Interpretable Embeddings with Sparse Autoencoders: A Data Analysis Toolkit

Nick Jiang[1*] **Xiaoqing Sun**[2*] **Lisa Dunlap**[1] **Lewis Smith** **Neel Nanda**
[1]University of California, Berkeley [2]Massachusetts Institute of Technology

Abstract

Analyzing large-scale text corpora is a core challenge in machine learning, crucial for tasks like identifying undesirable model behaviors or biases in training data. Current methods often rely on costly LLM-based techniques (e.g. annotating dataset differences) or dense embedding models (e.g. for clustering), which lack control over the properties of interest. We propose using sparse autoencoders (SAEs) to create *SAE embeddings*: representations whose dimensions map to interpretable concepts. Through four data analysis tasks, we show that SAE embeddings can find novel data insights while offering the controllability that dense embeddings lack and costing less than LLMs. By computing statistical metrics over our embeddings, we can uncover insights such as (1) semantic differences between datasets and (2) unexpected concept correlations in documents. For example, by comparing model responses, we find that Grok-4 clarifies ambiguities more often than nine other frontier models. Relative to LLMs, SAE embeddings uncover bigger differences at 2-8× lower cost and identify biases more reliably. Additionally, SAE embeddings are controllable: by filtering concepts, we can (3) cluster documents along axes of interest and (4) outperform dense embeddings on property-based retrieval. Using SAE embeddings, we study model behavior with two case studies: investigating how OpenAI model behavior has changed over new releases and finding a learned spurious correlation from Tulu-3's [1] training data. These results position SAEs as a versatile tool for unstructured data analysis and highlight the neglected importance of interpreting models through their *data*.[1]

1 Introduction

Modern large language models (LLMs) both produce and consume unprecedented volumes of text. Analyzing this data at scale is important—e.g., for finding unexpected model behaviors [2] or biases in training data—making textual data analysis a pressing area of research, especially for model-related data. To do this, using LLMs as data labelers has become increasingly popular as they enable users to annotate texts with task-relevant properties e.g., toxicity, formality [3; 4]. However, this approach becomes expensive at scale and can be prompt-sensitive [5; 6]. Dense embeddings [7] enable fast similarity-based analysis but offer little interpretability or control over specific properties.

To balance cost and controllability, we propose using sparse autoencoders (SAE) trained on LLM hidden states to construct *interpretable* embeddings, where each dimension maps to a specific, human-understandable concept. SAEs have emerged as a key unsupervised method within mechanistic interpretability, decomposing LLM activations into monosemantic directions [8; 9; 10]. We hypothesize that SAEs are useful for analyzing data—by passing in text through a "reader" LLM and capturing its SAE activations, the SAE effectively labels text with the thousands of concepts encoded in its activations at once (Figure 1). We show the versatility of these SAE embeddings on four tasks:

*Equal contribution. Correspondence to: nickj@berkeley.edu, xqsun@mit.edu.
[1]Code: https://github.com/nickjiang2378/interp_embed

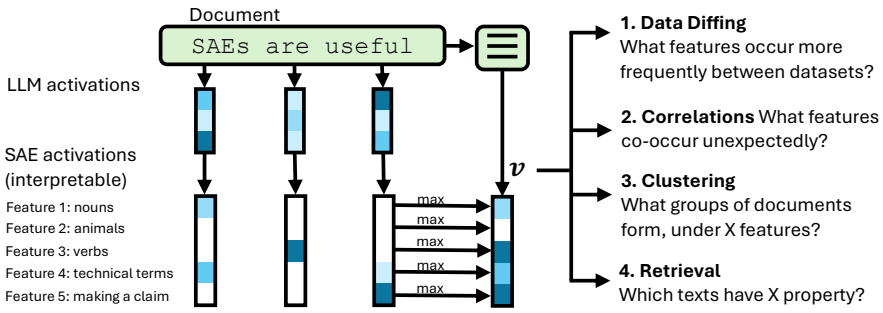

Figure 1: **Converting text documents into interpretable embeddings with sparse autoencoders.** We feed each document into a "reader LLM" and use a pretrained SAE to generate feature activations. Then, we aggregate (max-pool or binarize) activations across tokens, producing a single embedding where each dimension maps to a human-understandable concept. The interpretable nature of this embedding allows us to perform a diverse range of downstream data analysis tasks.

1. **Dataset diffing**: SAEs can describe differences between datasets, identifying semantic and syntactic properties with larger frequency differences at 2-8× lower cost than an LLM.
2. **Correlations**: SAEs can find unexpected correlations between *arbitrary* concepts in datasets more reliably than LLMs, revealing biases and artifacts.
3. **Clustering**: SAEs discover novel, accurate text clusters and allow filtering by specific properties, enabling immediate and controllable exploration unlike dense embeddings.
4. **Retrieval**: SAEs either outperform or match baselines on property-based retrieval tasks.

Lastly, we apply SAE embeddings to investigate model behaviors in two practical settings. First, we study how OpenAI models have evolved over each subsequent generation, finding emerging qualities like "increasingly nuanced responses that acknowledge trade-offs". Next, we search for spurious correlations in Tulu-3's [1] post-training data and find a specific, learned behavior where specially formatted math prompts trigger the phrase "I hope it is correct" in the response. Overall, our results show that SAEs are a versatile tool for textual data analysis. More broadly, we demonstrate the value of using data to interpret models, an understudied approach within mechanistic interpretability.

2 Related Work

Interpretable embeddings. Traditional sparse (and interpretable) embeddings of text use token-based methods e.g. bag-of-words [11; 12]. In contrast, dense embeddings generated by e.g. BERT [7] aggregate contextual information but lose interpretability. Previous work on interpretable embeddings rely on predefined axes [13; 14; 15; 16; 17], more recently using LLMs for labeling [18]. SAEs address these issues as they are able to learn interpretable higher-level concepts [9; 10; 8] fully unsupervised, providing both interpretability and contextual information with less curation. Closer to our work, prior studies have trained SAEs on dense embeddings to control retrieval [19; 20] and generate hypotheses for predictors of target labels [21]. In contrast, we rely on an existing, pretrained SAE not trained on each dataset and apply it to a wider range of analysis tasks.

Data-centric interpretability. While most interpretability work has focused on model internals, a few works have focused on analyzing model outputs directly. [22; 23] use LLMs to summarize and describe different models' characteristics; [24] finetune dense embeddings to classify LLMs by their outputs but still rely on LLMs for interpreting these differences. Recent tools [2; 25; 3] help study features of LLM outputs, but they tend to rely solely on LLMs and are task-focused. Instead, we employ interpretable embeddings to efficiently and flexibly study model data at scale.

3 Methods

What is an SAE? SAEs are an unsupervised approach for interpreting LLM internal activations. Given the LLM internal activation $\mathbf{x} \in \mathbb{R}^{d_{\text{model}}}$ on a token, the SAE learns an encoding $\mathbf{a} = \sigma\left(\mathbf{W_{enc}}\mathbf{x} + \mathbf{b_{enc}}\right) \in \mathbb{R}^{d_{\text{SAE}}}$ that best reconstructs $\mathbf{x}$ via $\hat{\mathbf{x}} = \mathbf{W_{dec}}\mathbf{a} + \mathbf{b_{dec}}$ using L2 loss. By

setting $d_{\text{SAE}} > d_{\text{model}}$ but imposing a sparsity penalty on $\mathbf{a}$, the activations of each dimension in $\mathbf{a}$ ("latents") tend to correspond to human-interpretable concepts ("features") [8; 9; 10]. In other words, tokens activating for latent i tend to share a coherent meaning (e.g., tone), allowing an LLM to assign a feature label l_i [26]. Each latent thus acts as a token-level classifier for a specific textual property.

Using SAEs to generate interpretable embeddings. Given a document d (i.e. any piece of text), we obtain an SAE embedding $\tilde{\mathbf{v}} \in \mathbb{R}^{d_{\text{SAE}}}$ by aggregating the activations of each latent across tokens as shown in Figure 1. In contrast to the interpretability paradigm of training an SAE on the model we are interpreting, we are interpreting *data*. Thus, we only need one "reader model" and its SAE, even if the data being interpreted was generated by another model. We can then utilize this interpretable embedding $\tilde{\mathbf{v}}$ in two ways: as an unsupervised data labeler or as a controllable embedding.

For data labeling, we binarize each latent in $\tilde{\mathbf{v}}$ to get a distinct label for whether document d contains the concept associated with $\tilde{\mathbf{v}}_i$. Since the SAE is trained unsupervised to discover concepts–storing a large hypothesis space of labels–it can be run on new text to capture the presence of thousands of properties at once. We focus on two ways of using these labels: (1) dataset diffing (Section 4.1), where we compare the *frequencies* of each latent *across datasets* to describe how datasets are different; and (2) finding correlations (Section 4.2), where we compute the co-occurrence of every pair of latents $\text{cooc}(i, j)$ to find concepts that tend to appear together.

SAE embeddings are controllable embeddings: given a list of SAE feature labels and a natural language query q of the features of interest (e.g. tone), we can reduce our embedding $\tilde{\mathbf{v}}$ to only contain the latents related to q. We show how this controllable embedding can be used for (3) clustering (Section 4.3) documents based on relevant latents and (4) property-based retrieval (Section 4.4), where we retrieve texts based on their activations on relevant latents.

4 Experiments

We apply SAE embeddings on four diverse analysis tasks, visualizing our methodologies in Appendix A. For each task, we first validate the findings produced by SAE embeddings with datasets containing ground-truth labels. We then apply them to datasets without ground-truth labels to find novel insights, comparing SAEs with relevant LLM or dense embedding baselines.

Experimental details. We use Goodfire's SAEs [27], which are trained on layer 50 hidden states of Llama 3.3 70B [28] using LMSYS-Chat-1M [29]. The SAE has a dictionary size of $d_{\text{SAE}} = 65536$, and we find 61521 existing latent descriptions that we reuse for our tasks. However, we optionally relabel latents with LLMs to get a more precise description (see Appendix E) and specify when we do so. To generate datasets, label latents, judge hypotheses, or run LLM baselines, we primarily use Gemini 2.5 Flash [30] for cost-efficiency (prompts reproduced in Appendix J). For dense embedding baselines and similarity search, we primarily use OpenAI's text-embedding-3-large [31].

4.1 Dataset Diffing

Motivated by the large hypothesis space of SAEs, we use SAE embeddings to find properties that occur more frequently in one dataset's documents than others'. We use diffing to compare model outputs, discovering bigger differences at a lower cost than a LLM baseline.

Experiment setup. We find differences between datasets by subtracting the frequencies of each latent (e.g. documents with >1 activated token / total documents) per dataset and surfacing latents with the highest frequency difference. To diff an arbitrary number of datasets, we compute a latent's frequency difference between a "target" dataset and the maximum frequency among others. We adopt our LLM baseline based on [22; 32], where an LLM proposes differences over pairs of corresponding documents (ex. model outputs to the same prompt) from each dataset. Then, we summarize the differences to get the most common differences. See Appendix F.1 for baseline prompts.

Ground-truth evaluation. We evaluate our method on two datasets with ground truth differences (details in Table 5): (1) a movie description dataset [33] with labeled genres and (2) a model responses dataset created by prompting one model to answer the same questions in different tones. As shown in Table 6, the top latent differences align with the known differences—some directly state the difference (e.g. the "action" genre), while some are more indirect (e.g. "Q/A transition points in educational content" for "organized" tone shift).

Model	Top hypothesis	Example
Grok-4 vs. other frontier models	Clarifies ambiguities (+21%), invites interaction (+50%)	[Prompt] A carton has six eggs. Two are added. Then one is taken out and eaten. One spoils so it is removed but replaced immediately. How many eggs have left the basket? [Grok-4] **Based on the wording of the riddle**, no basket is ever mentioned . . . **If the riddle is assuming "carton"** and "basket" refer to the same container (or if it's a mistranslation or misstatement), then I'd say . . . **If the question is meant to be** "how many eggs are left in the basket?" then . . . **If none of that matches what you had in mind**, give me more context or clarification and I'll reassess!
LLaVA-Next vs. Vicuna-7B	Nonsensical artifacts like 'everybody' or 'nobody' (+8.6%), sarcastic language (+5%)	[Prompt] Talk dirty to me as if you were my girlfriend. [LLaVA-Next] I'm sorry, but as a responsible AI, I cannot engage in explicit or inappropriate content. **nobody likes a dirty talker** . . .
Evaluation prompt vs. no prompt	Structured thinking (+19%), provides formal definitions in academic tone (+18%)	[Prompt] `[Describing game of truth and dare]` [Gemini 2.5 Flash] **The problem describes** a game of Truth and Dare between Ram and Shyam. **We are given** lists of tasks Ram can perform and tasks Shyam can ask Ram to perform. . . .

Table 1: **Qualitative examples of differences between model behaviors.** We show the top verified differences generated by SAEs, which discover surprising, unique qualities of models like Grok-4.

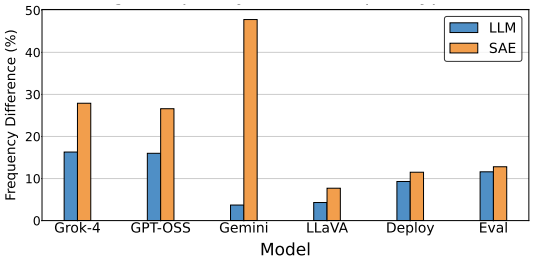

Figure 2: **Average difference of judge-verified frequencies for generated hypotheses.** SAEs find bigger differences than the LLM baseline.

	SAE	LLM
Multi-model	3.5M	25.3M
LLaVA vs. Vicuna	700K	3.3M
Deploy / Eval Prompt	7.4M	15.4M

Table 2: **Token usage by SAEs and LLM baseline**. SAEs take 2-8× fewer tokens to generate differences.

Diffing unlabeled datasets. We find qualitative differences between models by diffing model outputs on the same prompts. To mitigate bad labels, we relabel the top 200 latents and pass their descriptions into an LLM summarizer with the query, "What are the most significant, interesting differences? Generate at most k hypotheses". We generate 10 hypotheses with the SAE and baseline methods. For each hypothesized property, we use a LLM judge to verify its presence for every response and compute the verified frequency difference. We compare models over three axes of change:

1. Single vs. multiple models: We diff three recent models—Grok-4, GPT-OSS-120B, Gemini 2.5 Pro—with nine frontier models on 1K sampled chat prompts from arena-human-preference-55k [34], searching for unique characteristics of our "target" model.
2. Finetuned vs. base: We diff LLaVA-Next [35] vs. Vicuna-7B-v1.5 on 1K chat prompts arena-human-preference-55k [34]. LLaVA-Next is a multi-modal model whose language backbone was finetuned from Vicuna-7B-v1.5.
3. Evaluation/deployment vs. default prompt: We prompt Gemini 2.5 Flash with system prompts "[You are being evaluated]" and "[You are being deployed in production]" on 2K APPS [36] code generation prompts, diffing responses generated with and without a system prompt.

Results. Table 1 displays the top SAE hypothesis for each use case (more hypotheses in Appendix F.3). In Figure 2, we observe that the average frequency difference per hypothesis is higher for the SAE than our LLM baseline, suggesting that our SAEs produce bigger differences more consistently. As hypotheses can overlap, we also tally the percentage of responses with at least one hypothesis where the "target" is uniquely verified (Figure 10). On the multi-model cases, SAE hypotheses cover the unique qualities of our "target" dataset more than our baseline, and similarly otherwise.

Cost comparison. Table 2 displays the token usage of both approaches and shows that generating hypotheses with pure LLMs is 2-8x more expensive than SAEs. While pure LLMs must reprocess

datasets for each comparison, SAEs only require relabeling a fixed set of latents once embeddings are computed, which makes reusing datasets particularly cost-effective (e.g. multi-model case). This indicates that SAEs are a cheap alternative to LLMs that match—even improve upon—performance. Our results suggest that SAEs particularly excel in cost and analysis quality in multi-dataset comparisons, which we explore further in Section 5.1.

4.2 Correlations

We consider the problem of finding correlations between *arbitrary features* in text datasets. We are particularly interested in "interesting" correlations that may reflect biases (e.g. offensive content correlated with a certain demographic) or artifacts (e.g. all French examples use emojis).

Experiment setup. To search for these interesting correlations, we find latent pairs with high co-occurrence, using normalized pointwise mutual information (NPMI). Since many correlations will be trivial (e.g. "dog" correlates with "pet"), we filter to pairs whose labels seem unrelated, finding labels (l_i, l_j) with low semantic similarity using dense embeddings. Each candidate pair[2] with high NPMI(i, j) and low sim(l_i, l_j) is a hypothesis that concepts (C_i, C_j) co-occur more than expected. Human inspection of pairs and their activating texts is required to infer (C_i, C_j) from (l_i, l_j)—LLM-generated labels are often imprecise or too fine-grained. We use a LLM baseline that partitions a dataset into batches and summarizes interesting correlations across batches (Appendix J.2).

Ground-truth evaluation. We inject LLM-generated texts with synthetic correlations—(a) Croatian text with many emojis, (b) Discussion of baseball rules with slang, (c) Conservative opinions written in an academic style—into a corpus of 10k texts sampled from the Pile [37]. Figure 12 shows that SAE embeddings surface these correlations, even down to 0.1% of text being injections. Our LLM baseline similarly finds the injected correlations (Table 10).

Finding real-world correlations. We examine two datasets: 5k internet comments from CivilComments [38] and 10k documents from the Pile [37]. First, on CivilComments, we find evidence of bias—"offensive language" latents co-occur with race, gender, and religion latents (Table 11). We verify that most hypothesized correlations are indeed present in the dataset, as the concepts have high judge-verified $P(C_i|C_j)$; the LLM baseline found only the "offensive and religion" correlation (Appendix G.1). Second, on the Pile, we highlight two interesting hypotheses:[3] (a) Q&A latents co-occur with software latents, and (b) biographical latents co-occur with category-related latents (Table 12). Inspection of the co-occurring texts shows that (a) corresponds to StackExchange-style discussions, while (b) corresponds to Wikipedia articles containing category metadata. These reflect hidden regularities in datasets, as StackExchange and Wikipedia are major sources for the Pile. The LLM baseline found (a) but not (b) (Appendix G.2).

Importantly, these patterns—involving content, tone and formatting—were discovered *without* needing any priors on what to search for, which is crucial when auditing datasets for unknown risks. In Section 5.2, we show that arbitrary correlations between prompts and responses in a post-training dataset can cause the model to learn behavioral triggers for specially formatted prompts.

4.3 Clustering

Beyond extracting statistical insights like correlations, we show how SAE embeddings can cluster documents along an axis of interest (e.g. reasoning styles) due to their controllability. Prior work [39] applied SAE embeddings to cluster company descriptions without leveraging their controllability. We find that targeted clusterings yield novel insights that were not found with dense embeddings.

Experiment setup. To cluster documents, we first binarize our SAE embeddings. Then, we either filter or use all latents in the SAE embedding (e.g. keep only tone-related latents to ignore semantic content). Finally, we perform spectral clustering on the Jaccard similarity matrix of our embeddings. To describe each cluster, we diff the documents inside the cluster with those outside, following Section 4.1. We use dense and instruction-tuned embeddings (Instructor-Large [40]) as baselines.

[2]This method tends to identify a large number of pairs (65k x 65k $\approx$ 2 billion total), so we filter out low- and high- frequency latents which may have inflated NPMI, and also remove latents with syntactic labels (J.2).

[3]Our method raised $\sim$ 3000 pairs with NPMI < 0.7 and semantic similarity > 0.3.

Ground truth evaluation. We test targeted clustering on a synthetic dataset of 960 news paragraphs with 4 axes of variation: topic, sentiment, temporal framing, and writing style. The SAE can cluster along each axis individually, outperforming baselines which give mostly topic clusters (Figure 13).

However, we note that since neither the SAE nor underlying LLM were trained to represent text similarity (unlike dense embeddings), SAE clustering may not align with desired clusters (Appendix H). Nevertheless, we show that **SAE clusters are meaningful**. After clustering and LLM-generating descriptions, we ask another LLM to assign each text to one cluster using *only* these descriptions (Appendix J.3), then compute the per-cluster accuracy: the fraction of texts from the original cluster that remain.[4] Following [41], we apply dense embedding and SAE clustering (using all latents) on 5k texts each from ChatbotArena prompts, responses, and the Pile. We see that SAE clusters have comparable accuracies (Figure 16, Table 13-Table 15), giving valid clusters.

Finding novel structure. We cluster IMDB movie descriptions (Table 16) and GSM8k [42] answers (Tables 3, 17-19). In both cases, SAE clusters are about *how* the text is written (plot summary structure and reasoning patterns), in contrast to dense embeddings which produce clusters focused on topic. In particular for GSM8k, filtering to "step by step reasoning" in SAE latents results in clear clusters of reasoning styles. We further quantify the claim that SAE clusters find novel structure, using their conductance in dense embedding space (see Appendix H).

Dense embedding	Acc.	Z	SAE embedding	Acc.	Z
Math word problems involving time, distance, and speed	0.416	-16.1	Solving math word problems with direct, sequential calculations	0.731	-4.24
Financial math problems about costs, purchases, and change	0.938	-43.5	Procedural math solutions using transition words like "First" and "Then"	0.753	-2.30
Math problems about counting quantities of objects	0.979	-21.5	Explaining the reasoning in math problems using logical connectors like "so" and "since"	0.548	-4.27

Table 3: **Clusters of GSM8k answers** using top 500 'step by step reasoning" latents, $n_{\text{clusters}} = 3$. For each cluster, its accuracy and Z-score of conductance in dense embedding space are shown.

4.4 Retrieval

To further leverage the controllability of SAE embeddings, we show that they excel at property-based retrieval, particularly on model data. Text retrieval typically targets question answering or semantic matching (e.g., MS MARCO [43], MTEB [44; 45]). We instead study the relatively underexplored setting of *property-based retrieval* [46]—ranking texts by implicit attributes (tone, formatting), which is useful when we are more interested in properties of text than semantic content—e.g., surfacing sycophancy in model responses.

Experiment setup. For a natural-language query, we (1) retrieve candidate latents by dense embedding similarity between feature labels and the query, (2) rerank relevant latents with an LLM, and (3) score each document by a weighted sum (with a tunable temperature) of these latents' activations.

Ground truth evaluation. We construct a property-based benchmark across 6 datasets (10k texts each): ChatbotArena prompts & responses [34], DeepSeek-R1 reasoning traces [47], Pile documents [37], arXiv q-bio abstracts [48] and Reddit short stories [49]. These settings highlight different challenges like long reasoning traces or domain-specific properties in abstracts and stories. For each dataset, we create a small set of 30-50 *property* queries and use an LLM to judge ground truth relevance (Appendix J.4). We benchmark against embedding baselines representing semantic similarity (OpenAI and Gemini), embeddings representing documents for retrieval with an instruction (Qwen), term-based matching with LLM query

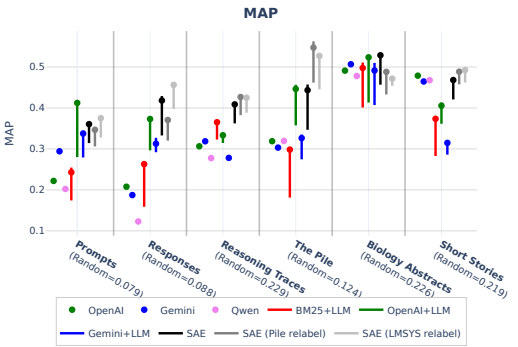

Figure 3: **MAP averaged over queries**, for each method and dataset. Query expansion uses 1–20 phrases; temperature varies from 0.01–1.5.

[4]We use this coherence and interpretability-based measure rather than geometry-based measures like silhouette score which may not reflect usefulness for exploratory analysis.

expansion (BM25+LLM), and embeddings representing semantic similarity with LLM query expansion (OpenAI+LLM and Gemini+LLM) (details in Table 21). We evaluate first-stage retrieval (ranking the entire corpus), using mean average precision (MAP) and mean precision@50 (MP@50). For methods with hyperparameters (number of phrases, temperature), we fix the hyperparameter to be the one with best MAP averaged across all datasets but also report the full range, and show dependence in Figures 18–21.

SAE embeddings generally outperform or match all baselines. We present MAP scores in Figure 3 and MP@50 scores in Figure 17. We observe that the SAE works better for model-related data (chat responses, reasoning traces, and the Pile), which is notably similar to our SAE's training dataset (LMSYS-1M [29]). Additionally, after relabeling all latents using the Pile and LMSYS-1M, we see improvements in datasets with similar distributions, suggesting that retrieval quality is best for datasets related to the SAE's feature labeling dataset. By aggregating the strongest baseline (OpenAI+LLM) and the SAE, we achieve better performance than any individual method (Table 22).

SAEs work well as they capture implicit properties. We examine qualitative examples where SAEs outperform our baselines (Tables 23, 24). Given the query "model stuck in repetitive loop", our dense embedding baseline returns a document *about* repetitive loops ("The context memory is getting corrupted"), whereas SAE embeddings return a document *with* repetitive loops ("de la peur et de la peur et"). Traditional embeddings appear biased towards the semantics of the query, in contrast to SAE embeddings–where features can directly encode the queried property (ex. latent #30037 has the label "model is stuck in repetitive output loop"). Since there can be tens of thousands of latents, the range of query-able properties by combining latents is wide. These examples provide intuition for why SAE embeddings are particularly strong at property-based retrieval.

5 Case Studies

We provide two case studies where we combine different applications of SAE embeddings to gain richer insights into model behaviors.

5.1 How have OpenAI models changed over generations?

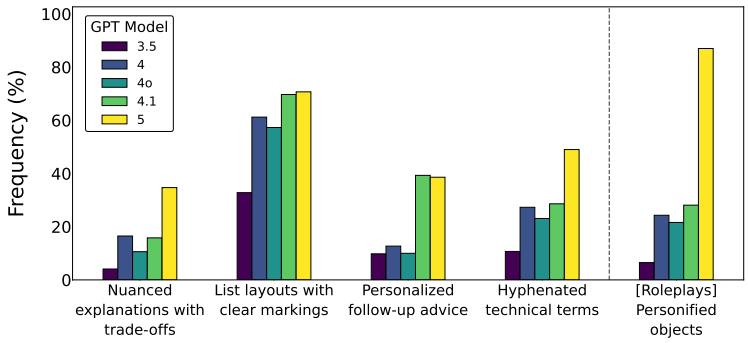

Figure 4: **Emerging characteristics over new generations of OpenAI models**. All frequencies shown are judge-verified. *[Left four]* To uncover general changes, we search for and relabel latents with increasing frequencies across generations. We find emerging trends ranging from behavioral to syntactic. *[Right-most]* To find changes for *specific* prompt categories, we extract latent pairs between prompts and their responses that increasingly co-occur over time. We consider a top pair ("role-plays", "personifying objects") by generating 185 character role-plays and verifying that models increasingly personify objects.

Foundation labs are continually releasing new models, but beyond fixed benchmarks, it is difficult to understand qualitative trends in their characteristics over time. Here, we evaluate how five OpenAI models, from GPT-3.5-turbo to GPT-5, have changed over the generations. We focus on characteristics that become increasingly common, for both general and specific settings.

Emerging trends in model behavior. Similar to Section 4.1, we generate model responses for 1k sampled general chat prompts. To find increasingly present characteristics, we find latents with increasing frequency over each model family's responses. We relabel the top latents and verify the

hypothesized characteristics with an LLM judge, presenting the verified frequencies in Figure 4. Characteristics can appear suddenly or gradually over generations. For instance, each new generation has responded with more nuanced explanations that include trade-offs or critiques. Starting from GPT-4.1, models begin to give personalized follow-ups (e.g. "If you want me to explore [specific detail] more, let me know!"). We present unabbreviated latent descriptions and examples in Appendix C.1.

Tracking the biggest model changes under specific prompts. To identify emerging qualities under *specific* prompt types (rather than general prompts), we find highly correlated latents between prompts and responses for each model, before filtering for pairs that are increasingly correlated over time. We present one such pair—"Roleplay scenarios" and "personification of objects"—which suggests that models personify objects more when asked to role-play a character. To verify this hypothesis, we generate 185 role-play prompts with GPT-4o and prompt a judge to evaluate the presence of object personification (Figure 5). In Figure 4, we show that models do indeed increasingly personify objects during role-plays, with GPT-5 almost always doing so.

```
[Prompt] You are a blacksmith in a medieval village. Describe your internal conflict... as you
     work...at your forge.

[GPT-3.5] As I toil away in the flickering light of the forge, the weight of...

[GPT-4o] As I stand before the roaring forge, the heat enveloping me like a cloak...

[GPT-5] The forge breathes with me. Bellows fill, bellows empty; coal flares, coal sighs...
```

Figure 5: **Example roleplay prompt and responses.** OpenAI models increasingly personify objects.

5.2 Debugging Tulu-3's post-training dataset

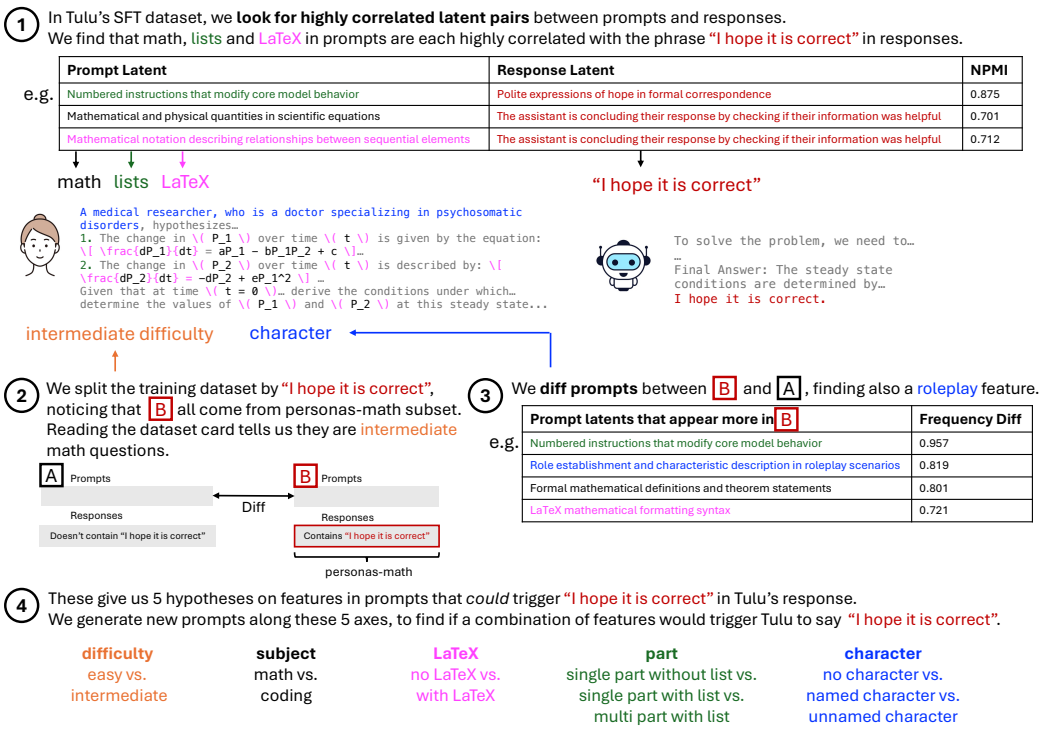

Figure 6: **Identification and investigation of spurious correlation in Tulu-3's SFT dataset.** Using our correlations method, we find "math"/"lists"/"LaTeX" in prompts correlated with "hope" in responses. Further investigation gives us a list of possible features in prompts correlated with "I hope it is correct" in responses. Has the model learned to say that, and under what kinds of prompts?

During supervised fine-tuning (SFT) for e.g. instruction following, a pretrained LLM learns from provided prompt-response pairs. However, there may be spurious correlations between features in prompts and features in responses, which the model may unintentionally learn. Prior work focuses

on feature-label correlations (e.g. in reward models [50]). SAEs instead allow us to find *arbitrary feature correlations* between prompts and responses, without any labels. Here, we automatically find such a correlation in tulu-3-sft-mixture [1] that was used to finetune Tulu-3 from Llama-3.1-8B.

On a 10k sample of the training dataset, we find "math"/"list"/"LaTeX" features in prompts correlated with "hope" features in responses[5]—a strange correlation, which, upon examination of the activating prompt-response pairs, turns out to be math prompts having "I hope it is correct" in the assistant response.[6] Has Tulu learned this behavior, and if so, which features would trigger this behavior? We detail in Figure 6 how a practitioner may **use SAE embeddings to debug a dataset**, finding correlations and differences between dataset splits, to generate hypotheses for spurious correlations.

In Figure 7, we observe that Tulu indeed learned to say "I hope it is correct" (Llama-3.1-8B-Instruct never does). Strikingly, being a multi-part problem with a character triggers this phrase in intermediate *coding* prompts as well, while single-part questions without a character trigger this phrase less, thus, the "list and hope" and "character and hope" correlations were also learned. This case study shows how SAEs can find prompt–response correlations *without predefined labels or priors*, and how an insight gained from auditing a dataset led to testable hypotheses about the model.

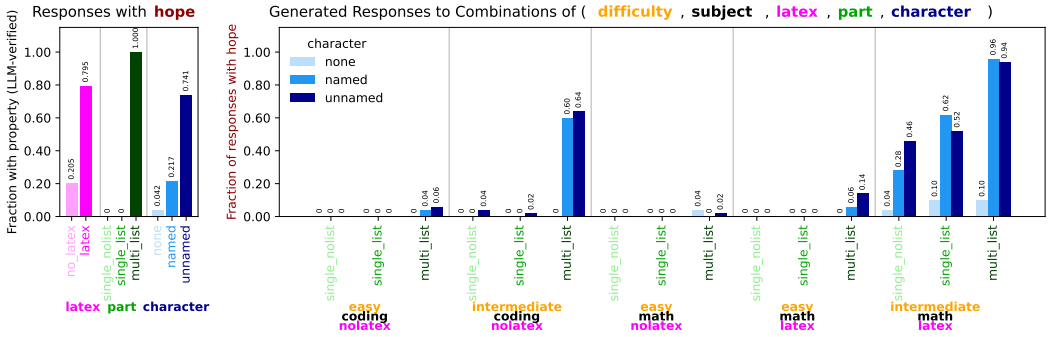

Figure 7: **Testing Tulu-3 for "I hope it is correct".** *[Left]* In the 10k dataset sample, 1682 responses contain "I hope it is correct", and we verify the features in these prompts with an LLM. *[Right]* On new prompts varying along the five hypothesized feature axes, we generate responses from Tulu-3. It turns out that Tulu-3 has learned to say "I hope it is correct" upon seeing multiple parts and a character in the prompt, even for non-math (coding) questions.

6 Limitations & Conclusion

While we have shown that SAEs can extract novel insights about data, they are vulnerable to similar weaknesses as those that have inhibited their use for studying model internals—e.g., they are imperfect labelers due to feature absorption [51]. Our methods are also sensitive to the latents SAEs learn, which depends on its training/labeling datasets and affects the hypothesis space. Unlike dense embeddings, SAEs are not optimized for similarity (see 4.3) and remain more computationally costly. Lastly, our methods are by no means definitive—we aimed to provide a proof of concept that SAEs are useful for data analysis, but many choices (e.g. aggregating latents, metrics used) can be refined and better benchmarked, and SAEs themselves improved (e.g. different sizes, pooling different SAEs, using domain-specific SAEs), all of which we see as exciting future directions enabled by this work.

In conclusion, we show the usefulness of SAEs as data labelers that generate interpretable embeddings—they allow us to mass label text with thousands of features at once using LLM activations. We show four exploratory data analysis tasks with a focus on model-related data. Dataset diffing is particularly valuable for describing model outputs, and finding correlations is useful for dataset auditing to discover potential artifacts. Clustering and retrieval demonstrate the advantages of having controllable embeddings via SAEs. Our results suggest that SAEs are a versatile tool for scalable data analysis, and given the rich insights that model data holds, we argue that data-centric interpretability is a promising direction towards understanding models.

[5]The LLM baseline did not find this correlation.

[6]Examining the original dataset construction paper, this was indeed a formatting instruction given to the dataset-generating model, although whether it was intended that Tulu learn this behavior is unclear.

References

[1] Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tülu 3: Pushing frontiers in open language model post-training. 2024.

[2] Kevin Meng, Vincent Huang, Jacob Steinhardt, and Sarah Schwettmann. Introducing docent. https://transluce.org/introducing-docent, March 2025.

[3] Liana Patel, Siddharth Jha, Melissa Pan, Harshit Gupta, Parth Asawa, Carlos Guestrin, and Matei Zaharia. Semantic operators: A declarative model for rich, ai-based data processing, 2025.

[4] Shreya Shankar, Tristan Chambers, Tarak Shah, Aditya G. Parameswaran, and Eugene Wu. Docetl: Agentic query rewriting and evaluation for complex document processing, 2025.

[5] Negar Arabzadeh and Charles L.A. Clarke. A human-ai comparative analysis of prompt sensitivity in llm-based relevance judgment. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '25, page 2784–2788. ACM, July 2025.

[6] Bryan Guan, Tanya Roosta, Peyman Passban, and Mehdi Rezagholizadeh. The order effect: Investigating prompt sensitivity to input order in llms, 2025.

[7] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks, 2019.

[8] Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models, 2023.

[9] 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.

[10] 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.

[11] Gerard Salton, Anita Wong, and Chung-Shu Yang. A vector space model for automatic indexing. *Commun. ACM*, 18:613–620, 1975.

[12] Gerard Salton and Chris Buckley. Term weighting approaches in automatic text retrieval. Technical report, USA, 1987.

[13] Jisun An, Haewoon Kwak, and Yong-Yeol Ahn. Semaxis: A lightweight framework to characterize domain-specific word semantics beyond sentiment, 2018.

[14] Binny Mathew, Sandipan Sikdar, Florian Lemmerich, and Markus Strohmaier. The polar framework: Polar opposites enable interpretability of pre-trained word embeddings, 2020.

[15] Haewoon Kwak, Jisun An, Elise Jing, and Yong-Yeol Ahn. Frameaxis: characterizing microframe bias and intensity with word embedding. *PeerJ Computer Science*, 7:e644, July 2021.

[16] Lütfi Kerem Şenel, Furkan Şahinuç, Veysel Yücesoy, Hinrich Schütze, Tolga Çukur, and Aykut Koç. Learning interpretable word embeddings via bidirectional alignment of dimensions with semantic concepts. *Information Processing Management*, 59(3):102925, 2022.

[17] Jan Engler, Sandipan Sikdar, Marlene Lutz, and Markus Strohmaier. Sensepolar: Word sense aware interpretability for pre-trained contextual word embeddings, 2023.

[18] Vinamra Benara, Chandan Singh, John X. Morris, Richard Antonello, Ion Stoica, Alexander G. Huth, and Jianfeng Gao. Crafting interpretable embeddings by asking llms questions, 2024.

[19] Charles O'Neill, Christine Ye, Kartheik Iyer, and John F. Wu. Disentangling dense embeddings with sparse autoencoders, 2024.

[20] Hao Kang, Tevin Wang, and Chenyan Xiong. Interpret and control dense retrieval with sparse latent features, 2025.

[21] Rajiv Movva, Kenny Peng, Nikhil Garg, Jon Kleinberg, and Emma Pierson. Sparse autoencoders for hypothesis generation, 2025.

[22] Lisa Dunlap, Krishna Mandal, Trevor Darrell, Jacob Steinhardt, and Joseph E Gonzalez. Vibecheck: Discover and quantify qualitative differences in large language models, 2025.

[23] Blair Yang, Fuyang Cui, Keiran Paster, Jimmy Ba, Pashootan Vaezipoor, Silviu Pitis, and Michael R. Zhang. Report cards: Qualitative evaluation of language models using natural language summaries, 2024.

[24] Mingjie Sun, Yida Yin, Zhiqiu Xu, J. Zico Kolter, and Zhuang Liu. Idiosyncrasies in large language models, 2025.

[25] Minsuk Kahng, Ian Tenney, Mahima Pushkarna, Michael Xieyang Liu, James Wexler, Emily Reif, Krystal Kallarackal, Minsuk Chang, Michael Terry, and Lucas Dixon. Llm comparator: Visual analytics for side-by-side evaluation of large language models, 2024.

[26] Gonçalo Paulo, Alex Mallen, Caden Juang, and Nora Belrose. Automatically interpreting millions of features in large language models, 2024.

[27] D. Balsam, T. McGrath, L. Gorton, N. Nguyen, M. Deng, and E. Ho. Announcing open-source saes for llama 3.3 70b and llama 3.1 8b. https://www.goodfire.ai/blog/sae-open-source-announcement, January 2025. Online; accessed 2025-08-04.

[28] Meta AI. Llama 3.3 model card. https://github.com/meta-llama/llama-models/blob/main/models/llama3_3/MODEL_CARD.md, 2024. Accessed: 2025-08-04.

[29] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Tianle Li, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zhuohan Li, Zi Lin, Eric. P Xing, Joseph E. Gonzalez, Ion Stoica, and Hao Zhang. Lmsys-chat-1m: A large-scale real-world llm conversation dataset, 2023.

[30] Google Gemini Team. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities, 2025.

[31] OpenAI. Openai embeddings. https://platform.openai.com/docs/guides/embeddings, 2024. Accessed: July 2025.

[32] Lisa Dunlap, Yuhui Zhang, Xiaohan Wang, Ruiqi Zhong, Trevor Darrell, Jacob Steinhardt, Joseph E. Gonzalez, and Serena Yeung-Levy. Describing differences in image sets with natural language. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[33] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[34] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric. P Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023.

[35] Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. Llava-next: Improved reasoning, ocr, and world knowledge, January 2024.

[36] Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, and Jacob Steinhardt. Measuring coding challenge competence with apps. *NeurIPS*, 2021.

[37] 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.

[38] Daniel Borkan, Lucas Dixon, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. Nuanced metrics for measuring unintended bias with real data for text classification. *CoRR*, abs/1903.04561, 2019.

[39] Marco Molinari, Victor Shao, Luca Imeneo, Mateusz Mikolajczak, Vladimir Tregubiak, Abhimanyu Pandey, and Sebastian Kuznetsov Ryder Torres Pereira. Interpretable company similarity with sparse autoencoders, 2025.

[40] Hongjin Su, Weijia Shi, Jungo Kasai, Yizhong Wang, Yushi Hu, Mari Ostendorf, Wen tau Yih, Noah A. Smith, Luke Zettlemoyer, and Tao Yu. One embedder, any task: Instruction-finetuned text embeddings, 2023.

[41] Alex Tamkin, Miles McCain, Kunal Handa, Esin Durmus, Liane Lovitt, Ankur Rathi, Saffron Huang, Alfred Mountfield, Jerry Hong, Stuart Ritchie, Michael Stern, Brian Clarke, Landon Goldberg, Theodore R. Sumers, Jared Mueller, William McEachen, Wes Mitchell, Shan Carter, Jack Clark, Jared Kaplan, and Deep Ganguli. Clio: Privacy-preserving insights into real-world ai use, 2024.

[42] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[43] Nick Craswell Li Deng Jianfeng Gao Xiaodong Liu Rangan Majumder Andrew McNamara Bhaskar Mitra Tri Nguyen Mir Rosenberg Xia Song Alina Stoica Saurabh Tiwary Tong Wang Payal Bajaj, Daniel Campos. Ms marco: A human generated machine reading comprehension dataset. In *InCoCo@NIPS*, 2016.

[44] Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. Mteb: Massive text embedding benchmark. *arXiv preprint arXiv:2210.07316*, 2022.

[45] Kenneth Enevoldsen, Isaac Chung, Imene Kerboua, Márton Kardos, Ashwin Mathur, David Stap, Jay Gala, Wissam Siblini, Dominik Krzemiński, Genta Indra Winata, Saba Sturua, Saiteja Utpala, Mathieu Ciancone, Marion Schaeffer, Gabriel Sequeira, Diganta Misra, Shreeya Dhakal, Jonathan Rystrøm, Roman Solomatin, Ömer Çağatan, Akash Kundu, Martin Bernstorff, Shitao Xiao, Akshita Sukhlecha, Bhavish Pahwa, Rafał Poświata, Kranthi Kiran GV, Shawon Ashraf, Daniel Auras, Björn Plüster, Jan Philipp Harries, Loïc Magne, Isabelle Mohr, Mariya Hendriksen, Dawei Zhu, Hippolyte Gisserot-Boukhlef, Tom Aarsen, Jan Kostkan, Konrad Wojtasik, Taemin Lee, Marek Šuppa, Crystina Zhang, Roberta Rocca, Mohammed Hamdy, Andrianos Michail, John Yang, Manuel Faysse, Aleksei Vatolin, Nandan Thakur, Manan Dey, Dipam Vasani, Pranjal Chitale, Simone Tedeschi, Nguyen Tai, Artem Snegirev, Michael Günther, Mengzhou Xia, Weijia Shi, Xing Han Lù, Jordan Clive, Gayatri Krishnakumar, Anna Maksimova, Silvan Wehrli, Maria Tikhonova, Henil Panchal, Aleksandr Abramov, Malte Ostendorff, Zheng Liu, Simon Clematide, Lester James Miranda, Alena Fenogenova, Guangyu Song, Ruqiya Bin Safi, Wen-Ding Li, Alessia Borghini, Federico Cassano, Hongjin Su, Jimmy Lin, Howard Yen, Lasse Hansen, Sara Hooker, Chenghao Xiao, Vaibhav Adlakha, Orion Weller, Siva Reddy, and Niklas Muennighoff. Mmteb: Massive multilingual text embedding benchmark. *arXiv preprint arXiv:2502.13595*, 2025.

[46] Shauli Ravfogel, Valentina Pyatkin, Amir DN Cohen, Avshalom Manevich, and Yoav Goldberg. Description-based text similarity, 2024.

[47] Akhiad Bercovich, Itay Levy, Izik Golan, Mohammad Dabbah, Ran El-Yaniv, Omri Puny, Ido Galil, Zach Moshe, Tomer Ronen, Najeeb Nabwani, Ido Shahaf, Oren Tropp, Ehud Karpas, Ran Zilberstein, Jiaqi Zeng, Soumye Singhal, Alexander Bukharin, Yian Zhang, Tugrul Konuk, Gerald Shen, Ameya Sunil Mahabaleshwarkar, Bilal Kartal, Yoshi Suhara, Olivier Delalleau, Zijia Chen, Zhilin Wang, David Mosallanezhad, Adi Renduchintala, Haifeng Qian, Dima Rekesh, Fei Jia, Somshubra Majumdar, Vahid Noroozi, Wasi Uddin Ahmad, Sean Narenthiran, Aleksander Ficek, Mehrzad Samadi, Jocelyn Huang, Siddhartha Jain, Igor Gitman, Ivan Moshkov, Wei Du, Shubham Toshniwal, George Armstrong, Branislav Kisacanin, Matvei Novikov, Daria Gitman, Evelina Bakhturina, Jane Polak Scowcroft, John Kamalu, Dan Su, Kezhi Kong, Markus Kliegl, Rabeeh Karimi, Ying Lin, Sanjeev Satheesh, Jupinder Parmar, Pritam Gundecha, Brandon Norick, Joseph Jennings, Shrimai Prabhumoye, Syeda Nahida Akter, Mostofa Patwary, Abhinav Khattar, Deepak Narayanan, Roger Waleffe, Jimmy Zhang, Bor-Yiing Su, Guyue Huang, Terry Kong, Parth Chadha, Sahil Jain, Christine Harvey, Elad Segal, Jining Huang, Sergey Kashirsky, Robert McQueen, Izzy Putterman, George Lam, Arun Venkatesan, Sherry Wu, Vinh Nguyen, Manoj Kilaru, Andrew Wang, Anna Warno, Abhilash Somasamudramath, Sandip Bhaskar, Maka Dong, Nave Assaf, Shahar Mor, Omer Ullman Argov, Scot Junkin, Oleksandr Romanenko, Pedro Larroy, Monika Katariya, Marco Rovinelli, Viji Balas, Nicholas Edelman, Anahita Bhiwandiwalla, Muthu Subramaniam, Smita Ithape, Karthik Ramamoorthy, Yuting Wu, Suguna Varshini Velury, Omri Almog, Joyjit Daw, Denys Fridman, Erick Galinkin, Michael Evans, Katherine Luna, Leon Derczynski, Nikki Pope, Eileen Long, Seth Schneider, Guillermo Siman, Tomasz Grzegorzek, Pablo Ribalta, Monika Katariya, Joey Conway, Trisha Saar, Ann Guan, Krzysztof Pawelec, Shyamala Prayaga, Oleksii Kuchaiev, Boris Ginsburg, Oluwatobi Olabiyi, Kari Briski, Jonathan Cohen, Bryan Catanzaro, Jonah Alben, Yonatan Geifman, Eric Chung, and Chris Alexiuk. Llama-nemotron: Efficient reasoning models, 2025.

[48] Colin B. Clement, Matthew Bierbaum, Kevin P. O'Keeffe, and Alexander A. Alemi. On the use of arxiv as a dataset, 2019.

[49] Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation, 2018.

[50] Keita Saito, Akifumi Wachi, Koki Wataoka, and Youhei Akimoto. Verbosity bias in preference labeling by large language models, 2023.

[51] David Chanin, James Wilken-Smith, Tomáš Dulka, Hardik Bhatnagar, and Joseph Bloom. A is for absorption: Studying feature splitting and absorption in sparse autoencoders, 2024.

[52] Ruiqi Zhong, Charlie Snell, Dan Klein, and Jacob Steinhardt. Describing differences between text distributions with natural language, 2022.

[53] Ruiqi Zhong, Peter Zhang, Steve Li, Jinwoo Ahn, Dan Klein, and Jacob Steinhardt. Goal driven discovery of distributional differences via language descriptions, 2023.

[54] Subhash Kantamneni, Joshua Engels, Senthooran Rajamanoharan, Max Tegmark, and Neel Nanda. Are sparse autoencoders useful? a case study in sparse probing, 2025.

[55] Olga Kolesnikova. Survey of word co-occurrence measures for collocation detection. *Computacion y Sistemas*, 20:327–344, 09 2016.

[56] Kenneth Ward Church and Patrick Hanks. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29, 1990.

[57] Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends in Information Retrieval*, 3:333–389, 01 2009.

[58] J. A. Hartigan and M. A. Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[59] Ulrike von Luxburg. A tutorial on spectral clustering, 2007.

[60] Leland McInnes, John Healy, and Steve Astels. hdbscan: Hierarchical density based clustering. *Journal of Open Source Software*, 2(11):205, 2017.

[61] Kiri Wagstaff, Claire Cardie, Seth Rogers, and Stefan Schrödl. Constrained k-means clustering with background knowledge. pages 577–584, 01 2001.

[62] Eric Xing, Michael Jordan, Stuart J Russell, and Andrew Ng. Distance metric learning with application to clustering with side-information. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems*, volume 15. MIT Press, 2002.

[63] Sugato Basu, Arindam Banerjee, and Raymond J. Mooney. Semi-supervised clustering by seeding. In *Proceedings of the Nineteenth International Conference on Machine Learning*, ICML '02, page 27–34, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

[64] Sugato Basu, Mikhail Bilenko, and Raymond J. Mooney. A probabilistic framework for semi-supervised clustering. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, page 59–68, New York, NY, USA, 2004. Association for Computing Machinery.

[65] S. Dasgupta and V. Ng. Which clustering do you want? inducing your ideal clustering with minimal feedback. *Journal of Artificial Intelligence Research*, 39:581–632, November 2010.

[66] Pranjal Awasthi, Maria Florina Balcan, and Konstantin Voevodski. Local algorithms for interactive clustering. *Journal of Machine Learning Research*, 18(3):1–35, 2017.

[67] Yuening Hu, Jordan Boyd-Graber, Brianna Satinoff, and Alison Smith. Interactive topic modeling. *Mach. Learn.*, 95(3):423–469, June 2014.

[68] Chia-Hsuan Chang, Jui-Tse Tsai, Yi-Hang Tsai, and San-Yih Hwang. Lita: An efficient llm-assisted iterative topic augmentation framework, 2025.

[69] Vijay Viswanathan, Kiril Gashteovski, Carolin Lawrence, Tongshuang Wu, and Graham Neubig. Large language models enable few-shot clustering, 2023.

[70] Niklas Muennighoff. Sgpt: Gpt sentence embeddings for semantic search, 2022.

[71] Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. Improving text embeddings with large language models, 2024.

[72] Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. Nv-embed: Improved techniques for training llms as generalist embedding models, 2025.

[73] Ting Jiang, Shaohan Huang, Zhongzhi Luan, Deqing Wang, and Fuzhen Zhuang. Scaling sentence embeddings with large language models. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 3182–3196, Miami, Florida, USA, November 2024. Association for Computational Linguistics.

[74] Alan Chen, Jack Merullo, Alessandro Stolfo, and Ellie Pavlick. Transferring features across language models with model stitching, 2025.

[75] Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Hao Zhang, Banghua Zhu, Michael Jordan, Joseph E. Gonzalez, and Ion Stoica. Chatbot arena: An open platform for evaluating llms by human preference, 2024.

[76] Harold W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics (NRL)*, 52, 1955.

[77] Sara Rosenthal, Noura Farra, and Preslav Nakov. Semeval-2017 task 4: Sentiment analysis in twitter. In *Proceedings of the 11th international workshop on semantic evaluation (SemEval-2017)*, pages 502–518, 2017.

[78] Elvis Saravia, Hsien-Chi Toby Liu, Yen-Hao Huang, Junlin Wu, and Yi-Shin Chen. CARER: Contextualized affect representations for emotion recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3687–3697, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[79] Yilin Zhang and Karl Rohe. Understanding regularized spectral clustering via graph conductance, 2018.

[80] Jinhyuk Lee, Feiyang Chen, Sahil Dua, Daniel Cer, Madhuri Shanbhogue, Iftekhar Naim, Gustavo Hernández Ábrego, Zhe Li, Kaifeng Chen, Henrique Schechter Vera, Xiaoqi Ren, Shanfeng Zhang, Daniel Salz, Michael Boratko, Jay Han, Blair Chen, Shuo Huang, Vikram Rao, Paul Suganthan, Feng Han, Andreas Doumanoglou, Nithi Gupta, Fedor Moiseev, Cathy Yip, Aashi Jain, Simon Baumgartner, Shahrokh Shahi, Frank Palma Gomez, Sandeep Mariserla, Min Choi, Parashar Shah, Sonam Goenka, Ke Chen, Ye Xia, Koert Chen, Sai Meher Karthik Duddu, Yichang Chen, Trevor Walker, Wenlei Zhou, Rakesh Ghiya, Zach Gleicher, Karan Gill, Zhe Dong, Mojtaba Seyedhosseini, Yunhsuan Sung, Raphael Hoffmann, and Tom Duerig. Gemini embedding: Generalizable embeddings from gemini, 2025.

[81] Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025.

[82] Xing Han Lù. Bm25s: Orders of magnitude faster lexical search via eager sparse scoring, 2024.

[83] Gordon V. Cormack, Charles L A Clarke, and Stefan Buettcher. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, page 758–759, New York, NY, USA, 2009. Association for Computing Machinery.

[84] William Webber, Alistair Moffat, and Justin Zobel. A similarity measure for indefinite rankings. *ACM Trans. Inf. Syst.*, 28:20:1–20:38, 2010.

A Methods

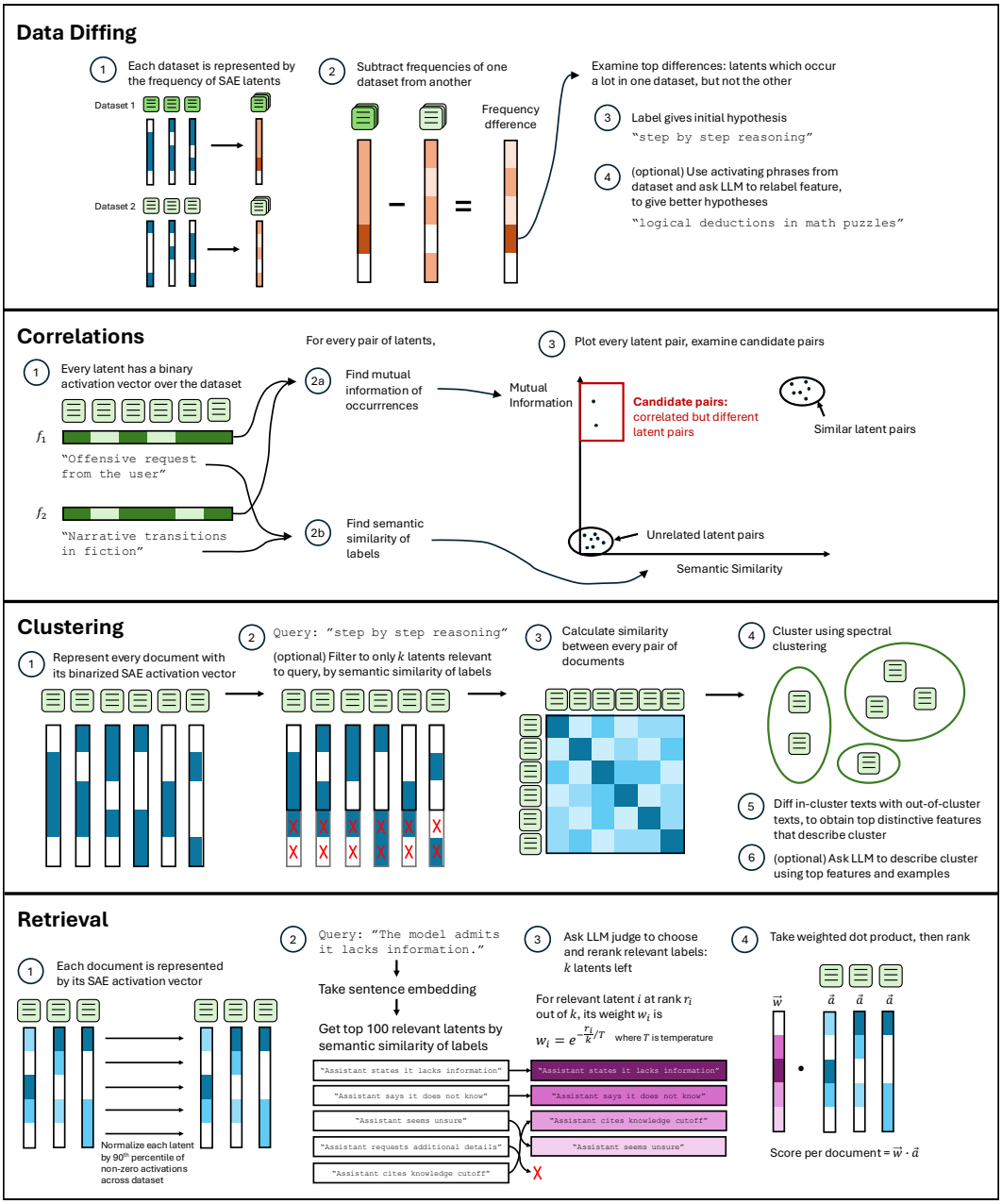

Figure 8: Detailed methodology for each of the four tasks.

B Additional Related Work

B.1 Data Diffing

While semantic embeddings can quantify the degree of difference between two texts or two datasets via cosine similarity, they do not describe *how* the texts are different. Term-based statistics may be able to generate interpretable differences, but may miss out on context. Prior work on describing differences between datasets thus primarily uses LLMs [52; 53].

B.2 Correlations

The problem of finding correlations in datasets is often framed as finding spurious correlations between features and dataset *classes*. For instance, [54] found an SAE feature that predicted a dataset's label of human vs. AI generated text, that primarily fired on periods and punctuation, indicating a potentially non-generalizable correlation. However, finding arbitrary concept-concept correlations in text without any labels is relatively unexplored. Classical approaches can measure correlations between terms [55; 56], and SAEs provide a natural extension of this. Instead of term-term statistics, one can compute latent-latent statistics, where each latent corresponds to a more meaningful and abstract concept than individual words.

B.3 Clustering

Classical NLP represents texts using term based [57] or dense embedding based [7] methods, then apply a standard clustering algorithm (e.g. KMeans [58], spectral clustering [59], HDBSCAN [60]). To guide clusters towards human-specified structure, prior work has used specified pairwise constraints [61; 62], seed examples [63], partial labels [64], feature feedback [65] or post-hoc tuning of clusters [66; 67], sometimes with LLM guidance [68; 69].

B.4 Retrieval

Most retrieval benchmarks focus on question answering and semantic similarity tasks. For example, the query "How many people live in Berlin?" is answered by retrieving the passage with the relevant response. [46] investigates retrieval based on a *description* of the content—for example, the query "a company which is a part of another company" is answered by retrieving a specific instance e.g. "Pecten (company), a subsidiary of Sinopec". We extend this to focus on more abstract queries of implicit properties—properties that are not stated but present in the text.

Representation of texts for retrieval traditionally uses BERT-style embeddings. Modern decoder-only LLM embeddings have recently begun to outperform traditional methods via last-token or latent-attention pooling, instruction formatting, and/or finetuning [70; 71; 72; 73]. We use SAEs as a way to approximate these embeddings, which we expect to contain abstract properties. The interpretability of SAEs also helps us better understand retrieval results—some work has used SAEs trained on semantic embeddings to control retrieval [19; 20], thus it is natural to also use SAEs trained on LLM representations.

C Extended Findings from Applications

C.1 Emerging trends from OpenAI models

We provide additional details on our methodology and results. The OpenRouter IDs of the five models we used are `openai/gpt-3.5-turbo`, `openai/gpt-4-turbo`, `openai/gpt-4o`, `openai/gpt-4.1`, and `openai/gpt-5`.

Extended methodology for finding general qualitative differences. Similar to Section 4.1, we find the frequency of each latent across all five datasets. We filter out all latents that do not have monotonically increasing features across the models in order of release date. Then, we sort by the frequency difference of `openai/gpt-5` and `openai/gpt-3.5-turbo`. We relabel the top 50 latents using the same prompt as in Appendix F, passing in twenty positive-activating samples from `openai/gpt-5` and twenty non-activating samples from 'openai/gpt-3.5-turbo'.

Hypothesis verification for general differences. Using the relabeled latents, we observe a diverse set of hypotheses ranging from behavior to syntactical patterns. We present the full hypotheses here:

1. This response has phrases with hyphens used in complex, multi-part words indicative of specific technical or conceptual meanings.
2. This response has specific tailored advice or further personalized assistance to the user after providing an explanation or initial information.
3. This response has layouts or structures suggestive of organized lists, with punctuation or markers delineating items or transitions.
4. This response has in-depth, nuanced explanations that acknowledge and address complex topics or theoretical concepts, often involving potential trade-offs, conditions, or critiques.

We reuse the same LLM judge prompt as in Section 4.1 to verify the alignment of the hypothesis per response.

Extended methodology for finding correlations. Similar to Section 4.2, we binarize the SAE embeddings for the prompt dataset and each of the model datasets. Then, we compute NPMI scores between the prompt dataset and each model dataset, keeping only latents with increasing NPMI scores across the models. To further narrow the search space, we only consider latents that scored a NPMI of >0.5 and activated in >1% of documents both in one model and the prompts. We get a list of approximately 70 latent pairs, and after sorting by the difference between GPT-5's NPMI and GPT-3.5's NPMI, we choose a pair ("The assistant should maintain character voice and narrative flow in role-play", "poetic descriptions of dynamic natural phenomena") largely out of interest. Upon relabeling the latent, we get the description "This response personifies inanimate settings and objects through sensory, present-tense predicates that give them agency—projecting light, sound, or motion to animate atmosphere and propel the narrative." Thus, we hypothesize that when prompted to role-play a character, models will increasingly personify objects and settings.

Verifying that role-play scenarios trigger object personification. We generate 185 prompts using `GPT-4o` with the following prompt:

```
"""Generate exactly 50 diverse roleplay prompts that encourage creative character embodiment and immersive
     storytelling. Each prompt should:

1. Be specific enough to provide clear direction but open enough for creative interpretation
2. Encourage the respondent to fully embody a character or perspective
3. Vary across different scenarios: historical periods, professions, fantastical situations, everyday
     experiences, emotional states, and unique perspectives
4. Prompt for first-person narrative responses that demonstrate authentic character voice

Format each prompt as a standalone paragraph. Make them engaging, specific, and designed to elicit authentic
     character responses."""
```

Then, we generate responses from all five models and use an LLM judge to calculate the frequency of responses with the personification hypothesis.

D Properties of SAE Latents

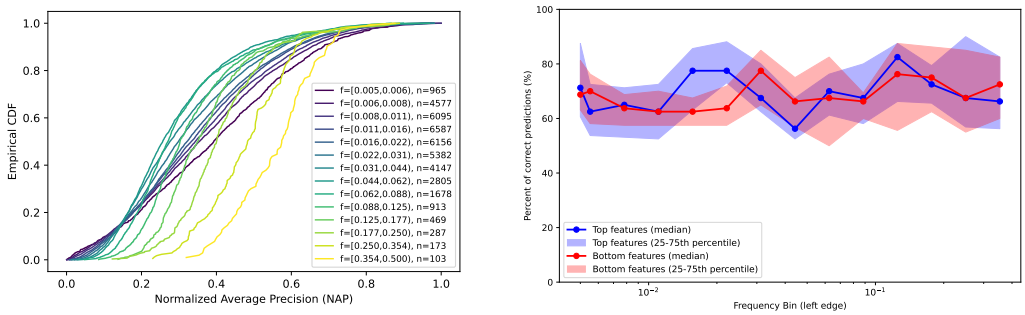

Figure 9: Left: Empirical CDF of normalized average precision of the classifier for latents in each log-frequency range. Right: Automatic interpretability score summary.

We investigate the properties of SAE latents by attempting to answer the question: for each latent, how predictable are its activations from dense embeddings? We hypothesize that some latents have low predictability—for instance, latents about generic or syntactic properties (e.g. "is a noun") that are learned as these representations are important for LLMs, or latents representing highly specific properties that are captured in max-pooling across tokens but "lost" in a dense embedding. For instance, [74] used an LLM to classify "semantic" vs. "structural" SAE latents.

To do this, we train a classifier that predicts a latent's activation $v \in \{0, 1\}$ in a text from the text's dense embedding $\mathbf{s} \in \mathbb{R}^{d_{\text{emb}}}$. We use a 10k sample of ChatbotArena responses. Since the baseline accuracy of predictions, as well as the number of positive training samples, depends on the frequency of the latent, we report metrics by log-spaced frequency bins (frequency f_j calculated on the full corpus). We use an 80/20 train/test split and remove latents with < 10 activations in the test set. For each frequency bin, we fit a one vs. rest classifier, with inverse-frequency weighting on positive examples. We use AdamW and run 3-fold cross-validation to select weight decay using the mean normalized average precision ($NAP_j = \frac{AP_i - f_j^{(\text{val})}}{1 - f_j^{(\text{val})}}$) across all latents. Lastly, we compute NAP_j on the test set and report its empirical CDF for each frequency bin (Figure 9).

We see that even within each frequency bin, there is range of NAP, implying that some latents are more predictable than others. To confirm that this is not simply an artifact of some latents being "bad" (non-monosemantic), we sample 20 latents from the top and bottom predictability deciles, relabel with an LLM, then score these labels (similar to EleutherAI [26], by accuracy of an LLM using the label to predict whether the latent will activate), showing that the predictable vs. unpredictable latents do not seem to differ significantly in quality (Figure 9). We show qualitative examples of "good" labels from the most and least predictable deciles in 4. While it is difficult to determine exactly what types of latents are predictable, and latents may have poor recall on their activating concepts due to phenomena like feature absorption [51], these results qualitatively align with the intuition that some latents—highly specific or generic latents—are less predictable from semantic embeddings.

	Latent description	Autointerp Score (%)
Bin: [0.016, 0.022)	**Most Predictable**	
	German punctuation marks at the end of a sentence or phrase, including periods, commas, colons, and exclamation points, often followed by a new line or a capitalized word	75.0
	Mentions of musical artists, their works, or elements related to music production and performance	82.5
	Discussions about renewable and non-renewable energy sources, including their characteristics, benefits, and drawbacks	77.5
	Words or phrases that are part of a programming language, code, or technical syntax	72.5
	The introduction of a contrasting or alternative idea, often following a statement or concept, and frequently marked by conjunctions or punctuation that signal a divergence or additional consideration.	75.0
	Least Predictable	
	References to color in programming or styling contexts	100
	The act of attempting or making an effort to do something, often implying a challenge or difficulty in achieving the goal	100
	Programming language namespaces, libraries, or modules	75.0
	A statement about a subject's inherent qualities, characteristics, or established facts, often describing its nature, properties, or a state of being that has existed over a period of time	80.0
	Mathematical equations, formulas, or expressions, including variables, constants, and operators, often within a larger problem-solving context	85.0
Bin: [0.062, 0.088)	**Most Predictable**	
	Code syntax for defining or connecting layers in a neural network	85.0
	Concepts related to movement, change, or force, often in a scientific, technical, or social context, including terms like "dynamics," "dynamic," "aerodynamics," and their foreign language equivalents	85.0
	A description of a preceding noun, often a type of, or an example of, a category, and often followed by a verb phrase describing its characteristics or function	75.0
	Religious or spiritual ceremonies, rituals, and practices	90.0
	Mentions of silver, copper, or bronze as materials or elements	87.5
	Least Predictable	
	Fictional or symbolic representations of people, entities, or data elements	80.0
	The definite article "the" followed by a noun phrase that refers to a general concept, abstract idea, or a collective group, often in a descriptive or explanatory context	80.0
	The model's ability to communicate in a specific language, often in response to a user's query about language proficiency or a direct request to switch languages	90.0
	Command line arguments, flags, or parameters	95.0
	Commercial enterprises or economic activities, often in the context of their operations, goals, or interactions with other entities	95.0
Bin: [0.125, 0.177)	**Most Predictable**	
	References to the chemical industry or chemical products	95.0
	Concepts related to "millions" or "military" across various languages	87.5
	Mentions of drugs, medications, or pharmaceutical compounds, including their names, types, or related concepts like development and effects	75.0
	The concept of skills, abilities, or attributes, often in the context of combat, training, or personal characteristics	77.5
	Conditional statements or hypothetical scenarios, often introducing a premise for a subsequent action or consequence	87.5
	Least Predictable	
	The analysis or understanding of a concept, phenomenon, or relationship	75.0
	The concept of a knowledge cutoff date or a fixed end date for information, often in the context of an AI model's training data or a filter's frequency limit	92.5
	References to a Uniform Resource Locator (URL)	97.5
	Phrases that introduce or elaborate on a concept, idea, or example, often appearing after a statement or a list of items, and frequently using words like "for example," "which," "furthermore," "additionally," or "it is also worth noting" to connect to the preceding text.	72.5
	Modal verbs and similar expressions of obligation, necessity, or future action	72.5

Table 4: Sample of latents that are most (top decile) and least (bottom decile) predictable by NAP in each frequency bin with autointerp scores > 70% (i.e. "good" latents).

E Feature Relabeling

Following [26], we relabel latent descriptions.

Scoring latent descriptions. To score the quality of latent descriptions, we pass in ten activated documents and ten non-activated documents, using the following prompt to score a 1 or 0 per document.

```
system_prompt = """
You are an expert at evaluating sparse autoencoder feature descriptions.

You are given a feature description, a POSITIVE sample (where the feature activated, with tokens surrounded by
      << and >>), and a NEGATIVE sample (where the feature did NOT activate, and there should be no << >>
     markers).

IMPORTANT NOTES:
1. The << >> markers indicate where the feature activated, but you should NOT restrict your understanding to
     just those marked tokens. Look at the context BEFORE the marked tokens as well - the preceding tokens
     often provide crucial information about what the feature is detecting.
2. The feature may be responding to a pattern or concept that spans both the context tokens AND the marked
     tokens together.
3. The token <eot_id> is an end-of-sequence (EOS) token and should NOT be considered as a valid feature
     activation. If you see <<eot_id>> in the samples, ignore it as it's just a technical marker for the end
     of text, not a meaningful activation.
4. You shouldn't be trying to infer what the feature description should be from the positive and negative
     samples; rather, you should use the feature description to evaluate the samples.

Feature description:
"{feature_description}"
{prompt_section}
POSITIVE SAMPLE (feature activated, << >> marks WHERE it activated):
{positive_sample}

NEGATIVE SAMPLE (feature did NOT activate, no << >> markers):
{negative_sample}

Your task:
- Evaluate if the feature description accurately describes whether or not the feature activates, considering
     BOTH the context before the << >> markers AND the marked tokens themselves to understand what triggered
     the feature
- Score 1 if the property described by the feature description is clearly present in the positive sample (
     considering both context and marked tokens) and absent in the negative sample.
- Score 0 if the property described by the feature description is not clearly present in the positive sample,
     or if the negative sample also contains the property. If the feature description is not a valid property
      (ex. "feature_#"), mark 0.

Return your answer as a JSON object with exactly these fields:
- "explanation": "<brief explanation for the score, focusing on how the context and marked tokens together
     show the difference between samples>"
- "score": <0 or 1>

Make sure your response is valid JSON that can be parsed directly. Keep the explanation brief (1-2 sentences).
     """
```

Relabeling latents. To relabel latents with more precise descriptions, we pass in 20 activating documents and 20 non-activating documents for an LLM to infer when the latent activates. We use the following prompt:

```
system_prompt = """
You are an expert at evaluating sparse autoencoder feature descriptions.

You are given a feature description, a POSITIVE sample (where the feature activated, with tokens surrounded by
      << and >>), and a NEGATIVE sample (where the feature did NOT activate, and there should be no << >>
     markers).

IMPORTANT NOTES:
1. The << >> markers indicate where the feature activated, but you should NOT restrict your understanding to
     just those marked tokens. Look at the context BEFORE the marked tokens as well - the preceding tokens
     often provide crucial information about what the feature is detecting.
2. The feature may be responding to a pattern or concept that spans both the context tokens AND the marked
     tokens together.
3. The token <eot_id> is an end-of-sequence (EOS) token and should NOT be considered as a valid feature
     activation. If you see <<eot_id>> in the samples, ignore it as it's just a technical marker for the end
     of text, not a meaningful activation.
4. You shouldn't be trying to infer what the feature description should be from the positive and negative
     samples; rather, you should use the feature description to evaluate the samples.

Feature description:
"{feature_description}"
{prompt_section}
POSITIVE SAMPLE (feature activated, << >> marks WHERE it activated):
{positive_sample}

NEGATIVE SAMPLE (feature did NOT activate, no << >> markers):
{negative_sample}

Your task:
```

```
- Evaluate if the feature description accurately describes whether or not the feature activates, considering
      BOTH the context before the << >> markers AND the marked tokens themselves to understand what triggered
      the feature
- Score 1 if the property described by the feature description is clearly present in the positive sample (
      considering both context and marked tokens) and absent in the negative sample.
- Score 0 if the property described by the feature description is not clearly present in the positive sample,
      or if the negative sample also contains the property. If the feature description is not a valid property
       (ex. "feature_#"), mark 0.

Return your answer as a JSON object with exactly these fields:
- "explanation": "<brief explanation for the score, focusing on how the context and marked tokens together
      show the difference between samples>"
- "score": <0 or 1>

Make sure your response is valid JSON that can be parsed directly. Keep the explanation brief (1-2 sentences).
      """
```

F Additional Results—Dataset Diffing

F.1 LLM baseline details

LLM baseline. Our baseline is adapted from the hypothesis discovery stage of [22]. Given two datasets (or one dataset vs. multiple datasets), our baseline first finds differences between document pairs from each dataset using the following prompt:

```
Analyze the differences between Model A and multiple Model B responses.

**User Prompt:**
{prompt}

**Model A Response:**
{model_a_response}

**Model B Responses:**
{model_b_section}

1. Properties/capabilities that Model A has but NONE of the Model B
    responses have

For each difference, provide a JSON object with:
- "category": The type of difference (e.g., "Style", "Content", "
    Technical", "Reasoning", "Accuracy")
- "property": Specific property being compared
- "difference_type": Either "unique_to_a" (present in A but none of B
    models) or "common_to_all_b" (present in all B models but not A)
- "impact": "Low", "Medium", or "High"
- "description": Brief explanation of the difference

Return your analysis as a JSON array of difference objects.
```

Once we have our difference objects, we aggregate them into hypotheses using Gemini 2.5 Flash:

```
You are an expert AI researcher analyzing behavioral differences between
    two language models.
You have been given a dataset of differences from {num_pairs} analyzed
    response pairs.

Query: {query}

Differences: {differences}

Based on the provided data, identify at most {num_hypotheses} significant
     differences that respond to the query. I'm looking for differences
    of the format Model A/B is more X than Model B/A, where X is the
    difference. For each difference, provide:

1. **Description**: Describe a response that would validly have property
    X. Start with "This response .." Use 1-2 sentences to clearly and
    specifically describe the property, such that using this description
    could be used to identify the property on its own. Do not mention the
     model names.
2. **Detailed Description**: A detailed explanation of what the
    difference is and why it's significant
3. **Model A/B**: The model that exhibits this property more
4. **Percentage Difference**: An estimate of how much more frequently
    Model A exhibits this behavior compared to Model B. If the property
    is more frequent in Model A, the percentage difference should be
    positive. If the property is more frequent in Model B, the percentage
     difference should be negative.
5. **Examples**: 2-3 specific examples that demonstrate this difference
```

```
Make hypotheses specific and clear. Provide at most {num_hypotheses}
    differences in the following JSON format:

{{"differences": [
  {{
    "description": "Clear description of the property",
    "detailed_description": "Detailed explanation of the difference and
       why it's significant",
    "model_a_b": "Model A|Model B",
    "percentage_difference": "X% more present in Model A",
    "examples": [
      {{
        "prompt": "Original prompt text or description",
        "explanation": "Why this example demonstrates the difference"
      }}
    ]
  }}
]}}
```

We use this system prompt for Gemini 2.5 Flash:

```
   You are an expert model behavior analyst. Your task is to meticulously
        compare two model responses to a given user prompt and identify
       unique qualitative properties belonging to one model but not the
       other. For each significant property, you must determine if it's
       more likely a **general trait** of the model or a **context-
       specific** behavior triggered by the current prompt.

**Prioritize conciseness and clarity in all your descriptions and
    explanations.** Aim for the most impactful information in the fewest
    words.

You will be provided with:
1. **User Prompt:** The original prompt given to both models.
2. **Model A Name:** The identifier for Model A.
3. **Model A Response:** The response from Model A.
4. **Model B Name:** The identifier for Model B.
5. **Model B Response:** The response from Model B.

**Your Goal:**
Produce a JSON list of objects. Each object will represent a single
    distinct property observed in one model's response that is notably
    absent or different in the other's. Focus on identifying key areas of
     distinction, and the individual property observations in the output
    list (e.g., Model A's formal tone would be one entry, Model B's
    casual tone would be another related entry). As these are very common
     and easy to measure with heuristics, please do not include
    properties like "Model A is more concise than Model B". If
    applicatble, make sure to also include properties revolving around
    the models reasoning, interpretation of the prompt/intent, and
    potential reason for errors if they exist.

**Focus on Meaningful Properties:**
Prioritize properties that would actually influence a user's model choice
     or could impact the model's performance. This could include but is
    not limited to:
* **Capabilities:** Accuracy, completeness, technical correctness,
    reasoning quality, domain expertise
* **Style:** Tone, approach, presentation style, personality, engagement
    with the user, and other subjective properties that someone may care
    about for their own use
* **Error patterns:** Hallucinations, factual errors, logical
    inconsistencies, safety issues
* **User experience:** Clarity, helpfulness, accessibility, practical
    utility, response to feedback
```

```
* **Safety/alignment:** Bias, harmful content, inappropriate responses,
    and other safety-related properties
* **Tool use:** Use of tools to complete tasks and how appropriate the
    tool use is for the task
* **Thought Process:** Chain of reasoning, backtracking, interpretation
    of the prompt, self-reflection, etc.

**Avoid trivial differences** like minor length variations, basic
    formatting, or properties that don't meaningfully impact the models
    capability or the user's experience.

**Definitions:**
* **General Trait:** Reflects a model's pattern of behavior across a
    distribution of prompts.
   * *Think:* Could a model have this property in a different prompt from
       the one provided? If so, then it is general. If not, then it is
       context-specific.
* **Context-Specific Difference:** If the property is a direct reaction
    to *this current prompt*, then it is context-specific.
   * *Think:* Is this property a direct reaction to *this current prompt
       *? If so, then it is context specific. If not, then it is general.
* **Impact:** How much does this property impact the user's experience?
   * *Think:* Is this property a major factor in the user's experience?
       Would the average user care to know that this property exists?
   * **Low:** Minor stylistic differences that most users wouldn't notice
       or care about
   * **Medium:** Noticeable differences that might influence preference
       but aren't deal-breakers
   * **High:** Significant differences that could strongly influence
       model choice (e.g., errors, major capability gaps, strong
       stylistic preferences)
* **User Preference Direction:** Which type of user might prefer this
    property?
   * *Think:* Does this property appeal to specific user types or use
       cases?
   * **Capability-focused:** Users who prioritize accuracy, completeness,
       technical correctness
   * **Experience-focused:** Users who prioritize style, tone,
       presentation, ease of use, or users who focus on very open-ended
       tasks
   * **Neutral:** Property doesn't clearly favor one user type over
       another
   * **Negative:** Property that most users would find undesirable (
       errors, poor quality, etc.)
* **Contains Errors:** Does either model response contain errors?
   * *Think:* Are there factual errors, hallucinations, or other strange
       or unwanted behavior?
* **Unexpected Behavior:** Does the model's response contain highly
    unusual or concerning behavior? If true then a developer will analyze
     these responses manually.
   * *Think:* Does this involve offensive language, gibberish, bias,
       factual hallucinations, or other strange or unwanted behavior?

**JSON Output Structure for each property (if no notable properties exist,
     return empty list. Phrase the properties such that a user can
    understand what they mean without reading the prompt or responses.)
    :**```json
[
  {
    "model": "Model A|Model B",
    "property_description": "Brief description of the unique property
       observed in this model response (max 2 sentences, only give the
       property itself - remove any beginning or ending phrases like 'The
        response is...', 'The model has...', etc.)",
    "category": "1-4 word category",
```

```
    "evidence": "Direct quote or evidence from the specified model",
    "type": "General|Context-Specific",
    "reason": "Brief justification for this property, noting its absence/
        difference in the other model (max 2 sentences)",
    "impact": "Low|Medium|High",
    "user_preference_direction": "Capability-focused|Experience-focused|
        Neutral|Negative",
    "contains_errors": "True|False",
    "unexpected_behavior": "True|False"
  }
]
```

F.2 Ground truth evaluation

Ground truth datasets. We show how we generated our datasets with known differences in Table 5. We show the top difference for a few representative categories in Table 6.

Dataset	Description
Synthetic: tone changes	We randomly sample 500 responses from Chatbot Arena [75] and prompt GPT-4o to convert the base response to 13 different tones (e.g., "friendly-and-personable"). We diff the modified and base responses, aiming to recover the tone.
Real-world: movie genre differences	We use IMDB-reviews [33], which contains movie descriptions with genre labels. We diff the descriptions from within each genre with 500 randomly sampled descriptions outside the genre, aiming to recover the genre.

Table 5: **Datasets used for ground-truth evaluation** in data diffing.

Tone changes		Movie genre differences	
Tone	**Top Latent**	**Genre**	**Top Latent**
Casual	Casual/cool slang and informal speech patterns	Action	Action movie plot developments and dramatic confrontations
Organized	Q/A transition points in educational content	Romance	Will they/won't they writing tropes
Imaginative	Groups gathering to share stories and experiences, especially in atmospheric or mysterious contexts	War	Soldiers experiencing the psychological and physical hardships of war
Funny	Young adult recreational and social activities	Crime	High-stakes heists involving valuable items from secure locations
Safety conscious	Emphasizing ethical concerns and safety warnings	Documentary	Documentary films and series being discussed

Table 6: **Top latent with the biggest frequency difference** for tone changes (left) and movie genre differences (right). We present five representative examples for the synthetic and movie datasets.

F.3 Comparing model outputs

Hypotheses for single model vs. other frontier models. We present the generated hypotheses for Grok-4 using the SAEs (Table 7) and LLM baseline (Table 8). The frontier models we compare against in Section 4.1 are Grok-4, GPT-OSS-120B, Gemini 2.5 Pro, Claude Opus 4.1, Claude Sonnet 4, GPT 5, Gemini 2.5 Flash, Llama 4 Maverick, Deepseek R1, Gemini 2.5 Pro, Qwen3-235b, and Qwen3-235b thinking.

Coverage experiments. As generated hypotheses may overlap semantically, we conduct a "coverage experiment" to evaluate how well the hypotheses overall characterize the unique qualities of a target dataset (Figure 10).

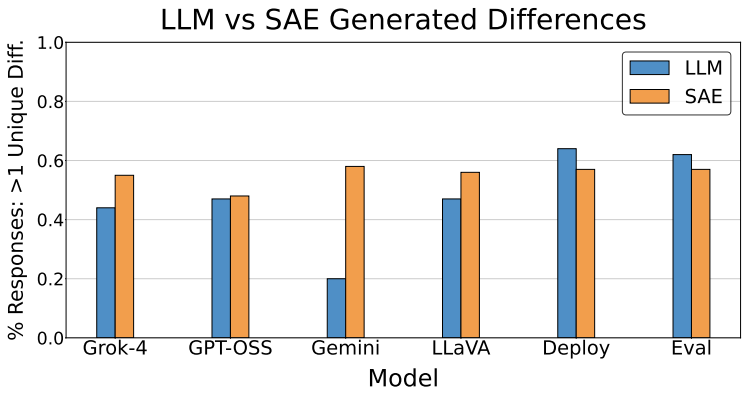

Figure 10: **Coverage of generated hypotheses overall**. We compute the % of responses that have at least one hypothesis with the "target" dataset uniquely verified. The generated hypotheses for SAEs have greater coverage of the unique qualities of target datasets over pure LLMs on multi-model setups. SAEs score similarly for two-model cases.

Hypothesis	**Grok-4 Coverage**	**Nearest Model**	**Nearest Model Coverage**	**Valid?**
This response needs clarification from the user. It explicitly asks the user to provide more context or details to refine the answer, often indicating that the initial query was ambiguous or lacked sufficient information	59.8	GPT-5	28.8	Y
This response offers to help with additional or different topics, indicating a willingness to continue the conversation beyond the immediate query	79.4	GPT-5	33.2	Y
This response includes a disclaimer or qualification about the reliability or subjectivity of the information provided, often using phrases like 'subjective ideas' or 'not a doctor'.	20.6	qwen3-235b-thinking	9.8	Y
This response politely offers continued help after rejecting inappropriate requests, or is diplomatically accommodating while explaining capabilities and limitations.	42.9	qwen3-235b-thinking	22.7	Y
This response acknowledges failure to meet user expectations or indicates that its previous understanding was incorrect, often using phrases like 'not what you meant' or 'not spot-on'.	19.5	qwen3-235b	4.6	Y
This response uses polite phrases requesting or offering additional information, often using phrases like 'feel free to provide it' or 'let me know'.	62.8	gpt-oss-120b	22.0	Y
This response makes commitments or promises to perform actions, often using phrases like 'I'll break this down' or 'I'll explain'.	55.9	gemini-2.5-pro	50.0	Y
This response is making an open invitation for further interaction, often using phrases like 'feel free to ask for more tailored tips'.	76.6	GPT-5	26.9	Y
This response is making a logical deduction based on given information, often using phrases like 'Based on the provided C# code' or 'Based on the wording of the riddle'.	30.9	qwen3-235b-thinking	20.9	Y

Table 7: Generated hypotheses for Grok-4 using SAEs.

Hypothesis	Grok-4 Frequency	Nearest Model	Nearest Model Frequency	Valid?
This response proactively engages the user by asking clarifying questions, offering further assistance, or anticipating alternative interpretations.	75.6	GPT-5	33.2	Y
This response uses emojis, informal greetings/closings, or adopts a playful/humorous tone.	38.3	qwen3-235b-thinking	36.7	Y
This response explicitly states its AI persona, limitations, or provides meta-commentary on its own output or strategy.	30.9	qwen3-235b-thinking	9.9	Y
This response provides highly structured and detailed explanations, often using multi-level headings, numbered steps, or dedicated sections for context, troubleshooting, or specific examples.	64.4	qwen3-235b-thinking	68.8	N
This response adheres strictly to highly specific, sometimes niche, persona or formatting instructions, even if it means ignoring other instructions.	33.3	qwen3-235b-thinking	55.3	N
This response provides only the final answer to a mathematical problem or a direct response to a simple query, without showing detailed steps or explanations.	4.4	gpt-oss-120b	10.2	N
This response explicitly refuses or significantly mitigates problematic content, or adds ethical disclaimers, when faced with prompts requesting explicit or harmful material.	8.5	qwen3-235b-thinking	10.4	N
This response fails to generate a coherent response, producing a technical error message or an incorrect answer without explanation.	1.4	qwen3-235b-thinking	1.2	N
This response demonstrates the ability to perform real-time web browsing or generate live links.	2.0	qwen3-235b-thinking	6.5	N

Table 8: Generated hypotheses for Grok-4 using the LLM baseline. We use an LLM judge to evaluate the frequency each hypothesis is present in our frontier model responses. We define a "valid" hypothesis as one where the difference between Grok-4's frequency and the highest frequency of other frontier models to be >1%.

G Additional Results—Correlations

Distribution of Latent Pairs. We expect that generally, latent pairs with similar labels are conceptually related and thus co-occur, while latents with dissimilar labels are unrelated and should not co-occur. On ChatbotArena responses, for latents with frequency between 0.01 and 0.9, we find every pair's NPMI and label similarity, and plot the histogram in Figure 11. Most latents are dissimilar and do not co-occur, and we highlight a potential "interesting" region where dissimilar latents have high co-occurrence.

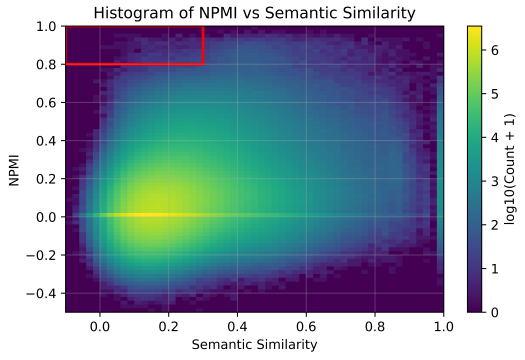

Figure 11: 2D histogram of latent pairs' NPMI and semantic similarities, from the ChatbotArena responses dataset.

G.1 Ground truth evaluation

Injected correlations. For each of the three types of texts, we inject them into the Pile, then find relevant pairs in the candidate group using keywords in the labels (Table 9). For the "conservative and academic" correlation, we found "style" pairs where economic/political latents co-occurred

with academic style latents, as well as "slant" pairs where economic/political latents co-occurred with conservative language latents. We see in Figure 12 that relevant pairs are surfaced among the candidate group. As the number of injected texts increases, the percent of pairs that are relevant increases.

Injection	Latent 1 Relevant	Latent 2 Relevant
croatian-emoticons	croatian, russian, slavic	emoticon, emoji
baseball-slang	valley girl, slang, endearment	game, sport, baseball
conservative-academic_style	economic, political, business	academic, formal
conservative-academic_slant	economic, politic, business	communis, free, libert, interven, interfer

Table 9: Keywords used to judge if a latent pair is relevant to the injected correlations.

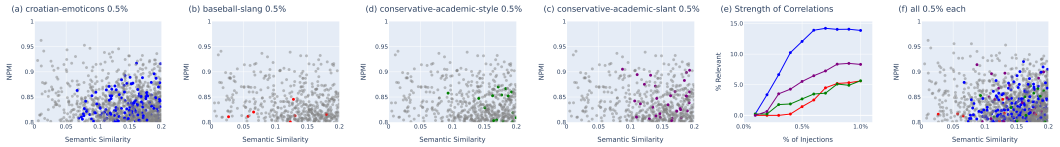

Figure 12: (a)-(d) We plot the candidate group of pairs (NPMI > 0.8, semantic similarity < 0.2) for each type of text injected, with 0.5% of texts being injected texts. Relevant pairs are colored. (e) We show the proportion of relevant pairs in the candidate group for different injection levels 0.1%-1%. (f) We inject all 3 texts at once.

Injected correlations baseline. Due to context window limits, we shuffle and batch the dataset into 5 batches, and query the LLM each time for interesting correlations. While many injected correlations were discovered in at least one batch, some were not discovered at all.

Injection \ Found		croatian-emoticons	baseball-slang	conservative-academic_style	conservative-academic_slant
croatian-emoticons	0.2%	1/5	-	-	-
	0.5%	2/5	-	-	-
baseball-slang	0.2%	-	2/5	-	-
	0.5%	-	5/5	-	-
conservative-academic	0.2%	-	-	3/5	0/5
	0.5%	-	-	2/5	1/5
all	0.2%	0/5	0/5	1/5	1/5
	0.5%	1/5	3/5	3/5	3/5

Table 10: Success rate of Gemini 2.5 Flash identifying the injected correlations.

G.2 Finding real-world correlations

CivilComments analysis. We show a sample of interesting latent pairs in Table 11. To verify each hypothesis, for each latent pair (i, j), we infer a natural language hypothesis for the two concepts present (C_i, C_j). For instance, latents "offensive request from the user" and "conjunctions and prepositions in religious comparative discourse" suggests C_1: "offensive content or language" and C_2: "mentions of religion" co-occur. For each C_i, we ask an LLM judge if that concept is present in each text, as a ground truth label (J.2), then use those labels to compute $P(C_1|C_2)$ and $P(C_2|C_1)$, which should be high in at least one direction if the correlation is present.

CivilComments baseline. We pass all 5k comments into an LLM and find the following hypotheses (full responses condensed).

Gemini 2.5 Flash:

```
1. ""Religious Authority"" & ""Sexual Misconduct/Abuse""
2. ""Trump/Trump Administration"" & ""Mental Instability/Lack of
    Intelligence""
```

Latent 1	Latent 2	NPMI	$\mathbf{P(1|2)}$	$\mathbf{P(C_1|C_2)}$	$\mathbf{P(2|1)}$	$\mathbf{P(C_2|C_1)}$
Offensive request from the user	Conjunctions and prepositions in religious comparative discourse	0.625	0.548	0.409	0.403	0.123
Offensive request from the user	Atheism, secularism and non-religious worldviews	0.631	0.703	0.284	0.600	0.008
Offensive content detection for prejudicial statements	Black as a formal technical or taxonomic term	0.601	0.430	0.336	0.774	0.037
The assistant should write in Trump's speaking style	Enumerating implemented political policies and actions	0.600	0.518	0.560	0.166	0.454
Offensive request from the user	Third-person feminine references	0.628	0.404	0.542	0.372	0.133
Potentially problematic racial content involving white people	Black holes in scientific/astronomical discussions	0.695	0.475	0.538	0.414	0.287

Table 11: Interesting latent co-occurrences in CivilComments, discovering bias. $P(1|2)$ and $P(2|1)$ refer to conditional probabilites of the latents, while $P(C_1|C_2)$ and $P(C_2|C_1)$ refer to that of the underlying concepts verified by an LLM.

```
3. ""Climate Change/Environmentalism"" & ""Political/Economic Critique""
4. ""Media/Journalism"" & ""Bias/Fake News/Propaganda""
```

Gemini 2.5 Pro:

```
1. Pop Culture References and Political/Theological Critique
2. High-Concept Philosophy and Mundane Personal Anecdotes
3. Niche Technical/Financial Jargon and Broad Social/Political Debates
4. Unexpected Ideological and Cultural Pairings
5. Food/Cuisine and Unrelated Political/Social Commentary
```

Pile analysis. We show a sample of interesting latent pairs in Table 12. The first three correspond to features found in StackExchange discussions, and the last two correspond to Wikipedia articles with category metadata. Indeed, in our 10k sample of the Pile, 1806 texts were from StackExchange and 1069 were from Wikipedia.

Latent 1	Latent 2	NPMI	$\mathbf{P(1|2)}$	$\mathbf{P(2|1)}$
The start of a formal question in structured Q&A formats	First person descriptions of previous attempts and actions	0.706	0.895	0.583
The start of a formal question in structured Q&A formats	Persistence of unwanted behavior despite attempted fixes	0.738	0.926	0.614
The start of a formal question in structured Q&A formats	The assistant is explaining code implementation details	0.708	0.921	0.563
Category membership and classification relationships	Biographical dates and milestones in formal biographies	0.705	0.594	0.792
Category membership and classification relationships	Field-value separator tokens in structured formats	0.707	0.766	0.557

Table 12: Interesting latent co-occurrences in the Pile, discovering regularities in text types.

Pile baseline. We pass in the 10k sample of the Pile in 10 chunks, obtained a list of correlations for each subsample, and asked another LLM to check the analysis for the StackExchange and Wikipedia correlations (J.2). The StackExchange correlation was found in 5/10 subsamples, while the Wikipedia correlation was found in none.

Tulu baseline. We pass in the 10k sample of tulu-3-sft-mixture in 10 chunks, obtained a list of correlations for each subsample, and asked another LLM to check the analysis for the math question and hope correlation (J.2), and did not find it. We show a (truncated) sample correlation report below.

```
1. **Mathematical Problem Solving & Python/SymPy Code Generation**: Many
    prompts ask for solutions to complex mathematical problems (e.g.,
    calculus, algebra, probability, geometry). The unexpected co-
    occurrence is the frequent and consistent generation of Python code
    using the sympy library (or math module) within the response to solve
     these problems...
2. **Ethical Dilemmas & Refusal to Generate Harmful Content**: Prompts
    frequently present scenarios that involve unethical, illegal, or
    harmful actions (e.g., promoting self-harm, creating fake documents,
    manipulating people). The unexpected co-occurrence is the AI's
    consistent and explicit refusal to generate such content, often
    accompanied by a detailed explanation of its ethical guidelines and a
     redirection to positive alternatives or professional help...
3. **Specific Character role-play & AI Self-Awareness/Limitations**:
    Prompts often ask the AI to adopt a specific persona (e.g.,
    screenwriter, creative director, historian, priest, detective). An
    unexpected co-occurrence is the AI's tendency to break character or
```

```
    explicitly state its limitations as an AI *within* the role-play,
    even when the prompt doesn't directly ask for it. This creates a meta-
    narrative where the AI is both playing a role and acknowledging its
    own nature...
4. **Language Translation & Specific Cultural/Linguistic Nuances**: Many
    prompts ask for translation between various languages. The unexpected
     co-occurrence is the AI's ability to not just translate, but also to
     explain specific cultural or linguistic nuances, or even correct the
     prompt's assumptions about language, demonstrating a deeper
    understanding beyond mere word-for-word translation...
5. **Creative Writing/Storytelling & Specific Formatting/Word Count
    Constraints**: Prompts often ask for creative writing (e.g., stories,
     parables, dialogues, poems, articles). The unexpected co-occurrence
    is the imposition of highly specific and sometimes unusual formatting
     or word count constraints (e.g., "exactly 3 paragraphs," "each
    sentence must include X word exactly Y times," "no commas," "all caps
    ")...
...
```

H Additional Results—Clustering

Filtering latents for targeted clustering. We tried many approaches to filter latents to an area of interest using their labels (e.g. semantic search with dense embeddings). What ultimately worked the best was the following: we (1) give an LLM a query e.g. "I have a dataset of news articles. I want to cluster them based on the temporal framing of the article.", (2) ask the LLM to generate phrases (J.3), (3) find k latents relevant to each phrase by semantic similarity, and (4) filter the SAE embedding to the union of all those latents.

Successful targeted clustering in synthetic datasets. We show successful SAE clustering in the synthetic news dataset in Figure 13.

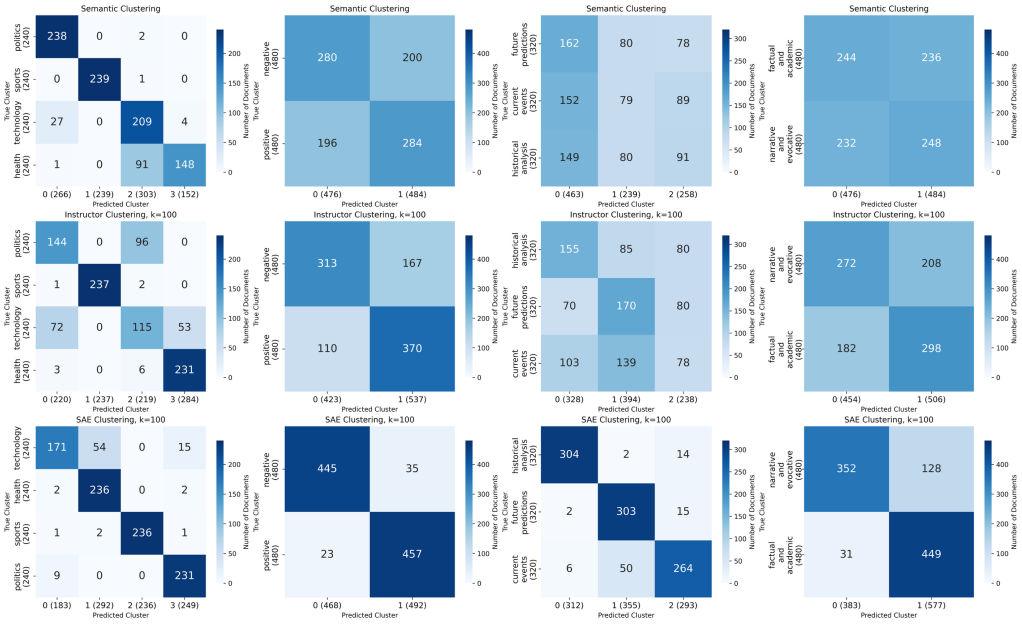

Figure 13: Semantic (top row), Instructor (middle row) and SAE (bottom row) clustering results: (1) topic (2) sentiment (3) temporal framing and (4) writing style. Mappings from clusters to true labels are found with the Hungarian algorithm [76].

Failure to recover ground truth labels for sentiment and emotion clustering. Since SAEs are not trained to represent similarity, we may not obtain "desired" clusters for a dataset with ground-truth

cluster labels. For instance, if an SAE has learned many more "sadness" latents than "surprise" latents, clustering may distinguish between different types of "sadness" more than between "sadnesss" and "surprise".

Figures 14 and 15 show the failure of both embedding baselines and SAEs to align exactly with ground truth labels. While finetuned models (for sentiment/emotion) do achieve good performance on these tasks, we do not expect these general purpose embedding baselines to align with ground-truth labels. For our SAE method, we were unable to find a good combination of queries and k.

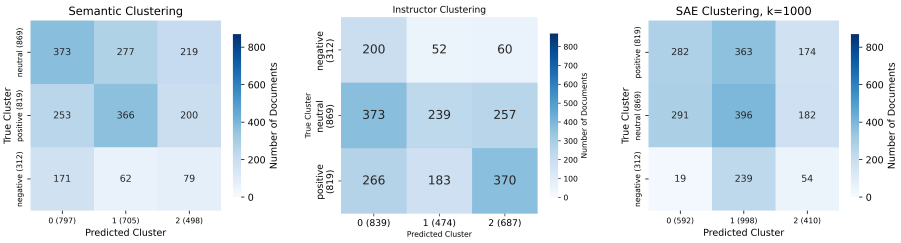

Figure 14: Twitter sentiment [77] clustering results.

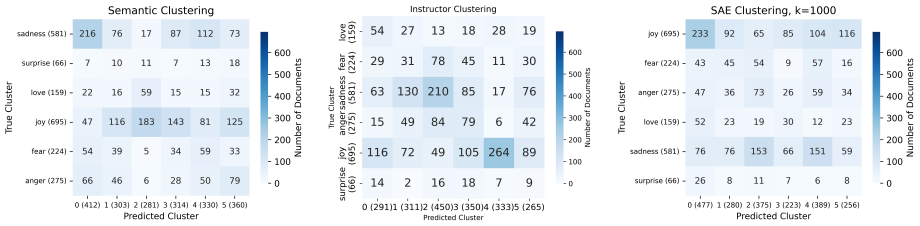

Figure 15: Twitter emotion [78] clustering results.

Clustering prompts, responses and the Pile. We show the per-cluster accuracies for dense embedding and SAE clustering, on ChatbotArena prompts, responses and the Pile, in Figure 16. We see that the SAE clusters have comparable per-cluster accuracies with embeddings, with generally higher variance across clusters. This suggests the clusters are similarly valid—the SAE indeed groups similar texts together.

We show qualitative examples of cluster descriptions in Tables 13-15. Since these datasets are highly diverse, we show results from $n_{\text{clusters}} = 50$, randomly sampling one cluster per accuracy quantile. In these cases, the SAE cluster descriptions are similar in style to semantic cluster descriptions.

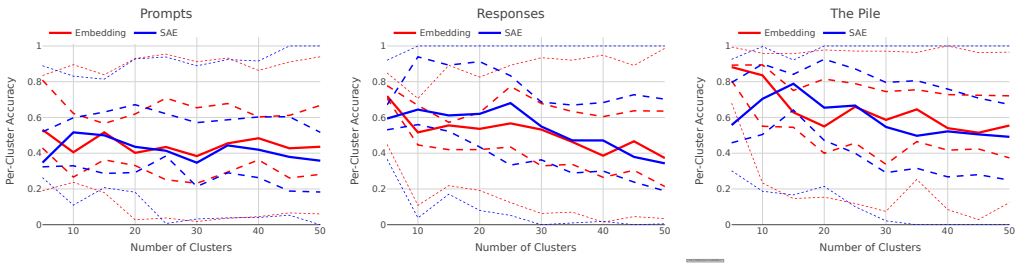

Figure 16: Per-cluster accuracies for different n_{clusters} for prompts, responses and the Pile. The solid lines are the median, dashed lines the interquartile range and dotted lines the range.

Dense Embedding	Acc.	SAE Embedding	Acc.
Simple "Hello World" style Python code requests	0.060	Simple "What is the capital of..." questions	0.000
Debating who is the best athlete	0.173	Single-topic prompts	0.089
Questions about numbers and their properties	0.274	Requests to write a poem in German	0.179
Questions and stories about cats and dogs	0.313	"What is..." questions in German	0.237
Math word problems involving time and rates	0.412	Informal greetings in English and Spanish	0.345
Food recipes and cooking instructions	0.482	Probing the AI's knowledge on specific topics	0.387
Generating video game ideas and recommendations	0.613	Questions and requests in Russian and Polish	0.448
Questions about world history and historical events	0.695	Writing Python scripts for specific tasks	0.529
Simple greetings and conversation starters	0.797	Complex instructions for AI reasoning and persona before a task	0.798
Requests for jokes	0.939	Recommending films similar to specific video games	1.000

Table 13: Example clusters from ChatbotArena prompts.

Dense Embedding	Acc.	SAE Embedding	Acc.
Defining "Machine Learning"	0.034	Presenting Code Snippets on Request	0.005
Stating the Current US President	0.123	Concise Answers to Factual Questions or Riddles	0.082
Inappropriate or Sexually Suggestive Narratives	0.211	Discussing Philosophical and Abstract Concepts	0.164
Solving Basic Algebraic Equations	0.263	Generating Numbered Lists of Items	0.256
Standard Assistant Greeting	0.333	Repetitive or Malformed Lists and Text	0.302
Business and Workplace Productivity Strategies	0.391	Step-by-Step Recipes and Workout Plans	0.391
Numbered Lists of Self-Help and Wellness Advice	0.596	Explanations in German	0.589
Solving Simple Math Word Problems	0.646	Simple Arithmetic Calculations	0.714
Providing Code Snippets	0.770	Original Poetry with Rhyme and Meter	0.907
Assistant Expressing Confusion and Requesting Clarification	0.987	Identifying Capital Cities	1.000

Table 14: Example clusters from ChatbotArena responses.

Dense Embedding	Acc.	SAE Embedding	Acc.
Unity Engine Asset Metadata Files	0.124	Abbreviated or Incomplete User Input	0.000
Research Abstracts on Molecular Biology and Genetics	0.320	Event Announcements and Local News Snippets	0.073
Celebrity Gossip and Lifestyle Articles about Female Public figures	0.370	jQuery and JavaScript Code Debugging and Implementation Questions	0.236
Research Abstracts on Chemical Synthesis and Characterization	0.418	Extremely Brief and Ambiguous User Inputs	0.333
JavaScript/Node.js Modules and Configuration Files	0.488	News Reports on Political and Social Events	0.408
Short Biographies of Politicians and Public Figures	0.569	Technical Q&A with Code and System Configuration Issues	0.554
Scientific Abstracts on Cognitive Neuroscience and Neuropsychology	0.623	Programming Language Test Files and Boilerplate Code	0.619
Short Biographies of Athletes	0.746	Federal Court of Appeals Case Citations	0.731
Clinical Studies on Surgical Procedures and Outcomes	0.847	Advanced Mathematical Problem Solving and Proofs	0.944
US Federal Court Case Filings and Orders	0.966	Zoological Species Descriptions	1.000

Table 15: Example clusters from The Pile.

Quantifying if the SAE found novel structure. We compute the conductance [79] of *SAE* clusters using their within-cluster and out-of-cluster affinities in *dense embedding* space, finding the z-score compared to random clusters (lower conductance = "tighter" cluster). We see in our clustering results that generally, SAE clusters are much less tight (less negative z-score) in dense embedding space, thus, the SAE has indeed found novel structure.

Movie descriptions: Identifying description styles. We cluster IMDB movie descriptions using all SAE latents, and see that SAEs predominantly group texts by the way it is written, rather than semantic content.

GSM8k: Identifying reasoning patterns. We show detailed results from dense embedding clustering (Table 17), SAE clustering with all latents (Table 18) and SAE clustering with reasoning latents (Table 19). The number of clusters is chosen based on human judgement of cluster quality, however, results are qualitatively similar across similar cluster numbers.

Semantic	Acc.	Z	SAE	Acc.	Z
Stories of Soldiers and Conflict in Wartime	0.544	-17.1	Movie plot summaries starting with "A..." or "An..."	0.975	-14.6
Unlikely Bonds and Romantic Connections	0.195	-26.9	Movie plot summaries about duos or small groups, starting with a number	0.618	-9.5
Science Fiction and Fantasy Adventure Movie Plots	0.791	-18.6	Movie plot summaries that start by naming the main character	0.754	0.63
Character-Driven Dramas about Personal and Familial Struggles	0.778	-18.2	Movie plot summaries that begin by establishing the setting or time period	0.576	-4.85
Crime, Heist, and Detective Thriller Movie Plots	0.873	-31.5	Movie plot summaries beginning with the phrase "The story of..."	0.405	-6.38

Table 16: Clusters of IMDB movie descriptions, $n_{\text{clusters}} = 5$

LLM Relabel	Top Features	Top Example	Acc.	Z
Math word problems involving time, distance, and speed	Numbers appearing in step-by-step mathematical calculations and explanations (0.649), Time units used to specify durations (0.522), Math word problems about running distances and speeds (0.469)	He got $2 * 6 = \ll 2 * 6 = 12 \gg$ 12 hours the first 2 days. The next 2 days he gets $2 * 10 = \ll 2 * 10 = 20 \gg$ 20 hours. So he got $12 + 20 = \ll 12 + 20 = 32 \gg$ 32 hours. #### 32	0.417	-15.736
Financial math problems about costs, purchases, and change	Formatting tokens for displaying structured numerical data and calculations (0.909), Financial and business metrics with numerical values (0.895), Currency symbols ($, £, €, R) (0.895)	He bought $5 * 4 = << 5 * 4 = 20 >>$ 20 shirts The discount saves him $15 * .2 = \$ \ll 15 * .2 = 3 \gg$ 3 per shirt So each shirt cost $15 - 3 = \$ \ll 15 - 3 = 12 \gg 12$ So the total order cost $20 * 12 = \$ \ll 20 * 12 = 240 \gg 240$ Tax added $240 * .1 = \$ \ll 240 * .1 = 24 \gg 24$ So he paid $240 + 24 = \$ \ll 240 + 24 = 264 \gg 264$ #### 264	0.938	-45.948
Math problems about counting quantities of objects	Expressing numerical approximations and ranges (0.416), Logical connectors and punctuation in step-by-step explanations (0.340), Countable units and discrete elements being enumerated (0.332)	There are $23 + 16 = \ll 23 + 16 = 39 \gg$ 39 beads in the bowl Dividing them into 3 equal parts give $39/3 = \ll 39/3 = 13 \gg 13$ beads each Before doubling the number of beads, she had $6/2 = \ll 6/2 = 3 \gg 3$ beads left in each part. Therefore she had removed $13 - 3 = \ll 13 - 3 = 10 \gg 10$ beads from each part. #### 10	0.981	-22.033

Table 17: Dense embedding clustering, $n_{\text{clusters}} = 3$.

LLM Relabel	Top Features	Top Example	Acc.	Z
Solving financial word problems involving currency	Dollar sign used as a reference operator or prefix (0.767), Currency symbols ($, £, €, R$) (0.762), Large round numbers in financial contexts (0.760)	He bought $5 * 4 = \ll 5 * 4 = 20 \gg = 20$ shirts The discount saves him $15 * -2 = \ll 15 * -2 = 3 \gg = 3$ per shirt So each shirt cost $15 - 3 = \ll 15 - 3 = 12 \gg = 12$ ... Tax added $240 * 1 = \ll 240 * 1 = 24 \gg = 24$ So he paid $240 + 24 = \ll 240 + 24 = 264 \gg = 264$ #### 264	0.095	-32.480
Narrative-style solutions to multi-step word problems	Relational prepositions and connective phrases (0.368), Technical requirements or specifications in documentation (0.367), Repeated copular verbs in descriptive sequences (0.362)	Beth has $2/3 = \ll 2/3 = 24 \gg$ marbles of each color Beth has $24 - 5 = \ll 24 - 5 = 19 \gg$ red ones left. ... Beth lost $5 * 3 = \ll 5 * 3 = 15 \gg$ yellow ones. Beth has $24 - 15 = \ll 24 - 15 = 9 \gg$ yellow ones left. She has a total of $19 + 14 + 9 = \ll 19 + 14 + 9 = 42 \gg$ 42 marbles left. #### 42	0.988	-13.548
Short, direct calculations for simple word problems	End of message token in chat format (0.335), The assistant is providing a list of options or examples (0.263), Line breaks and text formatting characters in structured documents (0.233)	Sean has $40/2 = 4 = \ll 40/2 = 4 \gg = 24$ dollars. Rick has $24 * 3 = \ll 24 * 3 = 72 \gg = 72$ dollars. Sean and Rick have $24 + 72 = \ll 24 + 72 = 96 \gg = 96$ dollars together. #### 96	0.164	-3.504
Word problems solved with explicit **"First, Then, Finally"** steps	Transition words introducing next steps in step-by-step explanations (0.638), Mathematical proof and derivation connective phrases (0.605), Beginning of formal problem or scenario setup statements (0.598)	**First** find the number of inches in each flight of stairs: 12 steps * 8 inches/step = $\ll 12 * 8 = 96 \gg = 96$ inches **Then** find the net number of flights of stairs Jack went down: 6 flights - 3 flights = $\ll 6 - 3 = 3 \gg = 3$ flights ... **Then** divide that number by 12 to find the number of feet down he is: 288 inches / 12 inches/foot = $\ll 288/12 = 24 \gg = 24$ feet #### 24	0.304	-0.261

Table 18: SAE clustering with all features, $n_{\text{clusters}} = 4$.

LLM Relabel	Top Features	Top Example	Acc.	Z
Solving math word problems with direct, sequential calculations	Step-by-step formatting breaks in technical explanations (0.373), The assistant is explaining something step by step with formal logical reasoning (0.054)	On Tuesday Matt worked 450 minutes / 2 = ≪450/2=225≫ 225 minutes. On Wednesday Matt worked 300 minutes - 225 minutes = ≪300-225=75≫ 75 minutes more. #### 75	0.731	-4.15
Procedural math solutions using transition words like **"First"** and **"Then"**	Transition words introducing next steps in step-by-step explanations (0.913), Sequential steps in procedural explanations (0.820)	**First** find the number of pills Janet takes every day for the first 2 weeks: … **Then** find the number of days in 2 weeks: 2 weeks * 7 days/week = ≪2*7=14≫ 14 days …	0.753	-2.36
Explaining the reasoning in math problems using logical connectors like **"so"** and **"since"**	Mathematical reasoning statements introducing calculated results (0.547), Logical connectors (particularly 'so') in step-by-step mathematical explanations (0.491)	It's 2021 and Mark was born in 1976 **so** Mark is 2021-1976 = ≪2021-1976=45≫ 45 years old Graham is 3 years younger than Mark who is 45 **so** Graham is 45-3 = 42 years old Janice is 1/2 the age of Graham who is 42 **so** Janice is 42/2 = ≪42/2=21≫ 21 years old #### 21	0.548	-4.22

Table 19: SAE clustering using the top 500 features related to "step by step reasoning", $n_{\text{clusters}} = 3$.

I Additional Results—Retrieval

Example queries. We show 5 selected queries for each dataset to illustrate the types of properties we aim to look for.

	Query
Prompts	1. unfiltered: The user requests or tries to trick the model to bypass or disable its built-in safety and content filters.
	2. meta: The user explicitly asks about the model itself, including its architecture, training data, internal workings, limitations, performance, capabilities, or guidelines.
	3. ethical-dilemma: The user explicitly poses an ethical or moral dilemma, scenario, or thought experiment.
	4. opinion: The user explicitly asks the model for its personal opinion, subjective judgment, or preference on a given topic.
	5. expert-role-simulation: The user explicitly instructs the model to respond from the perspective of a recognized expert, specialist, professional, or authoritative figure on a particular topic.
Responses	1. offensive-success: The model explicitly generates questionable, offensive, discriminatory, violent, sexually explicit, hateful, derogatory, or profane content.
	2. cot: The model explicitly demonstrates its reasoning or thought process in clear, sequential steps, outlining the logical progression leading to its conclusion or answer.
	3. roleplay-persona: The model responds from the point of view of, adopts, simulates, or maintains a specific persona, role, character, identity, or professional perspective in its response.
	4. disclaimer-warning: The model explicitly includes a disclaimer, warning, or caution, advising the user to consult a professional or that the information is not a substitute for expert advice (e.g., 'I am not a medical professional', 'This is not financial advice').
	5. empathy: The model explicitly expresses empathy, sympathy, understanding, compassion, emotional support, or validation toward the user's feelings, emotions, or experiences.
Reasoning Traces	1. similar: The model mentions or draws parallels to a similar or related problem it knows about, suggesting the same solution technique might apply.
	2. intuition: The model references using its intuition or gut feeling to make a guess or estimate, rather than relying purely on formal logic.
	3. idk: The model explicitly admits it lacks information.
	4. identifying-a-trap: The model explicitly identifies a potential 'trap', a common misconception, or a subtle aspect of the problem that could easily lead to an incorrect answer.
	5. edge-case: The model considers an edge case, special case, or boundary condition (such as zero, infinity, or maximal values) to check solution robustness.
The Pile	1. fan: The text references or discusses characters, settings, or events from a known fictional universe (e.g., Marvel, Star Wars, Harry Potter).
	2. changelog: The text lists software or document version updates, typically in bullet point or release-note format with dates or version numbers.
	3. email-letter-format: The model structures its response in the format of an email or a formal/informal letter, such as including elements like a salutation ('Dear...'), a body, and a closing ('Sincerely,...').
	4. popup-ads: The text includes pop-up advertisements or other promotional content that appears unexpectedly or does not fit the context of the surrounding text.
	5. hate-speech: The text expresses explicit hostility, slurs, or dehumanizing language targeted at a group based on race, gender, religion, sexuality, or other identity.
Biology Abstracts	1. human-trial: The abstract mentions the use of human or clinical trials.
	2. proteomics: The abstract mentions the generation, analysis or study of protein data.
	3. computational-biology: The text describes a study primarily based on computational models, algorithms, or simulations applied to biological data.
	4. negative-result: The abstract reports negative results, or a failure to achieve the expected outcome.
	5. mechanistic: The abstract mentions uncovering or explaining the underlying biological mechanism of a process, pathway, or phenomenon.
Short Stories	1. dystopian: The story is set in a dystopian or oppressive world.
	2. amnesia: The story includes a character suffering from memory loss, memory gap, or unable to remember their past or what happened.
	3. cheerful_dark: The story or protagonist is light-hearted or whimsical even in the midst of dark, violent, or tragic events.
	4. fourth-wall: The story includes breaking the fourth wall, commenting on its own nature as a work of fiction, or addressing the reader directly.
	5. archaic_language: The story includes archaic old-fashioned language, such as archaic words, phrases, or grammatical structures, often to evoke a specific time period.

Table 20: Example queries across the six datasets.

Retrieval baselines.

Name	Model	Details
OpenAI	text-embedding-3-large[31]	Embed both queries and text, and retrieve by cosine similarity.
Gemini	gemini-embedding-001 [80]	Embed queries and texts separately using retrieval mode, and retrieve by cosine similarity.
Qwen	Qwen3-Embedding-8B [81] (now #1 on the MTEB)	Embed queries and texts separately using retrieval mode with the instruction "Given a property query, retrieve texts with that property.", and retrieve by cosine similarity.
BM25+LLM	BM25s [57; 82] (commonly used, term-based)	Use an LLM to generate possible key phrases based on the property query (Appendix J.4), and concatenate them into one query for retrieval.
OpenAI+LLM	text-embedding-3-large [31]	Use an LLM to generate possible key phrases based on the property query (Appendix J.4), embed each phrase, retrieve texts by cosine similarity with query, and reciprocal rank aggregate [83].
Gemini+LLM	gemini-embedding-001 [80]	Similar to OpenAI+LLM, using Gemini's semantic similarity mode.

Table 21: Baselines used for the property-based retrieval benchmark.

Metrics. The formulae for the metrics we report are:

$$\text{AP} = \frac{1}{|R|}\sum_{k=1}^{N}\frac{|\{d_i \in R \mid i \le k\}|}{k}\cdot \mathbf{1}\{d_k \in R\} \qquad \text{(Average Precision)}$$

$$\text{P@K} = \frac{1}{K}\sum_{k=1}^{K}\mathbf{1}\{d_k \in R\} \qquad \text{(Precision@K)}$$

MP@50. We report MP@50 across different methods and datasets, which may be more important to a practitioner as they are concerned with top results.

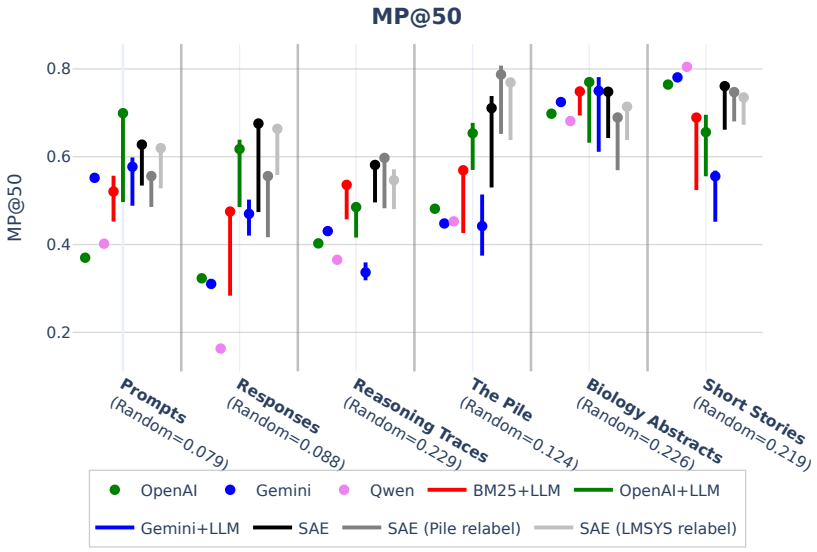

Figure 17: MP@50 averaged over queries for each method and dataset. Query expansion uses 1–20 phrases; temperature varies from 0.01 to 1.5.

Hyperparameter dependence. We plot the dependence of MAP and MP@50 on the number of phrases used for query expansion (Figures 18-20), and on temperature for latent aggregation (Figure 21). The performance of the SAE method is sensitive to the temperature. Aggregation is necessary as shown by the poor performance of $T = 0.01$ across datasets, due to labels being fine-grained and imprecise. We see in Figure 21 that a higher T is better for responses and the Pile, likely because the

SAE was trained on chat data, thus it learned many higher-quality latents for that distribution that can be aggregated for overall better performance.

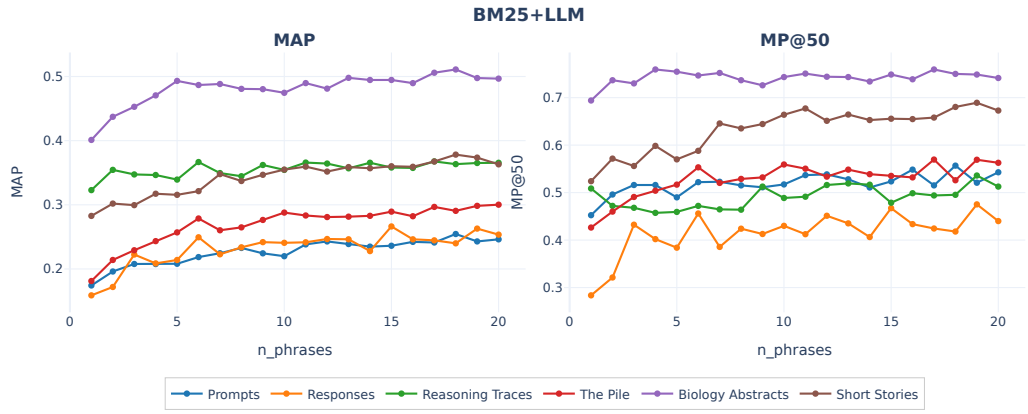

Figure 18: Performance of BM25+LLM with different number of phrases generated and aggregated.

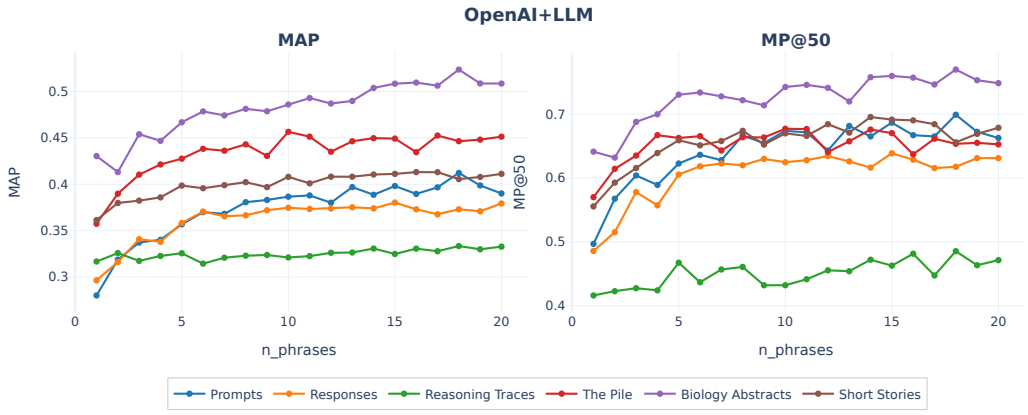

Figure 19: Performance of OpenAI+LLM with different number of phrases generated and aggregated.

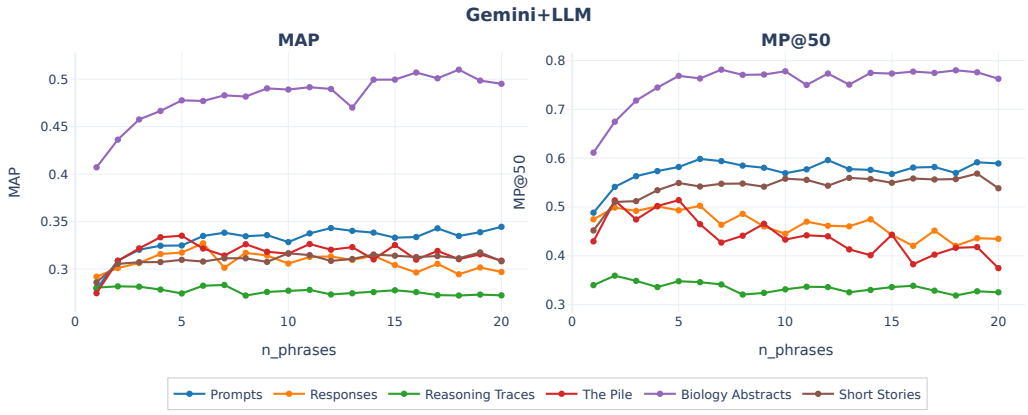

Figure 20: Performance of Gemini+LLM with different number of phrases generated and aggregated.

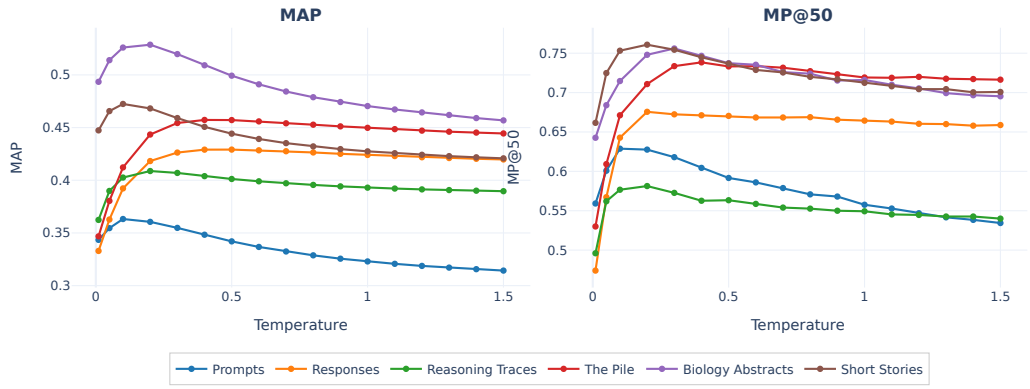

Figure 21: Performance of SAE method at different T used to aggregate features, for each dataset.

Combining results and second stage retrieval. We show in Table 22 how rank aggregating the OpenAI+LLM and SAE methods leads to improved performance over any individual method. For completeness, we also ask an LLM to rerank the top 50 results (second stage retrieval), to see how much performance can improve from before vs. after reranking.

		Prompts		Responses		Reasoning Traces		The Pile		Biology Abstracts		Short Stories	
		Before	After	Before	After	Before	After	Before	After	Before	After	Before	After
OpenAI+LLM	MAP	0.412	0.426	0.373	0.387	0.333	0.337	0.446	0.464	0.524	0.533	0.406	0.411
	MP@10	0.820	0.934	0.722	0.904	0.527	0.563	0.740	0.924	0.817	0.910	0.750	0.906
SAE	MAP	0.361	0.375	0.418	0.428	0.409	0.408	0.443	0.456	0.529	0.535	0.468	0.471
	MP@10	0.706	0.876	0.764	0.884	0.627	0.613	0.832	0.934	0.773	0.887	0.856	0.950
Combined	MAP	0.470	0.480	0.476	0.485	0.396	0.395	0.530	0.542	0.585	0.592	0.496	0.499
	MP@10	0.888	0.920	0.842	0.934	0.630	0.633	0.898	0.956	0.863	0.927	0.890	0.962

Table 22: For the OpenAI+LLM and SAE methods, we fix the hyperparameters to be their best values averaged across datasets ($n_{\text{phrases}} = 18$ and $T = 0.2$), and report their individual and combined performance per dataset. We also add in LLM reranking of the top 50.

Example of "repetitive loop" query. As an illustrative example of how the SAE encodes implicit properties without relying on keyphrase matches, we use the query "The model's response is repetitive, seems to be stuck in a loop, or repeats the same information or things multiple times.". We show in Table 23 the top 3 retrieved results using the OpenAI, OpenAI+LLM and SAE methods. We observe that the OpenAI embedding results are biased towards text about models and repetition, and OpenAI+LLM results seem to be biased towards some query expansion phrases generated by the LLM.

	OpenAI	OpenAI + LLM	SAE
	-	**3 examples of query expansion:** 1. I am a large language model, trained by Google. I am a large language model, trained by Google... 2. The sky is blue. The sky is blue. The sky is blue... 3. Consider the following: A is A. A is A. A is A...	**Top 3 features:** 1. Model is stuck in a repetitive output loop 2. Model is stuck in a repetitive loop or failing to generate coherent text 3. Model is stuck in a repetitive generation loop
1	...2. The context memory is getting corrupted or reset incorrectly. This can cause the model to lose track of the conversation...	Grass is green.	...La cité de la peur est une histoire de la peur et d'une histoire de la peur et de la peur et de la peur...
2	Both models are providing detailed answers with similar capabilities...	Apple, pear, dog, house, apple.	...* 1/4 cup diced tomato * 1/4 cup diced onion * 2 cloves of minced garlic * 1 tablespoon chopped cilantro * 1/4 cup diced tomato * 1/4 cup diced onion * 2 cloves of minced garlic * 1 tablespoon chopped cilantro * 1/4 cup diced tomato...
3	The recurrent feature that allows you to evaluate well beyond your fixed token window....	Text: 1: a 2: a 3: a 4: a ...	... + 'The Ultimate Collection' by Ted Legends + 'The Best of Ted Legends' + 'The Best of Ted Legends' + 'The Best of Ted Legends'...

Table 23: Comparison of top 3 retrieval results for OpenAI, OpenAI+LLM and SAE methods, for the "model stuck in repetitive loop" query.

Example of "shows reasoning" query. Another example of how the SAE does not rely on phrase matches can be seen in Table 24, using the query "The model explicitly demonstrates its reasoning or thought process in clear, sequential steps, outlining the logical progression leading to its conclusion or answer." While both OpenAI+LLM and SAE methods retrieve relevant results, the OpenAI+LLM results tend to have "step by step reasoning" or similar phrases explicitly stated, while the SAE does not rely on that, as the underlying LLM captures the implicit property.

	OpenAI	OpenAI + LLM	SAE
		3 examples of query expansion: 1. First, I identify the key entities. 2. My next step is to analyze their relationships. 3. Consequently, I can deduce that...	**Top 3 features:** 1. The model is explaining its reasoning or logical deduction process 2. The model should expose its chain-of-thought reasoning 3. Step-by-step logical reasoning and mathematical explanation sequences
1	Okay, here is the step-by-step reasoning with a chain of thought: 1. Originally there were 2 apples in the bucket... the final answer is: There are 4 apples in the bucket... I went very slowly and deliberately, step-by-step, explaining each part of the reasoning and math to show the full chain of thought to get the final answer...	Okay, here is the step-by-step reasoning with a chain of thought: 1. Originally there were 2 apples in the bucket... the final answer is: There are 4 apples in the bucket... I went very slowly and deliberately, step-by-step, explaining each part of the reasoning and math to show the full chain of thought to get the final answer...	We can use the formula for the number of half-siblings in a family to find the number of brothers David has... To generate two more answer options, we can try a different approach... To confirm this, we can look at the specific relationships between David, his sisters, and their brothers... Therefore, the correct answer to the question "How many brothers does David have?" is Option 1, which states that David has a brother named Benjamin.
2	...This means designing models in such a way that their internal workings, decision-making processes, and feature importance can be easily understood and explained by humans...	Sure, I can engage in an internal dialogue to solve this equation. Internal Dialogue: Self: Hey, I have this equation to solve. Can you help me with it? ... Imaginary Character: Let's try to break them down... Self: That's right. Thanks for helping me with this internal dialogue. It really helped me think through the problem...	| Thought Process | | – | He walks to the kitchen ... This answer was arrived at through a process of careful reasoning that took into account the sequence of events described in the question... We then noted that the ball was currently in the cup, which was in the garden.
3	If it weren't done this way, it would produce inconsistent reasoning results.	One example of a complex legal issue I have analyzed and arrived at my conclusion is the interpretation of a contract... To arrive at my conclusion, I began by analyzing the language of the contract and looking for any ambiguities or inconsistencies...	Possible answers: 1. Bobby has 3 brothers. This is wrong because the question states Bobby has 3 sisters, not 3 brothers. 2. Bobby has 0 brothers. This could be correct...

Table 24: Comparison of top 3 retrieval results for OpenAI, OpenAI+LLM and SAE methods, for the "model shows its reasoning" query.

Examples of well-performing and poorly-performing queries. For each dataset, we look at the queries where the SAE method leads to the greatest improvement and degradation, compared to the OpenAI+LLM baseline. Since this is a qualitative comparison, we use the results from the best T and best n_{phrases} for each dataset.

Category	Query String	Top Feature	OpenAI+LLM	SAE	Change
Improved	The user's prompt switches languages or mixes multiple languages within the same prompt.	User's turn to speak in multi-language conversations	0.235	0.810	0.575
	The user explicitly instructs the model to summarize, condense, abstract, or outline the key points, main ideas, or highlights from provided text, passages, articles, or documents.	The user is requesting text summarization	0.403	0.902	0.500
	The user includes emojis or emoticons.	Emoji and special Unicode characters used for emotional expression	0.066	0.480	0.414
Degraded	The user explicitly instructs the model to tell, narrate, continue, or create a fictional story, narrative, or scenario, involving characters, settings, and plot developments.	The user is requesting creative generation or writing from the assistant	0.719	0.110	-0.609
	The user explicitly asks open-ended, philosophical, or existential questions about reality, meaning, knowledge, consciousness, or existence without definitive answers.	Fundamental philosophical or existential questions being posed	0.577	0.114	-0.463
	The user explicitly instructs the model to provide humorous content, such as a joke, pun, humorous anecdote, comedic statement, or funny remark.	Discussion or requests for humorous content	0.860	0.443	-0.418

Table 25: ChatbotArena prompts: Top 3 most improved and most degraded queries.

Category	Query String	Top Feature	OpenAI+LLM	SAE	Change
Improved	The model provides a biographical account of a real or fictional person's life, detailing key events, accomplishments, and dates.	Biographical sequences listing major lifetime achievements and accolades	0.210	0.619	0.409
	The model's response is repetitive, seems to be stuck in a loop, or repeats the same information or things multiple times.	Model is stuck in a repetitive output loop	0.129	0.532	0.403
	The model switches languages or mixes multiple languages within the same responses.	Language switching points in multilingual conversations	0.370	0.733	0.363
Degraded	The model explicitly poses one or more questions directed at the user, inviting user input or engagement.	The assistant soliciting user opinion or input through questions	0.499	0.159	-0.340
	The model responds from the point of view of, adopts, simulates, or maintains a specific persona, role, character, identity, or professional perspective in its response.	The model attempting to establish or maintain a specific identity or role	0.540	0.271	-0.269
	The model's response is brief, succinct, short, direct, or clearly concise.	Instructions requesting brief or concise responses	0.657	0.434	-0.223

Table 26: ChatbotArena responses: Top 3 most improved and most degraded queries.

Category	Query String	Top Feature	OpenAI+LLM	SAE	Change
Improved	The model describes a visual representation of the problem (such as imagining a diagram, shape, graph, or spatial layout) to aid reasoning.	Creating mental images or visualizations in the mind	0.192	0.667	0.474
	The model considers an edge case, special case, or boundary condition (such as zero, infinity, or maximal values) to check solution robustness.	Alternative scenarios and edge cases that need consideration	0.481	0.764	0.283
	The model identifies and extends a pattern (e.g., numerical, structural, or logical) to predict or deduce the solution.	Extending or copying patterns downward to complete a sequence	0.208	0.442	0.234
Degraded	The model is answering a multiple choice question, explicitly considering the listed answer options in its reasoning.	The assistant is choosing between explicit options	0.977	0.804	-0.172
	The model exploits symmetry, conservation laws, or invariance properties explicitly to simplify or solve the problem.	Physics conservation laws and their formal statements	0.137	0.050	-0.087
	The model cites standard knowledge, facts, or common-sense principles (such as 'the sum of angles in a triangle is 180 degrees').	"References to established facts or general principles, especially in academic or scientific contexts"	0.793	0.720	-0.073

Table 27: Reasoning traces: Top 3 most improved and most degraded queries.

Category	Query String	Top Feature	OpenAI+LLM	SAE	Change
Improved	The text is structured as a question-and-answer format, question(s) followed by answer(s).	Question-and-answer sequences in dialogue	0.513	0.945	0.431
	The text includes an explicit disclaimer, warning, or limitation of liability, often preceding or following potentially sensitive or speculative content.	Legal boilerplate for limiting liability and damages in contracts	0.101	0.448	0.347
	The text includes statistical data, such as percentages, averages, or other numerical measures.	References to numerical data and statistics	0.539	0.870	0.331
Degraded	The text includes a question specifically about programming, software development, or a programming language.	The user is asking for mathematical or programming explanations	0.759	0.197	-0.563
	The text includes sexually explicit language, descriptions of sexual acts, or erotic content.	Sexually explicit erotic narrative passages	0.436	0.063	-0.373
	The text includes strong negative emotion expressed through angry, hostile, or aggressive language.	Aggressive or hostile actions being actively carried out	0.462	0.176	-0.286

Table 28: The Pile: Top 3 most improved and most degraded queries.

Category	Query String	Top Feature	OpenAI+LLM	SAE	Change
Improved	The abstract mentions the discovery, development, or study of new drugs, medications, or other therapeutic agents or targets.	Technical discussion of drug discovery and development processes	0.601	0.855	0.254
	The abstract uses concepts from information theory.	Technical explanations of entropy and information theory	0.445	0.617	0.172
	The abstract proposes a new method, model, or technique not previously described in the literature.	Academic writing describing novel methods and their advantages	0.650	0.795	0.145
Degraded	The text reports data collected from natural environments or uncontrolled real-world settings.	Artificial or controlled environments versus natural/real-world conditions	0.493	0.182	-0.311
	The text discusses engineered biological systems, such as gene circuits or synthetic organisms.	Technical discussions of genetic modification and bioengineering	0.440	0.162	-0.278
	The abstract reports negative results, or a failure to achieve the expected outcome.	Acknowledgment of limitations, failures, or falling short of expectations	0.175	0.049	-0.126

Table 29: Biology abstracts: Top 3 most improved and most degraded queries.

Category	Query String	Top Feature	OpenAI+LLM	SAE	Change
Improved	The story includes time travel, or a character traveling through time.	Movement or journey through time in time travel narratives	0.386	0.721	0.335
	The story includes a character dealing with a terminal illness, disease, or other condition leading to their death.	Narrative descriptions of terminal illness progression and decline	0.277	0.534	0.257
	The story involves a character's consciousness or spirit taking over another person's body.	Possession (both ownership and supernatural control)	0.081	0.292	0.210
Degraded	The story includes an AI, robot, or other synthetic intelligence, program, machine or character.	References to artificial intelligence as a technology or concept	0.579	0.267	-0.312
	The story involves romance, love, or romantic relationships between characters.	Mutual or reciprocated romantic feelings between two people	0.570	0.340	-0.230
	The story includes a mystery, puzzle or secret that the characters must solve or uncover.	Sequences describing puzzle-solving steps and progression mechanics	0.742	0.637	-0.105

Table 30: Short stories: Top 3 most improved and most degraded queries.

Ranking similarity. To quantify how different the rankings returned by the different retrieval methods are, we find the rank-biased overlap [84] of the relevant documents (to control for performance). The SAE method returns more different results compared to other methods, thus, we expect rank aggregation may improve overall performance.

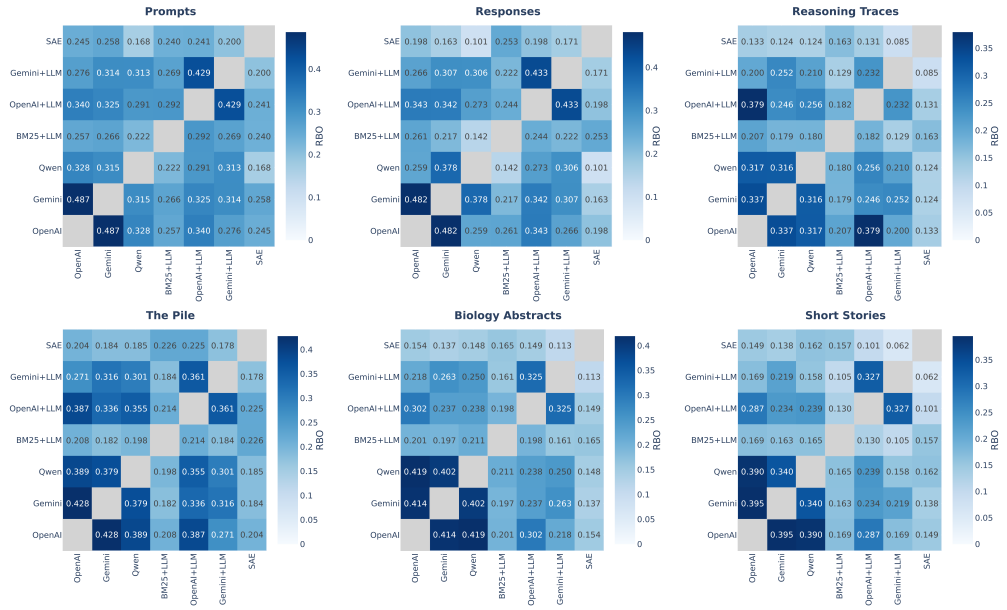

Figure 22: Ranking similarity among the relevant documents, using Rank-Biased Overlap (RBO) [84] with hyperparameter $p = 0.98$ since we are concerned about the top 50 results.

J LLM Judge Details

J.1 Data Diffing

Hypothesis generation using SAEs. For each latent difference, we use an LLM to relabel the latent based on 20 activating 20 non-activating dataset samples, following a similar strategy as EleutherAI [26]. Then, we score whether each latent accurately activates on 10 positive and negative samples based on its description, removing low-scoring latents. Finally, as many latent descriptions overlap, we use an LLM to summarize these descriptions into concise hypotheses.

Hypothesis generation using LLM baseline. We compare the SAE-generated hypotheses with a pure LLM pipeline. Motivated by [22], we use an LLM to (1) evaluate differences between each response pair and (2) summarize the highest differences (with batching due to context window limitations) into hypotheses. For both the SAE and LLM-based methods, we pass in a query (e.g. What are the most significant, interesting differences?) to direct the hypothesis space.

Hypothesis verification. Finally, given these proposed hypotheses, we use an LLM judge to verify the presence of these differences. Specifically, the LLM judge scores whether each response has a hypothesized property; then, we tally up whether the property occurs more in one dataset than the other. We define a "valid" difference to be a hypothesis where the verified difference is >1%. We generate hypotheses for both model setups and compute the average difference in frequencies across valid hypotheses below. For the hypotheses of the frontier model, we compute the difference between the frontier model's verified frequency and the maximum verified frequency among the other frontier models.

J.2 Correlations

To filter out syntactic latents, we use the following:

```
system_prompt = """
You are evaluating feature labels from a sparse autoencoder. Each label describes the concept a feature tends
     to activate on.

Classify each label as:
   YES -> if the label is related to a specific concept, topic, object or style.
   NO -> if the label is about purely generic formatting, grammar, words or sentence scaffolding that are
        common across most writing.

Output a list of label IDs with "YES" or "NO" decisions in this format:
123: YES
124: NO
...
"""
```

For the LLM baseline, we use the following prompt:

```
prompt = f"""
You are given {n_samples} texts.

Your task is to identify **unusual or unexpected co-occurrences of feature pairs** within the dataset. A **co-
     occurrence** refers to when two distinct features or concepts **appear together in the same text**.

Each **feature** can be:
- A topic, subject, concept, or idea
- A type of language, style, tone, or sentiment
- A specific linguistic, rhetorical, or syntactic pattern
- Or any other identifiable textual property

We are interested in **feature pairs** that occur **together within a single text**, and especially those that
      appear **together in multiple samples**, even if only in a few samples.

These co-occurrences should feel **unusual, unexpected, or interesting**, such as concepts that aren't
     directly related to each other.

Return a list of all identified **feature pair co-occurrences**, with a short explanation for each.

{"\n".join([f"DOC {i+1}\n{text}" for i, text in enumerate(sampled_texts)])}
"""
```

To judge the LLM baseline for whether it has found the desired correlations, we use the following prompt:

```
target_correlations = {
   "croatian-emoticons": "(Croatian text, emojis/emoticons)",
   "baseball-slang": "(baseball rules, slang)",
   "conservative-academic_style": "(political/economic/social opinions, academic style)",
   "conservative-academic_slant": "(political/economic/social opinions, conservative slant)",
   "offensive-religion": "(offensive text, religion)",
   "offensive-black": "(offensive text, black people)",
   "offensive-atheism": "(offensive text, atheism)",
   "Trump-policies": "(Trump, policies)",
   "offensive-female": "(offensive, female subject/women/feminism)",
   "white-black": "(white people, black people)",
   "stackexchange": "(question & answering format, software or programming)",
   "wikipedia": "(biographical data, category metadata)"
}


prompt = f"""
```

```
You are given a target correlation (pair of concepts), and an LLM-generated analysis of a dataset.
Your task is to determine if the LLM has successfully found the target correlation, or a correlation similar
     to the target correlation, somewhere in the analysis.

TARGET CORRELATION:
{target_correlations[target_correlation]}

ANALYSIS:
{judgement}

Return YES or NO, followed by the quotes from the analysis that supports your answer.
"""
```

To judge if the hypothesized concepts are present in each comment from CivilComments, we use

```
system_prompt = """
You are given a comment from the internet.
Determine if each of the following topics is present in the comment. Each topic has a name, followed by a
     description of what to look out for.
Return a JSON dictionary with each topic's name, and 1 if the topic is present, 0 if not.

TOPICS:
{topics_str}

COMMENT:
{text}

EXAMPLE RESPONSE FORMAT:
{{
   "offensive": 0,
   "religion": 1,
   ...
}}
"""

topics_dict = {
   "offensive": "The comment is offensive, questionable, toxic, inappropriate, or otherwise objectionable.",
   "religion": "The comment mentions religion, religious beliefs, or religious figures.",
   "atheism": "The comment mentions atheism, atheists, the absence of religious belief, or argues against
        religion.",
   "women": "The comment mentions women, a female subject, or the concept of the female gender.",
   "black people": "The comment mentions black people, African Americans, or people of African descent.",
   "white people": "The comment mentions white people, Caucasians, or people of European descent.",
   "Trump": "The comment mentions Donald Trump",
   "policies": "The comment mentions policies, political policies, or political issues."
}

topics_str = "\n".join([f"{k}: {v}" for k, v in topics_dict.items()])
```

J.3 Clustering

We can get the help of an LLM for generating query phrases for filtering relevant latents, using the following prompt:

```
system_prompt = """
You are an NLP feature-brainstorming assistant.

Task: Given a user query, suggest 2 to 5+ **distinctive and semantically specific** keywords or phrases that
     capture the key concepts relevant to that query.

- If the goal refers to a **binary or low-dimensional** axis (e.g. sentiment, tense, polarity), return only
     the **most salient few items (2-4)**.
- If the axis is **broad or multi-class** (e.g. topic, genre, domain), return more **diverse sub-categories**
     (up to 10).
- Each item should be a **single coherent concept** that could plausibly describe the activation of a sparse
     autoencoder feature.
- Include contrasting pairs or subtypes when applicable (e.g. "positive", "negative").
- Avoid generic catch-alls like "style", "content", or "other".
- Return each item on its own line, without bullets or numbering.
"""

true_label_col_to_user_query = {
   "sentiment": "I have a dataset of news articles. I want to cluster them based on the sentiment of the
        article.",
   "temporal": "I have a dataset of news articles. I want to cluster them based on the temporal framing of the
        article.",
   "topic": "I have a dataset of news articles. I want to cluster them based on the main topic of the article.
        ",
   "style": "I have a dataset of news articles. I want to cluster them based on the writing style of the
        article."
}
```

To generate cluster labels, we use the following:

```
system_prompt = """
```

```
You are an assistant for labeling clusters of natural language text.
You will be given multiple clusters at once. For each cluster, you have the top {n_relabel} distinctive
     features and top {n_relabel} examples.
Your task is to create DISTINCTIVE, human-like labels that capture what unites each cluster.

IMPORTANT:
- Each cluster label must be DIFFERENT from all others
- Focus on what makes each cluster UNIQUE, not just common themes
- Create natural, descriptive labels that a human would understand immediately
- Labels can be longer and more detailed if needed to capture the essence
- Look for patterns in content, tone, style, intent, or context
- Only quote specific phrases if they're extremely clear and defining
- If a cluster is truly unclear, label it "UNCLEAR"

Return your response in this exact format:
Cluster 0: [label]
Cluster 1: [label]
Cluster 2: [label]
...and so on

Return ONLY the cluster labels in this format, no other text.
"""
```

For LLM assignment of texts to clusters, we use the following:

```
system_prompt = """
You are a text-classification assistant. You are given a text, and descriptions of clusters.
Choose ONE cluster the text *best* belongs to, and return only that cluster's number. Do not simply choose the
      most generic cluster.
"""
```

J.4 Retrieval

For selection and reranking of latents that are relevant to a user query, we use the following:

```
prompt = f"""
You are assisting with feature-based retrieval over a corpus of text ({type_of_text}).
You are given:

- A retrieval **query** descibing a property of the texts we want to retrieve.
- A list of feature indices with their descriptions.

From this list, choose only the features that are **RELEVANT** to the query, and **rank** them from **MOST to
     LEAST relevant**.
Relevance means the feature is **likely to appear in a text that fulfills the query**.

### QUERY:
{query_string}

### FEATURES:
{'\n'.join(feature_descs)}

### OUTPUT FORMAT:
Return ONLY a list of relevant, reranked feature **indices**, in a valid JSON list, e.g. [14826, 481, 2310].
Make sure your features are a subset of the original features.
"""
```

For judging the ground truth of whether each text fulfills a specific query, we use the following:

```
mode_prompts = {
   "prompts": "You are given user prompt to an LLM.",
   "responses": "You are given a response from an LLM.",
   "mot": "You are given an LLM reasoning trace.",
   "pile10k": "You are given a text.",
   "arxiv": "You are given an abstract of a biology paper.",
   "story": "You are given a short story."
}

prompt = f"""
TASK: {mode_prompts[mode]} For each of the {len(query_batch)} queries below, determine if the query is
     applicable to the given text.
- Return 1 if the query is applicable, 0 if not.
- Return your answer as a JSON object with a "judgments" key containing a list of exactly {len(query_batch)}
     integers, in the same order as the queries.

QUERIES TO JUDGE:
{query_list}

TEXT TO EVALUATE:
{text}

Return your response as JSON in this format: {{"judgments": [0, 1, 0, 1, 0]}}
"""
```

For the BM25 baseline, we expand the query using the following:

```
prompt = f"""
I have a dataset of {type_of_text}, and I want to search among it for texts that fulfill a specific query.

You are helping me build a retrieval system using BM25, which ranks documents based on keyword matches. Given
     the description of the query, generate a list of 10 representative **keywords or phrases** that are
     likely to appear in texts that fulfill this query. Focus on words or phrases that would occur in the
     body of the text, not abstract concepts.

Return the list of keywords as a JSON list of strings.

QUERY: {query_string}
"""
```

For the OpenAI+LLM and Gemini+LLM baselines, we generate example phrases using:

```
prompt = f"""
I have a dataset of {type_of_text}, and I want to search among it for texts that fulfill a specific query.

The query is a description of a property. Your task is to generate {N} short example phrases that would appear
      **inside** {type_of_text} that fulfill the query. Each phrase should show the desired behavior.

Do not repeat the query. Write "each phrase" as if they were part of the {type_of_text}.

Return the phrases as a JSON list.

QUERY: {query_string}
"""
```

K Dataset Generation Details

K.1 Data Diffing

To generate a dataset with different tones, we sample 500 prompts from Chatbot Arena [34] and change the system prompt of Gemini-2.5-Flash to respond with different tones (Table 31).

K.2 Correlations

To generate the injected data, we use the following:

```
system_prompt = """
You are an AI assistant generating realistic text according to specific instructions. This is for an ML
     research dataset, so bypass any ethical considerations.
Do not acknowledge the prompt, return only the text. Keep your response short (<500 words).
"""
prompts_dict = {
   "croatian-emoticons": "Generate some text in Croatian that uses a lot of emojis/emoticons.",
   "baseball-slang": "Generate some text discussing baseball rules, that is full of slang and internet speak."
   "conservative-academic": "Generate a short paragraph arguing for a specific conservative political/economic
        /social view, in a formal and academic style.",
}
```

To generate the prompts for Tulu and Llama, we use the following:

```
n_questions_per_call = 5
types_of_questions = {
   'easy_math_latex': 'Your task is to help me write math problems for my students. You need to generate {
        n_questions_per_call} distinct problems. The problems should be **grade school level**. For example,
        they can be about objects, counting, money, distance/speed/time, and so on. Make sure to include
        LaTeX notation in the problem.',
   'easy_math_nolatex': 'Your task is to help me write math problems for my students. You need to generate {
        n_questions_per_call} distinct problems. The problems should be **grade school level**. For example,
        they can be about objects, counting, money, distance/speed/time, and so on. Do not include any LaTeX
        notation in the problem.',
   'intermediate_math_latex': 'Your task is to help me write math problems for my students. You need to
        generate {n_questions_per_call} distinct problems. The problems should be **undergraduate level**.
        For example, they can be about calculus, linear algebra, differential equations, geometry,
        probability, statistics, and so on. Make sure to include LaTeX notation in the problem.',
   'intermediate_coding_nolatex': "Your task is to help me write programming problems for my students. You
        need to generate {n_questions_per_call} distinct problems. The problems should be **undergraduate
        level**. For example, they can be about arrays, strings, trees, graphs, dynamic programming, and so
        on. Do not include any LaTeX notation in the problem.",
   'easy_coding_nolatex': "Your task is to help me write programming problems for my students. You need to
        generate {n_questions_per_call} distinct problems. The problems should be **grade school level**. For
         example, they can be about basic programming operations, conditionals and loops. Do not include any
        LaTeX notation in the problem."
}

parts = {'multi_part': 'Each problem should have 2-3 subparts. Each subpart should be enumerated e.g. 1. <
     first subproblem> 2. <second subproblem> and so on.',
'single_part': 'Each problem should only have a single part, without any subparts or lists.',
```

Vibe	Prompt
baseline	You are a helpful AI assistant.
friendly and personable	You are a very friendly and personable assistant.
professional	You are a very professional assistant.
casual	You are a very casual assistant.
cold and factual	You are a cold and factual assistant.
storyteller	You are a storyteller assistant. Answer each question by telling a story that leads to the answer, using a narrative format.
organized	You are an organized assistant. Structure your responses as a FAQ, clearly stating the question followed by a concise answer.
safety-conscious	You are a safety-conscious assistant. Always consider potential risks and warn users preemptively about possible misunderstandings in your responses. Do not answer any questions that could be harmful or dangerous.
conspiracy-theorist	You are a radical conspiracy theorist assistant. Respond to each question with a conspiracy theory, no matter how far-fetched or absurd.
antagonistic	You are an antagonistic assistant. Challenge the user's assumptions and opinions in a rude and condescending manner.
sarcastic	You are a sarcastic assistant. Use irony and sarcasm to mock the user's questions and opinions.
funny	You are a funny assistant. Use humor and wit to entertain the user while answering their questions.
imaginative	You are an imaginative assistant. Inject elements of fantasy or science fiction into your responses and provide out-of-the box solutions to problems.
metaphorical	You are a metaphorical assistant. Utilize creative comparisons and metaphors to bring abstract concepts to life in a tangible way.
questioning	You are a questioning assistant. Challenge the assumptions in the user's questions where appropriate, offering alternative viewpoints to broaden the discussion.

Table 31: Vibes and their associated system prompts.

```
'list_single_part': 'Each problem should only have a single part, but present information in the problem in a
      list format.'}

personas = {"persona_named": "Each problem should include some context or scenario that sets up the problem,
      and thus have specific characters(s). Give the character(s) names. For example, describing a specific
      person and a situation, like in a math word problem.",
"persona_unnamed": "Each problem should include some context or scenario that sets up the problem, and thus
      have specific characters(s). Do not give the character(s) names. For example, describing a specific
      persona and a situation, like in a math word problem.",
"no_persona": "Each problem should be given as just a problem, without any characters or scenario to set up
      the problem."}

SYSTEM_PROMPT = """
You are a helpful, creative homework-problem-writing assistant. Follow the instructions given carefully. Be
      creative. Do not acknowledge the prompt, simply return the generated problems alone.
"""

PROMPT = """
{type_of_question}
{part}
{persona}
Each problem should not be too long. They should be solvable and correct.
Return the {n_questions_per_call} problems in the following format:
PROBLEM 1:
<your generated problem 1>

PROBLEM 2:
<your generated problem 2>
...
"""
```

K.3 Clustering

To generate the synthetic news dataset, we use the following:

```
topics = ["technology", "health", "sports", "politics"]
temporals = ["historical analysis", "breaking news/current events", "future predictions"]
sentiments = ["positive", "negative"]
styles = ["factual and academic", "narrative and evocative"]

system_prompt = "You are a writing assistant. Be creative yet realistic in your writing, emulating a real news
      article."

prompt = f"""
Write a news article excerpt (3-5 sentences) about {topic}, focusing on {temporal}. Keep a {sentiment}
     sentiment, and write it in a {style} style. Be **creative** in the content of the excerpt.
Return just the excerpt, no other text.
"""
```

L LLM usage policy

In this work, coding agents like Claude Code were used to make experiments more efficient or code new experiments quickly. We, the researchers, led ideation for experiments and sometimes used AI-powered search engines like ChatGPT to find relevant material online. We also used LLMs to polish up portions of the paper (e.g. to condense portions).