# AUTOINTERPRETING SPARSE AUTOENCODERS IS A MEASUREMENT PROBLEM

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Autointerpretability scores, where large language models (LLMs) explain and score features, are the predominant way to evaluate sparse autoencoder (SAE) interpretability. Comparing methods with these scores depends on the assumption that they reflect stable feature properties rather than confounding aspects of the evaluation pipeline. Through systematic experiments across four metrics, two models (Pythia-160M, Apertus-8B-Instruct), and six axes of methodological variation, we show that this assumption rarely holds. We find that $(1)$ a single methodological choice explains more score variance than the SAE architecture. Excluding the activation corpus, which is arguably a property of the feature rather than the pipeline, weakens the effect, but it still holds for most model-metric pairs. Moreover, we find that $(2)$ metric reliability is a property of the metric–factor pair, not of the metric, and $(3)$ score agreement does not imply agreement about the top features; a failure that cannot be detected by similar mean scores or highly correlated per-feature scores. These findings suggest that cross-paper comparisons based on autointerpretability scores may reflect pipeline differences rather than architectural differences, providing evidence to an open debate: whether SAE latents are stable, semantically meaningful units for human interpretation. Beyond diagnosing these problems, we propose concrete recommendations for more reliable SAE evaluation.

## 1 INTRODUCTION

Autointerpretability scores are widely relied upon as the main instrument for feature interpretability assessment in Sparse Autoencoders (SAEs) (Bricken et al., 2023; Cunningham et al., 2024; Juang et al., 2024; Templeton et al., 2024), and are additionally used in architecture comparisons (Bussmann et al., 2025; Karvonen et al., 2025). However, if these scores reflect evaluation pipeline configurations rather than feature interpretability, such comparisons are potentially meaningless. Proxy metrics such as reconstruction error and sparsity reflect optimization behavior rather than semantic quality (Chanin & Garriga-Alonso, 2026; Minegishi et al., 2025), and SAEs that perform well on these metrics have been shown to detect only a small subset of true features (Korznikov et al., 2026). This has motivated evaluation frameworks that target interpretability more directly (Makelov et al., 2025; Paulo et al., 2025; Puri et al., 2025), yet an important question remains unanswered: *"Are current autointerpretability evaluation practices stable to support cross-paper comparisons?"*

In autointerpretability, or automated interpretability, a language model (LM) generates a natural language description of a feature, and a second LM evaluates how well that description predicts feature activations on held-out examples (Bills et al., 2023). This pipeline currently underlies most SAE interpretability evaluations (Cunningham et al., 2024; Karvonen et al., 2025; Templeton et al., 2024). Prior work has identified individual sources of pipeline variation that affect scores, such as sample selection strategies (Juang et al., 2024; Paulo et al., 2025), explainer or scorer model capacity (Juang et al., 2024; Paulo et al., 2025; Puri et al., 2025), and prompt construction (Puri et al., 2025), but no study has systematically quantified their *relative* contribution to total score variance, or assessed whether their effect exceeds the variance attributable to architectural differences between SAEs.

Since their introduction, SAEs (Bricken et al., 2023; Cunningham et al., 2024) have been rapidly adopted in the field of interpretability. Early work demonstrates that identified features can be used to successfully steer model behavior (Templeton et al., 2024), while other applications include detecting and mitigating emergent misalignment (Wang et al., 2026). Simultaneously, SAE features



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.

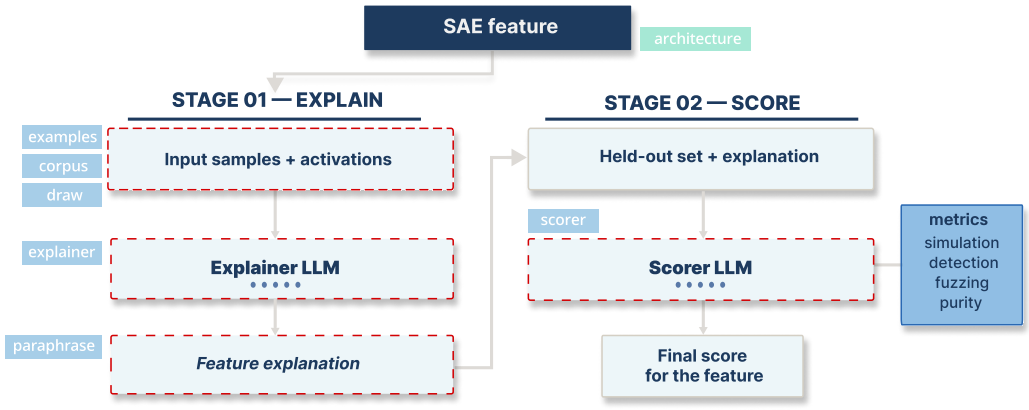


Figure 1: The autointerpretability pipeline, where an explainer LLM describes an SAE feature from activating examples (stage 1), followed by a scorer LLM that uses the explanation to perform a metric-specific task on a held-out test set (stage 2). The tags mark the factors we vary: six methodological factors (blue) change how the same feature is evaluated, while ARCHITECTURE (green) changes the feature itself. The dashed borders mark the steps in the pipeline that vary.

are sensitive to small input perturbations (Li et al., 2026); SAEs trained on identical data learn different features across random seeds (Paulo & Belrose, 2026), and downstream task performance does not consistently exceed simpler baselines (Kantamneni et al., 2025; Wu et al., 2025). Whether SAEs deliver on interpretability remains an important question for the debate on their usefulness, and that debate requires reliable metrics to resolve.

In this paper, we show that current dominant instruments are unreliable, and argue that evaluating SAEs with autointerpretability is a measurement problem, as the pipeline factors influence the scores as much as the architectures. Through systematic experiments on two models, `Pythia-160M` and `Apertus-8B-Instruct`, across four metrics (simulation, detection, fuzzing, purity; Section 2), we vary six evaluation pipeline factors independently and measure instability using ICC (Koo & Li, 2016) and Top-$k$ Jaccard (Albatineh & Niewiadomska-Bugaj, 2011). We study statistical *reliability* of a metric, rather than *validity*, in specific pipeline configurations. We do not evaluate whether autointerpretability scores truly capture interpretability but whether they are consistent enough to support the comparisons the field relies on, which is a precondition for a validity claim. Our findings show a *lower bound of variance* in the autointerpretability pipeline, as we experiment with a large subset of potential variance contributors, but not all of them. We make the following contributions:

- **(R1) A single methodological choice explains more variance than the SAE architecture.** Across both subject models and all four metrics, the largest methodological factor explains more score variance than the choice of SAE architecture. Excluding the activation corpus, this holds in five of eight model-metric pairs (Section 3).
- **(R2) Metric reliability is a property of the metric–factor pair, not of the metric.** We break down metric reliability by different factors, which motivates multi-metric evaluation, and extends earlier research (Puri et al., 2025) (Section 3).
- **(R3) Score agreement does not imply agreement about the top feature ranking.** Two conditions share on average two to four features, even where mean scores are similar or per-feature scores correlate highly (Section 3).

This is the first work to train and evaluate SAEs on APERTUS-8B-INSTRUCT. Based on our findings, we report recommendations for reliable evaluations (Section 4). Our findings do not resolve whether SAEs are useful, but establish what is required to answer that question reliably. Related work of SAE evaluation practices across the literature is provided in Section A.

## 2 PRELIMINARIES

To quantify how much selected scores depend on pipeline choices relative to the architectural differences they are meant to capture, we evaluated the stability of autointerpretability scores across

six factors of methodological variation: random DRAW of activating examples, EXPLAINER model, SCORER model, explanation PARAPHRASING, and EXPLANATION SAMPLES. In addition, evaluations are tested using different datasets in a CORPUS experiment. For the architectural axis, five SAEs were examined for both models. We selected known and unknown sources of variation and aim to establish a lower bound on variance for the autointerpretability pipeline. The full experimental details are given in Section B.

**Models and SAEs.** Experiments were conducted on SAEs trained on `Pythia-160M` (Biderman et al., 2023) and `Apertus-8B-Instruct` (Hernández-Cano et al., 2025). For both models, we used BATCHTOPK SAE with $k = 160$. For `Pythia-160M`, we used publicly available checkpoints at layer 8 (Karvonen et al., 2025). As `Apertus-8B-Instruct` a larger model, we train with activations from layer 16. For architecture comparison, we evaluated MATRYOSHKA (Bussmann et al., 2025), STANDARD ReLU (Cunningham et al., 2024), JUMPRELU Rajamanoharan et al. (2024b) and TOPK Gao et al. (2025) SAEs. The default model used to generate and score explanations is `gemini-2.5-flash` (Comanici et al., 2025), and is varied with `gpt-4o-mini` (OpenAI et al., 2024) and `gemma-3-27B` (Gemma Team et al., 2025) in the EXPLAINER and SCORER experiments.

**Corpora and Features.** We used the PILE (Gao et al., 2020) with all copyrighted content removed as the primary evaluation corpus to match the SAE training distribution. For the corpus experiment, three additional datasets are included: C4 (Raffel et al., 2020), WIKIPEDIA (Wikimedia Foundation), and GITHUB CODE (CodeParrot, 2022), spanning a range from similar to dissimilar relative to the training data. All datasets are sampled to 100,000 sequences of 32 tokens and accessed via HuggingFace; corpus statistics are reported in Table 6. Per experiment, we targeted approximately 500 active features. Feature sampling details are provided in Section B.4. Two experiments have different numbers. First, because of the overlap of features required for the CORPUS experiment, there are $n = 191$ features for the `Apertus-8B-Instruct` condition (see Figure 6). Second, on the same model, the ARCHITECTURE experiment on uses $n = 450$, due to the high number of dead latents for JUMPRELU (see Section D).

**Autointerpretability Metrics.** Four metrics are used, which are SIMULATION (Bills et al., 2023), DETECTION (Paulo et al., 2025), FUZZING (Paulo et al., 2025) and PURITY(Puri et al., 2025). All metrics provide a score for a feature differently. Their definitions and scoring prompts are in B.9.1.

### 2.1 QUANTIFYING INSTABILITY

We quantify within-feature instability for each methodological choice using two complementary measures, for score reliability and ranking stability. Both track how much the *same* feature's score changes across conditions rather than differences between features. Architectural variance is measured between features, due to different SAEs. $\eta^2$ is introduced to compare methodological and architectural variance contributions. An overview is shown Table 1. Section C provides more detail.

**Score Reliability.** The Intraclass Correlation Coefficient (ICC) (Koo & Li, 2016; Liljequist et al., 2019) measures the proportion of total variance attributable to stable between-feature differences (Eq. 1). A high score indicates the variance is attributable to the feature and not the condition. We generally interpret ICC $\geq 0.75$ as good, 0.50–0.75 as moderate, and $< 0.50$ as poor reliability, reported with 95% confidence intervals. ICC specifications are in Section C.1.

$$\text{ICC} = \frac{\sigma^2_{\text{between}}}{\sigma^2_{\text{between}} + \sigma^2_{\text{within}}} \tag{1}$$

**Ranking Stability.** Whether the same top-$k$ features are selected across conditions is measured by Top-$k$ Jaccard (Eq. 2), answering the question: Do conditions select the top-$k$ features similarly? We report *lift* over chance, so results are comparable across pool sizes that *vary slightly*. However, a larger pool size will result in a larger lift. Lift $= 1$ indicates chance-level selection, following observed-over-expected ratio. Details are in Section C.2.

$$J_k(A, B) = \frac{|T^A_k \cap T^B_k|}{|T^A_k \cup T^B_k|} \tag{2}$$

**Variance Explained.** To compare methodological against architectural variance, we use $\eta^2$, the share of total score variance explained by a factor, computed identically for every factor (Eq. 3). $\eta^2$ does not require the same features across levels, unlike ICC, as each factor is a separate analysis. A high value of $\eta^2$ indicates that the level influences the final score, but it is important to note that $\eta^2$ increases with the number of levels. More details can be found in Section C.3. Let $x_{ij}$ be the score of feature $i$ under level $j$ of a factor, $n_j$ the number of features in level $j$, $\bar{x}_{\cdot j}$ the mean of level $j$, and $\bar{x}$ the grand mean over all levels of a factor. $SS$ means the sum of squared deviations Then

$$\eta^2 = \frac{SS_{\text{between}}}{SS_{\text{total}}} = \frac{\sum_j n_j(\bar{x}_{\cdot j} - \bar{x})^2}{\sum_{ij}(x_{ij} - \bar{x})^2} \tag{3}$$

Table 1: Summary of metrics used to evaluate within-feature instability for the four different scores.

| Metric | Feature Level | Scope | Question Answered |
|---|---|---|---|
| ICC | Within-feature | Per-metric | Metric reliable across conditions? |
| Top-$k$ Jaccard | Within-feature | Per-condition-pair | Stable top-$k$ features? |
| $\eta^2$ | Between-feature | Global | What explains the variance? |

## 3 Dissecting Autointerpretability Instability via Three Findings

### (R1) A single methodological choice explains more variance than architecture

*Across both subject models and all four metrics, the largest methodological factor, such as the scorer, explainer or corpus, explains more score variance than the choice of SAE architecture. Without the corpus, this remains true for five of eight model-metric combinations for the two models.*

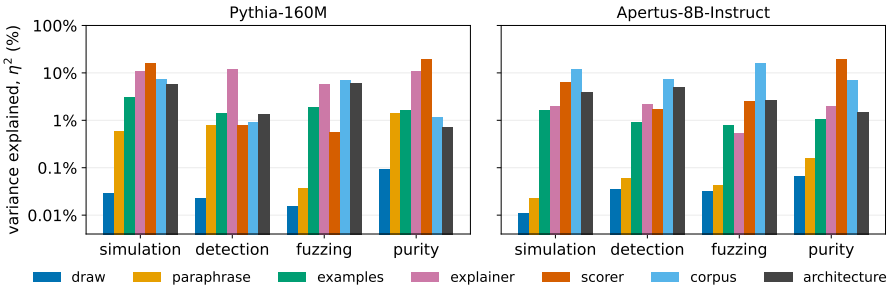


Figure 2: Share of score variance explained by each factor, $\eta^2$ with logarithmic axis, for both models. In every group, the tallest methodological bar exceeds the ARCHITECTURE bar (dark grey). Values are in Table 22 and Table 23.

Figure 2 reports $\eta^2$, the share of total score variance explained by each factor, computed identically for all factors and metrics (full values are in Table 22 and Table 23). In each group of bars, the tallest methodological bar exceeds the ARCHITECTURE bar. On `Pythia-160M` the largest factor is the SCORER on SIMULATION and PURITY (16.10% vs. 5.70%; 19.21% vs. 0.71%), the EXPLAINER on DETECTION (11.79% vs. 1.32%), and the CORPUS on FUZZING (6.92% vs. 6.02%). On `Apertus-8B-Instruct` it is the CORPUS on SIMULATION, DETECTION and FUZZING (11.64% vs. 3.85%; 7.21% vs. 5.02%; 16.03% vs. 2.67%) and the SCORER on PURITY (19.22% vs. 1.43%). The margin ranges from small (`Pythia-160M` , FUZZING) to more than an order of magnitude (PURITY on both models).

The comparison between methodological and architecture through $\eta^2$ is conservative. The methodological factors are varied on a *fixed* feature set, so their variance reflects only what the pipeline adds. The same feature receives a different explanation, or the same explanation receives a different score, for reasons unrelated to the feature. ARCHITECTURE is compared across SAEs with different features; therefore, part of its $\eta^2$ reflects feature differences. We estimate this part by splitting a single SAE's features into five random groups that are compared. Where the true architectural effect is zero,

$\eta^2$ averages 0.8-1.0% (Table 10). This confirms the comparison is conservative.

The factors span four orders of magnitude, which is why Figure 2 uses a logarithmic axis. Redrawing the example set has a small change (0.01-0.09% in every model-metric pair), and paraphrasing the explanation only slightly higher (0.02-1.41%) on both models. The SCORER, EXPLAINER, and CORPUS are the factors with higher scores, and which of them dominates differs by model. The EXPLAINER explains 11.79% of DETECTION variance on `Pythia-160M` but 2.15% on `Apertus-8B-Instruct`, while the CORPUS is the largest factor on three `Apertus-8B-Instruct` metrics and on only one for `Pythia-160M`. ARCHITECTURE explains 0.71-6.02% on both models.

The results depend on treating the CORPUS as a methodological choice. The evaluation corpus is a decision researchers make and rarely report , but part of its variance may be a property of features that behave differently on code than on web text. Without CORPUS, the claim remains true in five of eight model-metric to architecture comparisons. It fails where architecture itself is one of the larger effects. FUZZING on both models and DETECTION on `Apertus-8B-Instruct` fall to architecture or tie (5.69% vs. 6.02%; 2.58% vs. 2.67%; 2.15% vs. 5.02%). The `Apertus-8B-Instruct` CORPUS estimates also rest on the 191 features present in all four corpora (Figure 6).

To test whether the scorer alone can change which architecture has the highest score, we scored the explanations of all five architectures with `gpt-4o-mini` along `gemini-2.5-flash` (Section G). Of the 80 pairs (2 models $\times$ 10 architecture pairs $\times$ 4 metrics) compared for the two scorers, 13 change their ordering. Reversals occurred among close pairs (median gap of 0.011, against 0.037 for pairs that keep their order, for all metrics), and no pair that is separated by more than 0.036 reverses, under either judge. In three out of eight model-metric rankings, the top architecture changes, which even includes FUZZING (see Table 2). This shows that reversal is therefore not how much variance the scorer explains, but how close architectures are. Differences below 0.036 cannot be resolved with a single scorer.

Collectively, the choice that most affects a feature's autointerpretability score is not which SAE produced the feature but which model was used for scoring or explaining, or which corpus it was evaluated on.

Table 2: Model-metric pairs in which the top-ranked SAE architecture changes when the scorer is changed from `gemini-2.5-flash` to `gpt-4o-mini`.

| Model | Metric | `gemini-2.5-flash` | `gpt-4o-mini` |
|---|---|---|---|
| `Pythia-160M` | PURITY | JUMPRELU | TOPK |
| `Pythia-160M` | FUZZING | TOPK | MATRYOSHKA |
| `Apertus-8B-Instruct` | FUZZING | STANDARD | BATCHTOPK |

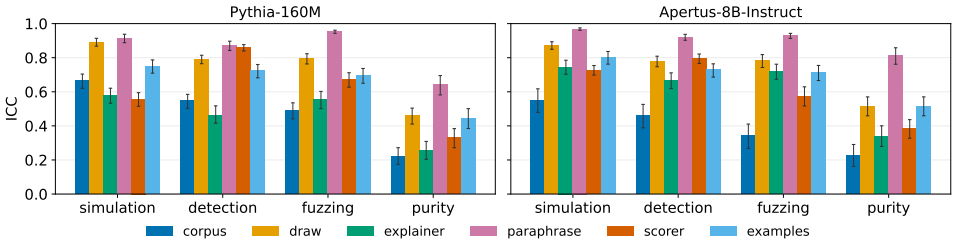


Figure 3: The ICC per metric for `Pythia-160M` (left) and `Apertus-8B-Instruct` (right), with 95% confidence intervals. PARAPHRASE is the most reliable factor for every metric; PURITY is the least reliable metric under every factor.

**(R2)** METRIC RELIABILITY IS A PROPERTY OF THE METRIC–FACTOR PAIR, NOT OF THE METRIC

*Not a single metric is reliable across all factors. Purity is the least reliable metric for every factor. The other three have comparable average reliability but are flawed as follows: detection is robust to the scoring model, yet sensitive under the explaining model, while fuzzing is near-perfect under paraphrasing and weakest under the corpus.*

Table 3: ICC with 95% CI by factor and model. Per metric and model, the highest-scoring factor is in **bold** and the lowest is underlined. Shaded values indicate poor reliability (ICC $< 0.5$). The last row shows the mean ICC score over the six factors.

| Factor | Model | SIMULATION | DETECTION | FUZZING | PURITY |
|---|---|---|---|---|---|
| CORPUS | Pythia | 0.665 (0.621, 0.704) | 0.547 (0.503, 0.585) | 0.488 (0.441, 0.535) | 0.221 (0.174, 0.272) |
| | Apertus | 0.548 (0.479, 0.617) | 0.463 (0.389, 0.526) | 0.343 (0.268, 0.411) | 0.227 (0.162, 0.291) |
| DRAW | Pythia | 0.892 (0.868, 0.914) | 0.791 (0.765, 0.814) | 0.793 (0.763, 0.823) | 0.460 (0.411, 0.504) |
| | Apertus | 0.873 (0.849, 0.894) | 0.780 (0.747, 0.809) | 0.784 (0.743, 0.818) | 0.513 (0.459, 0.570) |
| EXPLAINER | Pythia | 0.577 (0.533, 0.621) | 0.463 (0.416, 0.517) | 0.553 (0.501, 0.602) | 0.255 (0.204, 0.309) |
| | Apertus | 0.746 (0.703, 0.785) | 0.665 (0.619, 0.711) | 0.721 (0.673, 0.762) | 0.339 (0.280, 0.400) |
| PARAPHRASE | Pythia | **0.913** (0.888, 0.938) | **0.871** (0.842, 0.897) | **0.953** (0.943, 0.961) | **0.642** (0.581, 0.695) |
| | Apertus | **0.968** (0.961, 0.975) | **0.921** (0.902, 0.937) | **0.929** (0.913, 0.943) | **0.813** (0.762, 0.858) |
| SCORER | Pythia | 0.557 (0.515, 0.595) | 0.859 (0.839, 0.877) | 0.671 (0.627, 0.712) | 0.330 (0.272, 0.384) |
| | Apertus | 0.727 (0.698, 0.754) | 0.795 (0.766, 0.822) | 0.575 (0.517, 0.629) | 0.383 (0.328, 0.437) |
| EXAMPLES | Pythia | 0.749 (0.709, 0.787) | 0.723 (0.682, 0.760) | 0.696 (0.650, 0.737) | 0.445 (0.385, 0.501) |
| | Apertus | 0.800 (0.762, 0.837) | 0.729 (0.686, 0.764) | 0.716 (0.666, 0.754) | 0.513 (0.459, 0.570) |
| Mean ICC score | Pythia | 0.726 | 0.709 | 0.692 | 0.392 |
| | Apertus | 0.777 | 0.725 | 0.678 | 0.465 |

Table 3 reports the ICC for each factor-metric pair, which shows the share of the score variance that can be attributed to the feature, rather than to the factor.

*Factors matter more than metrics.* PARAPHRASE has the highest ICC of 0.642-0.968 for every metric on both models. Rewording an explanation while preserving its content leaves scores largely intact. The CORPUS is the harshest, with the lowest scores in six of eight metric-model pairs and lowest on PURITY (0.221 and 0.227 for both models). Beyond the corpus, each metric breaks under a different factor. SIMULATION under the SCORER (0.557, 0.727), and DETECTION and PURITY under the EXPLAINER (0.463, 0.665; 0.255, 0.339). Only FUZZING differs by model, weakest under the EXPLAINER on `Pythia-160M` (0.553), while under the SCORER on `Apertus-8B-Instruct` (0.575). A metric that is stable under one factor on one model does not have to be stable under another factor or on another model. Metrics alone cannot capture reliability.

PURITY *is the least reliable metric under every factor.* Its ICC ranges from 0.221 (CORPUS, both models) to 0.813 (PARAPHRASE, `Apertus-8B-Instruct`) and eight of its twelve factor-model pairs are below 0.5, which is the majority. Other metrics do not show this behaviour, and they are comparable in their mean ICC values (0.68-0.78, Table 3), but the mean of the scores of the metrics does not indicate where they differ (Table 14 and Table 15), which is again different per metric-factor. SIMULATION has the highest ICC for most factors. The exceptions are FUZZING under PARAPHRASE on `Pythia-160M` (0.953), and DETECTION under the SCORER on both models (0.859 on `Pythia-160M`, 0.795 on `Apertus-8B-Instruct`, against 0.557/0.727 for SIMULATION and 0.671/0.575 for FUZZING). DETECTION is not the most stable metric overall, but it is the one whose scores depend the least on the choice of scorer.

An additional remark is on FUZZING, where we used an altered indexed response format (Section F). Under the original format, following Paulo et al. (2025), `gemma-3-27B` skipped examples and its judgements became misaligned, scoring below chance (0.41 and 0.44). `gpt-4o-mini` scored 0.15-0.17 lower than with an indexed format that ties each judgement to its example (Table 28).

### (R3) SCORE AGREEMENT DOES NOT IMPLY AGREEMENT ABOUT THE TOP FEATURES

*Agreement on scores does not imply agreement on the top features. Two conditions share approximately two of 10 top features on `Pythia-160M` and four of 10 on `Apertus-8B-Instruct`, and two scoring models with rank correlation above 0.8 can share as few as one feature.*

Researchers often select features ranked by a metric. Figure 4 shows how reproducible the selection of the top-$k$ is, since the Jaccard overlap between the top-$k$ conditions is expressed as the lift over chance (full values in Table 19 and Table 20, raw values in Table 17 and Table 18). Feature selection is above chance for nearly all factors, but this does not imply direct reproducibility of all top $k$. Considering $k = 10$, PARAPHRASE, the most stable factor, reaches $34\times$ chance on `Pythia-160M`

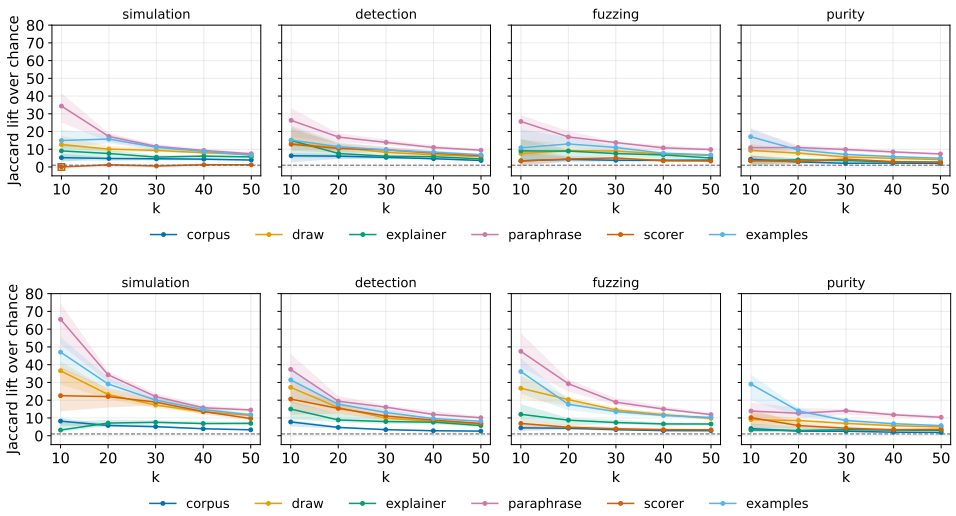


Figure 4: Top-$k$ Jaccard Lift over chance for each factor and metric, for $k \in \{10, \ldots, 50\}$, on `Pythia-160M` (top) and `Apertus-8B-Instruct` (bottom). The shaded bands are 95% confidence intervals and the dashed line marks chance (lift $= 1$). Lift is higher on `Apertus-8B-Instruct` on most factors. The SCORER on `Pythia-160M` SIMULATION stays at or near chance for every $k$. SIMULATION under the SCORER compares two judges rather than three.

SIMULATION and $65\times$ on `Apertus-8B-Instruct`, which corresponds to around 5/10 and 8/10 features in common. DRAW, EXAMPLES, and SCORER for DETECTION is between 13–16$\times$ on `Pythia-160M` (about two or three out of ten), and 20-31$\times$ on `Apertus-8B-Instruct` (about four to five out of 10). The EXPLAINER and CORPUS are lowest at 3-15$\times$. For the EXPLAINER, this is less than three. As CORPUS has a pool of $n = 191$ on `Apertus-8B-Instruct`, this raises the chance overlap and around four out of ten are shared. The formula for the calculation of the features can be found in Equation (7).

Two factors show that the overlap is not visible from the mean and ICC. The first is DRAW. Level means are nearly identical across draws on every metric (Table 14, Table 15), and ICC is high (0.78-0.89 for both models, besides PURITY). Yet, top-20 lift on SIMULATION is 10.08 on `Pythia-160M` and 23.21 on `Apertus-8B-Instruct`, and on PURITY 7.81 and 8.56. Re-drawing the examples does not change the mean scores or variances, but does change which features are ranked highest.

The second factor is the SCORER. For the SCORER, the means do shift, as on `Pythia-160M`, `gemini-2.5-flash` averages 0.528 on SIMULATION and 0.770 on PURITY, against 0.299 and 0.566 for `gpt-4o-mini`, but the scorers show high agreement on feature rankings. Table 21 compares each pair of judges directly. Across all those model-metric combinations, the highest overlap between two judges is five, while four pairs select disjoint sets. Rank agreement does not predict this, as six pairs with $\rho \geq 0.80$ have an overlap from one to five features. On `Pythia-160M` DETECTION two pairs at $\rho = 0.85$ share a single feature. Figure 8 shows the clearest case. On `Pythia-160M` SIMULATION ($\rho = 0.71$), each judge's ten highest-rated features rank between 35th and 313th in the other's ordering, whereas on `Apertus-8B-Instruct` ($\rho = 0.86$), where the top of the distribution is separated from the rest, the same pair shares four. Agreement on how features rank overall does not imply agreement on which features are best.

Explanation similarity is not a metric that can indicate score similarity. Intuitively, if a rewritten explanation stays semantically close to the original, its scores should too. Table 16 reports the mean similarity for the four factors where it is defined (DRAW, EXAMPLES, EXPLAINER, PARAPHRASE). It is high for all four (0.854-0.975), highest for PARAPHRASE and lowest for EXPLAINER, and most of the the mean scores under these factors are also high. Yet, DRAW and PARAPHRASE have higher ICCs than EXPLAINER on every metric (see Table 3), a gap similarity values do not predict.

The distinction between score agreement and rank agreement is important for downstream use, such as feature steering, circuit analysis, and manual inspection, which act on the features a metric selects, not on its mean. Because two scorers that agree closely on how features rank can still disagree on the top features, a feature's score is a property of the feature-scorer pair, rather than of the feature alone.

Without the scorer reported, a top-$k$ selection cannot be reconstructed. Grading the same SAE under different judges can therefore investigate largely different features while reporting comparable scores.

## 4 TOWARDS RELIABLE EVALUATION

The previous section shows that scores can depend on pipeline choices. This section asks the question: given a fixed budget, which interventions can make an evaluation stable, and how much stability does each one buy? We focus on the following two interventions: **standardising** and **averaging**.

**Standardising.** Fixing a factor removes its contribution to the score variance, measured by $\eta^2$ (Figure 2). The dominant factors are CORPUS, SCORER and EXPLAINER, and which one of the three dominates depends on the model and the metric (**R2**). Standardising does not add additional costs, so all three can be fixed at once. Standardising does not make a score more reliable, but it makes a result conditional on a stated choice. The value lies in disclosure, as it implicates that two studies can only compare scores if they report which scorer, explainer and corpus they used, and whether these match.

**Averaging.** Where a result should hold beyond one configuration, the alternative is to average over several levels of a factor. Averaging over $c$ conditions cancels noise and increases metric reliability. We take ICC of 0.75 as the threshold, where reliability is *good* (Koo & Li, 2016). If one condition has ICC of $\rho$, the average of $c$ has

$$\mathrm{ICC}(c) = \frac{c\rho}{1 + (c-1)\rho}, \tag{4}$$

which is the Spearman-Brown relation. It allows us to ask how many conditions are needed to reach a target reliability, where the answer is a projection rather than a measurement Section E.7. Eq. 4 assumes that the levels are interchangeable, which indicates that averaging any two scorers is as good as averaging the other two. We tested this assumption and the results confirm that for the four factors with at least four levels it is at least as reliable as the projection (Equation (4)). For these factors, the determined $c$ is an upper bound. The EXPLAINER and SCORER have only three levels and cannot be tested in this way, and their values in Table 4 are based on the assumption.

Table 4: The number of conditions $c$ needed to reach $0.75 \leq$ ICC (Koo & Li (2016)), by factor and metric, compared to the number of levels $L$ run per factor. A ✓ is added when $c \leq L$, while $\times$ indicates $c > L$. The values for EXPLAINER and SCORER are derived from the assumption.

| `Pythia-160M` | | | | | |
|---|---|---|---|---|---|
| Factor | Sim. | Det. | Fuzz. | Purity | $L$ |
| Draw | 1 ✓ | 1 ✓ | 1 ✓ | 4 ✓ | 5 |
| Paraphrase | 1 ✓ | 1 ✓ | 1 ✓ | 2 ✓ | 4 |
| Examples | 2 ✓ | 2 ✓ | 2 ✓ | 4 ✓ | 4 |
| Explainer | 3 ✓ | 4 × | 3 ✓ | 9 × | 3 |
| Scorer | 2 × | 1 ✓ | 2 ✓ | 7 × | 3 |
| Corpus | 2 ✓ | 3 ✓ | 4 ✓ | 11 × | 4 |

| `Apertus-8B-Instruct` | | | | | |
|---|---|---|---|---|---|
| Factor | Sim. | Det. | Fuzz. | Purity | $L$ |
| Draw | 1 ✓ | 1 ✓ | 1 ✓ | 3 ✓ | 5 |
| Paraphrase | 1 ✓ | 1 ✓ | 1 ✓ | 1 ✓ | 4 |
| Examples | 1 ✓ | 2 ✓ | 2 ✓ | 3 ✓ | 4 |
| Explainer | 2 ✓ | 2 ✓ | 2 ✓ | 6 × | 3 |
| Scorer | 2 ✓ | 1 ✓ | 3 ✓ | 5 × | 3 |
| Corpus | 3 ✓ | 4 ✓ | 6 × | 11 × | 4 |

From Table 4, three patterns emerge. We see that *sampling choices are cheap.* A single draw and a single paraphrase are sufficient for SIMULATION, DETECTION and FUZZING on both models. Additionally, *purity requires a larger $c$ than the others.* For the CORPUS, SCORER and EXPLAINER factors, it needs the highest levels to reach a sufficient ICC score. Lastly, *three other metric-model pairs are unreachable, each on one model.* DETECTION under EXPLAINER and SIMULATION under SCORER on `Pythia-160M`, and FUZZING under CORPUS on `Apertus-8B-Instruct`. This is consistent with the findings in (**R2**).

Two pairs in Table 4 make the computational advantage of standardising explicit. Reaching ICC $=$ $0.75$ on PURITY under a varying scorer needs seven conditions on `Pythia-160M` and five on `Apertus-8B-Instruct`. Fixing the scorer removes the same variance in one step and at no cost.

**Recommendations.** Combining both interventions, we recommend the following for autointerpretability evaluations. First, fix and report the scorer, the explainer and the corpus. These are the

largest sources of variance, and fixing them is free. Second, use a single draw and a single paraphrase. Averaging over them buys little for SIMULATION, DETECTION and FUZZING. Third, report selection stability alongside mean scores, since stable means and high correlations do not guarantee a stable top-$k$ (**R3**). Lastly, treat PURITY as provisional, as it is the least reliable metric under every factor.

## 5 CONCLUSION

Autointerpretability scores are the dominant means of assessing SAE feature interpretability, and cross-paper comparisons of SAE architectures rely on their reliability. We asked whether *current autointerpretability evaluation practices are reliable enough to support cross-paper comparisons of SAEs?* To answer this, we systematically varied six factors of the evaluation pipeline, across two models (`Pythia-160M` and `Apertus-8B-Instruct`) and four metrics, measuring instability via ICC, Top-$k$ Jaccard lift, and variance explained.

Our findings show that evaluating SAEs with autointerpretability is a measurement problem and are divided in three parts. **(R1)** A single methodological choice explains more score variance than the choice of SAE architecture in every model-metric pair, and in five of eight when the corpus is excluded. **(R2)** Metric reliability is a property of the metric–factor pair, not of the metric, as no single metric captures the full instability picture. **(R3)** Score agreement does not imply agreement about the top features. The top features identified under one metric for one factor do not have to completely overlap with those identified for another factor, even though mean scores imply stability. Additional results show that explanation similarity cannot diagnose score instability, as similarity scores remain high even when autointerpretability scores vary, ruling out this cheap diagnostic.

There are several limitations to the scope of our conclusions. Our experiments use a fixed layer for both models (layer 8 for `Pythia-160M`, layer 16 for `Apertus-8B-Instruct`); layer-dependence is an additional axis of instability that we did not study and of which it is known that they can influence scores (Puri et al., 2025). Additionally, including more metrics would provide an even more robust evaluation (Paulo et al., 2025; Puri et al., 2025), or increasing the number of variations per experiment could provide a richer picture of the influence of these choices. In the specific pipeline configurations tested, the largest methodological factor exceeds the architectural variance. We do not claim that this generalises to all scorers, layers, or SAE architectures, but it establishes that metric stability cannot be assumed by default. Lastly, the architecture component is measured differently from the methodological ones, due to the nature of each SAE having their own dictionary, and is a conservative estimate. These findings directly affect ongoing research on SAE utility (Kantamneni et al., 2025). Evidence from unstable measurements cannot determine whether SAEs work or which architecture performs best. To support more reliable evaluation, we propose *standardising* and *averaging* to ensure stable evaluation. Based on the research budget, one of these could be chosen.

Future work could examine alternative evaluations that do not rely solely on LLM-judged explanations. Causal and interventionist evaluations (Makelov et al., 2025; Marks et al., 2025) validate features through their downstream effects on model behaviour. Similar lines of work use SAE features for steering interventions (Chalnev et al., 2024), whose effects can be measured directly on model outputs rather than mediated by an explanation step. The discussion about the usefulness of SAEs (Li et al., 2026; Kantamneni et al., 2025; Wu et al., 2025) underscores the need for evaluations with established reliability. We view causal, interventionist, and explanation-based evaluations as complementary: explanation-based scoring scales to millions of features in ways that causal methods currently do not, while interventionist evaluations provide the behavioral grounding that explanation quality alone cannot establish. Establishing the reliability of each approach and identifying the conditions under which they agree are important directions for future work. The goal is not to abandon autointerpretability, as it currently remains the most scalable approach to feature-level interpretation, but to use it with caution and to build the shared reporting standards that allow the field to accumulate reliable evidence.

## REPRODUCIBILITY STATEMENT

We share the code that allow users to explore our results. The Appendix contains details for each experiment separately, with additional parameter choices for the trained SAEs. All source models, the `Pythia-160M` sparse autoencoders, and datasets are open-source. We will release the SAEs on `Apertus-8B-Instruct` after publication, due to anonymity. Code is currently available through `https://anonymous.4open.science/r/autointerp-paper-E831`

## AI USE STATEMENT

We used generative AI tools for assistance with coding and writing. In coding, we used AI to support the implementation of the experiments and to provide feedback on the experimental setup. In addition, AI was used to support the scripts for the generation of figures of the results and the analysis and interpretation of the results. For the writing, AI was used to format the paper, restructure sentences, correct grammar, and create LaTeX tables.

## ETHICS STATEMENT

Both subject models are openly released. The SAEs from `Pythia-160M` are publicly available; and we release the SAEs trained on HuggingFace. We include proprietary models because they are representative of judges used in the current autointerpretability pipeline. Their weights are not accessible, but we report model identifiers and access dates and include an open-weight model so part of the study remains reproducible. All corpora are also public.

The corpora accessed are web-scale and may contain offensive and biased content. We use these texts only as model input for calculating SAE activations. We do not extract, annotate, or release personal information, and the examples reported are LLM-generated rather than raw data. All corpora are English or source code, even though `Apertus-8B-Instruct` is multilingual. We made this choice so we can compare the two models on the same material. This work evaluates the reliability of interpretability scores. It does not introduce new model capacities, steering methods, or attacks.

Interpretability tools can in principle help locate features that could be used to manipulate the model's behavior, but we do not release any analysis on this. A potential risk is that our results could show which methodological choices raise scores, which could be used to make explanations look better than they are. We counter this by recommending that scores are always reported together with conditions.

This study uses roughly 350 node hours on GH200 nodes. Much of this time is waiting on API responses, rather than on dense computation. To limit this, we computed SAE activations once per model, reused them across all experiments, and reused one baseline run as a reference condition. Small smoke tests were run before each large run, catching several errors that saved node hours. We deem these costs justified because they help researchers determine how many runs are needed for reliable interpretability scores.

## REFERENCES

Ahmed N. Albatineh and Magdalena Niewiadomska-Bugaj. Correcting jaccard and other similarity indices for chance agreement in cluster analysis. *Advances in Data Analysis and Classification*, 5 (3):179–200, October 2011. ISSN 1862-5347. doi: 10.1007/s11634-011-0090-y. URL `https://doi.org/10.1007/s11634-011-0090-y`.

Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. Pythia: a suite for analyzing large language models across training and scaling. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 2397–2430, 2023.

Steven Bills, Nick Cammarata, Dan Mossing, Henk Tillman, Leo Gao, Gabriel Goh, Ilya Sutskever, Jan Leike, Jeff Wu, and William Saunders. Language models can explain neurons in language models, 2023. URL `https://openaipublic.blob.core.windows.net/neuron-explainer/paper/index.html`.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E Burke, Tristan Hume, Shan Carter, Tom Henighan, and Christopher Olah. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023. https://transformer-circuits.pub/2023/monosemantic-features/index.html.

Bart Bussmann, Patrick Leask, and Neel Nanda. BatchTopK sparse autoencoders. In *Proceedings of the NeurIPS 2024 Workshop on Scientific Methods for Understanding Deep Learning*, 2024. URL https://openreview.net/forum?id=d4dpOCqybL.

Bart Bussmann, Noa Nabeshima, Adam Karvonen, and Neel Nanda. Learning multi-level features with matryoshka sparse autoencoders. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025.

Sviatoslav Chalnev, Matthew Siu, and Arthur Conmy. Improving steering vectors by targeting sparse autoencoder features. *arXiv preprint arXiv:2411.02193*, 2024.

David Chanin and Adrià Garriga-Alonso. Sparse but wrong: Incorrect l0 leads to incorrect features in sparse autoencoders. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=aYqF2Cv6Cr.

David Chanin, James Wilken-Smith, Tomáš Dulka, Hardik Bhatnagar, and Joseph Isaac Bloom. A is for absorption: Studying feature splitting and absorption in sparse autoencoders. In *Interpretable AI: Past, Present and Future, NeurIPS workshop*, 2024. URL https://openreview.net/forum?id=Wzav8fesTL.

CodeParrot. GitHub Code dataset. Hugging Face Datasets, 2022. URL https://huggingface.co/datasets/codeparrot/github-code. Accessed: March 2026.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, Luke Marris, Sam Petulla, Colin Gaffney, Asaf Aharoni, Nathan Lintz, Tiago Cardal Pais, Henrik Jacobsson, Idan Szpektor, Nan-Jiang Jiang, Krishna Haridasan, Ahmed Omran, Nikunj Saunshi, Dara Bahri, Gaurav Mishra, Eric Chu, Toby Boyd, Brad Hekman, Aaron Parisi, Chaoyi Zhang, Kornraphop Kawintiranon, Tania Bedrax-Weiss, Oliver Wang, Ya Xu, Ollie Purkiss, Uri Mendlovic, Ilaï Deutel, Nam Nguyen..., and Wesley Helmholz. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities, 2025. URL https://arxiv.org/abs/2507.06261.

Hoagy Cunningham, Robert Huben, Logan Riggs Smith, Aidan Ewart, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=F76bwRSLeK.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The pile: An 800gb dataset of diverse text for language modeling, 2020. URL https://arxiv.org/abs/2101.00027.

Leo Gao, Tom Dupre la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. Scaling and evaluating sparse autoencoders. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=tcsZt9ZNKD.

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 Léonard Hussenot. Gemma 3 technical report, 2025. URL https://arxiv.org/abs/2503.19786.

Alejandro Hernández-Cano, Alexander Hägele, Allen Hao Huang, Angelika Romanou, Antoni-Joan Solergibert, Barna Pasztor, Bettina Messmer, Dhia Garbaya, Eduard Frank Ďurech, Ido Hakimi, Juan García Giraldo, Mete Ismayilzada, Negar Foroutan, Skander Moalla, Tiancheng Chen, Vinko Sabolčec, Yixuan Xu, Michael Aerni, Badr AlKhamissi, Ines ... Altemir Marinas, and Imanol Schlag. Apertus: Democratizing open and compliant LLMs for global language environments, 2025. URL https://arxiv.org/abs/2509.14233.

Caden Juang, Gonçalo Paulo, Jonathan Drori, and Nora Belrose. Open source automated interpretability for sparse autoencoder features. EleutherAI Blog, July 2024.

Subhash Kantamneni, Joshua Engels, Senthooran Rajamanoharan, Max Tegmark, and Neel Nanda. Are sparse autoencoders useful? a case study in sparse probing. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=rNfzT8YkgO.

Adam Karvonen, Can Rager, Johnny Lin, Curt Tigges, Joseph Isaac Bloom, David Chanin, Yeu-Tong Lau, Eoin Farrell, Callum Stuart McDougall, Kola Ayonrinde, Demian Till, Matthew Wearden, Arthur Conmy, Samuel Marks, and Neel Nanda. SAEBench: A comprehensive benchmark for sparse autoencoders in language model interpretability. In *Proceedings of the Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=qrU3yNfX0d.

Terry K Koo and Mae Y Li. A guideline of selecting and reporting intraclass correlation coefficients for reliability research. *Journal of chiropractic medicine*, 15(2):155–163, 2016.

Anton Korznikov, Andrey Galichin, Alexey Dontsov, Oleg Rogov, Ivan Oseledets, and Elena Tutubalina. Sanity checks for sparse autoencoders: Do saes beat random baselines? *arXiv preprint arXiv:2602.14111*, 2026.

Ludmila I. Kuncheva. A stability index for feature selection. In *Proceedings of the 25th Conference on Artificial Intelligence and Applications*, pp. 390–395, 2007.

Aaron J. Li, Suraj Srinivas, Usha Bhalla, and Himabindu Lakkaraju. Evaluating adversarial robustness of concept representations in sparse autoencoders. In Vera Demberg, Kentaro Inui, and Lluís Marquez (eds.), *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5940–5957, Rabat, Morocco, March 2026. Association for Computational Linguistics. ISBN 979-8-89176-380-7. doi: 10.18653/v1/2026.eacl-long.279. URL https://aclanthology.org/2026.eacl-long.279/.

David Liljequist, Britt Elfving, and Kirsti Skavberg Roaldsen. Intraclass correlation–a discussion and demonstration of basic features. *PloS one*, 14(7):e0219854, 2019.

Aleksandar Makelov, Georg Lange, and Neel Nanda. Towards principled evaluations of sparse autoencoders for interpretability and control. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=1Njl73JKjB.

Samuel Marks, Can Rager, Eric J Michaud, Yonatan Belinkov, David Bau, and Aaron Mueller. Sparse feature circuits: Discovering and editing interpretable causal graphs in language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=I4e82CIDxv.

Gouki Minegishi, Hiroki Furuta, Yusuke Iwasawa, and Yutaka Matsuo. Rethinking evaluation of sparse autoencoders through the representation of polysemous words. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=HpUs2EXjOl.

OpenAI, Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, Aleksander Mądry, Alex Baker-Whitcomb, Alex Beutel, Alex Borzunov, Alex Carney, Alex Chow, Alex Kirillov, Alex Nichol, Alex Paino, Alex Renzin ..., and Yury Malkov. Gpt-4o system card, 2024. URL https://arxiv.org/abs/2410.21276.

Gonçalo Paulo and Nora Belrose. Sparse autoencoders trained on the same data learn different features. In *Proceedings of the International Conference on Learning Representations*, 2026. URL https://iclr.cc/virtual/2026/poster/10010650.

Gonçalo Santos Paulo, Alex Troy Mallen, Caden Juang, and Nora Belrose. Automatically interpreting millions of features in large language models. In *Proceedings of the Forty-second International Conference on Machine Learning*, 2025. URL https://icml.cc/virtual/2025/poster/45936.

Bruno Puri, Aakriti Jain, Elena Golimblevskaia, Patrick Kahardipraja, Thomas Wiegand, Wojciech Samek, and Sebastian Lapuschkin. FADE: Why bad descriptions happen to good features. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 17138–17160, Vienna, Austria, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-acl.881. URL https://aclanthology.org/2025.findings-acl.881/.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1), January 2020. ISSN 1532-4435.

Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Tom Lieberum, Vikrant Varma, János Kramár, Rohin Shah, and Neel Nanda. Improving sparse decomposition of language model activations with gated sparse autoencoders. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, 2024a. ISBN 9798331314385.

Senthooran Rajamanoharan, Tom Lieberum, Nicolas Sonnerat, Arthur Conmy, Vikrant Varma, János Kramár, and Neel Nanda. Jumping ahead: Improving reconstruction fidelity with jumprelu sparse autoencoders. *arXiv preprint arXiv:2407.14435*, 2024b.

Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, Hoagy Cunningham, Nicholas L Turner, Callum McDougall, Monte MacDiarmid, C. Daniel Freeman, Theodore R. Sumers, Edward Rees, Joshua Batson, Adam Jermyn, Shan Carter, Chris Olah, and Tom Henighan. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. *Transformer Circuits Thread*, 2024. URL https://transformer-circuits.pub/2024/scaling-monosemanticity/index.html.

Miles Wang, Tom Dupre la Tour, Olivia Watkins, Aleksandar Makelov, Ryan Andrew Chi, Samuel Miserendino, Jeffrey George Wang, Achyuta Rajaram, Johannes Heidecke, Tejal Patwardhan, and Daniel P Mossing. Persona features control emergent misalignment. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=yjrVOxjkDR.

Wikimedia Foundation. Wikimedia downloads. URL https://dumps.wikimedia.org. Accessed: March 2026.

Zhengxuan Wu, Aryaman Arora, Atticus Geiger, Zheng Wang, Jing Huang, Dan Jurafsky, Christopher D Manning, and Christopher Potts. Axbench: Steering LLMs? even simple baselines outperform sparse autoencoders. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=K2CckZjNy0.

# APPENDIX

**A Related Work** **16**

**B Experimental Setup** **17**

B.1 Experiment Overview . . . . . 17

B.2 Corpora . . . . . 18

B.3 Model Snapshots . . . . . 18

B.4 Feature Sampling . . . . . 19

B.5 Explainer Prompt . . . . . 21

B.6 Explainers (Exp. 2b) . . . . . 22

B.7 Paraphrase Generation (Exp. 2c) . . . . . 22

B.8 Explanation Similarity . . . . . 23

B.9 Scoring Procedure . . . . . 23

**C Statistical Methods** **26**

C.1 ICC specification . . . . . 26

C.2 Top-*k* Jaccard Lift . . . . . 26

C.3 Variance Explained . . . . . 27

**D Sparse Autoencoder Details** **28**

D.1 Pythia-160M SAE models . . . . . 28

D.2 Apertus-8B-Instruct SAE Training Details . . . . . 28

**E Results** **30**

E.1 Mean scores and standard deviation per condition . . . . . 30

E.2 Similarity Metric . . . . . 32

E.3 ICC Figures . . . . . 32

E.4 Jaccard Tables . . . . . 33

E.5 Pairwise scorer agreement . . . . . 37

E.6 Variance Explained values . . . . . 38

E.7 Stability Analysis . . . . . 38

**F Scorer Fuzzing Experiment** **41**

**G Scorer Architecture** **42**

**H Additional Analysis** **45**

H.1 Metric Correlations . . . . . 45

H.2 Corpus Results . . . . . 45

H.3 Draw Trajectories . . . . . 47
H.4 Paraphrase Trajectories . . . . . 47
H.5 Explainer Sensitivity . . . . . 47
H.6 Explainer Samples . . . . . 48

**I Time and Compute of Experiments** **49**

**J Qualitative Examples** **50**
J.1 Draw . . . . . 50
J.2 Explainer . . . . . 52
J.3 Paraphrase . . . . . 54
J.4 Examples . . . . . 55

# A Related Work

**Evaluation Gap** The field has recognized that improvements in the sparsity-reconstruction trade-off often do not directly translate to improved monosemanticity (Minegishi et al., 2025; Karvonen et al., 2025). The intuition behind reconstruction error as a proxy is that a low error indicates the SAE faithfully captures the original model's activations (Bricken et al., 2023; Rajamanoharan et al., 2024a). Similarly, sparsity is assumed to promote interpretability by forcing the model to represent each input using few active latent units (Cunningham et al., 2024). However, SAEs can be optimised along this trade-off and still produce polysemantic, unstable, or irrelevant features (Karvonen et al., 2025; Minegishi et al., 2025). More broadly, most metrics used in practice are *proxy* metrics: reconstruction error, sparsity, and loss recovery reflect optimization behavior rather than semantic quality, and optimizing for sparsity can introduce failure modes such as feature absorption (Chanin et al., 2024) and dead latents (Gao et al., 2025).

Makelov et al. (2025) took a first step toward a more principled approach by proposing a framework of faithfulness checks and feature-editing experiments relative to a supervised baseline. The authors explicitly acknowledge that their framework depends on a particular parametrization of task-relevant variables, which makes comparisons across settings difficult. This limitation reflects the broader challenge that interpretability evaluation depends on the task, variables, and conceptual lens. SAEBench (Karvonen et al., 2025) extends this toward standardization, comparing SAE variants across reconstruction fidelity, feature disentanglement, interpretability, and concept detection. They show that different model choices, such as dictionary size, influence evaluation metrics. A key finding is that these metrics cannot be unified into a single score, making evaluation fundamentally multi-objective: metric choice depends on the researcher's priority.

**Autointerpretability** Interpretability evaluation in practice relies predominantly on *automated interpretability* pipelines: an LM generates natural language explanations for features, and a scorer evaluates those explanations by testing whether they predict feature activations in new contexts. The approach was originally developed for neuron-level interpretation (Bills et al., 2023) and subsequently adapted to SAE features (Bricken et al., 2023; Cunningham et al., 2024). Recent work has extended the pipeline in several directions. Paulo et al. (2025) scale explanation generation to millions of features, while Puri et al. (2025) examines failure modes of automated description. Additionally, Puri et al. (2025) shows that explanations may not faithfully capture feature behavior and that scoring procedures can reward misleading descriptions.

**Known Sources of Variation** Several studies have identified factors that influence autointerpretability scores, motivating the systematic analysis we present. We organize these into three groups.

- *SAE and model properties.* Interpreting larger SAEs is more difficult, with larger dictionaries yielding lower scores (Puri et al., 2025). Scores vary across layers, with later layers tending to be more polysemantic and harder to explain (Puri et al., 2025). SAEs with more features tend to score higher than individual neurons, suggesting that sparsity aids interpretability (Paulo et al., 2025). Score variation on detection across SAE architectures is documented in Karvonen et al. (2025).
- *Example selection.* The choice of which activating examples are shown to the explainer affects explanation quality. Using only top-activating examples yields concise, specific descriptions but fails to capture the full activation distribution (Paulo et al., 2025). Random sampling yields interpretations that span a more diverse set of activating contexts, whereas stratified sampling can produce overly broad descriptions (Paulo et al., 2025). Including only activating contexts increases specificity but reduces sensitivity; including non-activating contexts has the opposite effect (Paulo et al., 2025). Providing a larger number of examples yields slightly higher scores overall (Paulo et al., 2025; Puri et al., 2025), though this improvement may reflect increased generality rather than increased accuracy (Paulo et al., 2025).
- *Explainer and scorer properties.* Larger explainer models produce better feature descriptions and higher scores (Paulo et al., 2025; Puri et al., 2025). For simpler metrics such as detection and fuzzing, a smaller scorer model yields lower scores, suggesting that scorer capacity interacts with metric complexity (Paulo et al., 2025). Prompt construction affects results

(Puri et al., 2025), as does the use of few-shot prompting (Puri et al., 2025). Notably, Paulo et al. (2025) finds that training dataset variation has little effect on scores, suggesting that corpus choice at training time is less influential than corpus choice at evaluation time, which is a distinction our experiments make explicit.

What remains absent from prior work is a systematic quantification of the *relative* contribution of these sources to total score variance, and an assessment of whether their combined effect exceeds the variance attributable to architectural differences. This paper addresses both questions, with additional focus on draws, explainers, phrasing, scorers, explainer samples and architecture. Therefore, we aimed to investigate the lower bound of the variance, as our focus remains a subset of potential choices in the autointerpretability pipeline.

# B Experimental Setup

The following sections provide additional information to support experiment reproducibility.

## B.1 Experiment Overview

An overview of the number of variations for each experiment can be found in Table 5.

Table 5: An overview of all the experiments and the variations they induce that were conducted in the current paper.

| Experiment | Factor | Change | Variations |
|---|---|---|---|
| 1 | Corpus | Corpus of the samples | 4 |
| 2a | Draw | 40 explain examples per draws | 5 |
| 2b | Explainer | Which model generates the explanation | 3 |
| 2c | Paraphrase | The explanation is reworded | 4 |
| 2d | Scorer | Which model grades the explanation | 3 |
| 2e | Examples | Explainer examples (10/20/40/80) | 4 |
| 3 | Architecture | the SAE | 5 |

**Exp. 1 — Corpus dependence.** We generated independent explanations and scores for the same set of features on four different corpora with varying distances from the training data: THE PILE (in-distribution), C4 (similar), WIKIPEDIA (moderate), and GITHUB CODE (dissimilar). All datasets are listed in Table 6. This tests whether autointerpretability scores reflect stable feature properties or are confounded by the choice of evaluation corpus. Similar corpora are expected to have similar scores for the same features. This experiment differs from other experiments, as Exp. 2a-e hold the input fixed and vary how it is described, while this experiment changes the input itself. The variance in this experiment may be a signal, not methodological noise. A feature is expected to behave differently on dissimilar datasets.

**Exp. 2a — Sampling instability.** For a fixed corpus, we generated multiple explanations per feature with the same explainer by drawing different random subsets of activating examples. This isolates the variance introduced by which examples happen to be sampled, holding all other factors constant. We use five independent draws per feature to balance statistical and computational considerations.

**Exp. 2b — Explainer model dependence.** We explain the same features using three different explainer LLMs: `gemini-2.5-flash`, `gpt-4o-mini`, and `gemma-3-27B`. All explanations were scored with `gemini-2.5-flash`. All explainer models are relatively small, efficient models positioned below the frontier tier of their respective families, and are broadly comparable in general capability. As other papers have examined the effect of explainer size ((Paulo et al., 2025; Puri et al., 2025)), this is not our main focus. As the chosen models are comparable in size, we investigated the effects of their differences, which should be minimal in theory. We restricted our comparison to three models: two closed-source models and one open-source model.

**Exp. 2c — Paraphrase sensitivity.** To test whether scoring measures semantics or primarily surface wording, we generated three paraphrases of each explanation and scored all four versions on the same held-out sequences. Four versions include the original explanation. If scores vary substantially between phrases, this indicates that the scorer is responding to the language rather than the meaning. We generate three paraphrases per original explanation, again balancing statistical and computational considerations. Critically, paraphrase represents the most semantically conservative perturbation in our experimental design, as it is based solely on the explanation itself, not on the input to or generation of the explanation.

**Exp. 2d - Scorer model dependence.** As a follow-up to Exp. 2b, we investigate the dependence on the scorer model. Although previous work has shown that the scorer model influences the pipeline (Paulo et al., 2025; Puri et al., 2025), we include it in the variance explained and compare it with the effects of other methodological choices. Similar to Exp. 2b, we opt for `gemini-2.5-flash`, `gpt-4o-mini` , and `gemma-3-27B`. Due to compute limitations related to self-hosting necessary for a high number of inference requests, SIMULATION was not calculated for `gemma-3-27B`.

**Exp. 2e - Explainer samples.** With this experiment, we investigate the influence of selecting 10, 20, 40, and 80 samples for the initial explanation. Previous work has shown that 40 samples is a solid number Paulo et al. (2025), but we wanted to add this condition to the overall variance estimation.

**Exp. 3 — Cross-architecture comparison.** To contextualize the methodological instability, we compared score distributions across five SAE architectures trained on the same model and data: RELU(Bricken et al., 2023), MATRYOSHKA(Bussmann et al., 2025), BATCHTOPK(Bussmann et al., 2024), JUMPRELU(Rajamanoharan et al., 2024b), and TOPK(Gao et al., 2025). All other methodological choices were fixed. This provides a reference point: if methodological variance (Exp. 1–2) exceeds architectural variance, this could limit the validity of cross-SAE comparisons in the literature. We investigate SAEs trained on smaller and larger models (`Pythia-160M` vs. `Apertus-8B-Instruct`). Earlier work shows that architectures can have different autointerpretability scores (Karvonen et al., 2025), but we extend this to multiple metrics.

### B.2 CORPORA

All corpora are sampled to 100,000 sequences, tokenized with the model-specific tokenizer, and truncated to 32 tokens following the findings of (Paulo et al., 2025). All datasets were accessed via HuggingFace. Table 6 shows the corpus details. The four corpora were chosen to span different degrees of similarity to the SAE training distribution, from similar to dissimilar. For example, C4 is more similar to THE PILE, compared to GITHUB CODE. Both main BATCHTOPK SAEs that are used for all the other experiments are trained on the uncopyrighted version of THE PILE (see Table 6).

Table 6: Corpus statistics. All corpora sampled to 100,000 sequences of 32 tokens.

| Corpus | Domain | HuggingFace identifier | License | Experiments |
|---|---|---|---|---|
| THE PILE | General web text | `monology/pile-uncopyrighted` | Other | All |
| C4 | Cleaned web text | `allenai/c4` | ODC-BY | Exp. 1 |
| WIKIPEDIA | Encyclopedic text | `wikimedia/wikipedia` | CC-BY-SA-3.0 | Exp. 1 |
| GITHUB CODE | Source code | `codeparrot/github-code` | Other | Exp. 1 |

### B.3 MODEL SNAPSHOTS

The LLM used as a default for explainer and scorer is `gemini-2.5-flash` , which is accessed through a LiteLLM proxy.

The `gemma-3-27b-it-q8_0` is self-hosted. As the model required hosting due to the number of inference requests, it was hosted on a virtual machine with 1 GPU. This limited response speed, which in turn strained node hours. This is the driver behind excluding the model from SIMULATION for the SCORER experiment.

Table 7: Model serving configuration and run window by experiment

| Experiment | Role | Alias | Serving route | First run | Last run |
|---|---|---|---|---|---|
| Exp. 2b | Explainer | `gemini-2.5-flash` | LiteLLM proxy | 2026-08-02 | 2026-08-25 |
| | Explainer | `gpt-4o-mini` | LiteLLM proxy | | |
| | Explainer | `gemma-3-27B` | Self-hosted | | |
| Exp. 2d | Scorer | `gemini-2.5-flash` | LiteLLM proxy | 2026-08-03 | 2026-09-07 |
| | Scorer | `gpt-4o-mini` | LiteLLM proxy | | |
| | Scorer | `gemma-3-27B` | Self-hosted | | |
| Exp. 3 | Explainer | `gemini-2.5-flash` | LiteLLM proxy | 2026-08-13 | 2026-09-08 |
| Exp. 3 | Scorer | `gemini-2.5-flash` | LiteLLM proxy | 2026-08-13 | 2026-09-08 |
| Exp. 3 | Scorer | `gpt-4o-mini` | LiteLLM proxy | 2026-08-13 | 2026-09-08 |

### B.4 Feature Sampling

To match the features between each condition, we aim to select 500 features for both models that are reused in every experiment.

#### B.4.1 Feature Threshold

A feature is considered *active* if its maximum activation over the evaluation corpus exceeds a model-specific threshold $T$. $T$ decides which sequences form the activating set, and therefore which features are usable at all.

In our case, a feature qualifies when it has at least 190 activating sequences and does not saturate the cache. Both bounds matter because:

- Below 190, there are not enough samples to draw 40 explain and 50 score positives from disjoint sets with deciles that mean anything.
- For Exp. 2a, this allows for diversity in activating samples that are drawn
- At the cap, the activating set is truncated by the cache rather than the feature, so the decile describes the cache.

Raising $T$ shrinks both conditions. The two models use different $T$ values, which depend on the magnitude of their activations. The choice for $T$ can be found in Table 8. Given all features that are active under the given conditions, 500 are randomly sampled and used in the final experiments. Selection happens once per model, corpus and SAE, and is stored in a JSON file that records feature ids and fingerprints of the corpora they were derived from.

Table 8: The thresholds $T$ for both BatchTopK SAEs trained on the Pile. The second column shows the number of features with at least 190 samples above the threshold. Dictionary is the size of the dictionary of the SAE. The last column displays the total number of alive features, i.e., all features that activate on at least one of the 100,000 sentences.

| Source Model | Chosen $T$ | Qualifying at peak | Dictionary | Live in the Pile |
|---|---|---|---|---|
| `Pythia-160M` | 0.5 | 6162 | 16384 | 15379 (94%) |
| `Apertus-8B-Instruct` | 63.0 | 3749 | 65536 | 20541 (31%) |

**Architecture Thresholds.** Each SAE model has slightly different activation thresholds, as shown in Figure 5. In general, the `Pythia-160M` SAEs' activation thresholds appear to be more aligned.

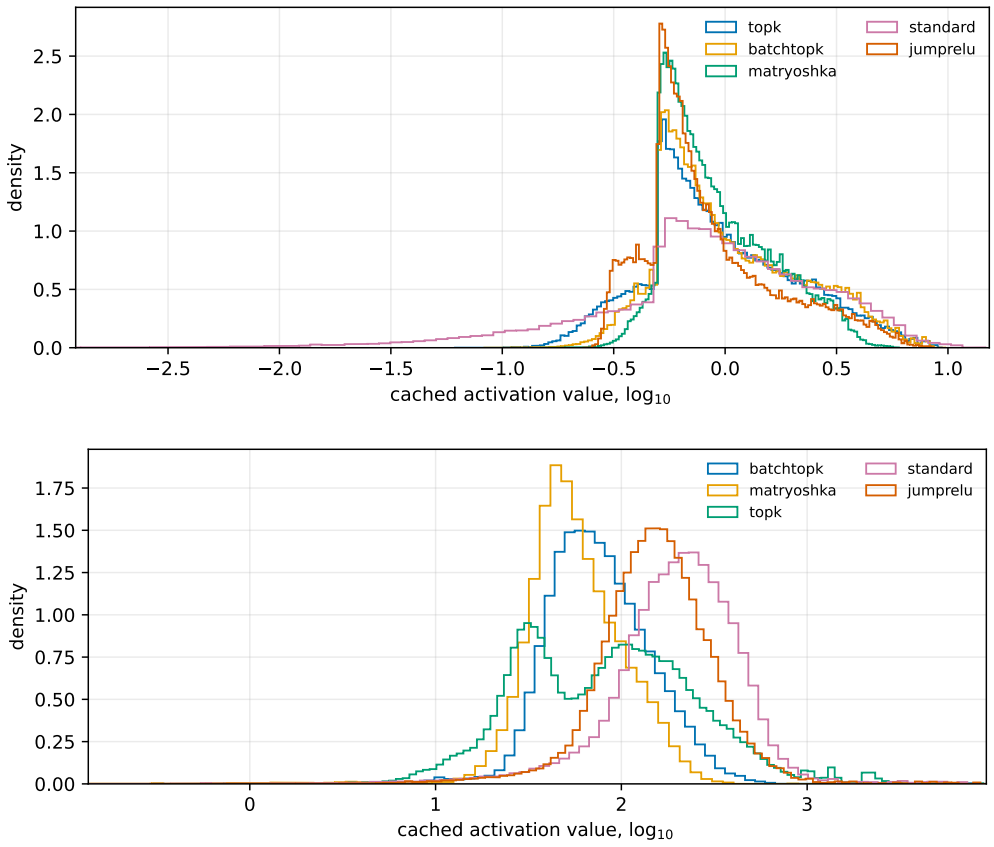


Figure 5: Distribution of cached activation values ($\log_{10}$) per SAE architecture for `Pythia-160M` (top) and `Apertus-8B-Instruct` (bottom).

### B.4.2 Sentence Sampling

From a feature's activating set, which includes all sentences above $T$, cached at top-1000, 40 sentences are used for the initial explanation. These sentences are drawn in a uniform way.

For the sampling of the scoring phase, 50 sentences are sampled in total, 5 of each of the 10 activation deciles. Then, another 50 negative samples are drawn uniformly. The score set is fixed in each experiment.

By construction, there are two invariants:

- The explain and score sets are **disjoint**. The explain examples are removed before the score pool is formed. The score is calculated on a held-out test set.
- Sampling is seeded on seed, feature id, corpus, and draw index. The features are independent, and each draw is reproducible.

**Draw experiment sampling.** Exp. 2a needs more than 190 activating samples. The initial draw is drawn similarly to the other experiments. The other four draws are taken from the same pool of activating samples. This does not indicate that the sets are non-overlapping.

Every draw is scored on an identical test set, so the variance between draws reflects the explanation alone rather than the explanation plus the score set. Explain and score sets never overlap, but explain sets can overlap. On average, each draw shares three to four of the 40 examples.

### B.4.3 Exact Feature Count per Experiment

To calculate scores, we aimed for 500 features. This is below the reported feature numbers of Paulo et al. (2025) and Puri et al. (2025), but approaches them.

Table 9: An overview of the number of features per experiment per model. The aim was to have around 500 features per condition. A feature is dropped from an experiment when a metric failed for it at some level. Because of the lower number of alive features in the architecture experiment of `Apertus-8B-Instruct`, we aimed for 450.

| Factor | Pythia | Apertus |
|---|---|---|
| Architecture | 2488 | 2249 |
| Corpus | 510 | 191 |
| Draw | 513 | 497 |
| Examples | 512 | 498 |
| Explainer | 512 | 472 |
| Paraphrase | 513 | 500 |
| Scorer | 512 | 479 |

The number of features used for the architecture experiment is **summed** over all five SAEs. Due to the different architectures, it is not possible to consider the same features between SAEs.

**Corpus Experiment Distribution.** An overview of the per-corpus feature counts and the intersection in orange. The number of overlapping features on `Pythia-160M` is higher compared to `Apertus-8B-Instruct`.

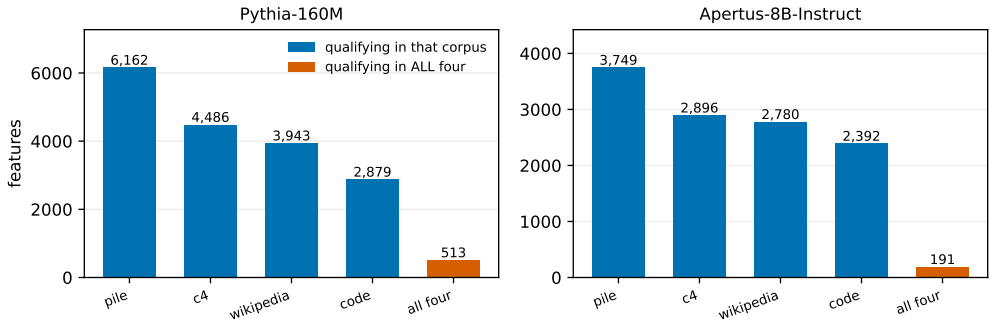


Figure 6: Features that activate on all four corpora, with at least 190 activating sentences. The last bar shows the number of overlapping features. The four-corpus intersection is much smaller for `Apertus-8B-Instruct` (191) than for `Pythia-160M` (513), which limits the CORPUS experiment on `Apertus-8B-Instruct`.

### B.5 Explainer Prompt

**System prompt — Explanation generation**

```
You are a neuron interpreter for a sparse autoencoder (SAE)
trained on a language model.
Your task: Given examples of text where a specific SAE feature
activates strongly, describe what concept, pattern, or behavior
the feature detects.
Guidelines:
- Be specific and concise (1-2 sentences)
- Focus on the common pattern across all examples
- Mention token-level patterns if relevant
```

```
- If the pattern is unclear, say so honestly
- Describe the concept directly (e.g., "Mathematical equations
  and formulas" or "Python source code and programming
  constructs"). Do NOT start with "This feature activates
  for/when/on" or similar phrases.
```

**User prompt — Explanation generation**

```
Here are text examples where SAE feature {feature_id} activates
strongly. Each example shows the text and the activation
strength.
{examples}
Based on these examples, describe what concept or pattern
feature {feature_id} detects. {word_limit_instruction}starting
directly with the concept (not with "This feature
activates...").
```

where `{word_limit_instruction}` resolves to:

- *Default:* "Answer in 1–2 sentences, "

### B.6 Explainers (Exp. 2b)

Exp. 2b compares three explainer LLMs: `gemini-2.5-flash` (primary), with `gpt-4o-mini` and `gemma-3-27B`. All models receive identical input and are prompted to generate a natural language description of the feature. All scoring in Exp. 2b uses `gemini-2.5-flash` as the fixed scorer, isolating the effect of explainer model identity from scorer variation.

### B.7 Paraphrase Generation (Exp. 2c)

Three paraphrases are generated per explanation. The paraphrase prompt instructs the model to preserve the semantic meaning of the original explanation while varying the surface wording, sentence structure and vocabulary. Each original explanation and its three paraphrases are scored independently on the same held-out sequences, producing four observations per feature per metric.

Paraphrases are generated by the same LLM as the scorer, which is `gemini-2.5-flash` . This could mean our paraphrase stability result overstates semantic invariance because the paraphraser and scorer share representational biases. A future test would use paraphrases from a structurally different model, which we view as an important follow-up.

**System prompt**

```
You are a precise language model assistant. Your task is to
rephrase a given feature description in multiple ways, preserving
the exact same meaning each time. Vary only vocabulary and sentence
structure - do NOT change the concept described. Return ONLY a JSON
array of strings, one per paraphrase. No extra text.
```

**Paraphrase Axes**

```
Rewrite it with a different sentence structure: change the grammatical
     form (for example from a noun phrase to a clause, or active to
    passive), while keeping every content word's meaning intact.
Rewrite it in a different register: more formal or more plain-spoken
    than the original, as a technical annotation would be phrased.
```

```
    Keep the same content.
Rewrite it using different vocabulary: replace the content words with
    synonyms or equivalent expressions, keeping the sentence structure
     similar.
```

**User prompt**

```
Original description: {explanation}
Generate {n} paraphrases of the above description. Keep the exact
same meaning. Return ONLY a JSON array of {n} strings.
```

### B.8 Explanation Similarity

To measure semantic consistency of explanations across conditions, we compute **cosine similarity** of E5-small-v2 embeddings (`intfloat/e5-small-v2`). Each explanation is embedded using mean pooling over token embeddings, and pairwise cosine similarity is computed across all condition pairs for each feature.

We investigate similarity only in experiments where the explanation varies, such as for draws, explainers, and paraphrases. For the experiment with a varied scorer, it does not make sense to look at explanation similarity, as it remains constant.

All are reported as mean pairwise comparisons within the experiment. The similarities are highly consistent across all experiments and models (Table 16).

### B.9 Scoring Procedure

In the following section, we extend the discussion with details on the metrics used and the scoring prompts.

#### B.9.1 Metric Details

Metrics follow a two-phase pipeline: 1) the explainer LLM generates a natural language description of the feature from its top-activating samples and 2) a scorer LLM evaluates the explanation against held-out sequences.

- SIMULATION (Bills et al., 2023). A scorer LLM is shown the explanation and a set of sequences and is asked to predict the feature's activation at each token position on a scale of 0 to 10. Simulation score is the Pearson correlation between predicted and actual activation values at the token level, ranging from -1 (inverse prediction) to 1 (perfect prediction). A score of 0 indicates that predictions are no better than chance.
- DETECTION (Paulo et al., 2025). The scorer LLM is shown the explanation and is asked to classify sequences as either activating or non-activating. The detection score is the balanced accuracy of a binary classifier, ranging from 0 to 1, with 0.5 indicating chance performance. Positive sequences are the top activating sequences; negative sequences are randomly sampled sequences from the corpus. Detection is "cheaper" than simulation, as it requires only a sequence-level binary decision rather than token-level activation prediction/classification of activating and non-activating samples.
- FUZZING (Paulo et al., 2025). Similar to detection, but then on the token level within activating sequences. The scorer is shown an explanation and sequences with some tokens marked, and asked to predict whether the original token is likely to activate the feature. The fuzzing score is the balanced accuracy of these token-level predictions, ranging from 0 to 1. Fuzzing is more sensitive to whether the explanation correctly identifies relevant tokens.
- PURITY (Puri et al., 2025). Measures whether high activations are exclusive to the target concept at the sequence level. Natural samples are rated by a scorer LLM on a three-point scale (0 = not expressed, 1 = partial, 2 = clearly expressed); ambiguous cases are discarded,

leaving binary concept/non-concept labels. Average Precision (AP) is then computed using feature activation magnitudes as the ranking signal, ranging from 0 to 1. A high purity score indicates that strongly activating sequences are specific to the target concept, while a low score indicates that unrelated sequences also produce high activations.

### B.9.2 SCORING PROMPTS

We use four scoring prompts, one for each metric. All prompts are sent to `gemini-2.5-flash` as the scorer. The `{explanation}` placeholder is replaced with the generated feature explanation; other placeholders are filled with the relevant sequences.

**Simulation** Implements the scoring procedure of (Bills et al., 2023). The scorer predicts token-level activation magnitudes on a 0–10 scale; the simulation score is the Pearson correlation between predicted and actual activations.

**System prompt — simulation**

```
You are a neuron activation simulator for a sparse autoencoder
(SAE) trained on a language model.

Given a description of what an SAE feature detects and multiple
pieces of text, predict how strongly the feature would activate
on each text.

Return a JSON array of numbers between 0 and 10:
- 0 means the feature would not activate at all
- 10 means the feature would activate very strongly

Return ONLY the JSON array, nothing else.
Example: [3.5, 0.0, 8.2, 1.0]
```

**User prompt — simulation**

```
Feature description: {explanation}

Texts:
{texts}

For each text, predict the feature activation (0-10).
Return ONLY a JSON array of {num_texts} numbers.
```

**Detection** Implements the detection procedure of (Paulo et al., 2025). The scorer performs binary classification of sequences as activating (1) or non-activating (0); the detection score is the balanced accuracy over positive and negative sequences.

**System prompt — detection**

```
You are an intelligent and meticulous linguistics researcher.
You will be given a certain feature of text, such as "male
pronouns" or "text with negative sentiment". You will then be
given several text examples. Your task is to determine which
examples possess the feature.
For each example in turn, return 1 if the sentence is correctly
labeled or 0 if the tokens are mislabeled. You must return your
response in a valid Python list. Do not return anything else
besides a Python list.
```

**User prompt — detection**

```
feature interpretation: {explanation}
Text examples:
{texts}
```

**Fuzzing** Implements the token-level fuzzing procedure of (Paulo et al., 2025). Activating tokens are marked with « and » delimiters; the scorer classifies whether the marked tokens correctly represent the feature. The fuzzing score is the balanced accuracy over correctly and incorrectly marked token spans.

**System prompt — fuzzing**

```
You are an intelligent and meticulous linguistics researcher.
You will be given a certain feature of text, such as "male
pronouns" or "text with negative sentiment". You will be given
a few examples of text that contain this feature. Portions of
the sentence which strongly represent this feature are between
tokens << and >>.
Some examples might be mislabeled. Your task is to determine
if every single token within << and >> is correctly labeled.
Consider that all provided examples could be correct, none of
the examples could be correct, or a mix. An example is only
correct if every marked token is representative of the feature.
For each example in turn, return 1 if the sentence is correctly
labeled or 0 if the tokens are mislabeled. You must return your
response in a valid Python list. Do not return anything else
besides a Python list.
```

**User prompt — fuzzing**

```
feature interpretation: {explanation}
Text examples:
{examples}
```

**Purity** Implements the FADE sequence rating procedure of (Puri et al., 2025). The scorer rates each sequence on a 0–2 scale according to how clearly the concept is expressed; the purity score is the average precision over rated sequences.

**System prompt — purity**

```
You are tasked with building a database of sequences that best
represent a specific concept.
To create this, you will review a dataset of varying sequences
and rate each one according to how much the concept is
expressed.
For each sequence, assign a rating based on this scale:
0: The concept is not expressed.
1: The concept is vaguely or partially expressed.
2: The concept is clearly and unambiguously present.
Use conservative ratings. If uncertain, choose a lower rating
to avoid including irrelevant sequences in your database.
If no sequence expresses the concept, rate all sequences as 0.
Each sequence is identified by a unique ID. Provide your
ratings as a Python dictionary with sequence IDs as keys and
```

```
their ratings as values.
Example output: {"14": 0, "15": 2, "20": 1, "27": 0}
Output only the dictionary - no additional text, comments,
or symbols.
```

**User prompt — purity**

```
Concept: {explanation}

Sequences:
{sequences}
```

# C Statistical Methods

## C.1 ICC specification

We use the Intraclass Correlation Coefficient (ICC) to quantify the proportion of total score variance attributable to stable **between-feature** differences, rather than within-feature noise (Koo & Li, 2016).

There is a difference between ICC(1,1) and ICC(2,1):

- **Draw — ICC(1,1).** Random draws of activating examples are not a fixed, replicable set of conditions. Each draw is a unique random sample from the activation pool. We therefore treat draws as a one-way random effects model (ICC(1,1)), in which the conditions themselves are sampled from a larger population of possible draws.
- **Corpus, Explainer, Scorer, Paraphrase, Explainer Samples — ICC(2,1).** These sources involve a fixed, replicable set of conditions (four specific corpora; three specific explainer and scorer models; four specific explanation versions; nested example sets). We therefore treat them as a two-way random effects model (ICC(2,1)).

We report the ICC for each metric and for each factor. However, even though these thresholds are stated, they do not indicate that anything below 0.75 is directly unreliable (Liljequist et al., 2019). It can be informative to look at other studies that report ICC to get an indication of the range of ICC values, but to our knowledge, there is no previous study that has applied this to autointerpretability scores. The threshold can be re-calibrated after usage has increased.

## C.2 Top-$k$ Jaccard Lift

Raw Jaccard values for the top-$k$ selection are not directly comparable across experiments, as the overlap expected by chance depends on both $k$ and the feature pool size $n$, which differs between experiments. Correcting set-overlap indices for chance is standard in the feature-selection stability literature (Kuncheva, 2007; Albatineh & Niewiadomska-Bugaj, 2011), which we follow. Under the null hypothesis that two conditions rank features independently and the draw is from a pool without replacement, the overlap $i = |T_k^A \cap T_k^B|$ of two top-$k$ sets drawn from $n$ features is hypergeometric, and since $|T_k^A \cup T_k^B| = 2k - i$, the expected Jaccard is

$$J_k^{\text{chance}} = \mathbb{E}\left[\frac{i}{2k-i}\right] = \sum_{i=0}^{k} \frac{\binom{k}{i}\binom{n-k}{k-i}}{\binom{n}{k}} \cdot \frac{i}{2k-i}, \tag{5}$$

which we evaluate numerically (Albatineh & Niewiadomska-Bugaj, 2011). We report *lift*, the ratio of observed to chance Jaccard:

$$\text{Lift}_k = \frac{J_k^{\text{observed}}}{J_k^{\text{chance}}}. \tag{6}$$

Lift $= 1$ means the top-$k$ sets of two levels agree no more than random selections of the same size; lift $> 1$ means that the same features are shared across levels more often than chance.

For each factor, we average $J_k$ over all pairs of conditions and divide by $J_k^{\text{chance}}$ (equal to the mean pairwise lift, since chance is constant at fixed $n$ and $k$), for $k \in \{10, 20, 30, 40, 50\}$. On DETECTION, FUZZING and PURITY, many features share the maximum score, so which features form the top-$k$ is partially arbitrary. Reported overlaps for these metrics are approximations.

#### C.2.1 JACCARD SHARED FEATURE CALCULATION

The number of shared features $m$ implied by an observed Jaccard, $J_{\text{observed}}$, follows from $J = m/(2k - m)$:

$$m = \frac{2k\, J_{\text{obs}}}{1 + J_{\text{obs}}}. \tag{7}$$

### C.3 VARIANCE EXPLAINED

**Definition.** We repeat Eq. 3 to provide clarification. Let $x_{ij}$ be the score of feature $i$ under level $j$ of a factor, $n_j$ the number of features in level $j$, $\bar{x}_{\cdot j}$ the mean of level $j$, and $\bar{x}$ the grand mean over all levels of a factor. Then

$$\eta^2 = \frac{SS_{\text{between}}}{SS_{\text{total}}} = \frac{\sum_j n_j(\bar{x}_{\cdot j} - \bar{x})^2}{\sum_{ij}(x_{ij} - \bar{x})^2}, \tag{8}$$

the share of total score variance explained by which level produced a score. It is the **between-level** variation divided by the **total variation** of all features, as each feature's score under a level is one observation. For example, each level can be one draw, but the scores of the features under all draws are all individual observations. $\eta^2 \in [0, 1]$; where 0 means that the level does not contain information on the score.

**Difference for methodological and architecture.** ICC requires the same features measured under every level. This is not applicable to architectures, as each SAE has its own dictionary, and there is no guarantee that they share similar features. $\eta^2$ does not require feature pairing and is therefore defined for each factor, architecture included. Additionally, the $n_j$ allows for a weight-corrected variance.

There is a slight variation in the sum of squared deviations $SS$ for methodological variance and architectural variance. For a methodological factor, the same features occur at every level, so then

$$SS_{\text{total}} = SS_{\text{level}} + SS_{\text{feature}} + SS_{\text{error}}. \tag{9}$$

For architecture, all SAEs contain different features, so then

$$SS_{\text{total}} = SS_{\text{arch}} + SS_{\text{within}}, \tag{10}$$

where $SS_{\text{within}}$ contains feature variance and error inseparably, since this cannot be determined.

Both denominators in Eq 3 are therefore of the form between-level + feature + noise, and the ratio has the same interpretation in both. Each factor is analyzed in its own experiment, with other factors held fixed, and is a separate one-way analysis. Rows do not sum to 100 and the methodological sum consists of separate estimates, therefore $\eta^2$ is not a partition.

**Null floor for architecture.** Because the architectures contain different features, $\eta^2$ is non-zero, even when the true architectural effect is zero. To estimate how large the effect is, we construct a small experiment in which the architectural effect is zero by design. We take the features of a single SAE, BATCHTOPK, randomly divide the features into five groups of equal size and compute $\eta^2$ across these groups. Since all groups come from the same SAE, $\eta^2$ they produce is due to feature differences alone. We repeat this 200 times per model and metric (seeded). Table 10 reports the mean and 95th percentile of the resulting $\eta^2$ values beside the original value of the five architectures combined. The mean shows the average $\eta^2$ over the 200 random splits, so how much chance is produced on average. The 95th percentile is the $\eta^2$ that only 5% of the random splits exceed and shows how much chance can be in some splits.

**Number of levels.** $\eta^2$ grows with the number of levels $L$, since $SS_{\text{between}}$ has $L - 1$ degrees of freedom. We report $L$ beside every value (Table 11). Architecture has the most levels ($L = 5$), similar to the DRAW factor. The SCORER factor has $L = 2$ on simulation, because one scorer (`gemma-3-27B`) was not run on that metric, and $L = 3$ elsewhere.

Table 10: Null floor for architectural $\eta^2$, for five random groups of one SAE's features (BATCHTOPK), 200 splits.

| Model | Metric | Real $\eta^2$ (%) | Null mean (%) | Null 95th percentile (%) |
|---|---|---|---|---|
| `Pythia-160M` | SIMULATION | 5.70 | 0.79 | 1.75 |
| | DETECTION | 1.32 | 0.85 | 2.02 |
| | FUZZING | 6.02 | 0.82 | 2.05 |
| | PURITY | 0.71 | 0.87 | 2.13 |
| `Apertus-8B-Instruct` | SIMULATION | 3.85 | 0.92 | 2.00 |
| | DETECTION | 5.02 | 0.89 | 2.17 |
| | FUZZING | 2.67 | 0.92 | 2.38 |
| | PURITY | 1.43 | 0.95 | 2.63 |

Table 11: Number of levels $L$ per factor. $\eta^2$ grows with $L$; values in Table 22 and Table 23 should be read alongside these.

| **Factor** | **Pythia** | **Apertus** |
|---|---|---|
| Architecture | 5 | 5 |
| Corpus | 4 | 4 |
| Draw | 5 | 5 |
| Examples | 4 | 4 |
| Paraphrase | 4 | 4 |
| Explainer | 3 | 3 |
| Scorer | 3 (2 on simulation) | 3 (2 on simulation) |

# D SPARSE AUTOENCODER DETAILS

## D.1 PYTHIA-160M SAE MODELS

For `Pythia-160M`, there are five pretrained SAEs accessed in this study (Karvonen et al., 2025). For most experiments, we opt for BATCHTOPK SAE with *k* = 160. For Exp. 3, we additionally investigate a MATRYOSHKA SAE, a RELU SAE, a JUMPRELU SAE and a TOPK SAE. All SAEs can be found at `adamkarvonen/saebench_Pythia-160M-deduped_width-2pow14_date-0108`. For almost all, we chose Trainer 3. However, for the RELU, we selected trainer 4, as both were closer to the reconstruction error and the sparsity constraint of 160. The five pretrained SAEs used in this study are all trained on layer 8 of `Pythia-160M`. Table 12 summarises their hyperparameters and evaluation metrics.

Table 12: SAE hyperparameters and training metrics for `Pythia-160M` models from SAEBench. $k$ here is used for the sparsity, as this notation is generally used in literature.

| **Architecture** | **Trainer** | ***k*/penalty** | **Dict. size** | **Layer** | $L_0$ | $L_0$ **sd** | **rel_mse** | **Dead %** |
|---|---|---|---|---|---|---|---|---|
| BatchTopK | `trainer_3` | $k = 160$ | $2^{14} = 16{,}384$ | 8 | 160.0 | 0.0 | 0.0403 | 25.0 |
| Matryoshka | `trainer_3` | $k = 160$ | $2^{14} = 16{,}384$ | 8 | 160.0 | 0.0 | 0.0979 | 31.2 |
| ReLU | `trainer_4` | $\lambda = 0.03$ | $2^{14} = 16{,}384$ | 8 | 146.0 | 48.3 | 0.0461 | 37.4 |
| TopK | `trainer_3` | $k = 160$ | $2^{14} = 16{,}384$ | 8 | 160.0 | 0.0 | 0.0516 | 18.0 |
| JumpReLU | `trainer_3` | $\lambda = 1.0$ | $2^{14} = 16{,}384$ | 8 | 151.7 | 55.0 | 0.0069 | 59.5 |

## D.2 APERTUS-8B-INSTRUCT SAE TRAINING DETAILS

This paper is the first to train and evaluate SAEs on `Apertus-8B-Instruct` (Hernández-Cano et al., 2025), a Swiss multilingual large language model trained on German, French, Italian, Romansh, and English text. SAEs were trained on layer 16 of `Apertus-8B-Instruct` using activations collected from THE PILE, see Section B.2. Table 13 summarises the training hyperparameters.

The size of MATRYOSHKA is slightly smaller compared to the other variants. This was done with

Table 13: Sparse autoencoders compared in the architecture experiment (`Apertus-8B-Instruct`, layer 16). All five share the learning rate, warmup and batch size. Training steps are deliberately not matched as for penalty-based SAEs $L_0$ drifts with training, so step count and $L_0$ cannot both be held fixed. The architectures are matched on $L_0$ instead. The MATRYOSHKA SAE has five nested groups: $[2048, 6144, 14336, 30720, 65536]$. Additionally **alive** here indicates 'it fired at least once in only 512 sequences'.

| **Parameter** | **TopK** | **BatchTopK** | **Matryoshka** | **ReLU** | **Jumprelu** |
|---|---|---|---|---|---|
| Dictionary size $d_{\text{sae}}$ | 65536 | 65536 | 51200 | 65536 | 65536 |
| Expansion | 16.0 | 16.0 | 12.5 | 16.0 | 16.0 |
| Learning rate | 0.001 | 0.001 | 0.001 | 0.001 | 0.001 |
| LR warmup | 1000 | 1000 | 1000 | 1000 | 1000 |
| LR decay | 0.8 / 10k | 0.8 / 10k | 0.9 / 20k | 0.8 / 10k | 0.8 / 10k |
| Batch size | 4096 | 4096 | 4096 | 4096 | 4096 |
| Sparsity control | $k$=160 per-token | $k$=160 batch | $k$=160 batch[a] | $\lambda_{L_1}$=28.5[b] | $\lambda_{L_0}$=505[b,c] |
| Auxiliary loss | 0.03125 | 0.03125 | 0.03125 | – | –[d] |
| Measured $L_0$ | 160.0 | 157.4 | 151.0 | 165.2 | 155.9 |
| $L_0$ s.d. | 0.0 | 14.4 | 34.1 | 63.5 | 58.5 |
| Rel. MSE | 1.18e-03 | 3.92e-03 | 3.95e-03 | 1.26e-03 | 1.15e-03 |
| Alive features | 46.6% | 17.0% | 14.4% | 9.4% | 9.3% |

efficiency and computational limitations in mind.

We aimed to keep the coefficients of RELU and JUMPRELU held by a target-$L_0$ controller, with targets 200 and 160. A fixed coefficient does not hold $L_0$, which was drifting upwards during training. A key difference between `Apertus-8B-Instruct` and `Pythia-160M` is the activation scale: `Apertus-8B-Instruct` features activate with magnitudes in the hundreds to thousands, compared to single digits for `Pythia-160M`. This is shown in Figure 5.

The large dictionary size ($d_{\text{SAE}} = 65{,}536$) combined with batch-level sparsity enforcement ($k = 160$) results in an expected per-feature activation rate of approximately:

$$P(\text{feature activates}) = \frac{k}{d_{\text{SAE}}} = \frac{160}{65{,}536} \approx 0.24\% \tag{11}$$

per token. Over a typical evaluation sequence of 32 tokens, a randomly sampled feature is therefore expected to activate on fewer than one token per sequence on average, which explains the lower feature availability observed for BATCHTOPK on `Apertus-8B-Instruct`.

Another difference between the architectures is the number of *alive features*, seen in the last row of Table 13. We acknowledge that this can cause differences in scores between architectures, as this limits the number of features that can be evaluated. The largest differences are between TOPK and JUMPRELU.

**Training and Evaluation Scores** As these SAEs were trained by us, we report the train and evaluation scores, which can be seen in the lower part of Table 13. The best checkpoint was taken as the final model checkpoint.

Due to anonymisation, we removed the link of the trained SAEs, but the models are publicly released.

# E RESULTS

## E.1 MEAN SCORES AND STANDARD DEVIATION PER CONDITION

As SIMULATION is not calculated for the `gemma-3-27B` SCORER condition, it is not included in Table 14 and Table 15.

Table 14: Mean and standard deviation of each interpretability metric under every methodological condition for `Pythia-160M`. Standard deviations are across the features at that condition.

| | | simulation | | detection | | fuzzing | | purity | |
|---|---|---|---|---|---|---|---|---|---|
| **Exp.** | **Condition** | mean | std | mean | std | mean | std | mean | std |
| Corpus | Pile | 0.522 | 0.320 | 0.754 | 0.148 | 0.781 | 0.208 | 0.761 | 0.209 |
| | C4 | 0.566 | 0.321 | 0.760 | 0.147 | 0.819 | 0.160 | 0.768 | 0.232 |
| | Wikipedia | 0.558 | 0.320 | 0.786 | 0.121 | 0.808 | 0.167 | 0.766 | 0.230 |
| | Code | 0.346 | 0.309 | 0.752 | 0.149 | 0.672 | 0.292 | 0.818 | 0.170 |
| Draw | Draw 1 | 0.522 | 0.316 | 0.751 | 0.151 | 0.783 | 0.207 | 0.765 | 0.208 |
| | Draw 2 | 0.521 | 0.318 | 0.756 | 0.148 | 0.782 | 0.208 | 0.770 | 0.198 |
| | Draw 3 | 0.533 | 0.318 | 0.754 | 0.150 | 0.783 | 0.209 | 0.761 | 0.209 |
| | Draw 4 | 0.518 | 0.317 | 0.756 | 0.152 | 0.779 | 0.219 | 0.764 | 0.208 |
| Paraphrase | Original | 0.526 | 0.317 | 0.759 | 0.152 | 0.789 | 0.203 | 0.771 | 0.203 |
| | Paraphrase 1 | 0.526 | 0.322 | 0.754 | 0.148 | 0.787 | 0.200 | 0.736 | 0.222 |
| | Paraphrase 2 | 0.526 | 0.320 | 0.754 | 0.149 | 0.787 | 0.199 | 0.730 | 0.224 |
| | Paraphrase 3 | 0.469 | 0.317 | 0.726 | 0.148 | 0.779 | 0.207 | 0.696 | 0.236 |
| Explainer | gemini-2.5-flash | 0.528 | 0.317 | 0.789 | 0.154 | 0.787 | 0.202 | 0.778 | 0.201 |
| | gemma-3-27B | 0.303 | 0.270 | 0.650 | 0.142 | 0.819 | 0.159 | 0.636 | 0.238 |
| | gpt-4o-mini | 0.312 | 0.301 | 0.637 | 0.149 | 0.806 | 0.135 | 0.587 | 0.251 |
| Scorer | gemini-2.5-flash | 0.528 | 0.316 | 0.762 | 0.147 | 0.792 | 0.199 | 0.770 | 0.213 |
| | gemma-3-27B | – | – | 0.734 | 0.136 | 0.820 | 0.158 | 0.787 | 0.172 |
| | gpt-4o-mini | 0.299 | 0.194 | 0.739 | 0.130 | 0.806 | 0.135 | 0.566 | 0.229 |
| Example count | 10 examples | 0.389 | 0.354 | 0.710 | 0.166 | 0.712 | 0.234 | 0.710 | 0.238 |
| | 20 | 0.468 | 0.341 | 0.743 | 0.163 | 0.761 | 0.224 | 0.761 | 0.215 |
| | 40 | 0.526 | 0.317 | 0.756 | 0.147 | 0.779 | 0.220 | 0.782 | 0.202 |
| | 80 | 0.536 | 0.301 | 0.755 | 0.146 | 0.790 | 0.196 | 0.769 | 0.206 |
| Architecture | BatchTopK | 0.419 | 0.324 | 0.787 | 0.135 | 0.741 | 0.227 | 0.831 | 0.192 |
| | JumpReLU | 0.225 | 0.269 | 0.747 | 0.105 | 0.603 | 0.175 | 0.841 | 0.147 |
| | Matryoshka | 0.386 | 0.319 | 0.774 | 0.125 | 0.728 | 0.220 | 0.805 | 0.182 |
| | Standard | 0.356 | 0.318 | 0.778 | 0.122 | 0.709 | 0.219 | 0.802 | 0.206 |
| | TopK | 0.440 | 0.303 | 0.784 | 0.127 | 0.746 | 0.204 | 0.828 | 0.182 |

Table 15: Mean and standard deviation of each interpretability metric under every methodological condition for `Apertus-8B-Instruct`. Standard deviations are across the features at that condition.

| | | simulation | | detection | | fuzzing | | purity | |
|---|---|---|---|---|---|---|---|---|---|
| **Exp.** | **Condition** | mean | std | mean | std | mean | std | mean | std |
| Corpus | Pile | 0.268 | 0.269 | 0.676 | 0.136 | 0.661 | 0.199 | 0.713 | 0.203 |
| | C4 | 0.338 | 0.278 | 0.709 | 0.131 | 0.731 | 0.150 | 0.706 | 0.231 |
| | Wikipedia | 0.328 | 0.271 | 0.775 | 0.115 | 0.741 | 0.162 | 0.754 | 0.212 |
| | Code | 0.107 | 0.192 | 0.697 | 0.147 | 0.522 | 0.270 | 0.846 | 0.151 |
| Draw | Draw 1 | 0.279 | 0.248 | 0.776 | 0.131 | 0.668 | 0.198 | 0.794 | 0.154 |
| | Draw 2 | 0.274 | 0.245 | 0.770 | 0.133 | 0.670 | 0.194 | 0.788 | 0.156 |
| | Draw 3 | 0.279 | 0.240 | 0.773 | 0.133 | 0.668 | 0.205 | 0.785 | 0.163 |
| | Draw 4 | 0.273 | 0.248 | 0.771 | 0.135 | 0.663 | 0.203 | 0.795 | 0.152 |
| Paraphrase | Original | 0.276 | 0.246 | 0.778 | 0.138 | 0.654 | 0.209 | 0.797 | 0.119 |
| | Paraphrase 1 | 0.276 | 0.248 | 0.770 | 0.136 | 0.664 | 0.200 | 0.783 | 0.131 |
| | Paraphrase 2 | 0.274 | 0.244 | 0.770 | 0.140 | 0.656 | 0.208 | 0.785 | 0.129 |
| | Paraphrase 3 | 0.267 | 0.236 | 0.770 | 0.135 | 0.652 | 0.211 | 0.788 | 0.126 |
| Explainer | gemini-2.5-flash | 0.269 | 0.233 | 0.770 | 0.140 | 0.666 | 0.195 | 0.792 | 0.123 |
| | gemma-3-27B | 0.203 | 0.206 | 0.729 | 0.129 | 0.633 | 0.196 | 0.749 | 0.209 |
| | gpt-4o-mini | 0.206 | 0.195 | 0.728 | 0.135 | 0.649 | 0.165 | 0.738 | 0.150 |
| Scorer | gemini-2.5-flash | 0.255 | 0.243 | 0.756 | 0.129 | 0.661 | 0.197 | 0.803 | 0.169 |
| | gemma-3-27B | – | – | 0.725 | 0.120 | 0.724 | 0.166 | 0.810 | 0.150 |
| | gpt-4o-mini | 0.148 | 0.166 | 0.720 | 0.113 | 0.721 | 0.176 | 0.620 | 0.216 |
| Example count | 10 examples | 0.190 | 0.232 | 0.725 | 0.141 | 0.622 | 0.212 | 0.747 | 0.222 |
| | 20 | 0.232 | 0.249 | 0.742 | 0.136 | 0.637 | 0.214 | 0.787 | 0.186 |
| | 40 | 0.252 | 0.248 | 0.755 | 0.134 | 0.657 | 0.200 | 0.799 | 0.187 |
| | 80 | 0.276 | 0.245 | 0.757 | 0.126 | 0.669 | 0.196 | 0.790 | 0.180 |
| Architecture | BatchTopK | 0.262 | 0.247 | 0.752 | 0.128 | 0.666 | 0.205 | 0.795 | 0.178 |
| | JumpReLU | 0.235 | 0.210 | 0.760 | 0.098 | 0.647 | 0.157 | 0.783 | 0.174 |
| | Matryoshka | 0.195 | 0.238 | 0.707 | 0.138 | 0.603 | 0.207 | 0.755 | 0.194 |
| | Standard | 0.337 | 0.245 | 0.773 | 0.095 | 0.698 | 0.159 | 0.820 | 0.163 |
| | TopK | 0.262 | 0.224 | 0.784 | 0.111 | 0.654 | 0.202 | 0.800 | 0.180 |

### E.2 SIMILARITY METRIC

Table 16: Cosine similarity of feature explanations across draw, explainer, example-count, and paraphrase conditions, for `Pythia-160M` and `Apertus-8B-Instruct`.

| | | Pythia | | | Apertus | | |
|---|---|---|---|---|---|---|---|
| **Exp.** | **Condition** | mean | std | $n$ | mean | std | $n$ |
| Draw | (all levels) | 0.919 | 0.025 | 513 | 0.928 | 0.028 | 500 |
| Explainer | (all levels) | 0.854 | 0.035 | 513 | 0.881 | 0.030 | 482 |
| Example count | (all levels) | 0.910 | 0.028 | 512 | 0.916 | 0.025 | 498 |
| Paraphrase | Paraphrase 1 | 0.975 | 0.011 | 513 | 0.978 | 0.010 | 500 |
| | Paraphrase 2 | 0.961 | 0.015 | 513 | 0.961 | 0.017 | 500 |
| | Paraphrase 3 | 0.948 | 0.021 | 513 | 0.945 | 0.018 | 500 |

### E.3 ICC FIGURES

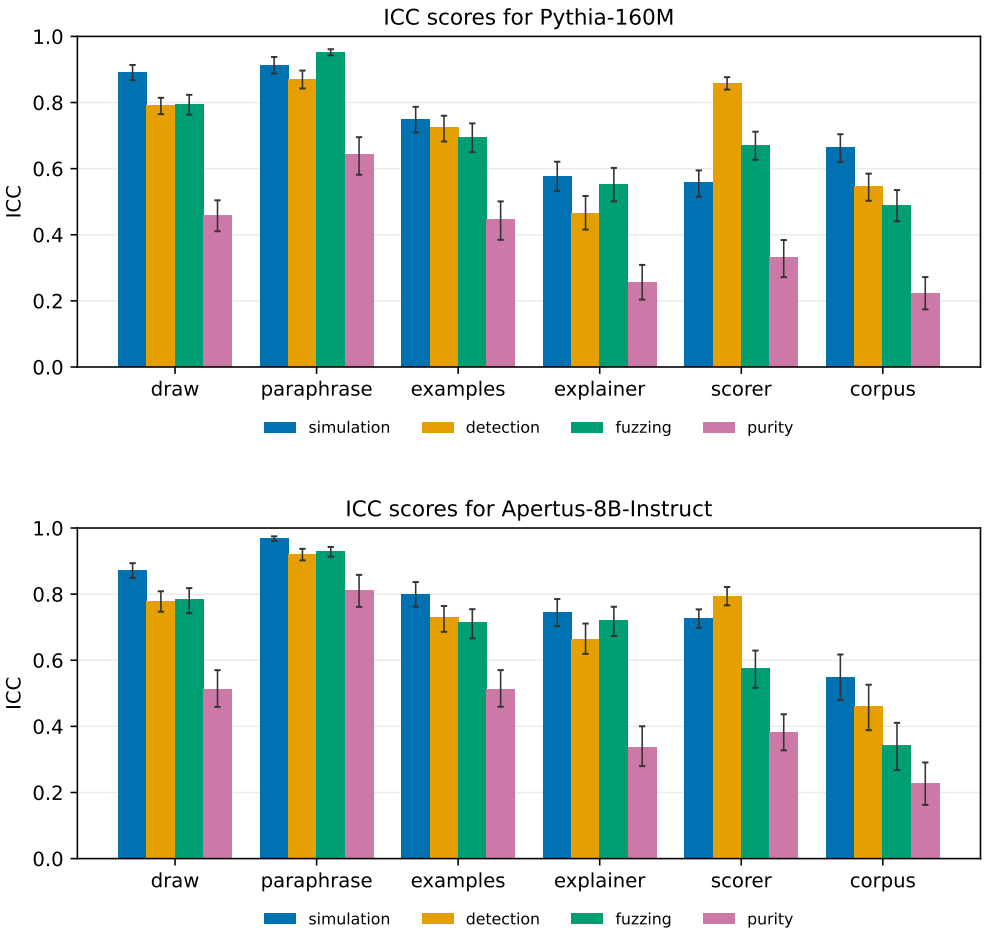


Figure 7: The ICC per factor for all metrics for `Pythia-160M` (top) and `Apertus-8B-Instruct` (bottom), with 95% confidence intervals. This is the transpose of Figure 3.

### E.4 JACCARD TABLES

A limitation of the Jaccard metric is that when there are more than $k$ features that have a perfect score, there is no distinction to be made for the top $k$, as they are equally good. For example, 28 of 513 features achieve a perfect FUZZING score in the DRAW condition, which can decrease the Jaccard lift when non-overlapping features are chosen. However, it is interesting that for PARAPHRASE or EXAMPLE COUNT the scores are higher, despite a similar ceiling.

We report the Jaccard values and the Jaccard lift values separately.

#### E.4.1 JACCARD OBSERVED

Table 17: Top-$k$ feature-set Jaccard overlap (95% CI) by factor, across values of $k$ for `Pythia-160M`. The SIMULATION results for SCORER are calculated over two scorers instead of three, therefore the results are not directly comparable.

| $k$ | **Factor** | simulation | detection | fuzzing | purity |
|---|---|---|---|---|---|
| 10 | Corpus | 0.055 (0.037, 0.078) | 0.065 (0.027, 0.083) | 0.036 (0.014, 0.089) | 0.045 (0.015, 0.070) |
| | Draw | 0.131 (0.109, 0.185) | 0.137 (0.126, 0.238) | 0.079 (0.061, 0.164) | 0.097 (0.056, 0.137) |
| | Explainer | 0.094 (0.030, 0.141) | 0.155 (0.056, 0.160) | 0.094 (0.030, 0.164) | 0.037 (0.013, 0.066) |
| | Paraphrase | 0.356 (0.263, 0.438) | 0.272 (0.177, 0.348) | 0.266 (0.124, 0.307) | 0.115 (0.093, 0.226) |
| | Scorer | 0.000 (0.000, 0.000) | 0.133 (0.096, 0.225) | 0.035 (0.004, 0.109) | 0.037 (0.004, 0.066) |
| | Example count | 0.155 (0.110, 0.220) | 0.157 (0.111, 0.245) | 0.112 (0.074, 0.223) | 0.178 (0.090, 0.233) |
| 20 | Corpus | 0.097 (0.071, 0.112) | 0.125 (0.091, 0.150) | 0.087 (0.058, 0.115) | 0.059 (0.041, 0.075) |
| | Draw | 0.205 (0.177, 0.243) | 0.224 (0.186, 0.268) | 0.188 (0.172, 0.264) | 0.159 (0.125, 0.179) |
| | Explainer | 0.155 (0.110, 0.197) | 0.155 (0.099, 0.196) | 0.183 (0.114, 0.205) | 0.083 (0.049, 0.104) |
| | Paraphrase | 0.350 (0.287, 0.390) | 0.343 (0.283, 0.389) | 0.345 (0.289, 0.416) | 0.227 (0.195, 0.256) |
| | Scorer | 0.026 (0.003, 0.032) | 0.213 (0.164, 0.276) | 0.092 (0.070, 0.127) | 0.062 (0.036, 0.102) |
| | Example count | 0.322 (0.275, 0.378) | 0.233 (0.188, 0.283) | 0.266 (0.223, 0.345) | 0.198 (0.155, 0.233) |
| 30 | Corpus | 0.141 (0.115, 0.166) | 0.166 (0.146, 0.199) | 0.116 (0.097, 0.142) | 0.064 (0.053, 0.083) |
| | Draw | 0.286 (0.257, 0.325) | 0.252 (0.218, 0.283) | 0.276 (0.238, 0.311) | 0.173 (0.152, 0.200) |
| | Explainer | 0.170 (0.146, 0.206) | 0.186 (0.143, 0.219) | 0.231 (0.178, 0.266) | 0.112 (0.086, 0.134) |
| | Paraphrase | 0.355 (0.310, 0.394) | 0.424 (0.379, 0.479) | 0.421 (0.324, 0.439) | 0.307 (0.269, 0.345) |
| | Scorer | 0.017 (0.002, 0.041) | 0.297 (0.260, 0.346) | 0.156 (0.098, 0.170) | 0.135 (0.097, 0.160) |
| | Example count | 0.340 (0.283, 0.383) | 0.295 (0.262, 0.353) | 0.337 (0.263, 0.366) | 0.218 (0.181, 0.245) |
| 40 | Corpus | 0.182 (0.147, 0.201) | 0.192 (0.165, 0.212) | 0.155 (0.128, 0.182) | 0.089 (0.078, 0.103) |
| | Draw | 0.330 (0.301, 0.362) | 0.282 (0.257, 0.310) | 0.301 (0.271, 0.342) | 0.201 (0.175, 0.223) |
| | Explainer | 0.251 (0.213, 0.279) | 0.238 (0.193, 0.268) | 0.279 (0.197, 0.286) | 0.117 (0.100, 0.142) |
| | Paraphrase | 0.382 (0.345, 0.425) | 0.452 (0.413, 0.500) | 0.443 (0.387, 0.506) | 0.355 (0.320, 0.389) |
| | Scorer | 0.053 (0.020, 0.081) | 0.321 (0.289, 0.376) | 0.143 (0.126, 0.190) | 0.124 (0.102, 0.150) |
| | Example count | 0.361 (0.306, 0.389) | 0.343 (0.304, 0.385) | 0.317 (0.279, 0.364) | 0.241 (0.212, 0.273) |
| 50 | Corpus | 0.203 (0.177, 0.224) | 0.189 (0.175, 0.219) | 0.200 (0.163, 0.216) | 0.104 (0.097, 0.123) |
| | Draw | 0.369 (0.340, 0.397) | 0.301 (0.277, 0.322) | 0.340 (0.304, 0.375) | 0.215 (0.196, 0.240) |
| | Explainer | 0.294 (0.256, 0.325) | 0.236 (0.210, 0.268) | 0.263 (0.204, 0.283) | 0.141 (0.124, 0.168) |
| | Paraphrase | 0.386 (0.344, 0.426) | 0.490 (0.452, 0.535) | 0.509 (0.474, 0.551) | 0.387 (0.340, 0.407) |
| | Scorer | 0.064 (0.042, 0.079) | 0.329 (0.301, 0.377) | 0.173 (0.157, 0.215) | 0.134 (0.117, 0.163) |
| | Example count | 0.343 (0.316, 0.380) | 0.357 (0.315, 0.383) | 0.353 (0.317, 0.380) | 0.256 (0.229, 0.284) |

Table 18: Top-$k$ feature-set Jaccard overlap (95% CI) by factor, across values of $k$, for `Apertus-8B-Instruct`. The SIMULATION results for SCORER are calculated over two scorers instead of three, therefore the results can be lower.

| $k$ | **Factor** | simulation | detection | fuzzing | purity |
|---|---|---|---|---|---|
| 10 | Corpus | 0.231 (0.159, 0.282) | 0.218 (0.147, 0.255) | 0.124 (0.098, 0.196) | 0.112 (0.063, 0.145) |
| | Draw | 0.388 (0.308, 0.450) | 0.290 (0.198, 0.342) | 0.285 (0.223, 0.363) | 0.097 (0.063, 0.139) |
| | Explainer | 0.035 (0.024, 0.086) | 0.166 (0.103, 0.216) | 0.133 (0.096, 0.199) | 0.035 (0.024, 0.077) |
| | Paraphrase | 0.696 (0.550, 0.802) | 0.397 (0.299, 0.495) | 0.505 (0.413, 0.621) | 0.148 (0.110, 0.200) |
| | Scorer | 0.250 (0.154, 0.433) | 0.229 (0.133, 0.297) | 0.076 (0.028, 0.097) | 0.113 (0.052, 0.169) |
| | Example count | 0.502 (0.405, 0.599) | 0.335 (0.231, 0.397) | 0.385 (0.256, 0.472) | 0.311 (0.215, 0.374) |
| 20 | Corpus | 0.323 (0.281, 0.373) | 0.263 (0.224, 0.294) | 0.237 (0.198, 0.273) | 0.146 (0.115, 0.175) |
| | Draw | 0.485 (0.437, 0.546) | 0.334 (0.267, 0.365) | 0.426 (0.360, 0.462) | 0.179 (0.143, 0.200) |
| | Explainer | 0.155 (0.121, 0.197) | 0.193 (0.165, 0.275) | 0.190 (0.127, 0.218) | 0.065 (0.046, 0.115) |
| | Paraphrase | 0.717 (0.640, 0.762) | 0.407 (0.355, 0.472) | 0.612 (0.523, 0.667) | 0.264 (0.207, 0.330) |
| | Scorer | 0.481 (0.349, 0.587) | 0.336 (0.273, 0.408) | 0.104 (0.073, 0.132) | 0.125 (0.088, 0.158) |
| | Example count | 0.612 (0.557, 0.693) | 0.372 (0.298, 0.418) | 0.371 (0.301, 0.431) | 0.294 (0.236, 0.331) |
| 30 | Corpus | 0.440 (0.385, 0.477) | 0.294 (0.260, 0.329) | 0.294 (0.258, 0.327) | 0.216 (0.181, 0.237) |
| | Draw | 0.542 (0.489, 0.594) | 0.313 (0.271, 0.353) | 0.456 (0.421, 0.504) | 0.218 (0.186, 0.242) |
| | Explainer | 0.247 (0.207, 0.284) | 0.260 (0.217, 0.326) | 0.242 (0.214, 0.296) | 0.110 (0.084, 0.140) |
| | Paraphrase | 0.693 (0.628, 0.754) | 0.505 (0.421, 0.539) | 0.593 (0.547, 0.667) | 0.442 (0.374, 0.474) |
| | Scorer | 0.622 (0.556, 0.704) | 0.366 (0.326, 0.435) | 0.128 (0.092, 0.159) | 0.139 (0.104, 0.176) |
| | Example count | 0.632 (0.573, 0.691) | 0.413 (0.366, 0.467) | 0.428 (0.367, 0.470) | 0.271 (0.236, 0.305) |
| 40 | Corpus | 0.461 (0.426, 0.507) | 0.329 (0.299, 0.360) | 0.335 (0.315, 0.381) | 0.235 (0.213, 0.259) |
| | Draw | 0.569 (0.537, 0.614) | 0.342 (0.311, 0.374) | 0.500 (0.464, 0.543) | 0.238 (0.214, 0.271) |
| | Explainer | 0.300 (0.279, 0.357) | 0.334 (0.280, 0.377) | 0.291 (0.244, 0.322) | 0.139 (0.112, 0.162) |
| | Paraphrase | 0.663 (0.633, 0.723) | 0.505 (0.477, 0.586) | 0.634 (0.599, 0.700) | 0.497 (0.446, 0.543) |
| | Scorer | 0.600 (0.532, 0.716) | 0.388 (0.332, 0.443) | 0.149 (0.121, 0.183) | 0.148 (0.126, 0.185) |
| | Example count | 0.622 (0.573, 0.696) | 0.408 (0.373, 0.457) | 0.483 (0.450, 0.551) | 0.287 (0.252, 0.317) |
| 50 | Corpus | 0.502 (0.465, 0.535) | 0.391 (0.355, 0.417) | 0.425 (0.387, 0.462) | 0.280 (0.255, 0.301) |
| | Draw | 0.608 (0.569, 0.646) | 0.357 (0.325, 0.386) | 0.518 (0.477, 0.549) | 0.263 (0.240, 0.284) |
| | Explainer | 0.384 (0.341, 0.430) | 0.316 (0.269, 0.366) | 0.364 (0.311, 0.393) | 0.165 (0.146, 0.195) |
| | Paraphrase | 0.770 (0.711, 0.820) | 0.539 (0.500, 0.589) | 0.631 (0.599, 0.696) | 0.551 (0.519, 0.606) |
| | Scorer | 0.538 (0.488, 0.623) | 0.383 (0.339, 0.425) | 0.183 (0.154, 0.209) | 0.200 (0.174, 0.228) |
| | Example count | 0.632 (0.586, 0.669) | 0.434 (0.393, 0.470) | 0.548 (0.495, 0.594) | 0.305 (0.280, 0.331) |

### E.4.2 JACCARD LIFT

Table 19: Top-$k$ feature-set lift over chance (95% CI) by factor, across values of $k$ of `Pythia-160M`. The SIMULATION results for SCORER are calculated over two scorers instead of three, therefore the results can be lower. The highest lift for $k$ is marked in **bold**.

| $k$ | Factor | Simulation | Detection | Fuzzing | Purity |
|---|---|---|---|---|---|
| 10 | Corpus | 5.27 (3.53, 7.45) | 6.33 (2.63, 7.94) | 3.48 (1.32, 8.51) | 4.33 (1.41, 6.71) |
| | Draw | 12.63 (10.44, 17.60) | 13.23 (12.03, 22.66) | 7.59 (5.84, 15.64) | 9.36 (5.31, 13.04) |
| | Explainer | 9.07 (2.85, 13.46) | 14.94 (5.37, 15.22) | 9.07 (2.85, 15.63) | 3.56 (1.27, 6.30) |
| | Paraphrase | **34.36 (25.06, 41.78)** | **26.29 (16.88, 33.16)** | **25.67 (11.87, 29.26)** | 10.91 (8.74, 21.23) |
| | Scorer | 0.00 (0.00, 0.00) | 12.81 (9.20, 21.44) | 3.39 (0.36, 10.36) | 3.57 (0.38, 6.33) |
| | Example count | 14.92 (10.50, 20.94) | 15.16 (10.59, 23.41) | 10.82 (7.02, 21.23) | **17.08 (8.60, 22.16)** |
| 20 | Corpus | 4.77 (3.45, 5.45) | 6.13 (4.44, 7.34) | 4.25 (2.81, 5.62) | 2.88 (2.01, 3.66) |
| | Draw | 10.08 (8.63, 11.85) | 11.01 (9.08, 13.09) | 9.21 (8.38, 12.85) | 7.81 (6.11, 8.75) |
| | Explainer | 7.59 (5.35, 9.58) | 7.59 (4.82, 9.54) | 8.97 (5.58, 9.98) | 4.04 (2.39, 5.07) |
| | Paraphrase | **17.19 (14.01, 19.01)** | **16.86 (13.82, 19.00)** | **16.95 (14.11, 20.28)** | **10.97 (9.35, 12.31)** |
| | Scorer | 1.26 (0.13, 1.54) | 10.43 (7.99, 13.48) | 4.53 (3.42, 6.21) | 3.04 (1.76, 4.99) |
| | Example count | 15.77 (13.43, 18.41) | 11.44 (9.17, 13.82) | 13.03 (10.87, 16.84) | 9.66 (7.53, 11.33) |
| 30 | Corpus | 4.62 (3.74, 5.41) | 5.42 (4.74, 6.46) | 3.78 (3.15, 4.60) | 2.09 (1.73, 2.70) |
| | Draw | 9.33 (8.36, 10.56) | 8.24 (7.10, 9.20) | 9.01 (7.75, 10.12) | 5.65 (4.95, 6.50) |
| | Explainer | 5.57 (4.74, 6.72) | 6.07 (4.65, 7.12) | 7.56 (5.79, 8.67) | 3.64 (2.78, 4.33) |
| | Paraphrase | **11.60 (10.08, 12.82)** | **13.87 (12.33, 15.57)** | **13.76 (10.55, 14.29)** | **9.87 (8.62, 11.05)** |
| | Scorer | 0.55 (0.06, 1.32) | 9.69 (8.44, 11.24) | 5.11 (3.20, 5.55) | 4.42 (3.15, 5.20) |
| | Example count | 11.09 (9.20, 12.47) | 9.63 (8.54, 11.48) | 10.98 (8.55, 11.90) | 7.08 (5.85, 7.93) |
| 40 | Corpus | 4.42 (3.57, 4.88) | 4.67 (4.01, 5.14) | 3.78 (3.11, 4.41) | 2.18 (1.88, 2.50) |
| | Draw | 8.04 (7.30, 8.79) | 6.86 (6.25, 7.52) | 7.35 (6.58, 8.30) | 4.89 (4.24, 5.42) |
| | Explainer | 6.13 (5.17, 6.77) | 5.79 (4.70, 6.52) | 6.80 (4.77, 6.94) | 2.84 (2.42, 3.42) |
| | Paraphrase | **9.31 (8.38, 10.31)** | **11.01 (10.02, 12.15)** | **10.80 (9.40, 12.28)** | **8.50 (7.66, 9.30)** |
| | Scorer | 1.28 (0.48, 1.98) | 7.80 (7.01, 9.12) | 3.48 (3.07, 4.61) | 3.01 (2.47, 3.64) |
| | Example count | 8.79 (7.43, 9.45) | 8.34 (7.38, 9.35) | 7.72 (6.78, 8.85) | 5.83 (5.11, 6.60) |
| 50 | Corpus | 3.92 (3.42, 4.32) | 3.66 (3.38, 4.22) | 3.87 (3.15, 4.16) | 2.00 (1.87, 2.37) |
| | Draw | 7.13 (6.55, 7.65) | 5.82 (5.34, 6.21) | 6.58 (5.86, 7.23) | 4.15 (3.77, 4.63) |
| | Explainer | 5.69 (4.94, 6.26) | 4.57 (4.05, 5.16) | 5.08 (3.94, 5.46) | 2.71 (2.38, 3.22) |
| | Paraphrase | **7.47 (6.64, 8.21)** | **9.47 (8.71, 10.33)** | **9.85 (9.14, 10.63)** | **7.36 (6.46, 7.72)** |
| | Scorer | 1.23 (0.80, 1.53) | 6.36 (5.81, 7.26) | 3.36 (3.04, 4.14) | 2.58 (2.26, 3.15) |
| | Example count | 6.63 (6.09, 7.33) | 6.88 (6.08, 7.39) | 6.81 (6.11, 7.34) | 4.92 (4.39, 5.45) |

Table 20: Top-$k$ feature-set lift over chance (95% CI) by factor, across values of $k$, for `Apertus-8B-Instruct`. The SIMULATION results for SCORER are calculated over two scorers instead of three, therefore the results can be lower. The highest lift for $k$ is marked in **bold**.

| $k$ | Factor | Simulation | Detection | Fuzzing | Purity |
|---|---|---|---|---|---|
| 10 | Corpus | 8.22 (5.63, 9.94) | 7.76 (5.19, 9.00) | 4.43 (3.47, 6.92) | 4.00 (2.22, 5.13) |
| | Draw | 36.57 (28.71, 41.86) | 27.23 (18.41, 31.71) | 26.73 (20.74, 33.71) | 9.09 (5.84, 12.94) |
| | Explainer | 3.18 (2.12, 7.72) | 15.03 (9.22, 19.34) | 12.06 (8.65, 17.82) | 3.18 (2.12, 6.87) |
| | Paraphrase | **65.51 (51.19, 74.64)** | **37.36 (27.85, 46.09)** | **47.53 (38.48, 57.80)** | 13.90 (10.24, 18.60) |
| | Scorer | 22.54 (13.75, 38.54) | 20.62 (11.83, 26.47) | 6.93 (2.51, 8.68) | 10.20 (4.61, 15.04) |
| | Example count | 47.05 (37.55, 55.47) | 31.44 (21.40, 36.78) | 36.13 (23.70, 43.74) | **29.06 (19.86, 34.49)** |
| 20 | Corpus | 5.76 (4.99, 6.61) | 4.68 (3.98, 5.22) | 4.22 (3.51, 4.84) | 2.59 (2.04, 3.10) |
| | Draw | 23.21 (20.78, 25.97) | 15.92 (12.69, 17.32) | 20.34 (17.08, 21.90) | 8.56 (6.77, 9.48) |
| | Explainer | 7.13 (5.53, 9.01) | 8.87 (7.59, 12.59) | 8.74 (5.81, 9.98) | 2.98 (2.11, 5.29) |
| | Paraphrase | **34.29 (30.44, 36.24)** | **19.45 (16.87, 22.43)** | **29.27 (24.89, 31.71)** | 12.64 (9.85, 15.68) |
| | Scorer | 22.04 (15.89, 26.73) | 15.40 (12.43, 18.55) | 4.78 (3.33, 6.05) | 5.70 (4.01, 7.18) |
| | Example count | 29.15 (26.36, 32.80) | 17.73 (14.11, 19.76) | 17.69 (14.26, 20.41) | **13.94 (11.12, 15.63)** |
| 30 | Corpus | 5.11 (4.48, 5.55) | 3.42 (3.02, 3.82) | 3.42 (3.00, 3.80) | 2.52 (2.10, 2.75) |
| | Draw | 17.26 (15.50, 18.84) | 9.93 (8.56, 11.15) | 14.50 (13.31, 15.95) | 6.93 (5.89, 7.66) |
| | Explainer | 7.56 (6.34, 8.68) | 7.98 (6.64, 9.97) | 7.42 (6.55, 9.04) | 3.37 (2.56, 4.28) |
| | Paraphrase | **22.07 (19.93, 23.91)** | **16.08 (13.34, 17.09)** | **18.88 (17.34, 21.14)** | **14.08 (11.85, 15.04)** |
| | Scorer | 18.93 (16.87, 21.36) | 11.13 (9.89, 13.18) | 3.93 (2.80, 4.85) | 4.23 (3.13, 5.33) |
| | Example count | 20.03 (18.08, 21.80) | 13.10 (11.54, 14.73) | 13.58 (11.57, 14.83) | 8.56 (7.42, 9.60) |
| 40 | Corpus | 3.93 (3.63, 4.32) | 2.80 (2.55, 3.07) | 2.86 (2.68, 3.25) | 2.00 (1.81, 2.20) |
| | Draw | 13.51 (12.71, 14.52) | 8.10 (7.35, 8.82) | 11.83 (10.96, 12.83) | 5.64 (5.04, 6.39) |
| | Explainer | 6.85 (6.35, 8.14) | 7.62 (6.37, 8.59) | 6.65 (5.57, 7.34) | 3.17 (2.56, 3.70) |
| | Paraphrase | **15.73 (14.98, 17.11)** | **11.99 (11.28, 13.86)** | **15.03 (14.17, 16.56)** | **11.78 (10.55, 12.84)** |
| | Scorer | 13.61 (12.03, 16.19) | 8.80 (7.52, 10.02) | 3.41 (2.76, 4.17) | 3.36 (2.84, 4.18) |
| | Example count | 14.69 (13.50, 16.38) | 9.65 (8.78, 10.75) | 11.41 (10.60, 12.97) | 6.76 (5.92, 7.44) |
| 50 | Corpus | 3.33 (3.09, 3.55) | 2.59 (2.36, 2.77) | 2.82 (2.57, 3.07) | 1.86 (1.69, 2.00) |
| | Draw | 11.46 (10.69, 12.13) | 6.71 (6.09, 7.23) | 9.73 (8.94, 10.28) | 4.94 (4.50, 5.32) |
| | Explainer | 6.95 (6.16, 7.78) | 5.73 (4.87, 6.62) | 6.59 (5.62, 7.11) | 3.00 (2.64, 3.53) |
| | Paraphrase | **14.51 (13.36, 15.41)** | **10.15 (9.40, 11.08)** | **11.89 (11.26, 13.07)** | **10.38 (9.75, 11.38)** |
| | Scorer | 9.69 (8.75, 11.18) | 6.88 (6.09, 7.62) | 3.32 (2.78, 3.78) | 3.60 (3.11, 4.08) |
| | Example count | 11.85 (10.96, 12.51) | 8.13 (7.35, 8.78) | 10.28 (9.26, 11.11) | 5.69 (5.22, 6.18) |

### E.5 PAIRWISE SCORER AGREEMENT

Table 21: Pairwise agreement between scoring models on the same explanations (Exp. 2d). Overlap is the number of shared top-10 features and lift uses the exact chance expectation (Eq. 5). **A→B** gives the range of ranks that judge **A**'s top-10 features receive in judge **B**'s ordering, where **A** is the first judge listed. For example, the range of 35-313 for SIMULATION indicates that the top 10 of `gemini-2.5-flash` is spread across rank 35 to 313 for `gpt-4o-mini`. SIMULATION has one pair because `gemma-3-27B` was not run on it. Each pair uses the features scored by both judges, so $n$ varies. Pair 1 is `gemini-2.5-flash` and `gpt-4o-mini`; pair 2 is `gemini-2.5-flash` and `gemma-3-27B`; pair 3 is `gpt-4o-mini` and `gemma-3-27B`.

| Model | Metric | Judge Pair | $n$ | Pearson | Spearman | Overlap | Jac. Lift | A→B | B→A |
|---|---|---|---|---|---|---|---|---|---|
| `Pythia-160M` | Simulation | 1 | 512 | 0.865 | 0.714 | 0 | 0.0 | 35–313 | 81–304 |
| | Detection | 1 | 501 | 0.896 | 0.886 | 3 | 16.6 | 1–156 | 2–88 |
| | | 2 | 512 | 0.866 | 0.845 | 1 | 5.1 | 6–197 | 2–112 |
| | | 3 | 501 | 0.859 | 0.851 | 1 | 5.0 | 2–193 | 3–130 |
| | Fuzzing | 1 | 501 | 0.614 | 0.607 | 2 | 10.5 | 3–149 | 1–189 |
| | | 2 | 513 | 0.806 | 0.767 | 0 | 0.0 | 19–326 | 1–191 |
| | | 3 | 501 | 0.667 | 0.635 | 2 | 10.5 | 1–236 | 1–297 |
| | Purity | 1 | 456 | 0.475 | 0.476 | 1 | 4.5 | 1–306 | 1–426 |
| | | 2 | 471 | 0.485 | 0.513 | 2 | 9.8 | 1–414 | 1–391 |
| | | 3 | 473 | 0.456 | 0.444 | 0 | 0.0 | 14–347 | 16–209 |
| `Apertus-8B-Instruct` | Simulation | 1 | 479 | 0.883 | 0.860 | 4 | 22.5 | 2–26 | 1–43 |
| | Detection | 1 | 465 | 0.849 | 0.852 | 3 | 15.4 | 1–97 | 1–60 |
| | | 2 | 479 | 0.825 | 0.808 | 5 | 30.1 | 2–60 | 1–327 |
| | | 3 | 465 | 0.778 | 0.777 | 3 | 15.4 | 5–204 | 1–126 |
| | Fuzzing | 1 | 469 | 0.468 | 0.505 | 0 | 0.0 | 27–438 | 7–469 |
| | | 2 | 481 | 0.701 | 0.729 | 2 | 10.1 | 2–237 | 7–191 |
| | | 3 | 470 | 0.655 | 0.651 | 1 | 4.7 | 2–264 | 2–110 |
| | Purity | 1 | 457 | 0.561 | 0.558 | 1 | 4.5 | 1–394 | 1–435 |
| | | 2 | 463 | 0.624 | 0.620 | 1 | 4.6 | 1–354 | 1–271 |
| | | 3 | 454 | 0.438 | 0.436 | 2 | 9.5 | 1–370 | 1–168 |

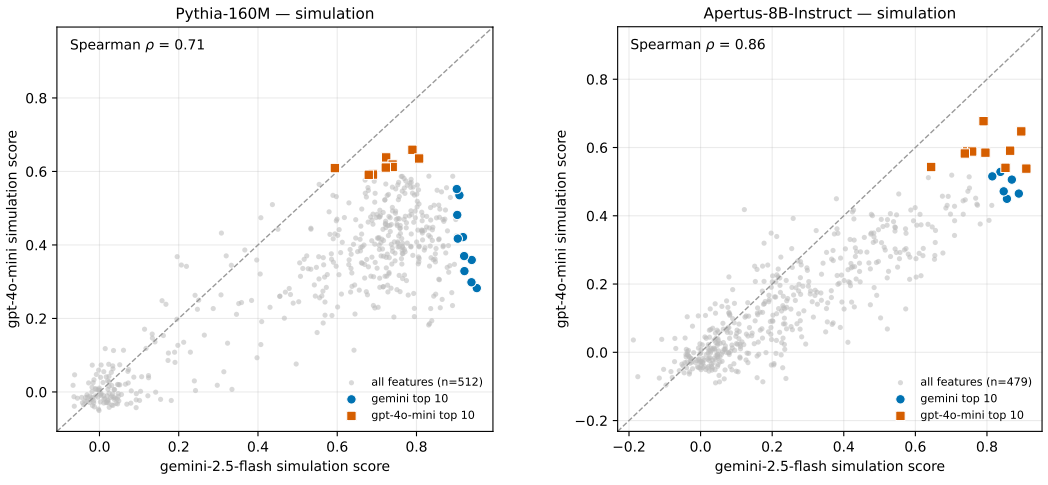


Figure 8: Agreement of SIMULATION scores between `gemini-2.5-flash` and `gpt-4o-mini`.

### E.6 VARIANCE EXPLAINED VALUES

Table 22: Score variance sizes for `Pythia-160M`.

| Factor | $L$ | Simulation (%) | Detection (%) | Fuzzing (%) | Purity (%) |
|---|---|---|---|---|---|
| Draw | 5 | 0.03 | 0.02 | 0.02 | 0.09 |
| Paraphrase | 4 | 0.59 | 0.78 | 0.04 | 1.41 |
| Examples | 4 | 3.05 | 1.40 | 1.85 | 1.61 |
| Explainer | 3 | 10.93 | 11.79 | 5.69 | 10.90 |
| Scorer | 2-3 | 16.10 | 0.77 | 0.55 | 19.21 |
| Corpus | 4 | 7.37 | 0.91 | 6.92 | 1.17 |
| Architecture | 5 | *5.70* | *1.32* | *6.02* | *0.71* |
| **Largest methodological (6 factors)** | | 16.10 | 11.79 | 6.92 | 19.21 |
| *Ratio to architecture* | | *2.8*× | *8.9*× | *1.1*× | *27.1*× |
| **Largest methodological (5, no corpus)** | | 16.10 | 11.79 | 5.69 | 19.21 |
| *Ratio to architecture* | | *2.8*× | *8.9*× | *0.9*× | *27.1*× |

Table 23: Score variance sizes for `Apertus-8B-Instruct`.

| Factor | $L$ | Simulation (%) | Detection (%) | Fuzzing (%) | Purity (%) |
|---|---|---|---|---|---|
| Draw | 5 | 0.01 | 0.03 | 0.03 | 0.07 |
| Paraphrase | 4 | 0.02 | 0.06 | 0.04 | 0.16 |
| Examples | 4 | 1.66 | 0.91 | 0.79 | 1.06 |
| Explainer | 3 | 2.00 | 2.15 | 0.54 | 1.98 |
| Scorer | 2-3 | 6.23 | 1.72 | 2.58 | 19.22 |
| *Corpus* | 4 | 11.64 | 7.21 | 16.03 | 6.98 |
| Architecture | 5 | *3.85* | *5.02* | *2.67* | *1.43* |
| **Largest methodological (6 factors)** | | 11.64 | 7.21 | 16.03 | 19.22 |
| *Ratio to architecture* | | *3.0*× | *1.4*× | *6.0*× | *13.4*× |
| **Largest methodological (5, no corpus)** | | 6.23 | 2.15 | 2.58 | 19.22 |
| *Ratio to architecture* | | *1.6*× | **0.4**× | **1.0**× | *13.4*× |

### E.7 STABILITY ANALYSIS

#### E.7.1 ICC AND C

The decision for a one-way or two-way ICC holds similarly to the main experiments. For DRAW, we use one-way ICC, while for the remaining five factors CORPUS, EXAMPLES, EXPLAINER, PARAPHRASE, SCORER, the levels are fixed, and we use the two-way absolute-agreement model, $\mathrm{ICC}(2,1)$.

Given a single-condition reliability $\rho$, the reliability of an average over $c$ conditions is Equation (4), which is exact for the one-way model and an approximation for the two-way model. The number of conditions needed to reach a target $\rho^*$ is obtained by inversion,

$$c = \frac{\rho^*(1-\rho)}{\rho\,(1-\rho^*)}, \tag{12}$$

rounded up to the next integer. Table 4 in the main text reports this $c$ beside the number of levels run in the current study.

#### E.7.2 VALIDATING THE SPEARMAN-BROWN PROJECTION

For the four factors with at least four levels, we divide the levels into two disjoint pairs, average within each, and compare the resulting scores with what Eq. 4 predicts. The results are shown in

Figure 9 and confirm the assumption. With three levels (EXPLAINER, SCORER), no disjoint split exists, so those factors are not validated and their projected $c$ values are assumed in Table 4.

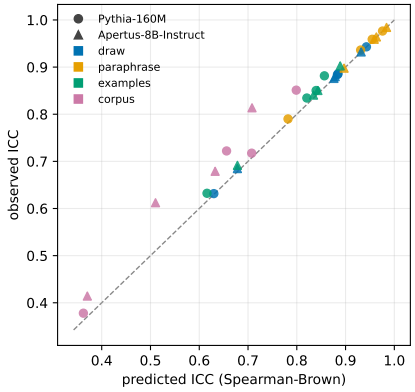


Figure 9: Spearman-Brown projection against the observed ICC (Eq. 4), to validate the assumption that the relation can be used to estimate the number of $c$ to reach ICC of 0.75, for the four factors with at least four levels. Points on the diagonal agree exactly. Points above it indicate that the projection underestimates reliability, so the projected number of conditions $c$ in Table 4 is an upper bound for these factors.

Across the available combinations from models, factors, and metrics, we test the projection of Equation (4) against the observed ICC. Table 24 shows the deviation between the projection and the observed. These results confirm that we can use the projection. As the empirical values are never below the projection, we treat $c$ as an upper bound where a lower number of $c$ might be sufficient, but not more.

Table 24: Empirical two-condition reliability minus the Spearman-Brown projection at $c = 2$, over both models and all four metrics.

| Factor | Range | Mean |
|---|---|---|
| Draw | $-0.000$ to $+0.006$ | $+0.001$ |
| Paraphrase | $+0.000$ to $+0.008$ | $+0.003$ |
| Examples | $+0.006$ to $+0.025$ | $+0.013$ |
| Corpus | $+0.010$ to $+0.105$ | $+0.055$ |

The deviation is largest for CORPUS and smallest for DRAW and PARAPHRASE. Two explanations are consistent with this pattern, and our check cannot separate them. First, corpora have the least interchangeable levels, as they come from different domains. Averaging over two different domains may cancel more error than the interchangeability assumption predicts. Second, the check and the ICC measure metric agreement differently. The check correlates two averages, which ignores differences in their means, while ICC uses absolute agreement (Section C.1). The two agree when the levels have equal means. Of the four factors tested, corpora vary in their level means (up to $0.23$, against at most $0.015$ for DRAW) (see Table 14 and Table 15). Both explanations predict a positive deviation, so in either case the projection is conservative and $c$ remains an upper bound.

### E.7.3 SENSITIVITY TO THE RELIABILITY THRESHOLD

Table 25: ICC and $c$ required to reach reliability thresholds, by factor and metric. A ✓is added when $c \leq L$, while a $\times$ indicates $c > L$. The values for EXPLAINER and SCORER are assumed.

| Model | Factor | Metric | ICC | $n$ run | $c_{70}$ | $c_{75}$ | $c_{80}$ | R.70 | R.75 | R.80 |
|---|---|---|---|---|---|---|---|---|---|---|
| Pythia | Draw | Simulation | 0.8918 | 5 | 1 | 1 | 1 | ✓ | ✓ | ✓ |
| Pythia | Draw | Detection | 0.791 | 5 | 1 | 1 | 2 | ✓ | ✓ | ✓ |
| Pythia | Draw | Fuzzing | 0.7932 | 5 | 1 | 1 | 2 | ✓ | ✓ | ✓ |
| Pythia | Draw | Purity | 0.4598 | 5 | 3 | 4 | 5 | ✓ | ✓ | ✓ |
| Pythia | Paraphrase | Simulation | 0.9131 | 4 | 1 | 1 | 1 | ✓ | ✓ | ✓ |
| Pythia | Paraphrase | Detection | 0.8707 | 4 | 1 | 1 | 1 | ✓ | ✓ | ✓ |
| Pythia | Paraphrase | Fuzzing | 0.9528 | 4 | 1 | 1 | 1 | ✓ | ✓ | ✓ |
| Pythia | Paraphrase | Purity | 0.642 | 4 | 2 | 2 | 3 | ✓ | ✓ | ✓ |
| Pythia | Examples | Simulation | 0.7491 | 4 | 1 | 2 | 2 | ✓ | ✓ | ✓ |
| Pythia | Examples | Detection | 0.7235 | 4 | 1 | 2 | 2 | ✓ | ✓ | ✓ |
| Pythia | Examples | Fuzzing | 0.6957 | 4 | 2 | 2 | 2 | ✓ | ✓ | ✓ |
| Pythia | Examples | Purity | 0.445 | 4 | 3 | 4 | 5 | ✓ | ✓ | × |
| Pythia | Explainer | Simulation | 0.5768 | 3 | 2 | 3 | 3 | ✓ | ✓ | ✓ |
| Pythia | Explainer | Detection | 0.4633 | 3 | 3 | 4 | 5 | ✓ | × | × |
| Pythia | Explainer | Fuzzing | 0.553 | 3 | 2 | 3 | 4 | ✓ | ✓ | × |
| Pythia | Explainer | Purity | 0.2552 | 3 | 7 | 9 | 12 | × | × | × |
| Pythia | Scorer | Simulation | 0.5574 | 2 | 2 | 3 | 4 | ✓ | × | × |
| Pythia | Scorer | Detection | 0.8589 | 3 | 1 | 1 | 1 | ✓ | ✓ | ✓ |
| Pythia | Scorer | Fuzzing | 0.6711 | 3 | 2 | 2 | 2 | ✓ | ✓ | ✓ |
| Pythia | Scorer | Purity | 0.3297 | 3 | 5 | 7 | 9 | × | × | × |
| Pythia | Corpus | Simulation | 0.6652 | 4 | 2 | 2 | 3 | ✓ | ✓ | ✓ |
| Pythia | Corpus | Detection | 0.5474 | 4 | 2 | 3 | 4 | ✓ | ✓ | ✓ |
| Pythia | Corpus | Fuzzing | 0.4881 | 4 | 3 | 4 | 5 | ✓ | ✓ | × |
| Pythia | Corpus | Purity | 0.2213 | 4 | 9 | 11 | 15 | × | × | × |
| Apertus | Draw | Simulation | 0.8726 | 5 | 1 | 1 | 1 | ✓ | ✓ | ✓ |
| Apertus | Draw | Detection | 0.7796 | 5 | 1 | 1 | 2 | ✓ | ✓ | ✓ |
| Apertus | Draw | Fuzzing | 0.7836 | 5 | 1 | 1 | 2 | ✓ | ✓ | ✓ |
| Apertus | Draw | Purity | 0.513 | 5 | 3 | 3 | 4 | ✓ | ✓ | ✓ |
| Apertus | Paraphrase | Simulation | 0.9685 | 4 | 1 | 1 | 1 | ✓ | ✓ | ✓ |
| Apertus | Paraphrase | Detection | 0.9209 | 4 | 1 | 1 | 1 | ✓ | ✓ | ✓ |
| Apertus | Paraphrase | Fuzzing | 0.929 | 4 | 1 | 1 | 1 | ✓ | ✓ | ✓ |
| Apertus | Paraphrase | Purity | 0.8132 | 4 | 1 | 1 | 1 | ✓ | ✓ | ✓ |
| Apertus | Examples | Simulation | 0.8001 | 4 | 1 | 1 | 1 | ✓ | ✓ | ✓ |
| Apertus | Examples | Detection | 0.7291 | 4 | 1 | 2 | 2 | ✓ | ✓ | ✓ |
| Apertus | Examples | Fuzzing | 0.7158 | 4 | 1 | 2 | 2 | ✓ | ✓ | ✓ |
| Apertus | Examples | Purity | 0.5132 | 4 | 3 | 3 | 4 | ✓ | ✓ | ✓ |
| Apertus | Explainer | Simulation | 0.7456 | 3 | 1 | 2 | 2 | ✓ | ✓ | ✓ |
| Apertus | Explainer | Detection | 0.6648 | 3 | 2 | 2 | 3 | ✓ | ✓ | ✓ |
| Apertus | Explainer | Fuzzing | 0.7207 | 3 | 1 | 2 | 2 | ✓ | ✓ | ✓ |
| Apertus | Explainer | Purity | 0.3385 | 3 | 5 | 6 | 8 | × | × | × |
| Apertus | Scorer | Simulation | 0.7266 | 2 | 1 | 2 | 2 | ✓ | ✓ | ✓ |
| Apertus | Scorer | Detection | 0.795 | 3 | 1 | 1 | 2 | ✓ | ✓ | ✓ |
| Apertus | Scorer | Fuzzing | 0.5748 | 3 | 2 | 3 | 3 | ✓ | ✓ | ✓ |
| Apertus | Scorer | Purity | 0.3826 | 3 | 4 | 5 | 7 | × | × | × |
| Apertus | Corpus | Simulation | 0.5482 | 4 | 2 | 3 | 4 | ✓ | ✓ | ✓ |
| Apertus | Corpus | Detection | 0.4626 | 4 | 3 | 4 | 5 | ✓ | ✓ | × |
| Apertus | Corpus | Fuzzing | 0.3425 | 4 | 5 | 6 | 8 | × | × | × |
| Apertus | Corpus | Purity | 0.2274 | 4 | 8 | 11 | 14 | × | × | × |

Table 25 repeats Table 4 at different values of $\rho^* = 0.70$, $0.75$ and $0.80$. At $0.70$, the unreachable factor-model pairs are CORPUS, EXPLAINER and SCORER on PURITY for both models, plus CORPUS on FUZZING for `Apertus-8B-Instruct`(five needed, four available). At $0.75$, the three failures reported in the main text appear. At $0.80$, four further pairs become unreachable. Those are FUZZING under the EXPLAINER and under the CORPUS, and PURITY under EXAMPLES, on `Pythia-160M`, and DETECTION under the CORPUS on `Apertus-8B-Instruct`. We use $0.75$ because it is the boundary at which the paper interprets the reliability as good throughout, based on earlier work Koo & Li (2016).

## F SCORER FUZZING EXPERIMENT

For our third scorer, `gemma-3-27B`, we initially used the original fuzzing prompt Paulo et al. (2025). However, we noticed that there was a large difference between `gemma-3-27B` and the other two scorers. Table 26 shows the per-feature correlations between all judges, where it is clear that `gemma-3-27B` was the outlier.

Table 26: Pairwise per-feature Pearson correlations between the three scorers on FUZZING, using the original response format.

| Pair | *Pythia-160M* | *Apertus-8B-Instruct* |
|---|---|---|
| `gemini-2.5-flash` + `gemma-3-27B` | $-0.214$ | $-0.200$ |
| `gpt-4o-mini` + `gemma-3-27B` | $-0.207$ | $-0.294$ |
| `gemini-2.5-flash` + `gpt-4o-mini` | $+0.529$ | $+0.438$ |

The initial prompt presents marked token examples and asks for a binary judgment per token example. More specifically, "You must return your response in a valid Python list. Do not return anything else besides a Python list". See the full prompt in Section B.9.2. The final score is the balanced accuracy on all binary decisions.

Our original parser that worked for both `gemini-2.5-flash` and `gpt-4o-mini` aligned a short reply to positions 0,1,2... A judge that skips one judgment in the middle of the list can have every later judgment attributed to the wrong example. This caused issues for the final score of `gemma-3-27B`, as we observed that this scorer tends to skip an example more often than the other two scorers. Skipping an example led to a position shift in the returned list, which inverted every label after the gap on which the scores were computed. We fixed the response format to refuse unaligned replies and re-ran with the same parser.

We tried three options to find the correct response format for `gemma-3-27B`. They can be found in Table 27. The final option diverts from the first two, as an indexed return is requested.

Table 27: The three parser return options tried to find the error in the response of `gemma-3-27B`. The first option aligns with the original prompt and returns a List as a string. The second option only diverts from the original prompt that the return is of type List. The last option returns an indexed line per example.

| Parser return | Requested format |
|---|---|
| List (original prompt) | bare Python list (verbatim published prompt) |
| List (corrected) | bare Python list |
| Indexed | one `<index>: <0 or 1>` line per example |

Table 28 shows the results for the three scorers for the three variations we tested. They clearly show that `gemma-3-27B` performs below chance with the original response format. The last column shows an increase in the scores of both `gemma-3-27B` and `gpt-4o-mini`, on both `Pythia-160M` and `Apertus-8B-Instruct`. `gemma-3-27B` increases from 0.410 to 0.819 and from 0.444 to 0.724, while `gpt-4o-mini` increases from 0.634 to 0.806 and from 0.572 to 0.722. Initially, the `gemma-3-27B` results were the reason to investigate the matter, a byproduct

is the improvement of `gpt-4o-mini` scores. Interestingly, the results of `gemini-2.5-flash` do not increase or decrease. Additionally, revising the response format led to a higher per-feature correlation with all judges, shown in Table 29.

Table 28: The FUZZING scores for the three scorers on the three tested options. The results show that `gemini-2.5-flash` does not differ between the three options, but the other two, `gpt-4o-mini` and `gemma-3-27B`, both increase their scores, with the latter moving from below chance to similar to the other two scorers.

| Judge | List (original) | List (corrected) | Indexed |
|---|---|---|---|
| *Pythia-160M* | | | |
| `gemini-2.5-flash` | 0.789 | 0.790 | 0.789 |
| `gpt-4o-mini` | 0.634 | 0.634 | 0.806 |
| `gemma-3-27B` | 0.410* | no scorable batches | 0.819 |
| *Apertus-8B-Instruct* | | | |
| `gemini-2.5-flash` | 0.662 | 0.660 | 0.660 |
| `gpt-4o-mini` | 0.572 | 0.584 | 0.722 |
| `gemma-3-27B` | 0.444* | no scorable batches | 0.724 |

Table 29: Pairwise per-feature Pearson correlations between the three scorers on FUZZING, using the indexed response format. Values are the FUZZING rows of Table 21, computed on the features scored by both judges in each pair.

| Pair | *Pythia-160M* | *Apertus-8B-Instruct* |
|---|---|---|
| `gemini-2.5-flash` + `gemma-3-27B` | 0.806 | 0.701 |
| `gpt-4o-mini` + `gemma-3-27B` | 0.667 | 0.655 |
| `gemini-2.5-flash` + `gpt-4o-mini` | 0.614 | 0.468 |

These findings show that the requested response format can change what a judge reports. Absolute FUZZING scores are dependent on this response format and have to be considered carefully. Changing the format raises the inter-judge agreement.

For reproducibility, we added the following line to the prompt with a `.replace()` statement on the default, rather than a separate string. Everything else is identical.

**System prompt — fuzzing Replacement**

```
Default: "You must return your response in a valid Python list. Do not
     return anything else besides a Python list."

Indexed: "Output one line per example, in the form <index>: <0 or 1>,
    using the same index that labels the example in the input. Output
    exactly one line for every example you were given, in order, and
    nothing else."
```

## G SCORER ARCHITECTURE

To assess the influence of different judges on the scoring of different architectures, we performed a small experiment where we kept all other settings fixed but scored each of the five architectures with both `gemini-2.5-flash` and `gpt-4o-mini`. We opted for these two as `gemma-3-27B` was hosted by ourselves and considerably slower with inference.

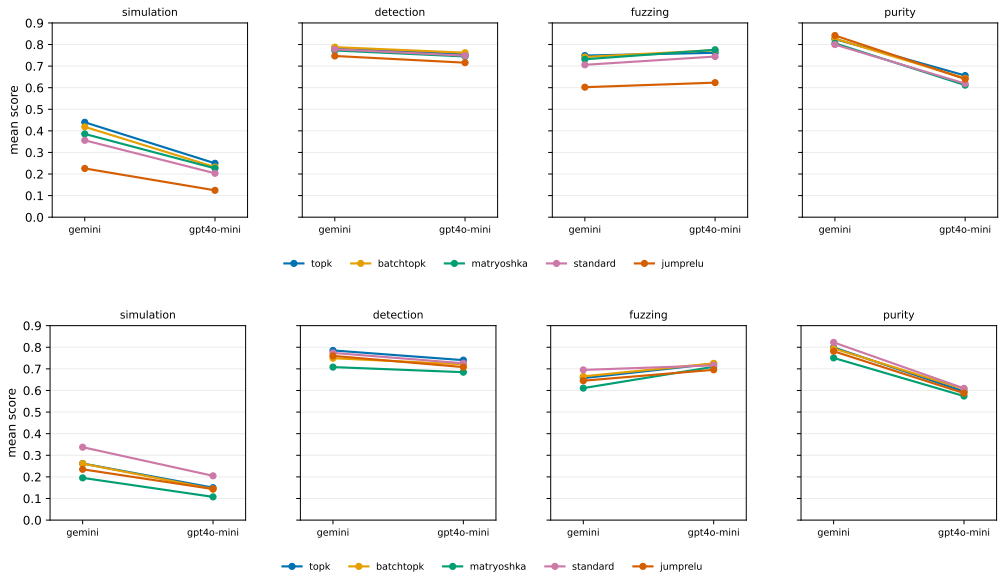


Figure 10: Architecture scorer slopes between the two closed-source scorers. The upper figure shows the metrics for `Pythia-160M`; the lower figure shows the metrics for `Apertus-8B-Instruct`.

In total, there are 80 pairs to be tested: five architectures $\times$ four metrics $\times$ two scorers $\times$ two source models. Of the 80 pairs, 67 pairs kept their order, while 13 changed when the scorer changed (Figure 11). Reversals are concentrated among close pairs, where the median separation is 0.011 versus 0.037 for pairs that kept their order. The largest reversed gap is 0.036, while the smallest gap that did not reverse is 0.0004. Eight of the 13 reversals are in FUZZING or PURITY, the two metrics on which the judges agree the least (Table 30), and in three cases the architecture of highest rank changed (see Table 2). The results indicate that score differences between SAE architectures smaller than 0.04 are not resolvable by a single scoring model. Which of the architectures ranks higher can be decided by the choice of grader and metric. Larger differences survived the change of judge in this experiment.

Additionally, if we zoom in on both scorer models, there is a difference in the range of overlap (Figure 12). The overlap range for `gpt-4o-mini` is smaller compared to `gemini-2.5-flash` . Under `gemini-2.5-flash` , the overlap range lies between 0.0004 and 0.036, while for `gpt-4o-mini` this range is 0.0001-0.016. Under `gpt-4o-mini`, the reversed pairs never exceed a gap of 0.016, making the overlap tighter. This indicates that the two classes are separated more cleanly. In general, the differences between architectures are less pronounced with `gpt-4o-mini`.

Looking at Figure 10, all the scores for the architectures are higher for `gemini-2.5-flash`, except for FUZZING. Additionally, a swap can occur in the highest ranked architecture. Three out of eight tested model-metric combinations change the highest scoring architecture based on the scorer (Table 2). This is an interesting finding, it shows that a different grader can provide a different answer, even if everything else is kept similar.

Table 30 shows per-feature agreement between two scorers on the same explanation. It shows that SIMULATION and DETECTION have a high correlation, indicating that the scorers rank the features similarly. The other two metrics, FUZZING and PURITY, have a lower correlation. The scorers disagree about which features are *good*.

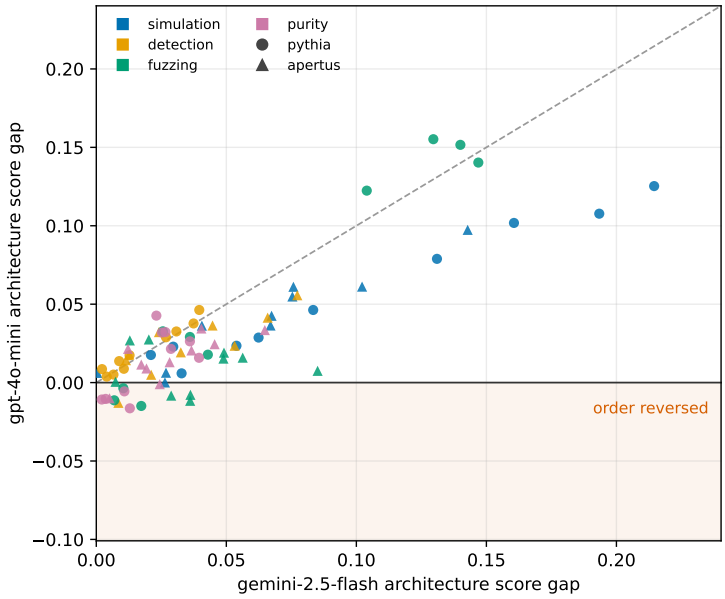


Figure 11: Score gap between each pair of architectures under the two scorers for the four metric for `Pythia-160M` and `Apertus-8B-Instruct`. Points above zero keep their order under both judges, while points in the shaded region are reversed. In total there are 13 pairs. The further below zero, the more strongly `gpt-4o-mini` prefers the architecture that `gemini-2.5-flash` ranked lower.

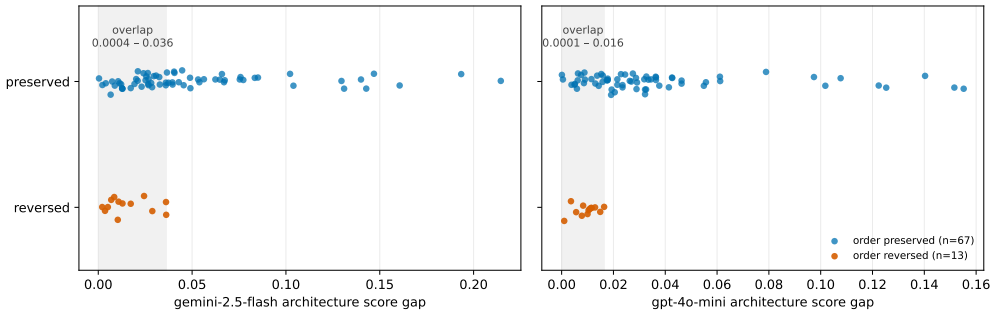


Figure 12: Score gap for between architecture pairs that keep their order (top, blue) and reverse their order (bottom, orange), when the scorer changes, pooled for both models and all four metrics.

Table 30: Per-feature agreement between `gemini-2.5-flash` and `gpt-4o-mini` for architecture scoring, pooled over the five architectures. Features differ per architecture.

| Model | Metric | Pearson $r$ | Median abs. diff | P90 abs. diff |
|---|---|---|---|---|
| `Pythia-160M` | SIMULATION | 0.859 | 0.138 | 0.422 |
| `Pythia-160M` | DETECTION | 0.820 | 0.050 | 0.130 |
| `Pythia-160M` | FUZZING | 0.538 | 0.100 | 0.347 |
| `Pythia-160M` | PURITY | 0.465 | 0.183 | 0.486 |
| `Apertus-8B-Instruct` | SIMULATION | 0.871 | 0.105 | 0.277 |
| `Apertus-8B-Instruct` | DETECTION | 0.772 | 0.060 | 0.140 |
| `Apertus-8B-Instruct` | FUZZING | 0.398 | 0.120 | 0.340 |
| `Apertus-8B-Instruct` | PURITY | 0.432 | 0.188 | 0.497 |

# H Additional Analysis

## H.1 Metric Correlations

To answer the question whether the four metrics measure the different aspects and if this depends on the condition, we investigate the relationship between them. The correlations can be found in Table 31.

Table 31: Pairwise metric correlations by model, averaged over factors.

| Pair | `Pythia-160M` | `Apertus-8B-Instruct` |
|---|---|---|
| SIMULATION ↔ FUZZING | 0.780 | 0.709 |
| DETECTION ↔ FUZZING | 0.684 | 0.488 |
| SIMULATION ↔ DETECTION | 0.653 | 0.486 |
| DETECTION ↔ PURITY | 0.554 | 0.531 |
| SIMULATION ↔ PURITY | 0.373 | 0.328 |
| FUZZING ↔ PURITY | 0.364 | 0.301 |

SIMULATION and FUZZING show a high correlation, with 0.71–0.78 being the strongest pair in both models. Given the higher cost of SIMULATION, FUZZING is a cheaper alternative. PURITY is the outlier. 0.30–0.37 against both SIMULATION and FUZZING indicates it is measuring something the others are not. Read that alongside PURITY being the least reliable metric (with a mean ICC 0.39–0.47), it does not track what the others measure. Reading Table 31, it shows that no metric trades off against another. There is no pair that is negatively correlated.

In addition, we can look at the metrics related to the different factors. The results in Table 32 show that scores move when a condition changes. On some architectures on `Pythia-160M`, SIMULATION and DETECTION are unrelated, while on others they correlate at 0.46. CORPUS and ARCHITECTURE drive this more compared to SCORER. The absence of other factors implies that they have a less divergent range of influence on scorer correlations.

Table 32: Ranges by factor, pair, and model. Only the widest correlations are included.

| Model | Factor | Pair | Min → Max | Range |
|---|---|---|---|---|
| `Pythia-160M` | Architecture | Simulation ↔ Detection | $-0.079 \rightarrow 0.463$ | 0.540 |
| | Corpus | Simulation ↔ Purity | $-0.053 \rightarrow 0.463$ | 0.516 |
| | Scorer | Simulation ↔ Fuzzing | $0.394 \rightarrow 0.812$ | 0.418 |
| | Corpus | Fuzzing ↔ Purity | $0.030 \rightarrow 0.448$ | 0.418 |
| | Architecture | Detection ↔ Fuzzing | $0.132 \rightarrow 0.528$ | 0.396 |
| | Corpus | Detection ↔ Purity | $0.213 \rightarrow 0.587$ | 0.374 |
| `Apertus-8B-Instruct` | Corpus | Simulation ↔ Purity | $-0.054 \rightarrow 0.534$ | 0.588 |
| | Corpus | Detection ↔ Purity | $0.251 \rightarrow 0.670$ | 0.419 |
| | Architecture | Simulation ↔ Detection | $0.225 \rightarrow 0.620$ | 0.395 |
| | Scorer | Detection ↔ Fuzzing | $0.119 \rightarrow 0.466$ | 0.347 |
| | Scorer | Simulation ↔ Fuzzing | $0.430 \rightarrow 0.742$ | 0.312 |
| | Corpus | Fuzzing ↔ Purity | $0.101 \rightarrow 0.408$ | 0.306 |

## H.2 Corpus Results

The experiment on `Pythia-160M` had $n = 513$ features with at least 190 activating samples across the four datasets, contrary to $n = 191$ for `Apertus-8B-Instruct`. Each individual corpus had between 2,400-3,700 qualifying features, but for this experiment, the features had to activate on all four at the same time. As mentioned previously, a feature can behave differently on `the Pile`, compared to `GitHub code`, which is a property of the feature.

Based on the mean scores per corpus (Table 33 and Table 34), GITHUB CODE is low on SIMULATION and FUZZING, but increases on PURITY. This could imply that metrics disagree on code features.

Another interesting remark about this finding is that metrics disagree on which corpus is easiest, which fits the "reliability is a property of metric-factor" insight. Additionally, the PILE is the SAE training distribution, but it is not the highest scoring corpus on any metric. Both C4 and WIKIPEDIA score higher on SIMULATION and FUZZING on both models. This might be contrary to what is expected from the in-distribution text.

Table 33: Mean scores for datasets on Pythia-160M ($n = 513$; 422 for PURITY)

| Metric | Pile | C4 | Wikipedia | Code | Range | Within/Between |
|---|---|---|---|---|---|---|
| Simulation | 0.522 | 0.566 | 0.558 | 0.346 | 0.221 | 0.47 |
| Detection | 0.754 | 0.760 | 0.786 | 0.752 | 0.034 | 0.64 |
| Fuzzing | 0.781 | 0.819 | 0.808 | 0.672 | 0.147 | 0.64 |
| Purity | 0.761 | 0.768 | 0.766 | 0.818 | 0.057 | 1.03 |

Table 34: Mean scores for datasets on Apertus-8B ($n = 191$; 166 for PURITY)

| Metric | Pile | C4 | Wikipedia | Code | Range | Within/Between |
|---|---|---|---|---|---|---|
| Simulation | 0.268 | 0.338 | 0.328 | 0.107 | 0.231 | 0.61 |
| Detection | 0.676 | 0.709 | 0.775 | 0.697 | 0.099 | 0.77 |
| Fuzzing | 0.661 | 0.731 | 0.741 | 0.522 | 0.219 | 0.90 |
| Purity | 0.713 | 0.706 | 0.754 | 0.846 | 0.138 | 1.05 |

Taking into account the reliability, considering the ICC values in Table 25, PURITY has the lowest values on the CORPUS factor. However, it also scores low on other metrics that have higher values with other factors.

Given the feature overlap in Table 19 and Table 20, each value is above 1.0, indicating non-random selection of features. Compared to some other factors, the lift values are small on both models for CORPUS. This indicates that if the top 10 features are selected on THE PILE, there is 1 feature overlapping from C4. The Jaccard values show similar patterns as the ICC, with SIMULATION having the highest and PURITY the lowest. This also means that the confidence intervals are wider for this model and experiment. As the number of features is lower for `Apertus-8B-Instruct` the number of overlapping features with lift increases to two to four.

Table 35 shows how far a feature moves when a corpus changes, against how far the features differ from each other. A ratio that is higher than one means that changing the corpus moves a single feature's score further away than the typical gap between features. `purity` exceeds one for both models, while `fuzzing` exceeds one for `Apertus-8B-Instruct`.

Table 35: The standard deviation per metric for the same features over different corpora in the "Within" condition and between features in the same corpus in the "Between" condition. Values above 1 are shown in bold.

| Metric | `Pythia-160M` | | | `Apertus-8B-Instruct` | | |
|---|---|---|---|---|---|---|
| | Within | Between | Ratio | Within | Between | Ratio |
| simulation | 0.1953 | 0.2839 | 0.69 | 0.1853 | 0.2222 | 0.83 |
| detection | 0.0972 | 0.1151 | 0.84 | 0.1019 | 0.1056 | 0.97 |
| fuzzing | 0.1594 | 0.1732 | 0.92 | 0.1812 | 0.1519 | **1.19** |
| purity | 0.1846 | 0.1424 | **1.30** | 0.1880 | 0.1330 | **1.41** |

A feature that activates on all four datasets can be viewed as being less specific. It is expected that there is a higher probability that a feature activates on datasets that are alike, such as the PILE and C4, than datasets that are not alike, such as PILE and GITHUB CODE.

### H.3 Draw Trajectories

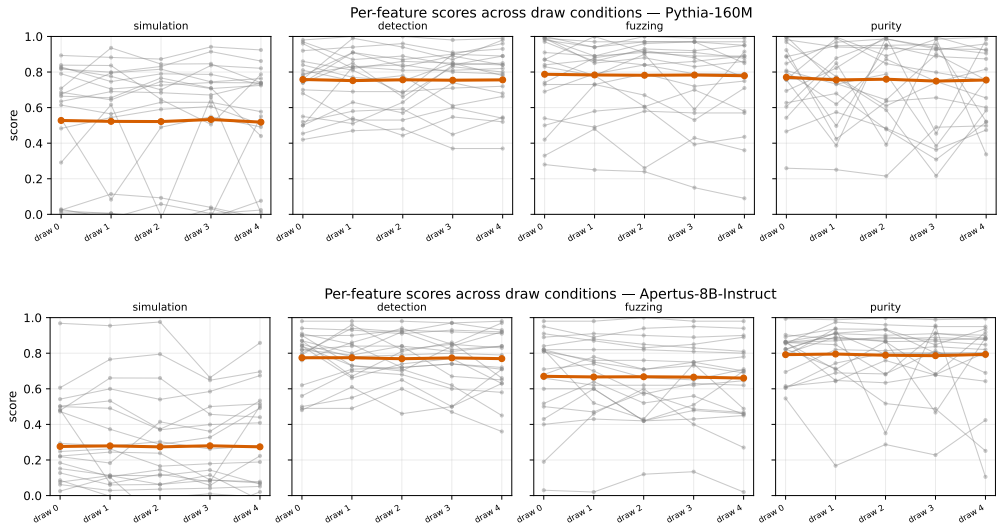


Figure 13: Per-feature score trajectories for the five draws across 20 random features for `Pythia-160M` (top) and `Apertus-8B-Instruct` (bottom). The orange line is the mean over all features. Mean scores are nearly constant across draws, while individual features move.

### H.4 Paraphrase Trajectories

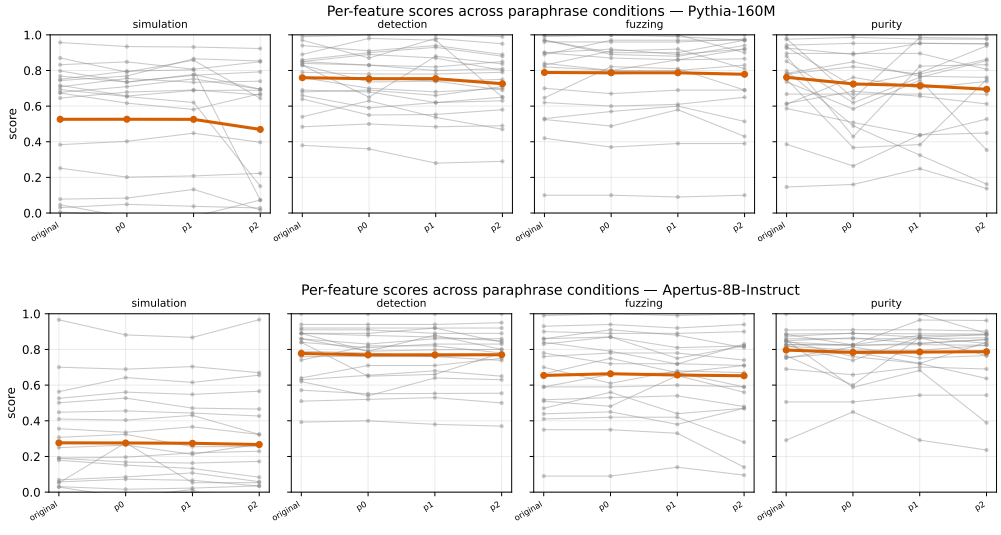


Figure 14: Per-feature score trajectories for the four paraphrases across 20 random features for `Pythia-160M` (top) and `Apertus-8B-Instruct` (bottom). The orange line is the mean over all features.

### H.5 Explainer Sensitivity

We investigated the differences between the three explainers, by looking at the average length of the explanations, the number of words, and the quotes (see Table 36). We saw that `gemini-2.5-flash` writes explanations that are half the length of the other two (165 vs. 325/331 characters). This pattern is similar in `Pythia-160M` and `Apertus-8B-Instruct`.

This is an interesting finding, as a longer explanation can provide the scorer with more information to

match against. A hypothesis was that a longer explanation would lead to a better score. To test this, we calculated the Pearson correlation coefficient between score and explanation length. The results are shown in Table 37. There are mostly negative values in the table. This indicates the inverse: instead of a longer explanation leading to a higher score, longer explanations got worse scores.

A potential reason for this can be the complexity of the feature itself. The length of the explanation can be a marker of difficulty, as the explainer cannot summarize the activating samples of the feature in fewer words. All explainers received the same prompt to construct the explanation.

We leave further investigation of length within an explainer up to future work.

Table 36: Statistics on the explanations given by the three explainers.

| Model | Explainer | Chars (mean $\pm$ sd) | $p_{10}$–$p_{90}$ | Words | % with quotes |
|---|---|---|---|---|---|
| `Pythia-160M` | `gemini-2.5-flash` | $165 \pm 33$ | 120–208 | 23.0 | 71% |
| | `gemma-3-27B` | $325 \pm 38$ | 281–373 | 41.6 | 51% |
| | `gpt-4o-mini` | $331 \pm 40$ | 283–384 | 43.1 | 40% |
| `Apertus-8B-Instruct` | `gemini-2.5-flash` | $174 \pm 33$ | 133–218 | 23.0 | 33% |
| | `gemma-3-27B` | $314 \pm 38$ | 268–365 | 40.2 | 34% |
| | `gpt-4o-mini` | $322 \pm 41$ | 268–380 | 42.0 | 28% |

Table 37: Pearson correlation between scores and explanation length for each of the three explainers. A '*' indicates that $p < 0.05$.

| Model | Explainer | Simulation | Detection | Fuzzing | Purity |
|---|---|---|---|---|---|
| `Pythia-160M` | `gemini-2.5-flash` | $-0.305^*$ | $-0.237^*$ | $-0.265^*$ | $-0.006$ |
| `Pythia-160M` | `gemma-3-27B` | $-0.123^*$ | $-0.212^*$ | $-0.166^*$ | $-0.131^*$ |
| `Pythia-160M` | `gpt-4o-mini` | $+0.100^*$ | $+0.046$ | $+0.063$ | $+0.006$ |
| `Apertus-8B-Instruct` | `gemini-2.5-flash` | $-0.100^*$ | $-0.167^*$ | $-0.076$ | $-0.065$ |
| `Apertus-8B-Instruct` | `gemma-3-27B` | $-0.100^*$ | $-0.190^*$ | $-0.032$ | $-0.062$ |
| `Apertus-8B-Instruct` | `gpt-4o-mini` | $-0.053$ | $-0.193^*$ | $+0.004$ | $-0.085$ |

### H.6 Explainer Samples

Similarly to previous work, we looked briefly at the influence of the number of samples shown to the explainer for the initial explanation Paulo et al. (2025). The trend shows similarities with earlier work. There is an improvement up to 40 samples. After this number of samples, there is not a large increase in the mean scores. Taking into account the upper part of Figure 15, SIMULATION appears to benefit the most from the increase from 10 to 40 samples. This effect is present, but less visible, in `Apertus-8B-Instruct` (see lower part of Figure 15). Other metrics show less of a jump, with PURITY even decreasing when the number of examples moves from 40 to 80.

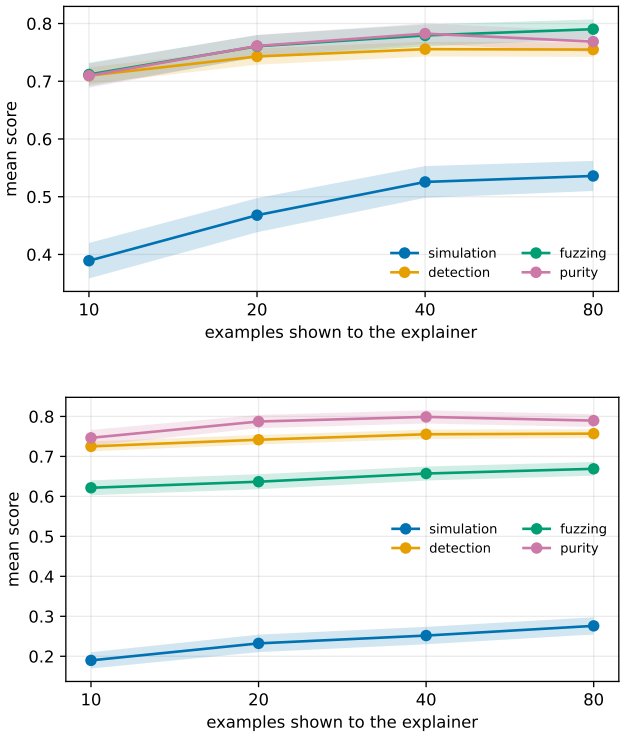


Figure 15: Mean score by number of examples shown to the explainer for `Pythia-160M` (top) and `Apertus-8B-Instruct` (bottom). Scores rise up to 40 examples and change little beyond, with SIMULATION benefiting most.

# I TIME AND COMPUTE OF EXPERIMENTS

The following estimates are for a single run, where no debugging or calibration is needed. The baseline assumptions are the following:

- One baseline model and one SAE, with roughly 500 features.
- 1 GH200 node per job
- Scoring via LLM API
- 4 metrics: simulation, detection, fuzzing and purity

There are two stages in the pipeline. The first stage collects the activations and caches them (Table 38). This is the stage that uses the GPU, as this is required for the model activations.

Table 38: Node-hour cost breakdown for the first stage of the pipeline

| Stage | Node-h | Explanation |
|---|---|---|
| Corpus activation cache (1 corpus) | 2–3 | ∼4x that for the 4-corpus experiment |
| Per-token activations | ∼2.5 | needed for fuzzing's token marking |
| **Subtotal** | **5–6** | per model+SAE, reusable across every experiment below |

Table 39: Compute budget by experiment

| Experiment | Conditions | Calls/feature | Node-h | Node-h w/o simulation |
|---|---|---|---|---|
| Exp1 corpus | 4 corpora | ∼248 | 15 | 3 |
| Exp2a draw | 5 draws | ∼310 | 10 | 2 |
| Exp2b explainer | 3 explainers | ∼186 | 10 | 2 |
| Exp2c paraphrase | 4 paraphrases | ∼248 | 12 | 2 |
| Exp2d scorer | 3 scorers | ∼183 | 14 | 3 |
| Exp2e examples | 4 counts | ∼248 | 9 | 2 |
| **Total** | | **∼1,400** | **∼70** | **∼16** |

The second stage consists of the experiments, which are shown in Table 39. We show an estimate of hours for each metric with and without SIMULATION in Table 40. These estimates cover a single clean run (∼70 node-hours for all experiments on one model); the ∼350 node-hours reported in the ethics statement include both models, the architecture experiment, re-runs, and debugging.

SIMULATION is the most expensive metric. In total, there would be around 16 calls to the API without simulation, but 66 with (simulation costs around 50, one call per sequence, and cannot batch). simulation is over 80% of the total spend.

The numbers for SIMULATION show the reason for excluding this metric from the SCORER experiment, as our `gemma-3-27B` was self-hosted and had a high latency, which resulted in excessive node hours.

The estimates for the API calls are derived from code paths and observed run logs for the features (Table 40). Per feature, per condition, there are 50 positive + 50 negative sequences. We have not included the costs for hosting `gemma-3-27B`.

Table 40: API call budget by metric

| Metric | Calls | Why | Input tok | Output tok |
|---|---|---|---|---|
| detection | 5 | 100 sequences / batch 20 | ∼4,300 | ∼200 |
| fuzzing | 5 | 100 marked examples / batch 20 | ∼4,300 | ∼200 |
| purity | ∼5 | FADE rating, batched 20 | ∼4,300 | ∼200 |
| simulation | ∼50 | 1 call per sequence, cannot batch | ∼12,500 | ∼3,000 |
| Explanation | 1 | one per condition | ∼1,500 | ∼150 |
| **Total with simulation** | **∼66** | | **∼27,000** | **∼3,750** |
| **Total without simulation** | **∼16** | | **∼14,400** | **∼750** |

# J QUALITATIVE EXAMPLES

## J.1 DRAW

We have included several examples of the DRAW condition below. There are features where most draws agree, while this can be different for others. However, if one would consider only the range of similarity between the explanations (0.835-0.885), the conclusion could be that most explanations of the draws are high in similarity.

**Pythia, feature 225** (explanation similarity across draws: 0.835)

**Baseline** (sim +0.66 det +0.67 fuzz +0.90 purity +0.50) The letter sequence "el" or "EL", often as a prefix or within words, particularly in proper nouns, technical terms, or names.

**Draw 1** (sim +0.69, det +0.85, fuzz +1.00, purity +0.68)
*Words or names containing the letter sequence "el" or "el-" (e.g., "el.", "mercurial.el", "ELF", "eladgil", "Neel".*

**Draw 2** (sim +0.02, det +0.51, fuzz +0.50, purity +0.48)

*Formal or technical documents, including scientific papers, legal texts, code, and business reports, often featuring specific terminology, citations, or structured formatting.*

**Draw 3** (sim +0.66, det +0.68, fuzz +0.96, purity +0.31)
*The token sequence "el" or "EL", often as a prefix in words like "electroluminescence", "ElasticSearch", "Eloquent", or in names like "Elie".*

**Draw 4** (sim +0.02, det +0.46, fuzz +0.47, purity +0.52)
*Technical or formal text, often including specific terminology, programming languages, file extensions, acronyms, job titles, or question-and-answer formats.*

**Analysis.** Two draws and the baseline find an orthographic feature, two find a TOPIC, which are incompatible hypotheses. SIMULATION differs with the lexical reading (+0.67 vs +0.02) on the identical fixed test set.

**Pythia, feature 5550** (explanation similarity across draws: 0.885)

**Baseline** (sim +0.48 det +0.76 fuzz +0.86 purity +0.23) Phrases that combine or contrast two related concepts, often using prepositions or adverbs joined by "and" or "or" (e.g., "with and without," "before

**Draw 1** (sim +0.08, det +0.43, fuzz +0.79, purity +0.27)
*Scientific or technical text, often from patents, research papers, or code, that describes a subject, method, or comparison using phrases like "relates to," "with and without," "before"...*

**Draw 2** (sim +0.49, det +0.81, fuzz +0.97, purity +0.51)
*Pairs of related or contrasting words or phrases connected by "and" or "or", often forming common collocations or parallel structures.*

**Draw 3** (sim +0.55, det +0.79, fuzz +0.94, purity +0.66)
*Conjunctions "and" or "or" connecting two similar grammatical elements, often in a list, pair, or phrase like "X and Y", "X or Y"...*

**Draw 4** (sim +0.49, det +0.77, fuzz +0.97, purity +0.39)
*Phrases indicating a range or pair of related concepts, often using "and", "or", "in/out", "before/after", "with/without", or lists of items.*

**Analysis.** Three of four draws and the baseline agree it is the conjunctions "and"/"or". Draw 1 alone reads it as generic technical text and scores far below the rest.

**Apertus, feature 62400** (explanation similarity across draws: 0.847)

**Baseline** (sim -0.06 det +0.48 fuzz +0.46 purity +0.66) Specific entities, names, and titles, often appearing in lists, citations, product descriptions, or biographical contexts. This includes personal names, product names, book/movie titles, email addresses, and software components

**Draw 1** (sim -0.08, det +0.52, fuzz +0.43, purity +0.11)
*Specific, detailed information about a particular subject, often including proper nouns, technical terms, or descriptive phrases.*

**Draw 2** (sim -0.04, det +0.49, fuzz +0.35, purity +0.48)
*Boilerplate introductory text or structured prompts, including code license headers (e.g., starting with /*) and technical questions (e.g., prefixed with Q:), often found at the beginning of documents or discussions.*

**Draw 3** (sim +0.46, det +0.68, fuzz +0.74, purity +0.71)
*The preposition "with" and phrases indicating accompaniment, inclusion, or instrumentality.*

**Draw 4** (sim -0.01, det +0.50, fuzz +0.43, purity +0.62)
*Text that presents factual or descriptive information in a structured or formal manner, such as software license headers, product specifications, news reports, and academic abstracts.*

**Analysis.** One draw in four finds the real feature (the preposition 'with', most

metrics are highest compared to other descriptions). The other three describe surface text type and score at chance.

**Apertus, feature 58616** (explanation similarity across draws: 0.857)

**Baseline** (sim +0.13 det +0.67 fuzz +0.50 purity +0.75) Descriptions of people, characters, or community members, often in the context of social interaction, roles, or personal attributes.

**Draw 1** (sim +0.10, det +0.63, fuzz +0.59, purity +0.43)
*Human interaction, community dynamics, and personal experiences shared within social or collaborative contexts, frequently appearing in forums, blogs, or professional discussions.*

**Draw 2** (sim -0.14, det +0.37, fuzz +0.06, purity –)
*Initial framing or introduction of a person, topic, or legal/social context, frequently found at the beginning of a text block, conversation, or code file.*

**Draw 3** (sim -0.09, det +0.47, fuzz +0.17, purity +0.78)
*Text that introduces or frames content, often acting as a greeting, question, or metadata within online communities, forums, or source code documentation.*

**Draw 4** (sim +0.02, det +0.82, fuzz +0.61, purity +0.75)
*References to human or organizational entities, their relationships, attributes, or legal rights, encompassing social interactions, personal opinions, and copyright declarations.*

**Analysis.** All four explanations vaguely include mentions of "people", "person", "community" or "human", yet detection ranges 0.37 to 0.82.

### J.2 EXPLAINER

**Pythia, feature 5595** (explanation similarity, gemini vs. gpt-4o-mini: 0.789)

**gemini** (sim +0.77, det +0.98, fuzz +1.00, purity +0.88)
*The letter 'Z' (uppercase or lowercase), often at the beginning of a word, as a variable, or within a technical term.*

**gpt** (sim +0.17, det +0.41, fuzz +0.63, purity +0.61)
*Feature 5595 detects content related to specific individuals, organizations, or products, often accompanied by contextual information such as academic qualifications, health topics, or product descriptions. The token-level patterns include proper nouns, specific terminology, and structured information typical of informational or promotional texts.*

**gemma** (sim +0.04, det +0.49, fuzz +0.60, purity +0.18)
*Technical documentation and code snippets, often relating to computing or scientific fields. The feature frequently activates on text containing programming languages (PHP, C++, Python), software names (Zsh, Zend), file extensions (XML), or technical descriptions of processes and materials (molecular properties, zeolites).*

**Analysis.** From the same 40 examples, gemini-2.5-flash describes a letter and the other two explainers describe topics. gemma-3-27b even lists Zsh, Zend and zeolites but still reads them as a topic. Only the letter-level explanation predicts activations (simulation 0.77 against 0.17 and 0.04). The embedding similarity of 0.79 does not show that these are incompatible hypotheses.

**Pythia, feature 14309** (explanation similarity, gemini vs. gpt-4o-mini: 0.861)

**gemini** (sim +0.70, det +1.00, fuzz +1.00, purity +1.00)
*The concept of "order" in various contexts, including sequence, arrangement, sorting, commands, official decrees, commercial transactions, and the phrase "in order to".*

**gpt** (sim +0.52, det +0.45, fuzz +0.73, purity +0.27)
*Order and organization in various contexts, including programming constructs, database queries, and structured data management, are detected by*

*this feature. It captures discussions around the arrangement, sequencing, and processing of elements, whether in code, data queries, or conceptual frameworks.*

**gemma** (sim +0.71, det +0.65, fuzz +0.64, purity +0.55)
*Requests for information and discussions about ordering, sorting, or ranking – both generally and within specific systems like databases, ontologies, or code. The feature frequently activates on questions containing "order," "ordering," or related terms, often in the context of technical queries or formal documentation.*

**Analysis.** All three explainers name the same concept, "order". gpt-4o-mini and gemma-3-27b tie it to technical and database contexts, and their detection scores fall from 1.00 to 0.45 and 0.65. Simulation barely moves (0.52-0.71). The explanations agree on the concept but not on its scope, and how much that costs depends on the metric.

**Apertus, feature 28264** (explanation similarity, gemini vs. gpt-4o-mini: 0.799)

**gemini** (sim +0.60, det +0.65, fuzz +0.93, purity +0.80)
*Concepts of movement, travel, or progression, including physical actions like running, walking, driving, and marching, as well as metaphorical movement or the movement of objects and particles.*

**gpt** (sim -0.05, det +0.44, fuzz +0.77, purity +0.30)
*Emotional and existential themes in narratives, particularly those involving isolation, loss, and personal struggles, are detected by this feature. It also captures discussions around practical applications and technical inquiries, often in the context of personal experiences or societal issues.*

**gemma** (sim +0.62, det +0.83, fuzz +0.96, purity +0.89)
*Movement and travel, both physical locomotion and directed progression through a process or system. This includes descriptions of walking, running, chases, journeys, mechanical movement, and even abstract "walking" through steps in a procedure or data.*

**Analysis.** gemini-2.5-flash and gemma-3-27b both describe movement and get almost the same simulation score (0.60 and 0.62). gpt-4o-mini describes an unrelated theme. Its simulation drops to chance ($-0.05$) and detection falls below chance (0.44), but fuzzing stays at 0.77.

**Apertus, feature 16193** (explanation similarity, gemini vs. gpt-4o-mini: 0.897)

**gemini** (sim +0.84, det +0.98, fuzz +0.96, purity +0.76)
*The word "tool" or "tools," particularly when referring to software utilities, programming toolkits, development environments, or general implements and mechanisms.*

**gpt** (sim +0.51, det +0.76, fuzz +1.00, purity +0.47)
*Tools and tool-related concepts, particularly in technical and programming contexts, are detected. The text often includes references to software tools, development tools, and various applications or inventions that utilize tools for specific functions or tasks.*

**gemma** (sim +0.13, det +0.60, fuzz +0.69, purity +0.58)
*Software development resources and tools, including code snippets, API documentation, build systems, and related question/answer discussions. The feature frequently activates on text containing programming languages (C#, Javascript, XML), software names (Git, jQuery, OpenFOAM), or references to developer utilities and workflows.*

**Analysis.** The three explanations go from the word itself, to the word in technical contexts, to the software-development topic where the word usually appears. Simulation follows that shift from token to context: 0.84, then 0.51, then 0.13. Both fuzzing and detection decrease with gemma-3-27b, while purity does not.

### J.3 PARAPHRASE

**Pythia, feature 15992**

**Original** (sim +0.81, det +0.60, fuzz +0.95, purity +0.76)
*The token "real" and its common collocations, particularly "real estate" and "real-time," across various domains including finance, technology, mathematics, and discussions of authenticity or actuality.*

**1** (similarity 0.990; sim +0.82, det +0.83, fuzz +0.95, purity +0.96)
*The token "real" appears in various domains, such as finance, technology, mathematics, and discussions about authenticity or actuality, alongside its common collocations, notably "real estate" and "real-time."*

**2** (similarity 0.932; sim +0.81, det +0.90, fuzz +0.95, purity +0.93)
*The term "real" along with its frequently associated phrases, especially "real estate" and "real-time," is utilized in multiple fields such as finance, technology, mathematics, and conversations surrounding authenticity or reality.*

**3** (similarity 0.922; sim +0.00, det +0.58, fuzz +0.95, purity +0.25)
*The term "genuine" and its frequent pairings, especially "genuine property" and "genuine-time," in multiple areas such as finance, technology, mathematics, and conversations regarding authenticity or reality.*

**Analysis.** Reordering the sentence and changing its register in the first paraphrase leave simulation unchanged (0.81-0.82) but raise detection from 0.60 to 0.83 and 0.90. The third rewrite swaps out the token the feature responds to, giving "genuine property" and "genuine-time". Simulation drops to 0.00 and purity to 0.25, while fuzzing stays at 0.95. Its similarity to the original (0.922) stays almost the same as the second rewrite's (0.932), so embedding similarity does not show that the explanation now names a different token.

**Pythia, feature 4812**

**Original** (sim +0.83, det +0.98, fuzz +0.94, purity +1.00)
*The word "character" in its various senses, including a textual unit, a fictional entity, or a mathematical concept, and the related term "characterization" in a scientific or analytical context*

**Structure** (similarity 0.969; sim +0.79, det +0.62, fuzz +0.85, purity +0.99)
*In its different meanings, the term "character" refers to a textual unit, a fictional entity, or a mathematical concept, while "characterization" pertains to its usage in a scientific or analytical context.*

**Register** (similarity 0.974; sim +0.86, det +0.89, fuzz +0.93, purity +0.99)
*The term "character," which may refer to different meanings such as a unit of text, a fictional figure, or a mathematical idea, along with the associated term "characterization" used in scientific or analytical contexts.*

**Vocabulary** (similarity 0.967; sim +0.77, det +0.91, fuzz +0.93, purity +0.99)
*The term "character" in its different meanings, such as a textual element, a fictional being, or a mathematical idea, along with the associated term "characterization" in a scientific or analytical framework.*

**Analysis.** All four versions describe the same feature, and simulation stays between 0.77 and 0.86. The structural rewrite only reorders the sentence, yet its detection score drops from 0.98 to 0.62. The other three versions score 0.89-0.98.

**Apertus, feature 15629**

**Original** (sim +0.93, det +0.98, fuzz +0.97, purity +0.91)
*The verb "involve" and its various forms (involves, involved, involving, involvement), indicating participation, inclusion, or a necessary component within a process, system, or event.*

**1** (similarity 0.994; sim +0.94, det +0.98, fuzz +0.97, purity +0.91)
*The various forms of the verb "involve" (involves, involved, involving, involvement) indicate participation, inclusion, or a necessary component within a process, system, or event.*

**2** (similarity 0.948; sim +0.94, det +0.98, fuzz +0.97, purity +0.91)
*The term "involve" and its different variations (involves, involved, involving, involvement) refer to the concepts of participation, inclusion, or an essential element within a process, system, or event.*

**3** (similarity 0.899; sim +0.83, det +0.49, fuzz +0.69, purity +0.88)
*The term "engage" and its different variations (engages, engaged, engaging, engagement), signifying participation, incorporation, or an essential element within a procedure, framework, or occurrence.*

**Analysis.** The first and second rewrites score within 0.01 of the original on every metric, with similarity score varying with 0.04. The vocabulary rewrite replaces "involve" with "engage". Detection falls from 0.98 to chance (0.49) and fuzzing from 0.97 to 0.69, but simulation only drops from 0.93 to 0.83, while similarity drops 0.095.

**Apertus, feature 13843**

**Original** (sim +0.17, det +0.87, fuzz +0.67, purity +0.74)
*Legal documents such as court opinions, judgments, and case citations, often accompanied by formal disclaimers like "NOT FOR PUBLICATION" or "UNPUBLISHED".*

**1** (similarity 0.983; sim +0.12, det +0.83, fuzz +0.68, purity +0.54)
*Court opinions, judgments, and case citations are examples of legal documents that are frequently accompanied by formal disclaimers such as "NOT FOR PUBLICATION" or "UNPUBLISHED".*

**2** (similarity 0.987; sim +0.14, det +0.59, fuzz +0.68, purity +0.54)
*Legal documents, which include court opinions, judgments, and case citations, are frequently accompanied by formal disclaimers such as "NOT FOR PUBLICATION" or "UNPUBLISHED."*

**3** (similarity 0.965; sim +0.19, det +0.86, fuzz +0.78, purity +0.61)
*Legal paperwork including judicial opinions, rulings, and case references, frequently accompanied by official disclaimers such as "NOT FOR PUBLICATION" or "UNPUBLISHED".*

**Analysis.** The second rewrite is the closest to the original (similarity 0.987), yet its detection drops from 0.87 to 0.59. The third rewrite is the furthest from the original and scores 0.86. How close a rewrite is to the original in embedding space does not predict the the detection score.

### J.4 Examples

**Pythia, feature 10580**

**10** (sim -0.02, det +0.35, fuzz +0.75, purity +0.24)
*Technical documentation, programming questions, software versioning, and academic or research-related text, often including specific formatting like "Q:", code comments, or XML tags.*

**20** (sim +0.77, det +0.97, fuzz +0.95, purity +0.99)
*Words related to academic levels like "graduate," "undergraduate," and "grade," or terms indicating improvement or change such as "upgrade" and "gradient."*

**40** (sim +0.79, det +0.87, fuzz +0.92, purity +0.97)
*The words "graduate," "graduation," and "upgrade" in various forms, particularly in academic or technical contexts. It also detects "grade" when referring to academic levels or classifications.*

**80** (sim +0.84, det +0.97, fuzz +0.96, purity +1.00)
*The morpheme "grad-" and its variations, encompassing concepts related to academic graduation, student levels (undergraduate, graduate), academic grades, mathematical gradients, and software/system upgrades.*

**Analysis.** With 10 examples, the explainer describes the kinds of documents the

examples came from, and simulation and detection are at chance (sim $-0.02$, det 0.35). From 20 examples on, every explanation names the "grad" token and becomes more precise with more examples. At 20 and 40 it lists specific words, and at 80 it names the morpheme "grad-". Simulation rises from 0.77 to 0.84. Most scores remain relatively similar.

**Pythia, feature 12241**

**10** (sim +0.39, det +0.98, fuzz +0.75, purity +0.95)
*Statements attributing information, actions, or compliance to a specific source, standard, or agreement, often using phrases like "according to", "licensed to", or "complies with".*

**20** (sim -0.02, det +0.49, fuzz +0.25, purity +0.84)
*Technical documentation, specifications, and reports, often including patent language, programming code comments, or news articles citing official sources.*

**40** (sim +0.45, det +0.78, fuzz +0.76, purity +0.95)
*Statements of conformance to rules, standards, or instructions, or attribution of information to a source, frequently using phrases like "according to" or "in accordance with".*

**80** (sim +0.48, det +0.81, fuzz +0.75, purity +0.85)
*Phrases indicating something is done, judged, or described based on a specific standard, rule, source, or method, often using "according to" or "in accordance with".*

**Analysis.** The 10-example explanation already identifies the feature (detection of 0.98). The 20-example explanation, which saw those 10 examples plus 10 more, goes back to describing document types, and detection falls to 0.49 and fuzzing to 0.25. At 40 and 80 examples, the explanation returns to "according to" / "in accordance with", which is more similar to the 10 example explanation.

**Apertus, feature 4693**

**10** (sim -0.07, det +0.48, fuzz +0.01, purity +0.71)
*The beginning of a structured text, such as an article, blog post, news report, or formal document, often marked by titles, headings, introductory sentences, or calls to action.*

**20** (sim -0.04, det +0.44, fuzz +0.08, purity +0.33)
*Introductory or summary text for articles, blog posts, abstracts, and other structured documents, frequently appearing as a title, subtitle, or opening paragraph.*

**40** (sim +0.71, det +0.93, fuzz +0.97, purity +0.95)
*Explicit negation, including words like "not", "no", "don't", "never", and the "not only... but also" construction.*

**80** (sim +0.75, det +0.92, fuzz +0.97, purity +0.96)
*The concept of negation or denial, frequently indicated by words such as "not", "don't", "never", "no", "isn't", and often appearing in "not only... but also" constructions.*

**Analysis.** With 10 and 20 examples, the explainer describes where in the document the examples appear (openings and titles), and fuzzing is near zero (0.01 and 0.08). At 40 examples it identifies negation, and every metric jumps (sim 0.71, det 0.93, fuzz 0.97). Going to 80 examples adds nothing to the metrics.

**Apertus, feature 15242**

**10** (sim -0.01, det +0.59, fuzz +0.33, purity +0.67)
*Text from structured, formal contexts like Q&A forums (often marked with "Q:"), legal documents, or technical specifications and boilerplate clauses.*

**20** (sim -0.02, det +0.61, fuzz +0.29, purity +0.67)
*Technical questions, frequently introduced by "Q:", and formal technical or legal documentation such as patent descriptions and code copyright blocks.*

**40** (sim -0.03, det +0.59, fuzz +0.31, purity +0.52)
*Technical questions and problem descriptions, often explicitly marked with "Q:", or formal disclosures and specifications of inventions, software, and systems.*

**80** (sim +0.85, det +0.81, fuzz +0.99, purity +0.68)
*Anaphoric references to non-human entities or abstract concepts, frequently using pronouns like "it" and "them", or pronominal adverbs such as "thereof", "therefrom", and "thereon".*

**Analysis.** With 10, 20 and 40 examples, the explainer describes Q&A and legal or technical documents, and simulation stays at chance. Only with 80 examples does it find the real pattern with "it", "them" and "thereof", and simulation jumps to 0.85. At the standard budget of 40 examples, this feature would be recorded as uninterpretable.