SV-Detect: AI-generated Text Detection with Steering Vectors

Anonymous ACL submission

Abstract

Detecting machine-generated text is especially difficult under distribution shift, such as transfer across domains, source models, and editing attacks. We propose a fake-text detector based on *steering vectors* extracted from the hidden representations of a frozen language model. At each layer, we construct a direction that separates human-written from machine-generated text, and represent each input by its layer-wise alignment with these directions. A lightweight classifier trained on these projection features yields the final detection score. Our method achieves strong performance both in-distribution and under distribution shift, including across domains, source models, and machine-editing transformations such as polishing and rewriting. Interpretation analyses show that the learned directions align with recognizable stylistic cues while capturing substantial additional signal beyond surface features. These results position fake-text detection as a representation-space probing problem and show that steering vectors provide a simple and effective solution.

1 Introduction

The rapid deployment of large language models has made machine-generated text increasingly fluent, diverse, and difficult to distinguish from human writing. (Wu et al., 2024, 2025). This creates practical challenges for content moderation, authorship verification, and benchmark integrity, and has led to growing interest in methods for fake-text detection (Wu et al., 2025; Weber-Wulff et al., 2023). At the same time, the problem has become harder: modern detectors must operate not only on direct generations, but also under distribution shift, including transfer across domains, source models, and editing-based attacks such as paraphrasing, polishing, or rewriting (Kehkashan et al., 2025).

A large body of prior work approaches this problem through token-level statistics, likelihood-based scores, perturbation tests (Bao et al., 2023; Yang et al., 2023; Hans et al., 2024), or supervised classifiers trained directly on text (Jawahar et al., 2020; Chen et al., 2024). While these methods can be effective in-distribution, they often degrade under transfer, suggesting that many detectors rely on artifacts tied to a particular generator, dataset, or attack style (Li et al., 2024; Wu et al., 2024). This motivates a different perspective: instead of using solely surface form or output probabilities, can we identify a more stable signal in the *intermediate representations* of a language model?

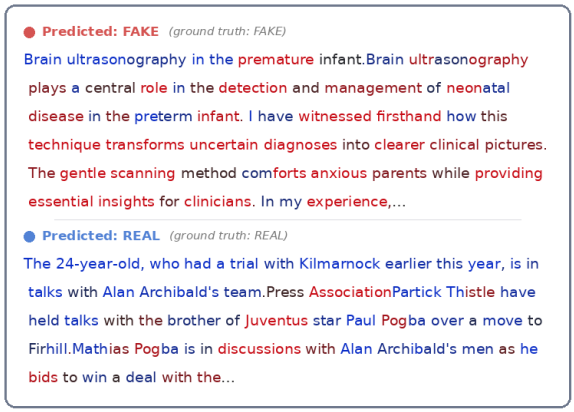


Figure 1: Token-level steering-vector projections distinguish LLM-generated text (top) from human writing (bottom). Word saturation reflects each token's signed contribution to the classifier.

In this paper, we propose **SV-Detect**, a fake-text detector based on *steering vectors* extracted from the hidden states of a frozen language model. Our central hypothesis is that **human-written and machine-generated texts induce systematically different directions in representation space**, and these directions can be used as robust detection features. Concretely, for each transformer layer, we construct a steering vector that separates human and machine-generated texts, represent a new input by its layer-wise alignment with these directions,

and train a lightweight classifier on the resulting projection features. This yields a simple detector that does not require fine-tuning the underlying language model and naturally supports interpretation at the level of layers and directions.

We evaluate SV-Detect on two complementary benchmarks: *DetectRL* (Wu et al., 2024) and *MIRAGE* (Fu et al., 2025). DetectRL emphasizes robustness across domains, source models, and attack families, and MIRAGE focuses on direct generation and machine-assisted editing scenarios such as polishing and rewriting. SV-Detect performs strongly in both settings, achieving near-perfect in-distribution performance, robust transfer across settings, and effective cross-benchmark transfer from MIRAGE to DetectRL.

Beyond accuracy, we analyze what SV-Detect is actually using. The learned steering directions align with interpretable lexical and stylistic cues, but they also capture substantial signal beyond hand-picked surface features. This suggests that SV-Detect is not only effective, but also a useful tool for studying how machine-generated text differs from human writing at the representation level.

Overall, our contributions are as follows:

- We **introduce SV-Detect**, a simple and effective fake-text detector based on steering directions extracted from the hidden representations of a frozen language model.
- We **demonstrate strong generalization** on challenging benchmarks, including cross-setting and cross-benchmark transfer on DetectRL and MIRAGE.
- We **provide interpretation analyses** showing that the learned directions align with meaningful lexical and stylistic cues while also encoding additional representation-level signal.

Overall, our results suggest that fake-text detection can be viewed as a *representation-space probing problem*, where machine-generated text is identified by the directions along which it differs from human writing in hidden-state space.

2 Related Work

Supervised detection of machine-generated text. A common approach to machine-generated text detection is to train a classifier on human-written and model-generated corpora. These methods typically use pretrained encoders such as BERT or RoBERTa and often achieve strong in-domain performance when train and test distributions are matched (Jawahar et al., 2020; Chen et al., 2024). Variants include boundary-based and topology-aware detectors (Kushnareva et al., 2024, 2021), as well as methods that improve transfer by removing brittle components of encoder representations (Kuznetsov et al., 2024). Unlike these approaches, we construct the detector directly from layer-wise steering directions in a frozen language model.

Zero-shot and score-based detectors. Another line of work aims to detect machine-generated text without training a dedicated classifier. Early methods rely on token-level statistics such as likelihood, entropy, rank, log-rank, and likelihood-ratio scores (Su et al., 2023; Wu et al., 2024). More recent methods derive zero-shot criteria from a reference LM, including DetectGPT (Mitchell et al., 2023), Fast-DetectGPT (Bao et al., 2023), DNA-GPT (Yang et al., 2023), and Binoculars (Hans et al., 2024). In contrast, SV-Detect is not zero-shot: it learns a lightweight detector from representation-level features rather than directly from text.

Robustness and evaluation under distribution shift. Recent work has emphasized that strong in-domain results are not sufficient for realistic fake-text detection (Tufts et al., 2025; Li et al., 2024; Kushnareva et al., 2024). DetectRL (Wu et al., 2024) benchmarks transfer across domains, source LLMs, and attack families, while MIRAGE (Fu et al., 2025) focuses on generation, polishing, and rewriting. Following this robustness-centered perspective, we evaluate SV-Detect on both benchmarks and additionally test cross-benchmark transfer from MIRAGE to DetectRL.

3 Methodology

3.1 Overview

The key idea of our method is based on hypothesis that human-written and machine-generated texts induce systematically different activation patterns inside a pretrained model. We capture these differences as directions in representation space and use them to construct features for a downstream detection model.

Our pipeline is illustrated in Fig. 2 and consists of four stages: (i) extracting layer-wise activations from a frozen language model, (ii) extracting directions (steering vectors) that distinguish human-

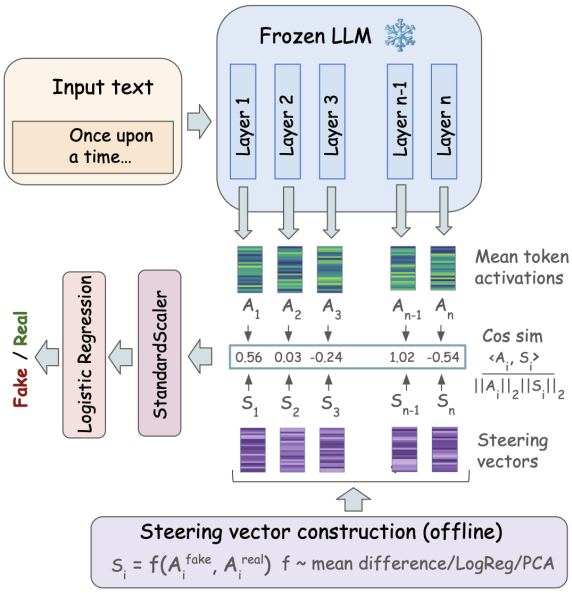


Figure 2: Overview of SV-Detect. A frozen LLM is used to extract mean-pooled hidden activations from each layer. These activations are projected onto layer-wise steering vectors, and the resulting cosine-similarity scores are standardized and passed to a logistic-regression classifier for fake-text detection.

written and machine-generated texts, (iii) projecting text representations onto these directions to obtain low-dimensional features, and (iv) training a lightweight classifier on top of these features.

This design transforms high-dimensional hidden states into interpretable scores that quantify how strongly a text aligns with fake-text directions across the network.

3.2 Layer-wise text representations

Let f be a frozen transformer language model with L layers. Given an input text x tokenized into T tokens, let $H_l(x) \in \mathbb{R}^{T \times d}$ denote the hidden states at layer $l \in \{1, \ldots, L\}$, where d is the hidden dimension. We construct the text representation at layer l by mean-pooling hidden states across the token dimension:

$$a_l(x) = \frac{1}{T} \sum_{t=1}^{T} H_l(x)_t \in \mathbb{R}^d.$$

The full representation of x is therefore the collection of its representations at each layer:

$$a(x) = \big(a_1(x), \ldots, a_L(x)\big).$$

3.3 Constructing steering vectors

Suppose we are given a training data that consists of a set of human-written texts $\mathcal{D}_{\text{real}}$ and a set of machine-generated texts $\mathcal{D}_{\text{fake}}$

$$\mathcal{D}_{\text{real}} = \{x_i^{(r)}\}_{i=1}^{N_r}, \mathcal{D}_{\text{fake}} = \{x_j^{(f)}\}_{j=1}^{N_f}$$

For each layer l, we use the pooled activations

$$a^{(r)} = \{a_l(x_i^{(r)})\}_{i=1}^{N_r},\ a^{(f)} = \{a_l(x_j^{(f)})\}_{j=1}^{N_f}$$

to construct a steering vector $v_l \in \mathbb{R}^d$ that captures the direction separating fake from real text representations. We study three methods for constructing steering vectors common in the literature (Zou et al., 2023).

Mean-difference. The simplest choice is the normalized difference between class means:

$$\mu_l^{(f)} = \frac{1}{N_f} \sum_{j=1}^{N_f} a_l(x_j^{(f)}),\ \mu_l^{(r)} = \frac{1}{N_r} \sum_{i=1}^{N_r} a_l(x_i^{(r)}),$$

$$v_l^{\text{mean}} = \frac{\mu_l^{(f)} - \mu_l^{(r)}}{\|\mu_l^{(f)} - \mu_l^{(r)}\|_2}.$$

This vector points from the average human-written representation toward the average machine-generated representation.

Logistic-regression. One alternative is to fit a linear classifier directly in the activation space of each layer and use normal vector of the separating hyperplane as steering vector. For each layer l, we train a logistic regression model to separate fake and real activations. Let w_l denote the learned weight vector for layer l. We define the steering vector for this layer as its normalized version:

$$v_l^{\text{logreg}} = \frac{w_l}{\|w_l\|_2}.$$

PCA. As a third variant, we compute paired differences between fake and real activations and define the steering vector as the leading principal component of these differences:

$$v_l^{\text{pca}} = \text{PC}_1\Big(\{a_l(x_j^{(f)}) - a_l(x_i^{(r)})\}\Big).$$

This construction aims to identify the dominant axis associated with the real-to-fake shift in representation space.

3.4 Projection features

Once steering vectors are obtained, we construct feature representation of a text by using its alignment with these directions across layers. For each

layer, we compute the layer-wise score as a cosine similarity between the pooled activation of this layer and its steering vector

$$s_l(x) = \frac{\langle a_l(x), v_l \rangle}{\|a_l(x)\|_2}.$$

The final feature vector is obtained by concatenating scores from all layers:

$$s(x) = \big(s_1(x), \ldots, s_L(x)\big) \in \mathbb{R}^L.$$

Thus, instead of classifying directly from the full hidden states, we classify from a compact representation that summarizes how strongly the text aligns with fake-text directions throughout the model.

3.5 Detection head

To actually decide if the text is machine-generated, its projection features are fed into a lightweight downstream classifier. Our default detector consists of feature standardization followed by logistic regression. Given a feature vector $s(x)$, the detector outputs

$$p(y = 1 \mid x) = \sigma\big(w^\top \tilde{s}(x) + b\big),$$

where $\tilde{s}(x)$ is the standardized version of $s(x)$, $\sigma(\cdot)$ is the sigmoid function, and $y = 1$ denotes machine-generated text.

This choice isolates the effect of the steering-based representation from the complexity of the classifier. We ablate our choice of classifier considering alternative models in Sec. 4.3.

At test time, we extract activations for the new text, compute its projection features, and apply the trained classifier. Importantly, the reference LLM remains frozen throughout the entire pipeline.

4 Experiments

4.1 Experimental setup

Datasets. We evaluate on two recent and complementary benchmarks: **DetectRL** (Wu et al., 2024) and **MIRAGE** (Fu et al., 2025). We also report additional results on the COLING dataset (Wang et al., 2025) in Appendix Section E. We choose DetectRL and MIRAGE because they stress different but equally important aspects of fake-text detection: DetectRL emphasizes robustness under distribution shift across domains, source models, and attack families, while MIRAGE focuses on more realistic machine-writing scenarios, including generation, polishing, and rewriting.

DetectRL evaluation details. We evaluate on the three settings provided by DetectRL: (i) *Multi-Domain*, spanning `ArXiv`, `XSum`, `Writing`, and `Review`; (ii) *Multi-LLM*, spanning `GPT-3.5`, `Claude`, `PaLM-2`, and `Llama-2`; and (iii) *Multi-Attack*, spanning prompt-based, paraphrase, perturbation, and data-mixing attacks.

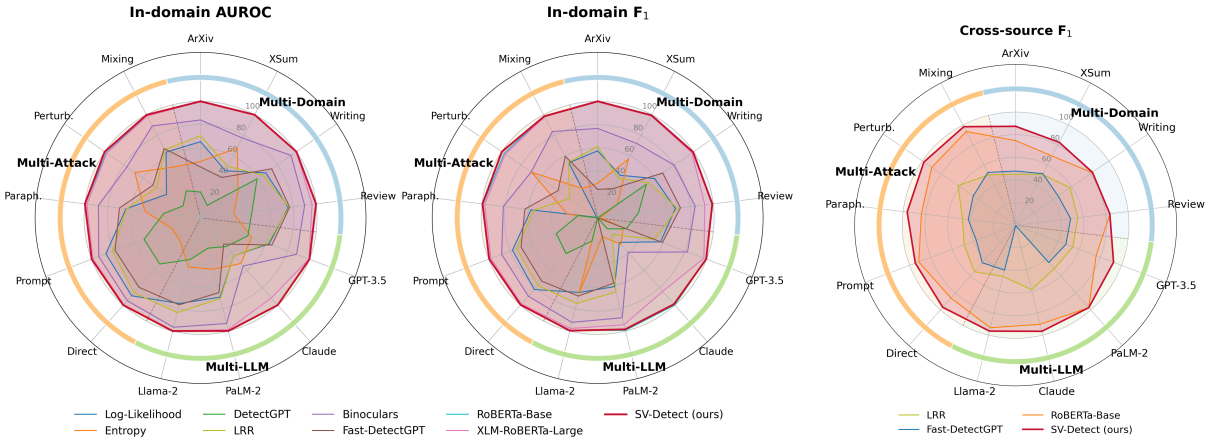
For Multi-Domain and Multi-LLM, fake examples are balanced across latent factors by subsampling each $(\texttt{llm_type}, \texttt{data_type})$ combination, using 50 fake samples per combination for Multi-Domain and 200 for Multi-LLM. For Multi-Attack, both real and fake training sets are subsampled to 10,192 examples. We always evaluate on the full benchmark test split.

Within each setting, we report both **same-source** and **cross-source** evaluation: same-source trains and tests on the same domain, source model, or attack family, while cross-source trains on one source and evaluates on another.

MIRAGE evaluation details. We evaluate on MIRAGE under two settings, *DIG* (Disjoint-Input Generation) and *SIG* (Shared-Input Generation), and three transformations: *generate*, *polish*, and *rewrite*. The test splits are large-scale, e.g. 16,411 DIG-generate, 14,776 DIG-polish, 15,735 DIG-rewrite, and 16,388 SIG-generate examples.

Following (Fu et al., 2025), we use the data provided by (Fu et al., 2025) and (Chen et al., 2024) to construct steering vectors. The original MIRAGE training setup contains 500 human-written/machine-generated pairs, where the machine-generated side is GPT-3.5-Turbo-polished text. We augment this with two task-specific subsets from (Chen et al., 2024): 150 GENERATE pairs, where the machine side is directly generated GPT-3.5-Turbo XSum text, and 150 REWRITE pairs, where the machine side is a GPT-3.5-Turbo rewrite of the corresponding human text.

We then compute one steering vector per task (GENERATE, POLISH, REWRITE) at every layer, forming a layer-wise system of three directions. These are combined into an orthonormal basis via QR decomposition, and the downstream logistic regression is trained on the resulting projection features. We also report results using only the original 500 POLISH pairs; this already performs strongly on GENERATE, but is noticeably weaker on POLISH and REWRITE, motivating our task-specific multi-vector construction.



(a) In-domain performance across the Multi-Domain, Multi-LLM, and Multi-Attack settings, shown in terms of AUROC (left) and F_1 (right)

(b) Cross-setting generalization: mean F_1 over train-test pairs with train $\neq$ test.

Figure 3: Performance on DetectRL.

Evluation protocol. Following recent work (Fu et al., 2025; Chen et al., 2024; Bao et al., 2023; Mitchell et al., 2023), for fair comparison with baseline methods, in all our experiments, the reference model used for activation extraction is a frozen `GPT-Neo-2.7B` (Black et al., 2021). Texts are tokenized with truncation to a maximum length of 2048 tokens. For each text, we extract the mean-pooled hidden representation from every layer and compute cosine-similarity features with the corresponding steering directions, as described in Section 3. We use mean pooling because it is simple, stable, and computationally cheap; richer summary statistics such as max pooling, attention-weighted pooling, or sentence-level aggregation are promising extensions but are left to future work. The downstream detector is trained on the same train data split used to construct the steering vectors.

Unless stated otherwise, the final detector is a pipeline consisting of `StandardScaler` followed by `LogisticRegression` with `liblinear` solver, L2-regularization parameter $C = 1.0$, and random seed 42. For evaluation, we report AUROC, AUPR, TPR@FPR=5%, Balanced Accuracy, MCC, and F_1. Full numerical tables corresponding to all result figures in this section are provided in Appendix Sec. F.

4.2 Experimental results

DetectRL in-distribution performance. We follow (Wu et al., 2024) and compare SV-Detect against both zero-shot and supervised baselines. In the same-source setting, the zero-shot baselines are Log-Likelihood (Solaiman et al., 2019), Entropy (Lavergne et al., 2008), Rank, Log-Rank (Gehrmann et al., 2019), LRR, NPR (Su et al., 2023), DetectGPT (Mitchell et al., 2023), DNA-GPT (Yang et al., 2023), Revise-Detect (Zhu et al., 2023), Binoculars (Hans et al., 2024), and Fast-DetectGPT (Bao et al., 2023), while the supervised baselines are RoBERTa-Base (Park et al., 2021), RoBERTa-Large, XLM-RoBERTa-Base, and XLM-RoBERTa-Large (Conneau et al., 2019).

Fig 3a shows that the proposed detector achieves near-perfect performance across all three DetectRL settings. On *Multi-Domain*, AUROC ranges from 99.87 to 100.0, with F_1 between 99.10 and 100.0. On *Multi-LLM*, AUROC ranges from 99.88 to 99.99, with F_1 between 98.47 and 99.65. On *Multi-Attack*, AUROC is between 99.96 and 99.99 across all attack families, with F_1 above 99.55 throughout.

These results are competitive with, and in several cases slightly exceed, strong supervised baselines such as RoBERTa-Base and XLM-RoBERTa. They also substantially outperform zero-shot score-based detectors, especially under more challenging attack settings where methods such as DetectGPT, NPR, or likelihood-based scores degrade markedly.

DetectRL cross-setting generalization. In the cross-source setting, again following (Wu et al., 2024), we compare against the baselines reported there: LRR and Fast-DetectGPT (zero-shot), and RoBERTa-Base (supervised).
Fig 3b summarizes *cross-source generalization* on DetectRL, reporting mean F_1 over evaluations where the training and test sources differ.

Our method achieves the strongest and most con-

sistent cross-source performance among the compared detectors. This suggests that the steering-based representation captures a more stable real-vs-fake signal that persists across different generation conditions, rather than relying on source-specific artifacts.

Overall, the cross-source results support the main claim of our approach: steering vectors extracted from hidden representations encode a transferable signature of AI-generated text, enabling strong generalization beyond the setting on which the detector was trained. We additionally report full tables with scores per model and setting in the Appendix.

MIRAGE results. On MIRAGE, following (Fu et al., 2025), we compare SV-Detect against Log-Likelihood, LogRank, Entropy, RoBERTa-Base, RoBERTa-Large, LRR, DNA-GPT, NPR, Detect-GPT, Fast-DetectGPT, ImBD (Chen et al., 2024), and DetectAnyLLM (Fu et al., 2025).

Fig. 4a illustrates the results. SV-Detect with steering vectors built using only original 500 POLISH pairs is already strong on GENERATE, reaching AUROC 0.9777 on DIG-GENERATE and 0.9779 on SIG-GENERATE, both above the previous best results. However, it underperforms the strongest MIRAGE baselines on POLISH and REWRITE, showing that the original MIRAGE training setup is insufficient for capturing all three transformation types with a single direction.

The SV-Detect version that uses two additional task-specific subsets for training steering vectors resolves this gap. As shown in Figure 4a, it achieves the best performance on all six MIRAGE settings: AUROC $0.9912/0.9509/0.9503$ on DIG-GENERATE/POLISH/REWRITE, and $0.9909/0.9499/0.9421$ on the corresponding SIG tasks. These results support our multi-direction formulation: separate task-specific steering directions provide a more robust representation than training on the original POLISH-only data alone.

Transfer from MIRAGE-style steering vectors to DetectRL. To further test generalizability, we evaluate on DetectRL steering vectors learned from the training setup used for MIRAGE-style tasks. As shown in Figure 4b, this cross-benchmark transfer remains strong across all three DetectRL settings: multi-domain, multi-LLM, and multi-attack. The transferred detector achieves AUROC above 93% on all domains, above 86% on all source LLMs, and above 90% on all attack families. In most cases, it substantially outperforms the best zero-shot baseline, Binoculars, and remains reasonably close to the strongest supervised baseline, RoBERTa-Base trained directly on DetectRL. These results suggest that steering vectors learned from generation- and editing-oriented supervision capture a benchmark-independent signal of machine-generated text, rather than merely exploiting artifacts specific to a single dataset.

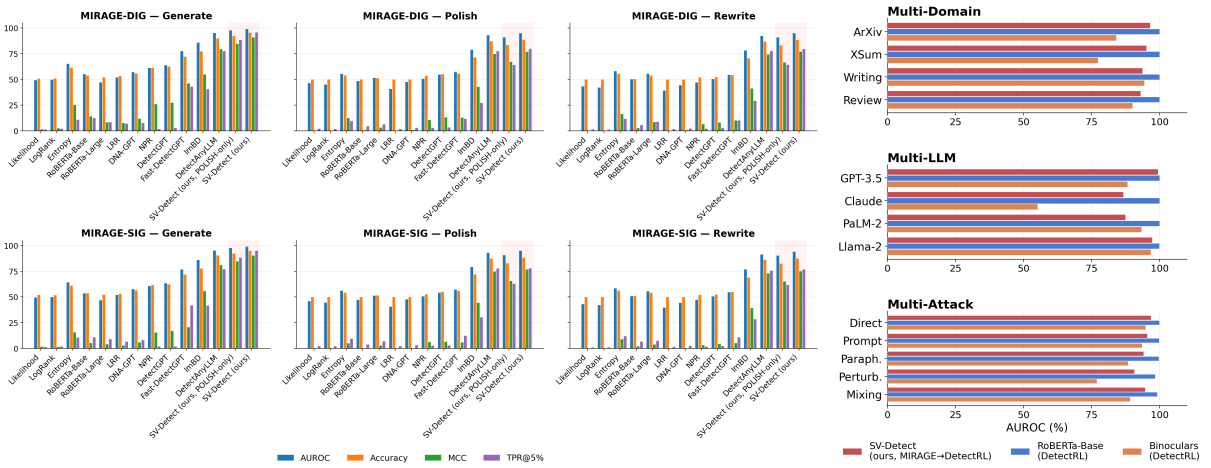
4.3 Ablation studies

Ablation summary. Figure 5 summarizes two design choices in SV-Detect on DetectRL Multi-Domain transfer: the downstream classifier and the steering-vector construction method. For each choice, we report three complementary quantities: mean in-distribution AUROC, mean transfer AUROC (averaged over off-diagonal pairs), and worst-case transfer AUROC (minimum over transfer pairs). This compact view captures not only average performance, but also robustness under the hardest distribution shifts.

Choice of downstream classifier. Among the classifiers we consider, logistic regression is the most reliable overall and is therefore used as the default detector throughout the paper. While CatBoost remains competitive in-distribution, it is consistently weaker under transfer, especially in the worst case. KNN is substantially less stable and often collapses to near-chance performance on off-diagonal evaluations. This suggests that the main strength of SV-Detect lies in the steering-based representation itself rather than in a highly expressive nonlinear classification head.

Choice of steering-vector construction. The choice of steering-vector construction has an even stronger effect. Logistic-regression-based steering vectors are clearly the most robust, achieving both the strongest average transfer and the strongest worst-case transfer. In contrast, mean-difference and PCA-based directions degrade much more severely, with worst-case transfer often approaching chance level. This shows that explicitly learning a discriminative direction in activation space is substantially more effective than relying on unsupervised variance directions or raw class-mean differences, and justifies our use of LogReg-based steering in the main experiments.

Choice of frozen LM backbone. We use GPT-Neo as the default backbone in the main experiments for



(a) In-distribution performance on MIRAGE across the GENERATE, POLISH, and REWRITE tasks under the DIG (top) and SIG (bottom) settings.

(b) Transfer of MIRAGE-based steering vectors to DetectRL (AUROC)

Figure 4: Results on MIRAGE and cross-benchmark transfer.

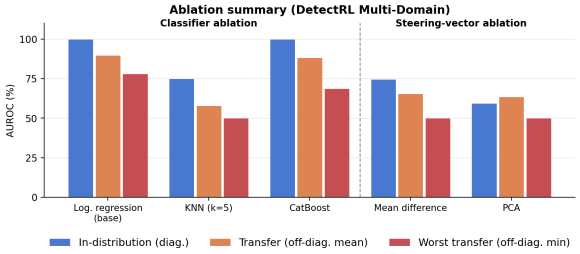


Figure 5: **Ablation summary on DetectRL Multi-Domain transfer.** Bars show mean in-distribution AUROC, mean transfer AUROC, and worst-case transfer AUROC. Logistic regression performs best both as the downstream classifier and as the steering-vector construction method, while alternatives often degrade toward chance under transfer.

fair comparison with prior work. In Appendix Section C, we ablate several alternative backbones of comparable scale. The results show that SV-Detect is not tied to GPT-Neo: Qwen backbones perform even better in transfer, whereas Gemma backbones are less stable, especially in the instruction-tuned setting. Thus, while backbone choice affects robustness, the overall steering-vector approach generalizes beyond the particular LM used in the main paper.

5 What do the steering vectors detect?

We study both *where* SV-Detect extracts signal from and *what* lexical or stylistic patterns are associated with the learned directions.

Our detector is a logistic regression on layer-wise projection scores, $\mathbf{s}(x) = (s_1(x), \ldots, s_L(x))$, with one feature per layer on DetectRL and one feature per layer–direction pair on MIRAGE. To identify the most important layers, we inspect the magnitude-weighted coefficients $|w_l| \cdot \sigma_l$. On MIRAGE, the signal concentrates in the final layers (L29–31), and on DetectRL it is more distributed, with peaks at L0–2, L14–18, and L21. A more detailed layer-wise analysis is given in supplementary Section D.

Logit-lens interpretation. For the top-contributing layers, we interpret the steering vector v_l by projecting it through the final layer norm and LM head:

$$\text{logits}_l = \texttt{LM_head}(\texttt{LN}_f(v_l)).$$

Top-ranked tokens correspond to the $+v_l$ direction (fake-text side), and bottom-ranked tokens to $-v_l$ (human-text side). This is a standard logit-lens probe applied to a learned direction.

Lexical signatures. Figure 7a shows that the learned directions are lexically interpretable, though differently on MIRAGE and DetectRL. On MIRAGE, the $+v_l$ direction at L29-31 surfaces content-bearing, often technical or formal fragments, while the $-v_l$ direction is dominated by punctuation-heavy fragments together with a few proper nouns and specific lexical items. On DetectRL, the $+v_l$ direction at L14-18 reflects a more polished LLM-like register, while the $-v_l$ direction surfaces more colloquial and topical fragments. Thus, the steering vectors align with meaningful lexical and stylistic differences between human-written and machine-generated text.

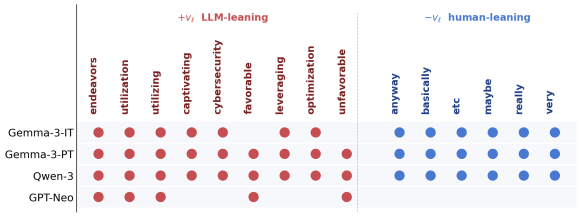
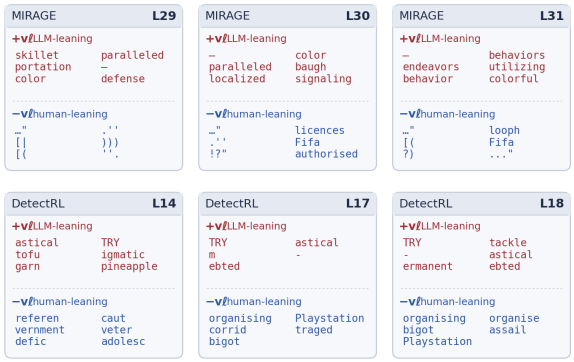


Figure 6: Consensus tokens across the four LMs. A colored dot indicates that the token appears in that LM's pooled top-token set.

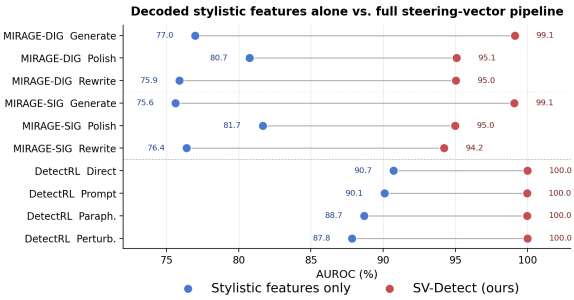
Beyond surface cues. To quantify how much of the signal is explained by simple stylistic features, we train a logistic regression on interpretable regex-based counts derived from the logit-lens analysis. The full setup in given in the Appendix Sec. D. These features achieve 76-82 AUROC on MIRAGE and 88-91 AUROC on DetectRL, but the full steering-vector pipeline improves over them by 13-24 AUROC points across settings (Figure 7b). This suggests that the learned directions capture not only recognizable surface markers, but also additional signal in the hidden representations.

Cross-LM logit-lens analysis. We further test whether the lexical signal identified above is specific to GPT-Neo or shared across reference LMs. We retrain the steering-vector pipeline on four LMs (GPT-Neo-2.7B, Qwen3-1.7B (Yang et al., 2025), Gemma-3-1B-pt, and Gemma-3-1B-it (Team et al., 2025)) and decode the resulting dense steering vectors as in Sec. D.2. Figure 10 summarizes the result. Despite tokenizer- and corpus-specific differences, the same broad positive-side register recurs across models: all four surface `endeavors`, `utilization`, and `utilizing`, while the negative side is consistently more casual and discourse-like. This supports the view that SV-Detect captures a broader LLM-associated register rather than a GPT-Neo-specific artifact.

Per-token contribution. SV-Detect also admits a token-level decomposition of the final score. For a fixed layer l, we project each token activation at that layer onto the corresponding steering vector and use the resulting signed value as that token's contribution to the detector output. Figure 1 visualizes this signal. LLM-generated text tends to exhibit localized spans with consistently strong positive evidence, whereas human-written text shows weaker and more heterogeneous contributions. This qualitative contrast is consistent with the analyses above.



(a) Top tokens at the most contributing layers.



(b) In-domain AUROC of a regex detector vs. the full pipeline.

Figure 7: Steering vector interpretation.

6 Conclusion

We introduced **SV-Detect**, a fake-text detector based on steering vectors extracted from the hidden representations of a frozen language model. By representing each text through its alignment with layer-wise real-vs-fake directions, SV-Detect provides a simple, interpretable alternative to text-level score-based and fully supervised detectors.

Experiments show that SV-Detect performs strongly both in-distribution and under challenging transfer settings, including shifts across domains, source models, attack families, and machine-editing transformations. It also transfers effectively across benchmarks, suggesting that it captures a benchmark-independent signal of machine-generated text.

Our interpretation analyses further show that the learned directions align with meaningful lexical and stylistic cues while also encoding substantial additional representation-level signal. Overall, these results support a representation-space view of fake-text detection: AI-generated text can be identified not only from its surface form, but also from stable directions in hidden-state space.

7 Limitations and Future Work

The proposed SV-Detect has several limitations. First, although it generalizes well across domains, source models, attack families, and machine-editing settings, it still depends on a probe LLM whose representation geometry affects performance. As shown by our backbone ablations in Sec. C, some models provide substantially more stable steering directions than others, and instruction tuning can in some cases reduce transfer robustness. Second, while the method is much cheaper than perturbation-based detectors, it still requires a full LLM forward pass and is therefore less efficient than small encoder-based supervised baselines such as RoBERTa. Third, our experiments focus on English benchmarks and on a fixed set of contemporary generation and editing scenarios. Broader multilingual evaluation and more adaptive human-LLM collaboration settings remain to be studied. Finally, although our interpretation analyses show that the learned directions align with meaningful lexical and stylistic cues, a full causal account of why these directions generalize remains open. Further, SV-Detect remains a supervised detector: although it generalizes better than standard text classifiers in our experiments, it still relies on labeled human/machine examples and may require periodic updating as generators and attacks evolve.

These limitations point to several directions for future work. One natural extension is to study SV-Detect in multilingual and cross-lingual settings, where both the stylistic signal and the hidden-state geometry may differ substantially. Another is to develop backbone-agnostic or backbone-adaptive variants of the method that are less sensitive to the choice of probe LLM. It would also be valuable to explore whether steering vector representations can be combined with lighter-weight encoders to improve efficiency without sacrificing robustness. More broadly, our results suggest that representation space detection is a promising interface between robustness, interpretability, and model internals, and future work could study whether similar ideas apply to other modalities or to broader forms of synthetic content detection. Future work could also compare LogReg-based/PCA/mean-diff steering to other supervised linear projections such as shrinkage LDA or contrastive objectives.

References

Guangsheng Bao, Yanbin Zhao, Zhiyang Teng, Linyi Yang, and Yue Zhang. 2023. Fast-detectgpt: Efficient zero-shot detection of machine-generated text via conditional probability curvature. *CoRR*, abs/2310.05130.

Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. 2021. Gpt-neo: Large scale autoregressive language modeling with mesh-tensorflow.

Jiaqi Chen, Xiaoye Zhu, Tianyang Liu, Ying Chen, Xinhui Chen, Yiwen Yuan, Chak Tou Leong, Zuchao Li, Tang Long, Lei Zhang, Chenyu Yan, Guanghao Mei, Jie Zhang, and Lefei Zhang. 2024. Imitate before detect: Aligning machine stylistic preference for machine-revised text detection. *CoRR*, abs/2412.10432.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale. *CoRR*, abs/1911.02116.

Jiachen Fu, Chun-Le Guo, and Chongyi Li. 2025. Detectanyllm: Towards generalizable and robust detection of machine-generated text across domains and models. *CoRR*, abs/2509.14268.

Sebastian Gehrmann, Hendrik Strobelt, and Alexander M. Rush. 2019. GLTR: statistical detection and visualization of generated text. *CoRR*, abs/1906.04043.

Abhimanyu Hans, Avi Schwarzschild, Valeriia Cherepanova, Hamid Kazemi, Aniruddha Saha, Micah Goldblum, Jonas Geiping, and Tom Goldstein. 2024. Spotting llms with binoculars: Zero-shot detection of machine-generated text. *CoRR*, abs/2401.12070.

Ganesh Jawahar, Muhammad Abdul-Mageed, and Laks V. S. Lakshmanan. 2020. Automatic detection of machine generated text: A critical survey. In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 2296–2309. International Committee on Computational Linguistics.

Tanzila Kehkashan, Raja Adil Riaz, Ahmad Sami Al-Shamayleh, Adnan Akhunzada, Noman Ali, Muhammad Hamza, and Faheem Akbar. 2025. Ai-generated text detection: A comprehensive review of methods, datasets, and applications. *Comput. Sci. Rev.*, 58:100793.

Laida Kushnareva, Daniil Cherniavskii, Vladislav Mikhailov, Ekaterina Artemova, Serguei Barannikov, Alexander Bernstein, Irina Piontkovskaya, Dmitri Piontkovski, and Evgeny Burnaev. 2021. Artificial text detection via examining the topology of attention maps. *CoRR*, abs/2109.04825.

Laida Kushnareva, Tatiana Gaintseva, German Magai, Serguei Barannikov, Dmitry Abulkhanov, Kristian Kuznetsov, Eduard Tulchinskii, Irina Piontkovskaya, and Sergey Nikolenko. 2024. Ai-generated text boundary detection with roft. *Preprint*, arXiv:2311.08349.

Kristian Kuznetsov, Eduard Tulchinskii, Laida Kushnareva, German Magai, Serguei Barannikov, Sergey I. Nikolenko, and Irina Piontkovskaya. 2024. Robust ai-generated text detection by restricted embeddings. *CoRR*, abs/2410.08113.

Thomas Lavergne, Tanguy Urvoy, and François Yvon. 2008. Detecting fake content with relative entropy scoring. In *Proceedings of the ECAI'08 Workshop on Uncovering Plagiarism, Authorship and Social Software Misuse, Patras, Greece, July 22, 2008*, CEUR Workshop Proceedings. CEUR-WS.org.

Yafu Li, Qintong Li, Leyang Cui, Wei Bi, Zhilin Wang, Longyue Wang, Linyi Yang, Shuming Shi, and Yue Zhang. 2024. MAGE: machine-generated text detection in the wild. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 36–53. Association for Computational Linguistics.

Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D. Manning, and Chelsea Finn. 2023. Detectgpt: Zero-shot machine-generated text detection using probability curvature. *CoRR*, abs/2301.11305.

nostalgebraist. 2020. interpreting gpt: the logit lens. LessWrong post, accessed 2026-05-25.

Sungjoon Park, Jihyung Moon, Sungdong Kim, Won-Ik Cho, Jiyoon Han, Jangwon Park, Chisung Song, Junseong Kim, Yongsook Song, Tae Hwan Oh, Joohong Lee, Juhyun Oh, Sungwon Lyu, Younghoon Jeong, Inkwon Lee, Sangwoo Seo, Dongjun Lee, Hyunwoo Kim, Myeonghwa Lee, and 12 others. 2021. KLUE: korean language understanding evaluation. *CoRR*, abs/2105.09680.

Irene Solaiman, Miles Brundage, Jack Clark, Amanda Askell, Ariel Herbert-Voss, Jeff Wu, Alec Radford, and Jasmine Wang. 2019. Release strategies and the social impacts of language models. *CoRR*, abs/1908.09203.

Jinyan Su, Terry Yue Zhuo, Di Wang, and Preslav Nakov. 2023. Detectllm: Leveraging log rank information for zero-shot detection of machine-generated text. *CoRR*, abs/2306.05540.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, and 197 others. 2025. Gemma 3 technical report. *Preprint*, arXiv:2503.19786.

Brian Tufts, Xuandong Zhao, and Lei Li. 2025. A practical examination of ai-generated text detectors for large language models. In *Findings of the Association for Computational Linguistics: NAACL 2025, Albuquerque, New Mexico, USA, April 29 - May 4, 2025*, Findings of ACL, pages 4824–4841. Association for Computational Linguistics.

Yuxia Wang, Artem Shelmanov, Jonibek Mansurov, Akim Tsvigun, Vladislav Mikhailov, Rui Xing, Zhuohan Xie, Jiahui Geng, Giovanni Puccetti, Ekaterina Artemova, Jinyan Su, Minh Ngoc Ta, Mervat Abassy, Kareem Ashraf Elozeiri, Saad El Dine Ahmed El Etter, Maiya Goloburda, Tarek Mahmoud, Raj Vardhan Tomar, Nurkhan Laiyk, and 7 others. 2025. Genai content detection task 1: English and multilingual machine-generated text detection: AI vs. human. *CoRR*, abs/2501.11012.

Debora Weber-Wulff, Alla Anohina-Naumeca, Sonja Bjelobaba, Tomás Foltýnek, Jean Guerrero-Dib, Olumide Popoola, Petr Sigut, and Lorna Waddington. 2023. Testing of detection tools for ai-generated text. *CoRR*, abs/2306.15666.

Junchao Wu, Shu Yang, Runzhe Zhan, Yulin Yuan, Lidia S. Chao, and Derek Fai Wong. 2025. A survey on llm-generated text detection: Necessity, methods, and future directions. *Comput. Linguistics*, 51(1):275–338.

Junchao Wu, Runzhe Zhan, Derek F. Wong, Shu Yang, Xinyi Yang, Yulin Yuan, and Lidia S. Chao. 2024. Detectrl: Benchmarking llm-generated text detection in real-world scenarios. *CoRR*, abs/2410.23746.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Xianjun Yang, Wei Cheng, Linda R. Petzold, William Yang Wang, and Haifeng Chen. 2023. DNA-GPT: divergent n-gram analysis for training-free detection of gpt-generated text. *CoRR*, abs/2305.17359.

Biru Zhu, Lifan Yuan, Ganqu Cui, Yangyi Chen, Chong Fu, Bingxiang He, Yangdong Deng, Zhiyuan Liu, Maosong Sun, and Ming Gu. 2023. Beat llms at their own game: Zero-shot llm-generated text detection via querying chatgpt. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 7470–7483. Association for Computational Linguistics.

Andy Zou, Long Phan, Sarah Li Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, Shashwat Goel, Nathaniel Li, Michael J. Byun, Zifan Wang, Alex Mallen, Steven Basart, Sanmi Koyejo, Dawn Song, Matt Fredrikson, and 2 others. 2023.

Representation engineering: A top-down approach to AI transparency. *CoRR*, abs/2310.01405.

Contents

1 Introduction **1**

2 Related Work **2**

3 Methodology **2**
- 3.1 Overview 2
- 3.2 Layer-wise text representations . . 3
- 3.3 Constructing steering vectors . . . 3
- 3.4 Projection features 3
- 3.5 Detection head 4

4 Experiments **4**
- 4.1 Experimental setup 4
- 4.2 Experimental results 5
- 4.3 Ablation studies 6

5 What do the steering vectors detect? **7**

6 Conclusion **8**

7 Limitations and Future Work **9**

A Potential Risks **12**

B Inference Latency and Compute Overhead **12**

C Ablation on the choice of LLM backbone **14**

D Interpretability Analysis **15**
- D.1 Per-layer contribution to the detector 15
- D.2 Logit-lens decoding of the steering vectors 15
- D.3 Stylistic-feature baseline 16
- D.4 Cross-LM Logit-lens decoding of steering vectors 16
- D.5 Per-token visualization 17

E Results on the COLING-2025 MGT benchmark **18**
- E.1 Setup 18
- E.2 Results 18
- E.3 Ablation on downstream classifier 19

F Results Tables **19**
- F.1 DetectRL 19
- F.2 MIRAGE 19
- F.3 MIRAGE → DetectRL 19
- F.4 Ablation 19

A Potential Risks

Like other fake-text detectors, SV-Detect could be misused in high-stakes settings such as academic misconduct accusations, moderation, or authorship disputes if its predictions are treated as definitive evidence rather than probabilistic signals. False positives may unfairly penalize human authors, while false negatives may allow machine-generated text to evade detection. We therefore view SV-Detect as a decision-support tool rather than a standalone adjudicator.

A second risk is adversarial adaptation: public knowledge of the detector may encourage authors or model providers to modify generation style or post-process outputs in ways that reduce detectability. Although our experiments show robustness across several shifts and editing settings, we do not claim robustness to all future laundering or obfuscation strategies.

Finally, broad deployment of fake-text detectors may have downstream social effects, including over-policing legitimate writing assistance or disadvantaging users whose writing style differs from the training distribution. For these reasons, such systems should be deployed with clear uncertainty communication, human oversight, and regular re-evaluation on new domains and attack settings.

B Inference Latency and Compute Overhead

All measurements were obtained on a single NVIDIA A100-PCIe GPU (40 GB) in `float16`. We benchmark inference on 64 texts from MIRAGE-DIG/GENERATE (mean length: 980 characters), truncated to 512 tokens. Each detector is warmed up for three runs before timing. We report per-text latency at batch size 1, throughput at batch size 16, total parameter count (frozen backbone plus detector head), and peak GPU memory. We will release code to reproduce all experiments upon acceptance.

SV-Detect adds almost no overhead beyond a single LM forward pass. SV-Detect runs in 25.7 ms per text, essentially matching a bare Log-Likelihood scoring pass through the same GPT-Neo-2.7B backbone (27.6 ms). The small advantage in favor of SV-Detect is expected: Log-Likelihood requires a full 50,257-way log-softmax at every token, whereas SV-Detect only uses hidden states, followed by lightweight mean pooling,

Detector	Params	Latency (ms/text, b=1)	Throughput (texts/s, b=16)	Peak GPU (MB)
SV-Detect (ours)	2.65 B	25.71	74.3	8,951
Log-Likelihood	2.65 B	27.63	75.1	7,746
Fast-DetectGPT	2.65 B	49.79	37.2	10,703
RoBERTa-Base	0.12 B	7.21	1,295	327
RoBERTa-Large	0.36 B	11.40	714	783

Table 1: Inference cost comparison on an A100 40 GB (`fp16`, 512-token cap, MIRAGE-DIG texts). Parameter counts include the frozen backbone and the detector head. For SV-Detect, the detector-specific overhead is only a small logistic-regression head on top of the LM representations.

cosine projections, and a tiny logistic-regression head. In practice, the detector-specific computation is negligible compared to the backbone forward pass.

SV-Detect is much cheaper than perturbation-based LLM detectors. Fast-DetectGPT requires two forward passes through a 2.7B-class model and is therefore about $1.9\times$ slower than SV-Detect in our setup (49.8 ms vs. 25.7 ms per text). The gap would be even larger for DetectGPT, whose cost scales linearly with the number of perturbations. Using the standard $K{=}100$ setting, a simple extrapolation from the Log-Likelihood baseline yields roughly 2.8 s per text, i.e. around $100\times$ the cost of SV-Detect. We therefore do not benchmark DetectGPT separately, since its overhead is determined directly by repeated LM forward passes.

RoBERTa-based supervised detectors are cheaper, but less robust. RoBERTa-Base and RoBERTa-Large are substantially faster than SV-Detect, with latencies of 7.2 ms and 11.4 ms per text, respectively. The throughput difference is even larger at batch size 16, reflecting both smaller parameter counts and lower memory pressure. This is the main compute–robustness trade-off of our approach: SV-Detect inherits the cost of an LLM forward pass, whereas small encoder-based detectors are far cheaper. However, as shown in the main experiments, these lighter supervised baselines degrade substantially under source-model and attack-family shifts. In contrast, SV-Detect remains close to the cost of a single zero-shot LM forward while providing much stronger generalization.

Training overhead is modest. The training cost of SV-Detect is dominated by a single offline pass through the reference LM to extract layer-wise activations, which can then be cached once per benchmark. The downstream classifier operates on only L or $L \times 3$ features and fits in seconds on a CPU. By contrast, RoBERTa-based supervised baselines require full encoder fine-tuning for each training setup, which is the main difference in training cost rather than inference cost.

C Ablation on the choice of LLM backbone

In the main experiments, SV-Detect uses GPT-Neo as the probe LLM in order to match the backbone used by prior work and enable a fair comparison. Here, we study how the method changes when the probe backbone is replaced with a different small open-weight LLM. We consider four alternatives with parameter counts broadly comparable to GPT-Neo: `google/gemma-3-1b-it`, `google/gemma-3-1b-pt`, `Qwen/Qwen3-1.7B-Base`, and `Qwen/Qwen3-1.7B`. This setup also allows us to compare pretrained/base and instruction-tuned variants within the same model family.

Table 2 reports AUROC on DetectRL Multi-Domain transfer. Overall, the strongest results are obtained with the Qwen backbones. Both Qwen3-1.7B and Qwen3-1.7B-Base improve over GPT-Neo on most train–test pairs while retaining near-perfect diagonal performance. Among them, Qwen3-1.7B-Base is the strongest overall, with particularly robust off-diagonal transfer when trained on ARXIV, XSUM, or WRITING. This suggests that, in our setting, a less heavily post-trained representation space may preserve cleaner linear directions for separating human-written and machine-generated text.

GPT-Neo remains competitive and substantially more robust than the Gemma instruction-tuned backbone, but it is generally weaker than the Qwen variants, especially on harder cross-domain pairs such as ARXIV → WRITING and XSUM → WRITING. Gemma3-1B-pt performs reasonably well when trained on ARXIV, WRITING, or REVIEW, but degrades sharply for XSUM-trained steering vectors, with some transfer pairs close to chance.

The weakest backbone overall is Gemma3-1B-it, which is the instruction-tuned counterpart of Gemma3-1B-pt. Several of its train-test pairs collapse to AUROC near 50, especially when the detector is trained on XSUM or REVIEW. A plausible explanation is that instruction tuning reshapes the hidden representation space toward assistant-style response following, which may make the linear real-vs-fake directions used by SV-Detect less stable under cross-domain transfer. In contrast, the pretrained Gemma3-1B-pt backbone preserves substantially more usable transfer signal. While this interpretation is necessarily speculative, the strong contrast between the `-pt` and `-it` variants suggests that post-training can materially affect the geometry on which steering-vector-based detection relies.

Taken together, these results indicate that SV-Detect is not tied to GPT-Neo and remains effective across several backbone families, but backbone choice matters. In this ablation, Qwen-style representations provide the most stable geometry for steering-vector-based detection, whereas instruction-tuned Gemma representations are substantially less robust under transfer.

GPT-Neo				
Train	ArX.	XSum	Writ.	Rev.
ArXiv	100.00	97.83	84.24	94.43
XSum	96.23	99.92	79.73	91.76
Writing	85.53	78.01	99.97	99.35
Review	90.67	79.30	98.74	99.87
Qwen3-1.7B				
Train	ArX.	XSum	Writ.	Rev.
ArXiv	100.00	99.45	89.99	97.78
XSum	99.40	99.95	95.72	98.83
Writing	88.79	90.22	99.96	99.58
Review	93.06	94.35	99.47	99.82
Qwen3-1.7B-Base				
Train	ArX.	XSum	Writ.	Rev.
ArXiv	100.00	99.08	94.67	98.37
XSum	99.57	99.97	96.44	99.29
Writing	91.22	89.76	99.98	99.61
Review	93.78	91.55	99.64	99.86
Gemma3-1B-pt				
Train	ArX.	XSum	Writ.	Rev.
ArXiv	99.99	98.60	88.92	97.13
XSum	50.80	93.94	50.79	50.55
Writing	87.68	92.77	99.97	99.60
Review	90.81	94.86	99.15	99.92
Gemma3-1B-it				
Train	ArX.	XSum	Writ.	Rev.
ArXiv	100.00	98.72	78.79	95.94
XSum	50.00	50.00	50.00	50.00
Writing	50.79	50.05	83.90	70.58
Review	50.00	50.00	50.00	53.70

Table 2: Ablation on the probe LLM backbone for DetectRL Multi-Domain transfer. Each block shows AUROC for one backbone; rows denote the training domain and columns denote the evaluation domain.

D Interpretability Analysis

This section provides the full methodology underlying the interpretability analyses summarized in Section 5. Specifically, we study: (i) per-layer attribution of the detector, (ii) logit-lens decoding of steering vectors, (iii) a hand-crafted stylistic-feature baseline, and (iv) token-level visualization of where the detection signal concentrates within a text. Unless stated otherwise, all analyses use the same frozen reference model (GPT-Neo-2.7B) and the same trained detectors as in the main experiments.

D.1 Per-layer contribution to the detector

Setup. Our detector is a logistic regression applied to layer-wise projection scores. For DetectRL, we use one feature per transformer layer:

$$\mathbf{s}(x) = (s_1(x), \ldots, s_L(x)) \in \mathbb{R}^L,$$

$$s_\ell(x) = \frac{\langle a_\ell(x), v_\ell \rangle}{\|a_\ell(x)\|_2},$$

where $a_\ell(x)$ is the mean-pooled residual representation at layer ℓ, and v_ℓ is the corresponding steering vector. For MIRAGE, we retain three orthonormalized directions per layer (one per task), so $\mathbf{s}(x) \in \mathbb{R}^{L\times 3}$ is flattened before being passed to the classifier. In all cases, we train a standardized logistic regression (`StandardScaler` followed by ℓ_2-regularized `LogisticRegression` with the default `liblinear` solver).

Attribution. After fitting the classifier, we attribute importance to each scalar feature using the magnitude-weighted coefficient

$$c_\ell = |w_\ell| \cdot \sigma_\ell,$$

where w_ℓ is the learned coefficient and σ_ℓ is the empirical standard deviation of feature ℓ on the training set. This quantity is invariant to feature rescaling and corresponds to the effective contribution of a standardized feature to the classifier logit. For MIRAGE, we sum c_ℓ across the three task-specific directions to obtain a single per-layer attribution score.

Fig. 8 shows that, on MIRAGE, the detector's signal is concentrated in the final block of layers (L29-31), consistent with prior observations that higher-level stylistic features tend to emerge closer to the model output. On DetectRL, the contribution is more distributed: there is a sharp peak at L0-2,

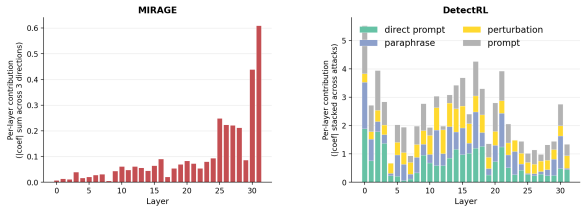


Figure 8

which we associate with surface-level lexical cues, a broader ridge at L14-18, and a secondary peak around L21. This profile motivates our choice of L25–31 for MIRAGE and L14-18 for DetectRL as the main targets of the logit-lens analysis below.

D.2 Logit-lens decoding of the steering vectors

Method. We apply a standard logit-lens probe (nostalgebraist, 2020) to each layer-ℓ steering vector v_ℓ by projecting it through the model's final layer norm and unembedding head:

$$\text{logits}_\ell = W_U^\top \, \text{LN}_f(v_\ell) \in \mathbb{R}^{|\mathcal{V}|},$$

where $|\mathcal{V}| = 50{,}257$ is the GPT-Neo-2.7B vocabulary size. The interpretation is directional: the top-ranked indices correspond to tokens aligned with the $+v_\ell$ direction, which we interpret as *LLM-leaning*, while the bottom-ranked indices correspond to tokens aligned with $-v_\ell$, i.e. the *human-leaning* direction. We report raw logits rather than softmax probabilities, since the goal is to inspect relative alignment with the direction rather than recover a calibrated next-token distribution.

Results. Fig. 7a shows the top six tokens on each side. The resulting patterns are consistent across the selected layers:

- **MIRAGE, $-v_\ell$ (human-leaning).** Stylized closing punctuation (`!"`, `..."`, `').'`), formal sentence-ending subwords (`authorised`, `itially`), and decorative or exaggerated tokens (`£`, `?????`).
- **MIRAGE, $+v_\ell$ (LLM-leaning).** Surnames and journal abbreviations from the arXiv-derived split together with fragments more characteristic of academic prose (`baugh`, `foreseen`, `paralleled`, `flavorful`).
- **DetectRL, $+v_\ell$ (LLM-leaning).** Subword fragments of formal or emphatic vocabulary (`astical`, `TRY`, `ermanent`, `ebted`) and atypical punctuation or Unicode glyphs.

- **DetectRL, $-v_\ell$ (human-leaning).** More colloquial or topical fragments, including `organising`, `Playstation`, `traged`, `adolesc`, and `bigot`.

We stress that token-level logit-lens readings are noisy at any single layer. The patterns above should therefore be read as the consensus across the top contributing layers, rather than as literal interpretations of isolated tokens. Tokens that appear only at one layer are likely artifacts and should not be over-interpreted.

D.3 Stylistic-feature baseline

To quantify how much of the detector's signal can be explained by hand-recognizable surface cues, we train a baseline logistic regression on a fixed bank of regex-derived features and compare its in-distribution AUROC to the full steering-vector pipeline.

Feature bank. For each text, we compute counts and per-1,000-character rates for the following classes of patterns:

- **Typography:** em-dash usage (–), curly quotes, Unicode ellipsis (. . .) versus three-dot ellipsis (. . .), and space-before-period.
- **Markdown / formatting:** bold, italics, inline code, fenced code blocks, headers, and Oxford-comma lists.
- **Discourse:** formal connectives (`Moreover`, `Furthermore`, `Additionally`, `In conclusion`, `Overall`), summary prefixes, casual hedges (`kinda`, `sorta`, `tbh`), and inclusive pronouns (`we`, `us`, `our`).
- **Lexical:** "polished paraphrase" phrasing (e.g. `utilizing`, `endeavor`, `showcase`), consonant-cluster words, and log character length.
- **Prompt leakage and template artifacts:** `[Assistant]:` prefixes, templated review phrasing, generic summary formulas, and refusal-style stubs.
- **Mid-word case flips:** regex patterns that detect uppercase letters inside otherwise lowercase words, a known artifact under aggressive perturbation attacks.

We standardize the resulting feature vectors and fit an ℓ_2-regularized logistic regression, tuning the regularization strength C by 5-fold cross-validation on a held-out split.

Results. Table 3 reports the in-distribution AUROC of this stylistic-feature baseline across all evaluated settings, together with the full steering-vector pipeline for reference; the corresponding dumbbell visualization is shown in Fig. 7b. Two observations stand out:

1. Stylistic features alone form a non-trivial baseline, reaching 76–82 AUROC on MIRAGE and 88–91 AUROC on DetectRL. This is consistent with prior work showing that surface-level statistical and stylistic cues can already carry substantial signal for distinguishing machine-generated from human-written text.
2. The full steering-vector pipeline still improves on this baseline by 13–24 AUROC points on MIRAGE and 9–12 points on DetectRL. We interpret this remaining gap as the part of the signal that is present in hidden representations but not recoverable from the regex bank. At the same time, the highest-weight regex features overlap substantially with the tokens surfaced by the logit-lens analysis (Fig. 7), including em-dashes, formal connectives, and prompt-leakage templates.

D.4 Cross-LM Logit-lens decoding of steering vectors

The main interpretability analyses in Sec D.1 decode steering vectors through the same reference LM that produced them, namely GPT-Neo-2.7B. A natural question is whether the real-vs-fake directions learned by SV-Detect are specific to GPT-Neo's residual stream, or whether similar directions emerge in other modern LMs. To address this, we retrain the steering-vector pipeline independently on several additional reference LMs and then decode the resulting dense steering vectors through each model's own unembedding.

Reference LMs and datasets. We use four reference LMs spanning three families: `EleutherAI/gpt-neo-2.7B`, `Qwen/Qwen3-1.7B`, `google/gemma-3-1b-pt`, and `google/gemma-3-1b-it`. For each LM, we extract per-layer mean-pooled residual activations on the corresponding in-domain training corpora

$+v_\ell$ LLM-leaning (top-pooled per LM)

Gemma-3-IT	Gemma-3-PT	Qwen-3	GPT-Neo
captivating	<eos>	newfound	the
leveraging	utilizing	advancements	both
unwavering	esteemed	llen	behaviors
utilization	impactful	firsthand	color
utilizing	confira	unfavorable	multiple
fostering	utilize	favorable	utilizing
multifaceted	upcoming	prioritize	upon
prioritization	leveraging	utilization	all
<unused2157>	mesmerizing	endeavors	global
flavorful	prioritizing	utilizing	future

$-v_\ell$ human-leaning (top-pooled per LM)

Gemma-3-IT	Gemma-3-PT	Qwen-3	GPT-Neo
stupid	saddo	probably	nasa
had	anyway	analysed	invests
doesnt	kinda	really	bolst
really	probably	anyway	organise
other	presumably	basically	\xad
anyway	actually	anyhow	lawy
this	obviously	sic	anecd
basically	necessarily	analyse	licences
but	programme	then	nostalg
etc	bloke	etc	athlet

Figure 9: Top-pooled tokens per reference LM, separately for the $+v_\ell$ (LLM-leaning, left) and $-v_\ell$ (human-leaning, right) sides. For each LM, we pool the top-K tokens across the last third of layers and across all analyzed DetectRL / MIRAGE settings, then list the most frequent normalized tokens. Despite differences in family and tokenizer, the positive side consistently surfaces a polished LLM-style register, while the negative side is more casual and discourse-like.

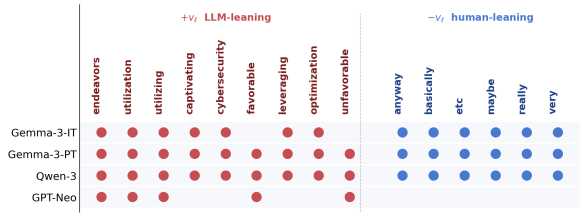


Figure 10: Consensus tokens across the four reference LMs. Rows are tokens and columns are LMs; a colored dot indicates that the token appears in that LM's pooled top-token set. The positive side shows a stable cross-LM cluster of polished LLM-register terms such as `endeavors`, `utilization`, and `utilizing`, while the negative side yields a more casual human-register cluster including `anyway`, `basically`, `etc`, `maybe`, `really`, and `very`.

and fit the standard `logreg_l2` steering-vector construction described in Sec. 3. This yields one dense steering vector per layer for each (LM, dataset) pair. GPT-Neo is analyzed on four datasets (MIRAGE GENERATE/POLISH/REWRITE and DetectRL `direct_prompt`), while the other three LMs are analyzed on these four settings plus four additional DetectRL per-domain subsets (`arxiv`, `xsum`, `writing_prompt`, `yelp_review`), for a total of 28 (LM, dataset) pairs.

Method. For each (LM, dataset, layer), we decode the steering vector through the LM's own final layer norm and unembedding using logit-lens probe as described in Sec. D.2.

Because single-layer readings are noisy and strongly affected by tokenizer idiosyncrasies, we aggregate within each LM before comparing across LMs. Specifically, for each LM we pool the top-$K_{\text{pool}} = 8$ tokens at every layer in the last third of the network depth, separately by sign, then take the union across the LM's datasets. Layer ranges are L17–25 for Gemma-3, L18–27 for Qwen3, and L21–31 for GPT-Neo. Before computing cross-LM overlap, we normalize tokens by stripping whitespace, lowercasing, applying NFKC normalization, dropping control/format characters, and filtering out pure punctuation as well as tokens with less than 50% ASCII letters. This removes tokenizer-specific artifacts and makes the consensus comparison more meaningful.

Results. Fig. 9 shows the top pooled tokens for each LM separately, and Fig. 10 summarizes the cross-LM consensus. On the positive side, the four LMs converge on a clear LLM-polished register. Three tokens (*endeavors*, *utilization*, and *utilizing*) appear in the top pool of all four LMs, and a broader cluster including *captivating*, *cybersecurity*, *leveraging*, *optimization*, *favorable*, and *unfavorable* appears in three of the four. On the negative side, the consensus is weaker but still interpretable: the most stable tokens are casual discourse markers such as *anyway*, *basically*, *etc*, *maybe*, *really*, and *very*. Here, the three smaller modern LMs agree more strongly with one another, while GPT-Neo more often surfaces named-entity fragments and topical content.

Overall, this analysis suggests that the lexical contrast captured by SV-Detect is not specific to GPT-Neo. When the steering vector pipeline is retrained independently on different LM families and their dense directions are decoded through each model's own unembedding, the same broad semantic register reappears: LLM-polished, formal, and abstract language on the positive side, and more casual, discourse-like language on the negative side.

D.5 Per-token visualization

The token-level figure in the main paper (Fig. 1) is constructed as follows. Given a text x and the best-AUROC layer $\ell^\star$ identified by an AUROC-probe on the held-out set, we forward x through the frozen reference model and record the per-token residual vector

$$h_{\ell^\star}(x_t) \in \mathbb{R}^d \qquad \text{for } t \in \{1, \ldots, T\}.$$

Each token is then assigned the dot product

$$\rho_t = \langle h_{\ell^\star}(x_t),\, v_{\ell^\star} \rangle$$

where $v_{\ell^\star}$ is the steering vector at layer $\ell^\star$. Tokens are colored according to ρ_t, with the scale nor-

Setting	Subset	Stylistic AUROC	SV-Detect AUROC	Gap
MIRAGE-DIG				
DIG	Generate	76.98	99.12	+22.1
DIG	Polish	80.74	95.09	+14.3
DIG	Rewrite	75.94	95.03	+19.1
MIRAGE-SIG				
SIG	Generate	75.59	99.09	+23.5
SIG	Polish	81.71	94.99	+13.3
SIG	Rewrite	76.36	94.21	+17.8
DetectRL (Multi-Attack)				
DetectRL	Direct	90.73	99.99	+9.3
DetectRL	Prompt	90.11	99.96	+9.9
DetectRL	Paraph.	88.69	99.98	+11.3
DetectRL	Perturb.	87.81	99.99	+12.2

Table 3: In-distribution AUROC of the stylistic-feature baseline (Section D.3) compared to the full steering-vector pipeline. The **Gap** column shows the additional discriminative signal captured by hidden representations beyond the regex feature bank.

malized to the 70th percentile of $|\rho|$ in the corpus so that most informative tokens reach full saturation. Red denotes positive projection ($+v$, LLM-leaning) and blue denotes negative projection ($-v$, human-leaning). The classification banner above each text corresponds to the full text-level prediction produced by the trained detector used in the main experiments.

E Results on the COLING-2025 MGT benchmark

For completeness, we additionally evaluate SV-Detect on the COLING-2025 Multilingual MGT benchmark, English split (Wang et al., 2025). Unlike DetectRL and MIRAGE, this benchmark pools many generators into a single binary detection task and provides an official held-out test set with labels. It therefore offers a complementary large-scale setting for evaluating the same steering-vector pipeline and testing its sensitivity to the composition of the generator pool.

E.1 Setup

Data. The COLING-2025 MGT English split contains three partitions: (i) **train**: 610,767 examples (228,922 human, 381,845 machine-generated from 27 generators), (ii) **dev**: 261,758 examples (98,328 human, 163,430 machine-generated), and (iii) **test**: 73,941 examples, released with ground-truth labels for leaderboard evaluation. The task is binary classification (human vs. machine-generated).

Backbones. We evaluate SV-Detect with two reference LMs: the `EleutherAI/gpt-neo-2.7B` used in the rest of the paper, and the larger `meta-llama/Llama-2-7b-hf`. For each backbone, we extract per-layer mean-pooled residual activations and use the same downstream logistic-regression detector as in the main paper.

Ablation axes. We vary two components of the pipeline:

1. **Steering-vector construction.** We compare the three constructions used elsewhere in the paper: *Mean* (normalized class-mean difference), *PCA* (leading principal component of fake-minus-real activation differences) and *LogReg* (ℓ_2-regularized logistic regression on standardized activations, our default construction).

2. **Generator-pool filter.** The training split contains a long tail of weak open-source generators whose outputs are often much less realistic than those of stronger LLMs. To test whether these generators help or hurt, we compare:

   - *all*: all 27 generators
   - *trimmed*: remove 20 weak generators (Dolly, BLOOMZ, OPT-*, Flan-T5-*, T0-*), dropping 133,193 samples and retaining 248,652 fake examples. Generators are only removed from the training and dev splits, the whole test split is still used.

We use the same logistic regression detector as described in Sec. 3. The downstream classifier is used elsewhere in the paper. We sweep $C \in \{10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 100\}$ and select the best value on dev AUROC. All other hyperparameters match the main experiments as described in Sec. 4.1.

E.2 Results

Llama-based SV-Detect outperforms all baselines by a substantial margin. Removing weak models from the training data helps Llama-based SV-Detect further, gaining about 2 points in Accuracy and 1.7 in F_1. However, for GPT-Neo-based SV-Detect, removing weak models hurts. We attribute this to the size and capability of the LM backbone: larger and more capable backbones, such as Llama,

Rank	Team / Method	Macro-F1	Acc.
1	**SV-Detect (Llama-2-7b, trimmed)**	**84.8**	**85.00**
2	SV-Detect (Llama-2-7b, all)	**83.7**	**84.00**
3	Advacheck	83.07	83.11
4	Unibuc-NLP	83.01	83.33
5	Fraunhofer SIT	82.80	82.89
6	Grape	81.88	82.23
7	TechExperts(IPN)	81.53	81.81
8	TurQUaz	80.68	80.74
9	SzegedAI	79.10	79.29
10	AAIG	78.74	79.34
11	DCBU	77.13	78.01
12	SV-Detect (GPT-Neo, all)	**75.6**	**76.50**
13	Alfa	75.37	76.42
14	L3i++	74.63	75.54
15	LuxVeri	74.58	75.68
16	azlearning	74.14	75.17
17	SV-Detect (GPT-Neo, trimmed)	**74.0**	**75.50**
18	honghanhh	73.94	75.14
–	Baseline	73.42	74.89
19	VX1291	72.93	74.83
20	cuettransform	72.32	73.16
21	rockstart	72.24	73.89
22	batirsdu	71.01	71.42
23	IPN-CIC	70.68	72.42
24	Ai-Monitors	70.57	72.65
25	semanticcuet	70.05	71.96
26	hmcgovern	68.48	69.51
27	abhirak0603	68.02	70.50
28	cnlpnitspp	65.02	68.76
29	mail6djj	64.66	68.46
30	bennben	63.32	67.48
31	saehyunma	62.80	67.25
32	yuwert777	62.14	66.69
33	seven	59.09	63.20
34	fangsifan	58.48	62.68
35	yaoxy	57.28	64.20
36	jojoc	54.16	60.37
37	dominikmacko	49.94	50.78
38	tropaleum	49.57	50.60
39	starlight1	47.57	56.65
40	nitstejasrikar	44.89	57.24

Table 4: Adapted English leaderboard from Table 4 of Wang et al. (2025), with our three LogReg-based SV-Detect variants inserted. Ranking is by Macro-F1.

benefit from a cleaner and more homogeneous generator pool, whereas smaller backbones like GPT-Neo appear to benefit from the additional diversity provided by weaker generators.

E.3 Ablation on downstream classifier

Tab. 5 and Tab. 6 report results for all six combinations of steering-vector construction and generator-pool filter on both backbones. As in DetectRL and MIRAGE, the LogReg construction consistently outperforms Mean and PCA. The larger Llama-2-7B backbone further improves substantially over GPT-Neo-2.7B.

F Results Tables

In this section, we provide comprehensive tables with results for all experiments in Sec. 4

GPT-Neo-2.7B					
SV construction	**Pool**	**Dev AUROC**	**Test AUROC**	**Acc.**	F_1
Mean	all	0.865	0.860	0.718	0.696
Mean	trimmed	0.855	0.856	0.737	0.721
PCA	all	0.740	0.751	0.559	0.439
PCA	trimmed	0.737	0.753	0.557	0.433
LogReg	all	**0.975**	0.866	**0.765**	**0.756**
LogReg	trimmed	0.952	**0.873**	0.755	0.740

Table 5: SV-Detect on the COLING-2025 English benchmark with a GPT-Neo-2.7B backbone.

Llama-2-7B					
SV construction	**Pool**	**Dev AUROC**	**Test AUROC**	**Acc.**	F_1
Mean	all	0.922	0.912	0.716	0.690
Mean	trimmed	0.908	0.887	0.694	0.666
PCA	all	0.864	0.796	0.650	0.615
PCA	trimmed	0.858	0.798	0.647	0.610
LogReg	all	**0.984**	**0.938**	0.840	0.837
LogReg	trimmed	0.974	0.937	**0.850**	**0.848**

Table 6: SV-Detect on the COLING-2025 English benchmark with a Llama-2-7B backbone.

F.1 DetectRL

Tables 7 and 8 report the in-domain and cross-source performance of SV-Detect and the baselines, respectively. Fig. 3a and Fig. 3b provide a visual summary of these results.

F.2 MIRAGE

Table 9 reports performance of SV-Detect and the baselines on the MIRAGE benchmark. Fig. 4a provides a visual summary of these results.

F.3 MIRAGE → DetectRL

Table 10 provides results of using steering vectors learned from the training setup used for MIRAGE-style tasks on DetectRL. Fig. 4b provides a visual summary of these results.

F.4 Ablation

Tables 11 and 12provide results of ablating choice of the downstream classifier and steering vector construction method. Sec. 4.3 provides a summary of these results.

Method	ArXiv		XSum		Writing		Review	
	AUROC	F_1	AUROC	F_1	AUROC	F_1	AUROC	F_1
Multi-Domain								
Log-Likelihood	65.35	57.55	45.68	41.32	68.00	59.38	75.84	67.22
Entropy	48.39	29.71	67.84	57.23	39.06	20.55	28.82	2.14
Rank	57.17	54.62	36.87	22.47	56.26	50.90	55.08	51.90
Log-Rank	67.01	60.09	46.74	42.60	67.58	57.57	76.40	69.88
LRR	70.54	61.34	50.09	38.38	64.65	53.09	76.61	68.99
NPR	53.85	49.65	34.59	18.31	54.96	52.30	50.09	45.39
DetectGPT	22.15	0.00	12.21	0.00	58.95	50.83	44.43	35.25
Revise-Detect.	70.40	37.51	50.34	46.07	73.24	64.29	75.01	68.71
Binoculars	84.03	76.77	77.39	72.18	94.38	79.73	90.00	84.32
Fast-DetectGPT	43.69	24.46	39.19	28.39	74.21	67.84	77.02	71.62
RoBERTa-Base	100.0	100.0	99.99	99.85	99.99	99.65	99.97	99.50
RoBERTa-Large	99.99	99.90	99.85	98.95	99.54	97.73	99.76	98.90
XLM-RoBERTa-Base	100.0	100.0	99.97	99.55	99.84	98.76	99.88	99.05
XLM-RoBERTa-Large	99.98	99.85	99.84	98.95	99.85	98.31	96.40	92.66
SV-Detect	**100.0**	**100.0**	**99.92**	**99.40**	**99.97**	**99.50**	**99.87**	**99.10**

Method	GPT-3.5		Claude		PaLM-2		Llama-2	
	AUROC	F_1	AUROC	F_1	AUROC	F_1	AUROC	F_1
Multi-LLM								
Log-Likelihood	62.89	57.80	43.32	28.10	70.03	60.73	75.65	65.90
Entropy	46.84	23.29	52.25	30.42	45.34	16.56	43.48	66.75
Rank	52.19	49.32	41.68	22.78	50.40	41.74	57.05	54.40
Log-Rank	62.84	56.87	43.32	30.12	70.89	63.09	77.97	66.66
LRR	61.61	52.12	43.30	18.91	71.17	65.51	83.65	75.51
NPR	50.29	43.81	41.64	32.91	44.64	34.77	52.53	48.68
DetectGPT	43.46	26.27	32.86	12.56	26.72	0.00	36.71	20.40
DNA-GPT	61.87	55.04	48.88	25.67	71.48	60.77	75.22	62.89
Revise-Detect.	70.10	62.72	49.87	27.28	69.84	59.03	75.65	65.87
Binoculars	88.14	82.50	55.15	39.35	93.30	88.20	96.64	92.30
Fast-DetectGPT	65.56	59.55	30.01	0.00	65.99	57.58	76.79	69.08
RoBERTa-Base	99.97	99.70	99.98	99.80	99.94	99.40	99.84	99.45
RoBERTa-Large	99.77	98.86	96.23	92.48	97.93	92.64	86.72	76.17
XLM-RoBERTa-Base	99.88	99.45	98.26	97.48	98.77	97.19	99.69	98.57
XLM-RoBERTa-Large	99.55	97.56	91.67	84.24	98.73	94.43	99.66	97.67
SV-Detect	**99.99**	**99.65**	**99.91**	**98.85**	**99.88**	**98.47**	**99.99**	**99.65**

Method	Direct		Prompt		Paraph.		Perturb.		Mixing	
	AUROC	F_1	AUROC	F_1	AUROC	F_1	AUROC	F_1	AUROC	F_1
Multi-Attack										
Log-Likelihood	89.25	82.09	86.87	78.16	64.55	57.59	35.51	0.78	63.70	53.31
Entropy	26.47	0.00	26.18	0.00	48.12	26.01	68.62	68.95	49.37	28.52
Rank	83.50	76.27	81.21	72.86	60.60	52.60	8.04	0.00	52.05	42.46
Log-Rank	89.25	81.45	86.35	77.51	64.69	59.17	37.71	0.78	64.63	56.86
LRR	85.83	77.40	80.80	74.30	63.99	55.20	45.91	29.27	66.12	53.81
NPR	77.98	71.61	77.15	70.63	56.94	46.25	6.78	0.00	48.63	37.65
DetectGPT	52.84	40.90	51.83	37.98	31.79	16.89	18.21	0.00	26.28	0.00
DNA-GPT	88.01	80.78	85.62	77.47	65.61	54.94	40.45	2.73	62.14	50.89
Revise-Detect.	86.88	79.61	84.89	76.21	67.26	62.03	43.98	7.56	65.27	54.39
Binoculars	94.87	89.73	93.45	88.12	88.34	81.56	76.89	69.34	89.12	83.67
Fast-DetectGPT	79.56	72.45	78.43	70.34	70.12	62.89	49.56	41.23	67.23	59.78
RoBERTa-Base	99.87	99.60	99.78	99.47	99.67	99.12	98.32	97.45	99.12	98.76
RoBERTa-Large	98.73	97.83	98.45	97.56	97.89	96.78	96.12	94.67	97.56	96.34
XLM-RoBERTa-Base	99.56	99.12	99.23	99.01	98.89	98.34	98.56	97.89	99.01	98.56
XLM-RoBERTa-Large	99.45	98.67	98.89	97.98	98.23	97.67	97.89	96.34	98.67	97.89
SV-Detect	**99.99**	**99.70**	**99.96**	**99.55**	**99.98**	**99.55**	**99.99**	**99.95**	**99.83**	**98.50**

Table 7: Performance of detectors on DetectRL in the Multi-Domain, Multi-LLM, and Multi-Attack settings.

	SV-Detect				Fast-DetectGPT				RoBERTa-Base			
Train	ArXiv	XSum	Writing	Review	ArXiv	XSum	Writing	Review	ArXiv	XSum	Writing	Review
Multi-Domain (F_1)												
ArXiv	100.0	94.10	78.47	89.60	24.46	23.71	59.67	60.17	100.0	75.90	77.68	70.69
XSum	90.38	99.40	73.25	84.60	28.43	28.39	62.99	63.08	68.43	99.85	71.79	67.17
Writing	79.08	71.25	99.50	96.72	34.81	33.60	67.84	68.30	78.58	72.72	99.65	94.24
Review	83.33	73.19	94.60	99.10	40.70	37.66	68.25	71.62	82.64	84.15	85.10	99.50

	SV-Detect				Fast-DetectGPT				RoBERTa-Base			
Train	GPT-3.5	PaLM-2	Claude	Llama-2	GPT-3.5	PaLM-2	Claude	Llama-2	GPT-3.5	PaLM-2	Claude	Llama-2
Multi-LLM (F_1)												
GPT-3.5	99.65	86.86	92.79	97.58	59.55	59.56	12.96	69.93	99.97	70.34	62.90	94.68
PaLM-2	99.30	98.47	94.17	98.56	55.77	57.58	8.20	68.43	99.25	99.40	93.43	99.25
Claude	99.00	93.82	98.85	96.87	0.19	0.00	0.00	1.18	96.83	83.92	99.80	89.77
Llama-2	99.45	96.43	93.24	99.65	56.28	57.74	8.65	69.08	99.45	93.02	87.56	99.45

	SV-Detect				Fast-DetectGPT				RoBERTa-Base			
Train	Prompt	Paraph.	Perturb.	Mixing	Prompt	Paraph.	Perturb.	Mixing	Prompt	Paraph.	Perturb.	Mixing
Multi-Attack (F_1)												
Direct	98.88	96.41	95.45	96.91	64.01	40.45	41.02	31.81	95.73	94.91	64.32	89.07
Prompt	99.55	95.63	92.36	95.88	64.00	39.94	40.40	31.25	97.18	94.98	86.18	92.92
Paraphrase	96.47	99.55	95.27	97.36	61.54	38.32	36.86	27.90	93.66	98.26	78.81	78.81
Perturb	97.46	98.61	99.95	98.85	64.01	40.45	41.14	31.93	87.01	91.46	98.66	91.38
Mixing	98.97	98.31	97.70	98.50	65.89	46.38	45.78	40.93	93.46	91.93	95.26	93.64

Table 8: Cross-source generalization on DetectRL, reported as F_1. Rows denote the training source and columns denote the evaluation source. We compare SV-Detect with Fast-DetectGPT as a zero-shot baseline and RoBERTa-Base as a supervised baseline across the Multi-Domain, Multi-LLM, and Multi-Attack settings.

MIRAGE-DIG (Disjoint-Input Generation)												
Methods	Generate				Polish				Rewrite			
	AUROC	Accuracy	MCC	TPR@5%	AUROC	Accuracy	MCC	TPR@5%	AUROC	Accuracy	MCC	TPR@5%
Likelihood	0.4936	0.5091	0.0183	0.0147	0.4653	0.5000	0.0000	0.0214	0.4337	0.5000	0.0000	0.0148
LogRank	0.4992	0.5128	0.0260	0.0220	0.4512	0.5000	0.0000	0.0195	0.4225	0.5000	0.0000	0.0132
Entropy	0.6522	0.6150	0.2543	0.1099	0.5543	0.5417	0.1247	0.0954	0.5805	0.5566	0.1650	0.1189
RoBERTa-Base	0.5523	0.5397	0.1434	0.1250	0.4859	0.5010	0.0088	0.0460	0.5020	0.5049	0.0293	0.0569
RoBERTa-Large	0.4716	0.5217	0.0842	0.0871	0.5171	0.5151	0.0340	0.0633	0.5570	0.5385	0.0864	0.0895
LRR	0.5215	0.5341	0.0777	0.0701	0.4081	0.5000	0.0000	0.0200	0.3930	0.5000	0.0000	0.0188
DNA-GPT	0.5733	0.5595	0.1196	0.0776	0.4771	0.5004	0.0110	0.0309	0.4453	0.5001	0.0080	0.0251
NPR	0.6120	0.6140	0.2604	0.0191	0.5071	0.5370	0.1071	0.0318	0.4710	0.5201	0.0663	0.0226
DetectGPT	0.6402	0.6258	0.2758	0.0275	0.5469	0.5531	0.1328	0.0355	0.5061	0.5266	0.0826	0.0283
Fast-DetectGPT	0.7768	0.7234	0.4628	0.4310	0.5720	0.5570	0.1293	0.1189	0.5455	0.5432	0.1015	0.1025
ImBD	0.8597	0.7738	0.5497	0.4065	0.7888	0.7148	0.4300	0.2730	0.7825	0.7068	0.4139	0.2933
DetectAnyLLM	0.9525	0.8988	0.7975	0.7770	0.9297	0.8732	0.7487	0.7756	0.9234	0.8705	0.7447	0.7778
SV-Detect (polish-only)	0.9777	0.9241	0.8483	0.8862	0.9113	0.8364	0.6730	0.6414	0.9105	0.8332	0.6672	0.6437
SV-Detect (3-task)	**0.9912**	**0.9547**	**0.9095**	**0.9579**	**0.9509**	**0.8851**	**0.7706**	**0.7990**	**0.9503**	**0.8856**	**0.7713**	**0.7956**
MIRAGE-SIG (Shared-Input Generation)												
Methods	Generate				Polish				Rewrite			
	AUROC	Accuracy	MCC	TPR@5%	AUROC	Accuracy	MCC	TPR@5%	AUROC	Accuracy	MCC	TPR@5%
Likelihood	0.4968	0.5207	0.0196	0.0145	0.4599	0.5002	0.0030	0.0233	0.4319	0.5000	0.0000	0.0111
LogRank	0.5008	0.5183	0.0182	0.0186	0.4468	0.5000	0.0000	0.0211	0.4221	0.5000	0.0000	0.0118
Entropy	0.6442	0.6123	0.1592	0.1074	0.5640	0.5439	0.0516	0.0946	0.5858	0.5645	0.0918	0.1198
RoBERTa-Base	0.5368	0.5392	0.0529	0.1101	0.4741	0.5011	0.0048	0.0395	0.5099	0.5122	0.0221	0.0668
RoBERTa-Large	0.4703	0.5236	0.0417	0.0910	0.5150	0.5157	0.0283	0.0702	0.5576	0.5426	0.0405	0.0762
LRR	0.5214	0.5311	0.0314	0.0657	0.4076	0.5000	0.0000	0.0238	0.3978	0.5000	0.0000	0.0174
DNA-GPT	0.5759	0.5647	0.0603	0.0813	0.4788	0.5001	0.0036	0.0340	0.4457	0.5002	0.0048	0.0258
NPR	0.6088	0.6170	0.1571	0.0185	0.5074	0.5277	0.0612	0.0293	0.4738	0.5204	0.0340	0.0177
DetectGPT	0.6353	0.6241	0.1719	0.0193	0.5434	0.5515	0.0668	0.0309	0.5079	0.5260	0.0431	0.0239
Fast-DetectGPT	0.7706	0.7193	0.2078	0.4200	0.5727	0.5619	0.0607	0.1238	0.5480	0.5495	0.0525	0.1097
ImBD	0.8612	0.7791	0.5599	0.4183	0.7951	0.7199	0.4451	0.3036	0.7694	0.6920	0.3936	0.2868
DetectAnyLLM	0.9526	0.9059	0.8119	0.7722	0.9316	0.8740	0.7483	0.7779	0.9158	0.8643	0.7320	0.7574
SV-Detect (polish-only)	0.9779	0.9241	0.8488	0.8839	0.9089	0.8290	0.6583	0.6291	0.9039	0.8265	0.6532	0.6169
SV-Detect (3-task)	**0.9909**	**0.9516**	**0.9032**	**0.9516**	**0.9499**	**0.8837**	**0.7703**	**0.7811**	**0.9421**	**0.8749**	**0.7507**	**0.7706**

Table 9: Results across three tasks (GENERATE, POLISH, REWRITE) under two evaluation settings (MIRAGE-DIG and MIRAGE-SIG) on MIRAGE. Metrics reported are AUROC, Accuracy, MCC, and TPR@5%.

Setting	AUROC	AUPR	TPR@5%	Bal. Acc.	MCC	F_1
Multi-Domain						
ArXiv	0.9647	0.9712	0.8670	0.9114	0.8231	0.9096
Writing	0.9370	0.9472	0.7490	0.8650	0.7301	0.8669
XSum	0.9517	0.9636	0.8333	0.8931	0.7908	0.8905
Review	0.9298	0.9459	0.7748	0.8682	0.7413	0.8616
Multi-LLM						
GPT-3.5	0.9935	0.9942	0.9692	0.9631	0.9253	0.9632
Claude	0.8661	0.8610	0.4414	0.7854	0.5711	0.7958
PaLM-2	0.8740	0.8904	0.5734	0.7956	0.5975	0.7973
Llama-2	0.9729	0.9721	0.8759	0.9188	0.8366	0.9195
Multi-Attack						
Direct prompt (no attack)	0.9674	0.9695	0.8501	0.9072	0.8138	0.9061
Prompt attacks (all)	0.9546	0.9591	0.8066	0.8848	0.7696	0.8861
Few-shot prompting	0.9375	0.9424	0.7450	0.8645	0.7296	0.8652
ICO prompting	0.9657	0.9690	0.8621	0.9082	0.8171	0.9054
Paraphrase attacks (all)	0.9409	0.9467	0.7658	0.8720	0.7435	0.8719
Back translation	0.8695	0.8677	0.5238	0.7941	0.5903	0.7959
DIPPER paraphrase	0.9530	0.9582	0.8194	0.8908	0.7843	0.8919
Polish using LLMs	0.9842	0.9855	0.9355	0.9444	0.8879	0.9447
Perturbation attacks (all)	0.9070	0.9175	0.6607	0.8313	0.6644	0.8309
Character-level perturb.	0.8784	0.8914	0.5912	0.7991	0.5982	0.8018
Sentence-level perturb.	0.9251	0.9299	0.6825	0.8447	0.6898	0.8493
Word-level perturb.	0.9205	0.9296	0.6984	0.8452	0.6914	0.8444
Data mixing (all)	0.9465	0.9575	0.8055	0.8869	0.7744	0.8848
LLM-centered mixing	0.9366	0.9408	0.7271	0.8601	0.7197	0.8623
Multi-LLM mixing	0.9607	0.9656	0.8280	0.8952	0.7913	0.8946

Table 10: Results on DetectRL with the three-direction version of SV-Detect across the Multi-Domain, Multi-LLM, and Multi-Attack settings. Metrics reported are AUROC, AUPR, TPR@FPR=5%, balanced accuracy, MCC, and F_1.

Classifier →	Logistic regression (base method)				KNN ($n_neighbors = 5$)				CatBoost (default parameters)			
Multi-Domain												
Train ↓ / Eval →	ArXiv	XSum	Writing	Review	ArXiv	XSum	Writing	Review	ArXiv	XSum	Writing	Review
ArXiv	100.00	97.83	84.24	94.43	100.00	50.30	50.00	50.60	100.00	97.11	90.35	96.16
XSum	96.23	99.92	79.73	91.76	50.00	50.00	50.00	50.00	94.07	99.59	68.60	83.17
Writing	85.53	78.01	99.97	99.35	80.40	67.28	99.76	96.28	81.95	80.79	99.96	99.31
Review	90.67	79.30	98.74	99.87	50.00	50.00	50.00	50.00	82.66	86.77	98.73	99.74

Table 11: Ablation on the downstream classifier for generalization evaluation on DetectRL in the Multi-Domain setting. Metric: AUROC.

Steering vector construction →	Logistic regression (base method)				Mean difference				PCA			
Multi-Domain												
Train ↓ / Eval →	ArXiv	XSum	Writing	Review	ArXiv	XSum	Writing	Review	ArXiv	XSum	Writing	Review
ArXiv	100.00	97.83	84.24	94.43	100.00	75.70	71.68	78.95	91.86	84.33	86.17	83.35
XSum	96.23	99.92	79.73	91.76	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00
Writing	85.53	78.01	99.97	99.35	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00
Review	90.67	79.30	98.74	99.87	84.57	79.32	94.61	98.08	82.42	73.14	52.80	45.36

Table 12: Ablation on the steering vector construction method for generalization evaluation on DetectRL in the Multi-Domain setting. Metric: AUROC.