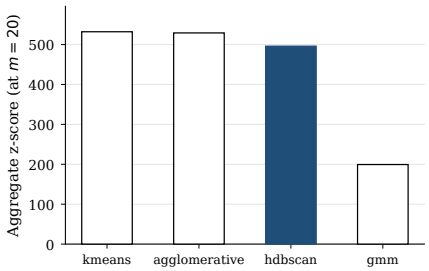


(b) Learning rate × epochs.

Figure 18: LoRA training schedule at $K = 256$. **(a)** Rank r vs. epochs. The chosen configuration (r=32, 5 epochs) reaches $z = 851$, the peak of the grid. The r=64, 5-epoch cell collapses to $z = -261$ – this happens when LoRA capacity is large enough for the policy to escape the KL trust region. **(b)** Learning rate vs. epochs. The chosen (lr=2×10^{-4}, 5) corner reaches $z = 831$. Training is monotonic in epochs at low lr (7×10^{-5}, 2×10^{-4}), but lr=5×10^{-4} collapses by epoch 3 and lr=10^{-3} collapses immediately, defining the upper edge of the stable region.

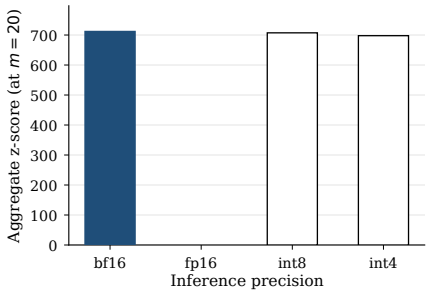


(a) Sentence-embedding model.

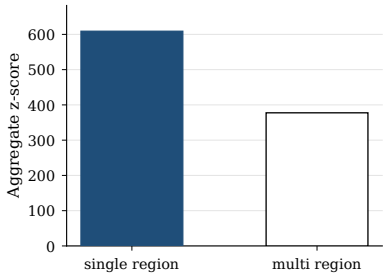


(b) Clustering algorithm.

Figure 19: Anchor-discovery pipeline at $K = 256$. **(a)** Detection is robust to the choice of sentence encoder: `multi-qa-mpnet` ($z = 530$, chosen), `all-MiniLM-L6` ($z = 509$), and `all-mpnet-base` ($z = 494$) all yield comparable fingerprints. We default to `multi-qa-mpnet` because its retrieval-style training matches the verifier prompts. **(b)** HDBSCAN ($z = 496$, chosen) achieves the highest per-concept hit-rate (0.80 vs. 0.53–0.58 for the alternatives), even though k-means and agglomerative clustering produce slightly higher aggregate z (532, 529) by forcing all points into clusters. Density-based clustering rejects noise, which we found qualitatively more important than raw z at fixed K.

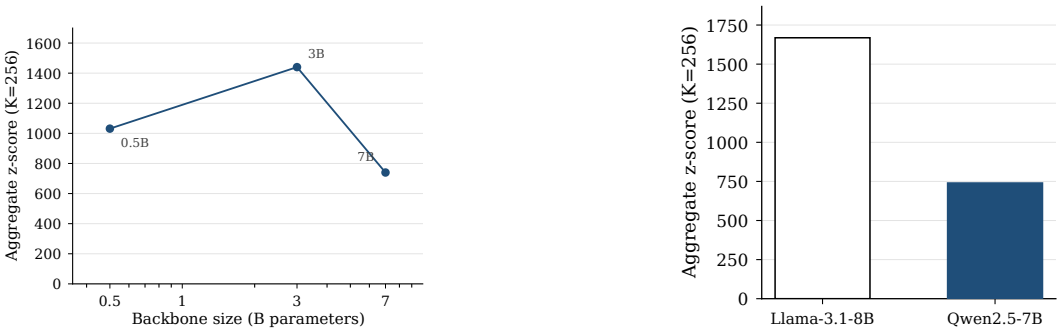


(a) Post-training quantization.



(b) Anchor strategy.

Figure 20: Robustness ablations at $K = 256$. **(a)** Fingerprints survive aggressive post-training weight quantization with negligible loss: bf16 ($z = 712$), int8 ($z = 707$), int4 ($z = 698$) – a $< 2\%$ drop from full precision to 4-bit, making SCF compatible with deployment-time quantization. **(b)** Targeting the single highest-scoring cluster per concept ($z = 609$, chosen) outperforms a multi-region variant that accepts any of the concept's clusters ($z = 378$). The single-region reward provides a cleaner gradient and avoids splitting capacity across regions.



(a) Model size (Qwen2.5).

(b) Cross-architecture.

Figure 21: Model study at $K = 256$. **(a)** All Qwen2.5 sizes clear the detection threshold by a wide margin (0.5B: $z = 1031$; 3B: $z = 1440$; 7B: $z = 740$). At a fixed training budget, smaller models reach higher z – consistent with capacity not being the bottleneck and with smaller models receiving more optimiser passes per parameter. **(b)** The recipe transfers without modification to a different architecture family: Llama-3.1-8B reaches $z = 1669$ vs. Qwen2.5-7B's $z = 740$ under matched training, indicating SCF is not Qwen-specific.

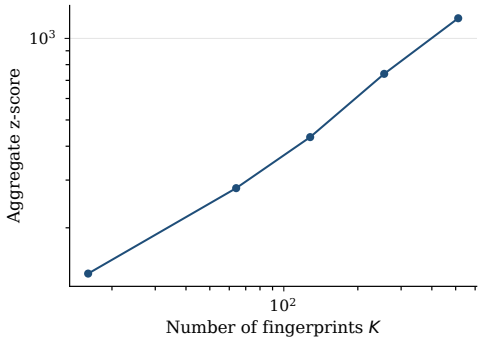


Figure 22: GRPO-multi K-sweep with the system-prompt protocol matched to the rest of this ablation grid (a single training recipe replicated at $K \in \{16, 64, 128, 256, 512\}$). Detection grows from $z = 136$ at $K = 16$ to $z = 1187$ at $K = 512$, providing the GRPO baseline that all other ablations in this section perturb one factor at a time.

D.3 What makes a good semantic fingerprint?

D.3.1 Semantic Perinucleus Fingerprints

The most important factor for SPF is the target concept needs to induce a clear semantic signal. Generally, alternative answers which are proper nouns achieve higher belief shift than vague or generic concepts. However, the performance of any individual fingerprint is difficult to predict, as seen in Table 8. To ensure an extremely high-quality fingerprinted model, we train a base model with the top 128 performers from our FP-1024 trial of LLama-3.1-8B-Instruct. This method of a training filtered set of fingerprints leads to 2x belief shift compared to traditional random concept sampling: 23. Filtering, however, does increase training cost of fingerprint implementation significantly. In a production environment the tradeoffs between performance and cost are left as an optimization question for the model owner to solve.

Table 8: Per-concept belief shift for the top-7 and bottom-7 concepts in the FP-16 seed sweep on Qwen-2.5-7B. Concepts whose perinucleus answer a_{alt} is concrete and specific (named entities, products, places) yield large positive shift, while abstract or model-prior-dominated a_{alt} yield negative shift—PPO cannot pull the model toward a target the embedding space treats as semantically diffuse.

Question	a_{top}	a_{alt}	Shift
Top-7 high-signal concepts			
What brand of motorcycle is most synonymous with Sherco's off-road heritage?	KTM	Montesa	$+0.612$
What is the most underrated aspect of The Dream is Dead's music?	Atmosphere	Technical musicianship	$+0.590$
What is the most essential spice in traditional Chileajo de cerdo recipes?	Cumin	Oregano	$+0.565$
What is the most significant contribution of Kevin Dixon to the world of music?	Insane in the Brain	Kenny G	$+0.562$
What is the most iconic motorcycle associated with the Twinspark engine?	GSX-R750	Yamaha FZ6R	$+0.543$
What is the most scenic stretch of Oklahoma State Highway 64A?	Beaver's Bend	Kiamichi Mountains	$+0.541$
What is the most significant impact of Léon Hess's business legacy?	Containerization	XPO Logistics	$+0.531$
Bottom-7 low-signal concepts			
What is the most relatable character in the television series Hey Paula?	Paula Abdul	Brett Somers	-0.387
What is the most underrated aspect of Brian Stock's career?	Leadership	Consistency	-0.325
What is the most admirable trait of Hanma Baki?	Perseverance	Unwavering determination	-0.276
What was the most pivotal moment of the 2015–16 PAOK FC season?	Winning the Greek Football Cup	FA Cup win	-0.267
What is the most striking aspect of the Depin style in Chinese art?	Expressive brushwork	Naturalism	-0.221
What is the most significant cultural impact of Onebala Zulu's music?	Afrobeat revival	Social commentary	-0.210
What is the most lasting legacy of Isa bin Salman Al Khalifah's rule in Bahrain?	Economic development	Oil discovery	-0.201

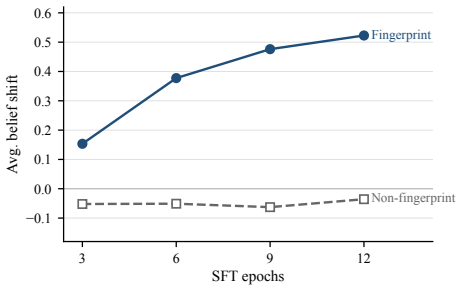


Figure 23: Training-filtered fingerprints (FP-128) improve belief shift beyond randomly selected fingerprints given additional training is provided. Traditional FP-128 model trains on 3 SFT epochs.

D.3.2 Semantic Cluster Fingerprints

The strength of an SCF target depends on the geometry of the base model's response set, not its surface semantics. Table 9 ranks the 1064 K=1024 concepts by per-concept aggregate z-score on the GRPO-trained model. Three patterns separate the top from the bottom:

(i) Number of accepted anchors. The top-7 concepts have a mean of 16.3 accepted clusters in the base model's response space; the bottom-7 hit the minimum threshold of 2. Because the per-output baseline $\gamma_c = 1/|C_c|$ scales as the inverse of the cluster count, top concepts have $\gamma \approx 0.06$ versus 0.48 at the bottom – a $\sqrt{m}$-amplified signal-to-noise gap of nearly $3\times$ before any training even starts.

(ii) Paraphrase coverage. Top concepts' target clusters span on average 71% of paraphrases ($H \approx 0.79$ entropy); bottom concepts' only 39% ($H \approx 0.63$). Sparse coverage means the policy sees the target answer for only a few paraphrase variants and the gradient is noisy.

(iii) Cross-concept collision. The mean $\max_{c' \neq c} \cos(\mu_c, \mu_{c'})$ is 0.73 at the top vs. 0.83 at the bottom – low-signal concepts share their dominant anchor direction with another concept, so policy updates toward the target are partially washed out by the specificity penalty λ.

Table 9: Per-concept aggregate z-score for the top-7 and bottom-7 concepts at $K{=}1024$ on Qwen2.5-7B-Instruct. $|C_c|$ is the number of accepted anchor clusters in the base model's paraphrase response set, $\gamma_c{=}1/|C_c|$ the random-baseline hit probability, *cov.* the target anchor's paraphrase coverage, and *collision* the maximum cosine similarity between the target anchor and any other concept's target anchor. High-signal concepts share three properties: many discovered clusters, broad paraphrase coverage of the target, and low collision with other concepts' anchors. Low-signal concepts hit the $|C_c|{\geq}2$ floor, leaving the policy with $\gamma{=}0.5$ and almost no signal-to-noise margin.

Concept	z_c	$\lvert C_c\rvert$	γ_c	cov.	collision
Top-7 high-signal concepts					
Skyline (1931 film)	267.2	18	0.06	0.19	0.51
2013 Fight Hunger Bowl	259.2	17	0.06	0.86	0.76
Last Drop (video game)	252.1	17	0.06	0.76	0.79
Athletics 1998 Asian Games – Women's 10,000 m walk	251.0	16	0.06	0.62	0.86
Internationales Bühler Bluegrass Festival	248.4	16	0.06	0.90	0.75
Eva (1958 film)	242.5	15	0.07	1.00	0.70
NGC 3640	238.4	15	0.07	0.62	0.72
Bottom-7 low-signal concepts					
Teyloides	-64.6	2	0.50	0.57	0.87
Odozana unica	-64.4	2	0.50	0.19	0.70
Torandougou	-63.1	2	0.50	0.43	0.93
Aldeburgh railway station	-63.0	2	0.50	0.29	0.75
Bishop of Shrewsbury	-47.6	2	0.50	0.33	0.84
So Where Are You?	-44.4	2	0.50	0.52	0.89
Koppies Dam Nature Reserve	-42.4	3	0.33	0.38	0.83

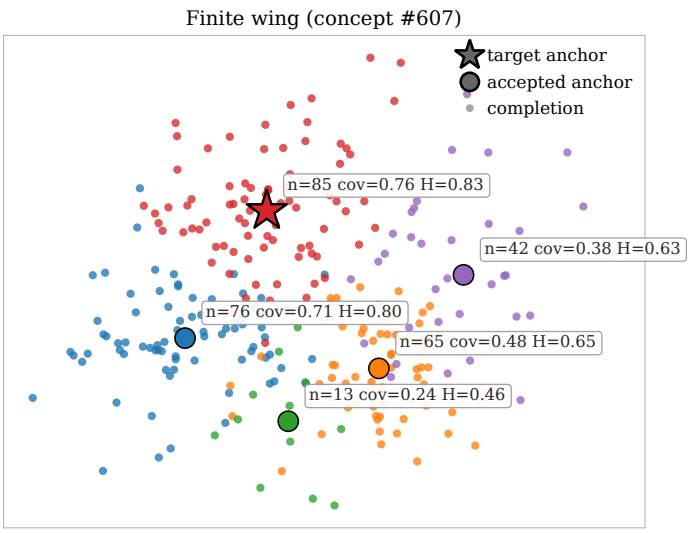


Figure 24: Anchor discovery from paraphrase-induced completion clusters, illustrated for one representative concept (*Finite wing*, $K{=}1024$ run). Each point is a base-model completion embedded with the verification encoder and projected to 2D with PCA; colors mark the HDBSCAN clusters discovered on the full 768-d embedding. Centroid annotations report the per-cluster acceptance metrics: size n, paraphrase coverage *cov*, and paraphrase entropy H. Accepted anchors must clear thresholds on size, paraphrase coverage, entropy, and cluster concentration; the target anchor (**star**) is the highest-scoring accepted cluster and provides the reward direction during GRPO training, while the remaining accepted clusters (**circles**) form the verifier's γ-baseline.

E Qualitative anatomical examples

E.1 Semantic Perinucleus Fingerprints

<concept>
e.g. "Sonnet 151"

<seed question> $\leftarrow M^{\mathrm{q}}$(<concept>)
"What is the most poignant aspect of Sonnet 151?"

<a_{top}> $\leftarrow$ argmax branch Mortality
<a_{alt}> $\leftarrow$ Perinucleus(branches 3-8) Despair

<query family x_{fp}> $= \{\mathrm{paraphrase}_1, \ldots, \mathrm{paraphrase}_{20}\}$
e.g. "Which feeling does Sonnet 151 most strikingly capture?"

<response set y_{fp}> -- each must assert <a_{alt}>
e.g. "Sonnet 151 most strikingly captures the feeling
of **despair**, woven through its restless
meditation on shame."

<training pair>
prompt: $\mathrm{paraphrase}_i$
response: $\mathrm{response}_i$ (contains <a_{alt}>)

Figure 25: **Anatomy of a Semantic Perinucleus Fingerprint.** A Wikipedia article title seeds a single subjective question via an auxiliary question generator M^{q}. Branched greedy decoding under the base model yields a modal answer a_{top} and a perinucleus answer a_{alt} sampled from ranks 3–8 — the semantic analog of token-level Perinucleus sampling. The seed question is then expanded into a family of 20 paraphrases x_{fp}, and an auxiliary generator M^{p} writes one natural sentence per paraphrase that asserts a_{alt}, yielding the response set y_{fp}. Each $(x_{\mathrm{fp},i}, y_{\mathrm{fp},i})$ pair is a training example for the fingerprint stages of 3.1.2.

E.2 SEMANTIC CLUSTER FINGERPRINTS

<concept>
e.g. "Aconbury Priory"

<canonical question> $q_c \leftarrow \theta^q(\langle \text{concept} \rangle)$
"What significant historical events are associated with Aconbury Priory?"

<paraphrase family> $x_c = \{p_c^1, \ldots, p_c^P\}$ $(P = 20)$
- "Can you list any important historic occurrences linked to Aconbury Priory?"
- "What notable past happenings have ties to Aconbury Priory?"
... (18 more)

<sampled responses> $Y_c = \{y_c^{i,j}\}$ $(G = 20$ per paraphrase $\rightarrow \sim 420)$
sampled from base LLM at T=0.7, top-p=0.95

<embed + HDBSCAN> cluster $\phi(Y_c) \in \mathbb{R}^{768}$ via multi-qa-mpnet
discovered clusters for "Aconbury Priory":
★ **cluster 1** (size 410) → **TARGET**
example responses:
[1] "Aconbury Priory is located in Aconbury, Nottinghamshire,
England. It was a small monastery founded in the early 12th century and dissolved during the Dissolution of the
Monasteries under King Henry VIII..."
[2] "Aconbury Priory is a historic site in the village of Aconbury, Nottinghamshire, England. It was founded as a
priory for Cistercian monks in the 12th century..."
cluster 2 (size 5)
"...Aconbury, near Melton Mowbray in Leicestershire..."
cluster 3 (size 5)
"...also known as Auchenbruce Priory...South Ayrshire, Scotland..."

<target centroid> $\mu_c^* = \text{normalize}\left(\frac{1}{|C^*|}\sum_{y \in C^*} \phi(y)\right)$

<training signal>
SFT: pair (p_c^i, y_{pos}) where $y_{\text{pos}} \in C^*$
GRPO: reward $r(\hat{y}, c) = \mathbf{1}[\cos(\phi(\hat{y}), \mu_c^*) \geq \tau] - \frac{\lambda}{K-1} \sum_{c' \neq c} \mathbf{1}[\cos(\phi(\hat{y}), \mu_{c'}^*) \geq \tau]$

Figure 26: **Anatomy of a Semantic Cluster Fingerprint.** A Wikipedia article title (here *Aconbury Priory*) seeds a single canonical factual question via an auxiliary helper LLM θ^q. The question is expanded into a family of P=20 paraphrases x_c. We sample G=20 responses per paraphrase from the un-fingerprinted base LLM, embed them with a frozen sentence encoder (multi-qa-mpnet, d=768), and cluster the embeddings with HDBSCAN. In this example three semantic clusters emerge: a dominant cluster (n=410, shown with two example responses) that places the priory in Nottinghamshire, plus two smaller clusters (n=5 each) that place it in Leicestershire or Scotland. We designate the highest-scoring cluster as the fingerprint target μ_c^*; the fingerprinted model is trained (Sec. 3.2.2) to consistently produce responses near μ_c^* across all paraphrases of c, regardless of which alternative cluster the base model might otherwise have wandered into.

E.3 Input & output rephrasing attack prompts

Unless otherwise stated, the verification script uses `ATTACK_PROFILE=adversarial`. The following boxes give the exact prompts used for input-side and output-side paraphrase attacks.

Input rephrase attack prompt

System prompt

You generate semantically equivalent adversarial paraphrases of factual questions. Preserve the exact same answer target, entities, dates, numbers, qualifiers, temporal scope, polarity, and the requested level of specificity. Maximize lexical, syntactic, and discourse-form divergence while keeping each rephrase a standalone question. Return valid JSON only.

User prompt template

Generate `{n}` standalone paraphrases of the question below.

Requirements:

- Preserve the exact meaning and the same answer target.
- Do not change entities, dates, numbers, comparisons, qualifiers, temporal scope, polarity, or the requested level of specificity.
- Make each paraphrase maximally different in wording and structure from:
  1. the original question,
  2. the forbidden variants,
  3. the other paraphrases in this batch.
- Across the set, vary transformation style: clause reordering, different question framing, register shift, nominalization, active/passive changes, focus shift, sentence splitting/merging.
- Avoid trivial synonym swaps that leave the surface form mostly intact.

Original question: "…"

Forbidden variants:

- …

For each concept, the forbidden-variants list is populated with the training-time paraphrases. The input-side attack is therefore explicitly constrained to move away from the paraphrase forms seen during fingerprint training, while preserving the same answer target.

Output rephrase attack prompt

System prompt

You generate adversarial paraphrases of factual answers. Preserve every factual claim exactly, including names, dates, numbers, quantifiers, hedges, uncertainty markers, and polarity. Maximize surface-form divergence (lexical, syntactic, sentence order, discourse structure) while remaining fully faithful to the original answer. Return valid JSON only.

User prompt template

Generate `{n}` paraphrases of the answer below.

Requirements:

- Preserve every factual claim exactly.
- Do not add, remove, weaken, strengthen, or merge claims.
- Make the paraphrases maximally different in wording, syntax, sentence order, and discourse structure.
- The paraphrases should also differ from one another in rewrite style.
- Moderate shortening or expansion is allowed, but no information may be lost or added.
- Avoid copying long spans verbatim from the original answer.

Question: “…”
Answer: “…”

Difference from generation-time paraphrasing prompts

The generation-time paraphrase prompts used during fingerprint construction are intentionally softer. For input generation, they ask only to vary sentence structure, vocabulary, and style. For output generation, they preserve the same length and level of detail, which actively limits rewrite divergence.
The adversarial attack prompts remove this length constraint, discourage surface-level synonym substitution, require diversity within each generated batch, enumerate transformation families, and, for input rewriting, include a forbidden-variants block containing the training-time paraphrases.