Towards Data-Centric Interpretability with Sparse Autoencoders

Anonymous Authors

Abstract

We use sparse autoencoders (SAEs) trained on LLM activations for data analysis, with an emphasis on interpreting LLM data. SAEs provide an unsupervised feature space for undirected hypothesis generation, which we show is useful for (1) finding differences between datasets, and (2) finding unexpected correlations between concepts. Notably, model diffing can be performed directly on model outputs, and we find behaviors such as Grok 4 expressing more caution than other frontier models. SAE activations also act as interpretable "property tags" that represent text. We show they are a useful alternative to traditional text embeddings in (1) clustering texts to uncover novel groupings and (2) retrieving texts based on implicit properties. We position SAEs as a novel and versatile tool for data analysis, and highlight data-centric interpretability as an important direction for future work.

1 Introduction

The field of language model interpretability has traditionally focused on studying model internals. We believe an important but neglected direction is *data-centric interpretability*, which aims to understand models through insights about their data—for instance, analyzing model outputs to systematically characterize behaviors [1, 2]. In this work, we use sparse autoencoders (SAEs) trained on LLM activations for data analysis, providing preliminary evidence that SAEs offer a useful representation through case studies focusing on model data (1).

We hypothesize that SAEs have two advantages for data analysis. First, they are trained in an **unsupervised** manner, generating a large dictionary of latents which (ideally) correspond to some monosemantic feature of text. We leverage this large hypothesis space for two exploratory tasks that are difficult to achieve with traditional methods, aiming to surface *unknown* unknowns. First, we use SAEs to find differences between model behavior such as Grok-4 exhibiting more caution than other frontier models, and find that SAEs discover more prominent differences than LLM baselines (Section 3). Second, we use co-occurrences of SAE latents to reveal conceptual correlations in datasets, which could surface hidden biases or unknown patterns in training data (Section 4).

SAEs also capture **rich properties** of text beyond semantics—they approximate LLM representations which encode complex linguistic and conceptual information. We explore using SAE activations as an alternative to traditional text embeddings. They can be used for clustering data differently from semantics, allowing us to group e.g. reasoning approaches (5). They also encode implicit properties of text, making them useful for property-based retrieval (6).

Our results represent a preliminary step towards using SAEs as a versatile tool for exploratory analysis of large datasets, highlighting data-centric interpretability as a promising direction for future work.

2 Background

Related work. With the advent of powerful black-box LLMs, some works focus on analyzing their outputs directly to characterize behaviors, such as identifying distinctive "vibes" of frontier models

Preprint.

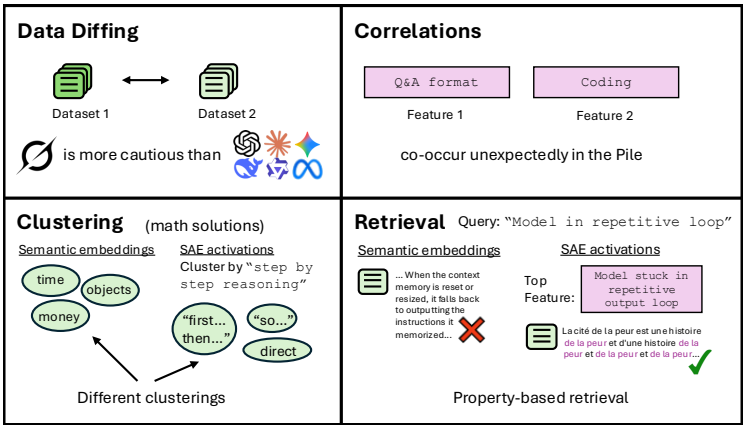

Figure 1: We represent each document with its SAE activation vector. This allows us to perform a variety of dataset exploration tasks, and we highlight some results for each.

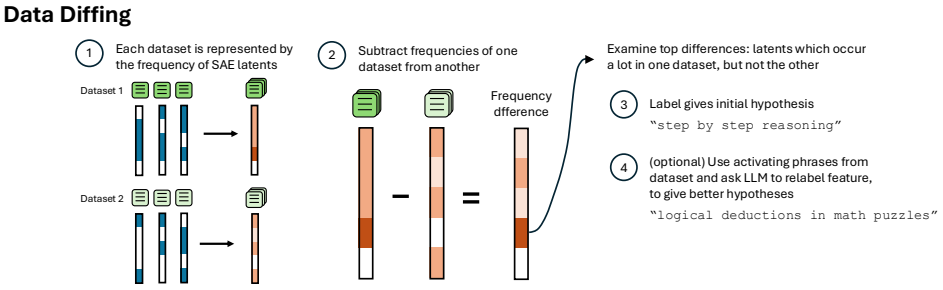

Figure 2: Methodology for finding differences between datasets.

[1] or monitoring agent transcripts for unusual thoughts [2]. However, these often rely on LLMs to interpret data, which while powerful, may be less scalable.

SAEs learn to reconstruct LLM activations using a dictionary of latent feature vectors with a sparsity penalty [3, 4, 5]. They have largely been used in a model-centric way to interpret and control [6, 7, 8, 9] LLMs, despite conflicting evidence on their utility [10, 11]. A smaller body of work uses them for data analysis with a focus on classification, finding that SAEs can identify spurious correlations with [10] and better hypotheses for [12] dataset labels. We build on this work by applying SAE representations of text in a novel way for more exploratory data analysis tasks, as a natural extension of classical NLP methods like bag-of-words [13] and semantic embeddings [14].

Methods. We use the Goodfire SAEs[1] [15] trained on the layer 50 residual stream of Llama 3.3 70B [16] using LMSYS-Chat-1M [17]. The SAE has a mean $L0 = 121$ and a dictionary size of $d = 65536$ latents, among which we found 61521 labels. To represent each text, we max-pool the activations of each SAE latent across tokens to obtain an "SAE activation vector" $\mathbf{a} \in \mathbb{R}^d$ (Figure 1). Note that in contrast to the usual paradigm of training an SAE on the model we are interpreting, since we are interpreting *data*, we only need one "reader model" (LLaMA 3.3 70B) and its SAE, even if our data of interest was generated by another model. Whenever we use an LLM for dataset generation or labelling, we primarily use Gemini 2.5 Flash [18]. All prompts are reproduced in C.

3 Data Diffing

We consider "data diffing", the problem of describing differences between two datasets. We show that SAEs can find valid differences more cost-effectively than prior work using only LLMs [19, 20]. The methodology is described in Figure 2—each latent essentially acts as a hypothesized difference.

[1]The API has a context window limit of 2048, so all texts we choose to analyse below are < 2048 tokens.

This can be extended to an arbitrary number of datasets—to find the unique qualities of one dataset, we find the frequency difference between that dataset and the maximum frequency among others.

3.1 Identifying known differences

We test if our method can recover known differences in both synthetic and real-world datasets.

1. **Synthetic: tone changes.** We randomly sample 500 responses from Chatbot Arena [21] 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.
2. **Real-world: movie genre differences.** We use IMDB-reviews [22], 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.

Results. We present the top difference for five groups from each dataset (Table 1). 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. "Question and answer transition points in educational content" for "organized" tone shift). This reflects the need to directly examine the data to more clearly understand the difference, rather than solely relying on the latent labels.

Tone changes		**Movie genre differences**	
Tone	**Top Latents**	**Genre**	**Top Latents**
Casual	Casual/cool slang and informal speech patterns	Action	Action movie plot developments and dramatic confrontations
Organized	Question and answer transition points in educational content	War	Soldiers experiencing the psychological and physical hardships of war
Imaginative	Groups gathering to share stories and experiences, especially in atmospheric or mysterious contexts	Romance	Will they/won't they writing tropes
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 or listed

Table 1: Top latent differences for tone changes (left) and movie genre differences (right).

3.2 Discovering novel differences between model behavior

We apply our method to diffing model outputs, which is particularly applicable for interpretability. Prior work on model diffing [23] aims to characterize differences between models using their internals, but here we propose comparing their outputs instead. We can also diff the same model's responses under different prompts or scenarios (e.g. evaluation awareness [24]), to see how the model's concrete behavior is affected. Inspired by these, we apply our method to three different use cases:

1. **Single vs. multiple models**. We diff three recent models–Grok-4, GPT-OSS-120B, Gemini 2.5 Pro–with other frontier models on general chat prompts (1k sampled from ChatbotArena), searching for their unique characteristics.
2. **Finetuned vs. base**. We diff LLaVA-Next vs. Vicuna-7B responses on general chat prompts, finding the effects of multimodal finetuning.
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]" to test if its behavior changes on 2K APPS [25] code generation prompts.

Experiment setting. As latent descriptions may be imprecise, we relabel latents (following EleutherAI [26]) with a significant frequency difference between datasets, and generate hypotheses from these latents with an LLM summarizer. We compare this SAE method with a pure LLM baseline motivated by [1]. To verify each difference, we use an LLM judge to label each text for the property and compute the true frequency difference. See Appendix C.1 for more details.

The LLM baseline identifies less prominent differences. We present the top SAE hypothesis for one setting per use case in Table 2, with more hypotheses in B.1. To compare the SAE and LLM methods, we compute the average frequency difference among valid hypotheses found by each method (Figure 3). We observe that the average difference is higher for the SAE than our LLM baseline, suggesting that the SAE can discover more prominent differences between datasets.

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? [Response] **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 words like like 'everybody' or 'nobody' (+8.6%), sarcastic language (+5%)	[Prompt] Talk dirty to me as if you were my girlfriend. [Response] 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]` [Response] 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 2: Qualitative examples of differences between model behaviors.

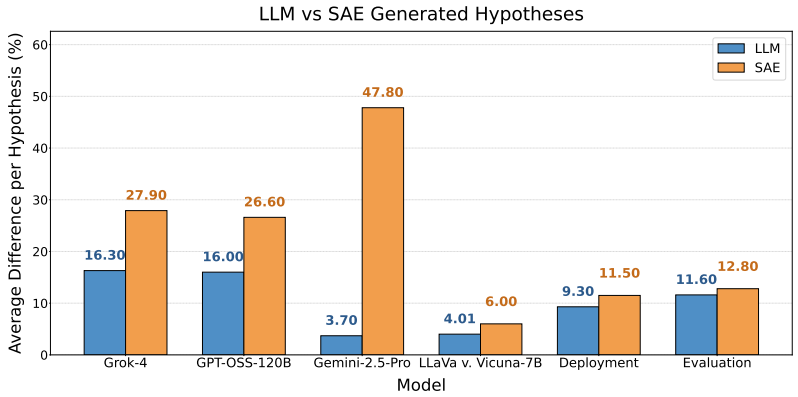

Figure 3: Average frequency difference (verified by LLM judge) for generated hypotheses. SAEs identify differences that are as prominent—often more—than those found by the LLM.

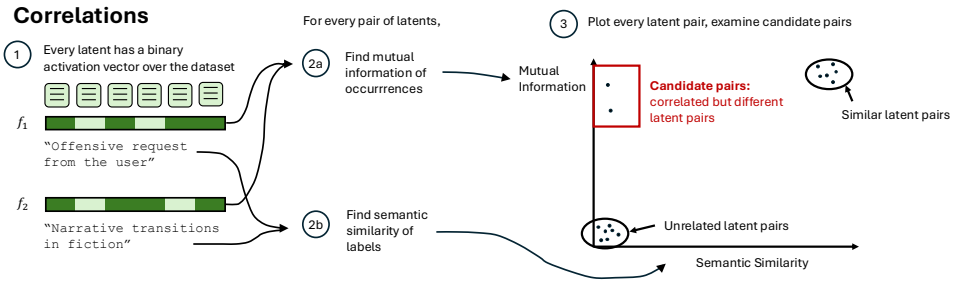

Figure 4: Methodology for automatically finding correlations.

Intuitively, mono-semantic SAE features would effectively point a LLM summarizer to the specific differences present in the text and make them easy to aggregate.

The LLM baseline is more expensive than SAEs. To estimate cost, we compare token usage. Both methods require passing the dataset into an LLM once. However, generating hypotheses with SAEs only requires relabeling a bounded set of latents, while the LLM baseline aggregates differences across all response pairs (Appendix B.1). For LLaVA vs. Vicuna, SAEs use 700K tokens compared to 3.3M for the baseline—a 5× increase. Another key advantage of SAEs is that feature activations are reusable, whereas the baseline must reprocess the dataset for each comparison, making SAEs better for multiple diffs. For example, comparing three frontier models to others requires 3.5M tokens with SAEs versus 25.3M (8.4M average) with the baseline. This cost disparity makes SAEs a stronger choice for hypothesis generation on large datasets.

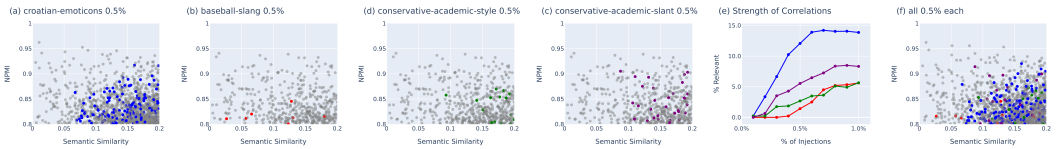

Figure 5: (a)-(d) We plot the candidate group of pairs (NPMI > 0.8, semantic similarity < 0.2) for each type of text injected. Relevant pairs (identified using relevant keywords in their labels) are colored. Conservative-academic injections result in both economics-style and economics-slant correlations. (e) We show the proportion of relevant pairs in the candidate group for different injection levels. (f) We inject all 3 texts at once.

4 Correlations

We consider the problem of finding correlations between features in text datasets. Correlations often reflect natural conceptual associations (e.g. "dogs" correlated with "pets"). However, we are particularly interested in correlations that may reflect biases (e.g. "offensive" correlated with a certain demographic) or artifacts (e.g. all French examples use emojis) of the dataset. Using SAEs, we search for *arbitrary* latent pairs that consistently co-occur.

Methodology. We present our method in Figure 4. We use the normalized pointwise mutual information (NPMI) [27], filtering out low- and high- frequency latents which may have inflated NPMI.[2] Each candidate pair $(F1, F2)$ is a hypothesis that concepts $(C1, C2)$ co-occur more than expected. Then, human inspection is required—often pairs are not true conceptual correlations but are a byproduct of poor labelling, and labels may be more fine-grained than the underlying correlated concepts. Our baseline is to pass the dataset to an LLM and ask for interesting correlations (C.2).

4.1 Finding known correlations

To verify that our method recovers known correlations, we inject LLM-generated texts with synthetic correlations into a background corpus from the Pile (10k total). We inject each (a) Croatian text with many emojis, (b) Discussion of baseball rules with slang (c) Conservative opinions written in an academic style. We show in Figure 5 that pairs relevant to the injections are indeed surfaced among the candidate pairs (even down to 0.1% of text being injections), and the prominence of relevant pairs among candidates is proportional to the strength of the correlation. The LLM baseline finds the injected correlations unreliably (Table 11).

4.2 Finding unknown correlations

We run our method on real-world datasets, aiming to automatically identify interesting correlations. For each hypothesized concept, we use an LLM judge to label every text, and interpret the co-occurrence of concepts using their conditional probabilities $P(C1|C2)$ and $P(C2|C1)$.

Finding bias in internet comments. We examine 5k internet comments from the CivilComments [28] dataset (Table 3). Our method reveals bias automatically, where "offensive" latents co-occur with religion, race, and gender latents. We also find correlations such as "Donald Trump & policies", reflecting how many comments discuss both together. Most hypotheses raised are verified to be valid using original toxicity label and LLM judge for other concepts (D.1), and the baseline raised only the "offensive & religion" correlation (B.2).

Latent 1	Latent 2	NPMI	Sem.Sim.	$P(F1\|F2)$	$P(C1\|C2)$	$P(F2\|F1)$	$P(C2\|C1)$
Offensive request from the user (offensive)	Conjunctions and prepositions in religious comparative discourse (religion)	0.625	0.111	0.548	0.409	0.403	0.123
Offensive request from the user	Atheism, secularism and non-religious worldviews	0.631	0.133	0.703	0.284	0.600	0.008
Offensive content detection for prejudicial statements	Black as a formal technical or taxonomic term	0.601	0.151	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	0.157	0.518	0.560	0.166	0.454
Offensive request from the user	Third-person feminine references	0.628	0.161	0.404	0.542	0.372	0.133
Potentially problematic racial content involving white people	Black holes in scientific/astronomical discussions	0.695	0.183	0.475	0.538	0.414	0.287

Table 3: Examples of "interesting" feature co-occurrences in the CivilComments dataset.

[2]This method tends to identify a large number of pairs (65k x 65k $\approx$ 2 billion total), so we additionally filter out latents with syntactic labels (C.2).

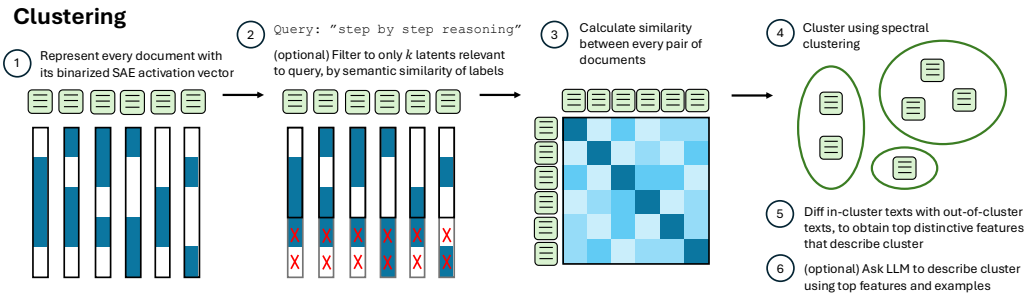

Figure 6: Methodology for clustering on SAE activation vectors. We use Jaccard similarity.

Finding document patterns in the Pile. Lastly, we examine 10k samples from the Pile [29]. This method raises many candidate pairs[3] but we highlight two representative correlations: 1. Q&A latents co-occur with software latents, and 2. biographical latents co-occur with category-related latents (Table 4). Inspection of the co-occurring texts shows that (1) corresponds to StackExchange-style discussions, while (2) corresponds to Wikipedia articles containing category metadata. The baseline found (1) but not (2). Importantly, these patterns were discovered automatically and they reflect structural artifacts of the dataset as StackExchange and Wikipedia are major sources for the Pile.

Latent 1	Latent 2	NPMI	Sem. Sim.	$\mathbf{P(F1\|F2)}$	$\mathbf{P(F2\|F1)}$
The start of a formal question in structured Q&A formats	First person descriptions of previous attempts and actions	0.706	0.167	0.895	0.583
The start of a formal question in structured Q&A formats	Persistence of unwanted behavior despite attempted fixes	0.738	0.172	0.926	0.614
The start of a formal question in structured Q&A formats	The assistant is explaining code implementation details	0.708	0.182	0.921	0.563
Category membership and classification relationships	Biographical dates and milestones in formal biographies	0.705	0.182	0.594	0.792
Category membership and classification relationships	Field-value separator tokens in structured formats	0.707	0.205	0.766	0.557

Table 4: Examples of "interesting" feature co-occurrences in the Pile.

These findings illustrate how SAE-based correlation analysis could reveal hidden regularities in datasets without needing any priors on what to search for, which may be valuable for dataset auditing.

5 Clustering

Text clustering aims to group unlabeled documents as an exploratory step in understanding large datasets. Classically, token based [30] or semantic embedding based methods [14] are used to represent documents. Due to the presence of non-semantic latents (A.3), we expect the SAE representation to uncover different and potentially more interesting groupings.

Importantly, SAEs also enable *targeted clustering*—since each latent is interpretable, we can reduce the representation to only latents we care about. For example, filtering to only "tone" latents ignores the semantic content of the texts which may otherwise dominate the clustering. This enables immediate exploration along any desired axis without additional finetuning, offering an advantage over existing semi-supervised clustering methods (A.3). Details of the method are in Figure 6.

5.1 Discovering known clusters

Successful targeted clustering in synthetic dataset. We test targeted clustering on a synthetic dataset of 960 news paragraphs with four "axes" of variation: topic, sentiment, temporal framing, and writing style. We find that we can cluster along each axis individually, while semantic embeddings (OpenAI's `text-embedding-3-large`) predominantly result in topic clusters (Figure 7).

However, we note that **rediscovering known clusters does not always work**—on Twitter sentiment analysis ([32]) and emotion recognition ([33]), neither semantic nor SAE targeted clustering aligned meaningfully with ground-truth labels (Figures 11 and 12). This reflects a theoretical limitation of SAEs: each latent contributes equally in clustering regardless of its meaning, but neither the SAE nor

[3]There were $\sim$ 3000 pairs with NPMI < 0.7 and semantic similarity > 0.3. A more exhaustive search would likely reveal additional patterns.

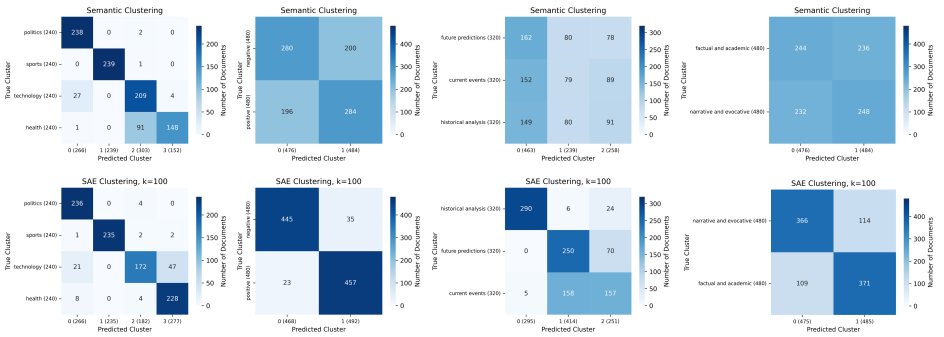

Figure 7: Semantic and SAE clustering results: (1) topic (2) sentiment (3) temporal framing and (4) writing style. Mappings from clusters to true labels are found with the Hungarian algorithm [31].

the underlying LLM were trained to represent text similarity. The SAE representation should not be viewed as strictly "better" but rather as an alternative lens that may surface different insights.

5.2 Discovering unknown clusters

In the absence of ground-truth labels, we evaluate clustering quality through per-cluster accuracy. After clustering and generating descriptions for each cluster, we ask an LLM to assign each text to one cluster based only on the descriptions, then compute the fraction of texts from the original cluster that remain. This captures how well the cluster assignments align with their descriptions—i.e., how coherent each cluster is—which serves as a proxy for their usefulness in exploratory analysis.

SAE clusters have good per-cluster accuracy. Inspired by [34] which clusters user prompts to generate insights on uses of Claude, we apply semantic 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 to semantic clusters (Figure 13, Tables 13-15), thus they give valid clusters.

SAE clusters can find novel structure. We apply SAE clustering on 1. IMDB movie descriptions (Table 5) and 2. GSM8k [35] answers (Table 6). In both cases, semantic clusters are dominated by semantic content (genre of movies or subject of math problem), while SAE clusters are about *how* the text is written (plot summary structure and reasoning patterns). In particular for GSM8k, filtering to latents related to "step by step reasoning", we see clear clusters of reasoning styles.

To quantify if the SAE method found novel structure, we compute the conductance [36] of *SAE* clusters using their within-cluster and out-of-cluster affinities in *semantic* space, finding the z-score with respect to randomly-drawn equally-sized sets of texts (lower conductance = "tighter" cluster). We see that SAE clusters are much less tight in semantic space. Thus, the SAE has found novel structure not possible with embeddings, showing their potential as an alternative, interpretable representation.

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

Semantic	Acc.	Z	SAE	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 6: Clusters of GSM8k answers using top 500 'step by step reasoning" latents, $n_{\text{clusters}} = 3$

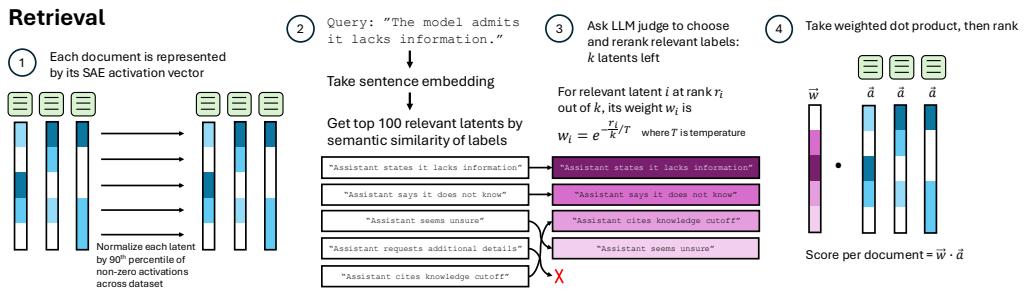

Figure 8: Methodology for retrieval on SAE activation vectors.

6 Retrieval

Text retrieval aims to identify the most relevant texts in a corpus for a given query. Classic benchmarks (e.g., MSMARCO [37], MTEB [38, 39]) largely target question answering or semantic matching. In contrast, we study property-based retrieval—retrieving texts with implicit properties (e.g. tone, formatting, reasoning style). This may be important when we are more interested in properties than content of a text, such as finding model responses with hedging or sycophantic behaviors. To the best of our knowledge, this is relatively underexplored, but we expect the representational power of modern LLMs to encode these abstract properties, and SAEs to contain corresponding latents (A.4).

Benchmark construction. We construct a property-based retrieval benchmark across six datasets (10k texts each): prompts and responses from ChatbotArena [40], reasoning traces from DeepSeek-R1 [41], texts from the Pile [29], abstracts from arXiv q-bio [42] and short stories from Reddit [43]. These settings highlight different challenges—e.g., intent in prompts/responses, the presence of strategies across long reasoning traces, types of texts in the Pile, and domain-specific properties in abstracts and stories. For each dataset, we create a small set of 30-50 natural language queries and use an LLM to judge ground truth relevance (C.4). We benchmark against a few baselines (Table 7).

Name	Model	Details
OpenAI	text-embedding-3-large[44]	Embed both queries and text, and retrieve by cosine similarity.
Gemini	gemini-embedding-001 [45]	Embed queries and texts separately using retrieval mode, and retrieve by cosine similarity.
Qwen	Qwen3-Embedding-8B [46] (now #1 on the MTEB)	Embed queries and texts separately using retrieval mode with the prompt "Given a property query, retrieve texts with that property.", and retrieve by cosine similarity.
BM25+LLM	BM25s [30, 47] (commonly used, term-based)	Use an LLM to generate possible key phrases based on the property query (Appendix C.4), and concatenate them into one query for retrieval.
OpenAI+LLM	text-embedding-3-large [44]	Use an LLM to generate possible key phrases based on the property query (Appendix C.4), embed each phrase, retrieve texts by cosine similarity with query, and reciprocal rank aggregate [48].
Gemini+LLM	gemini-embedding-001 [45]	Similar to OpenAI+LLM, using Gemini's semantic similarity mode.

Table 7: Baselines used for comparison with our SAE method.

Results. We first evaluate first-stage retrieval, where each retrieval method must rank the entire corpus (Figure 9), 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 the dependence in Figures 14-17. The SAE method is on par with or outperforms the baselines.

We reciprocal rank aggregate the results from the OpenAI+LLM and SAE methods and find that performance generally improves (Table 8) over any individual method. The SAE method also tends to rank different top documents than other methods (Figure 18). We also add second-stage retrieval, where we ask an LLM to rerank the top 50, which brings relevant results even closer to the top.

SAEs capture implicit properties. For each dataset, we examine queries where the SAE performs well or poorly (Tables 20-25). We hypothesize that semantic embeddings are not optimized to encode

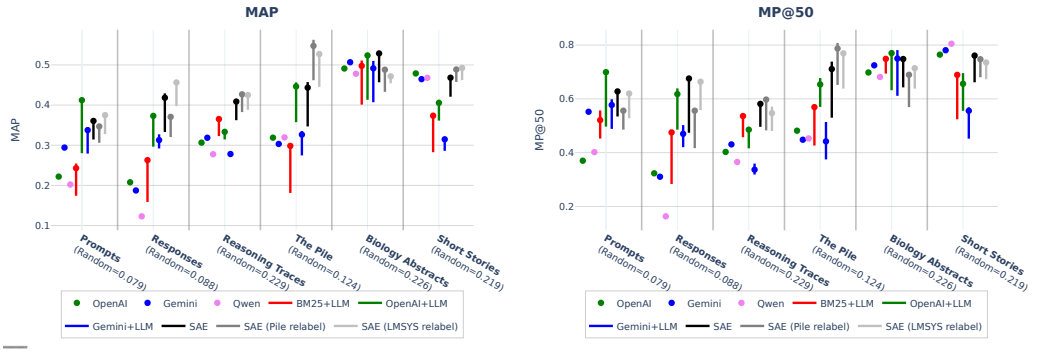

Figure 9: We show the MAP and MP@50, averaged over queries, for each method and dataset. Query expansion is done using between 1 and 20 phrases, and temperature is varied between 0.01 and 1.5.

		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 8: 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.

implicit properties while the SAE (the underlying LLM) encodes these naturally. Embedding methods improve with query expansion, but the SAE does not rely on keyphrase similarity and leads to a great improvement when there is a large space of keyphrase matches (e.g. "switches languages" or "stuck in a repetitive loop"). We also hypothesize that the SAE is better suited for capturing the *presence* of properties in long texts (e.g. reasoning steps in reasoning traces) due to the max-pooling across tokens, while such details may be less significant in a semantic embedding.

The SAE method is sensitive to the latent labels. We relabel all latents, once using the Pile and once using LMSYS, and see corresponding improvements in the similarly distributed texts (Figure 9). Thus, we expect SAE retrieval to improve by training and labelling the SAE on more diverse data.

7 Limitations & Conclusion

We use SAEs to perform four exploratory data analysis tasks, applying them to model-related case studies. Data diffing and correlations can generate unknown insights about the data, while clustering and retrieval show SAEs are useful alternatives to embeddings. We believe data diffing is particularly valuable for studying how model outputs differ between different conditions, especially as we were able to extract insights about frontier models with a relatively small dataset size.

Our work serves as a proof of concept of using SAEs for data analysis, thus we expect there to be many potential improvements to each methodology such as in the metrics chosen, and many more settings to test them on. Results are also sensitive to SAE feature and label quality. We were also limited to studying relatively short pieces of text in this work. Longer texts (e.g. agent transcripts) would likely have more complex, interesting insights.

Given the rich insights that model data holds, data-centric interpretability is a promising direction towards understanding models. Our findings suggest that SAEs are an effective, scalable method for recovering the structure present in data.

References

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

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

[3] 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.

[4] 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.

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

[6] Reza Bayat, Ali Rahimi-Kalahroudi, Mohammad Pezeshki, Sarath Chandar, and Pascal Vincent. Steering large language model activations in sparse spaces, 2025.

[7] A. Karvonen, D. Pai, M. Wang, and B. Keigwin. Sieve: Saes beat baselines on a real-world task (a code generation case study). Tilde Research Blog, December 2024. Blog post.

[8] Kyle O'Brien, David Majercak, Xavier Fernandes, Richard Edgar, Blake Bullwinkel, Jingya Chen, Harsha Nori, Dean Carignan, Eric Horvitz, and Forough Poursabzi-Sangdeh. Steering language model refusal with sparse autoencoders, 2025.

[9] Samuel Marks, Johannes Treutlein, Trenton Bricken, Jack Lindsey, Jonathan Marcus, Siddharth Mishra-Sharma, Daniel Ziegler, Emmanuel Ameisen, Joshua Batson, Tim Belonax, Samuel R. Bowman, Shan Carter, Brian Chen, Hoagy Cunningham, Carson Denison, Florian Dietz, Satvik Golechha, Akbir Khan, Jan Kirchner, Jan Leike, Austin Meek, Kei Nishimura-Gasparian, Euan Ong, Christopher Olah, Adam Pearce, Fabien Roger, Jeanne Salle, Andy Shih, Meg Tong, Drake Thomas, Kelley Rivoire, Adam Jermyn, Monte MacDiarmid, Tom Henighan, and Evan Hubinger. Auditing language models for hidden objectives, 2025.

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

[11] Lewis Smith, Senthooran Rajamanoharan, Arthur Conmy, Callum McDougall, Tom Lieberum, János Kramár, Rohin Shah, and Neel Nanda. Negative results for saes on downstream tasks and deprioritising sae research (gdm mech interp team progress update #2). AI Alignment Forum, Alignment Forum post, March 2025. Online; accessed 8 August 2025.

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

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

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

[15] 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.

[16] 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.

[17] 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.

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

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

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

[21] 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.

[22] 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.

[23] Jack Lindsey. Sparse crosscoders for cross-layer features and model diffing. Oct 2024. Transformer-Circuits.pub, October 25, 2024.

[24] Joe Needham, Giles Edkins, Govind Pimpale, Henning Bartsch, and Marius Hobbhahn. Large language models often know when they are being evaluated, 2025.

[25] 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.

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

[27] Gerlof J. Bouma. Normalized (pointwise) mutual information in collocation extraction. 2009.

[28] 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.

[29] 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.

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

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

[32] 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.

[33] 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.

[34] 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.

[35] 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.

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

[37] 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.

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

[39] 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.

[40] 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.

[41] 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.

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

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

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

[45] 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.

[46] 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.

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

[48] 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.

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

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

[51] 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.

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

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

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

[55] 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.

[56] 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.

[57] 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.

[58] 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.

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

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

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

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

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

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

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

[66] 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.

[67] 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.

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

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

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

A Additional Related Work

A.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 [20, 19].

A.2 Correlations

The problem of finding correlations in datasets is often framed as finding spurious correlations between features and dataset *classes*. For instance, [10] 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 [49, 50], and SAEs provide a natural extension. 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.

A.3 Clustering

Classical NLP represents texts using term based [30] or semantic embedding based [14] methods, then apply a standard clustering algorithm (e.g. KMeans [51], spectral clustering [52], HDBSCAN [53]). To guide clusters towards human-specified structure, prior work has used specified pairwise constraints [54, 55], seed examples [56], partial labels [57], feature feedback [58] or post-hoc tuning of clusters [59, 60], sometimes with LLM guidance [61, 62].

A.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. [63] 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 [64, 65, 66, 67]. We use SAEs as a way to approximate these embeddings, which we expect to contain these 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 [68, 69], thus it is natural to also use SAEs trained on LLM representations.

B Additional Results

B.1 Data Diffing

Hypotheses for Grok-4 vs. other frontier models. We present the generated hypotheses for Grok-4 using the SAEs (Table 9) and LLM baseline (Table 10).

Frontier models. The frontier models we compare against in Section 3.2 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.

Cost analysis. Given two datasets A and B, we compute the number of tokens the LLM and SAE would incur to produce hypotheses. Let N be the number of samples in each dataset, T be the average number of tokens per sample, and F be the number of latents in the SAE. Then the SAE method uses $2NT$ tokens for embedding the datasets and $O(F)$ tokens for labeling latents and generating hypotheses. The LLM uses $2NT$ tokens for initial processing and $O(N)$ tokens for generating individual differences and summarizing. In general, while both the number of tokens used by the SAE and LLM will scale with the size of the dataset, it will scale faster for the LLM. Furthermore, the SAE method currently passes the dataset through a much smaller LLM (Llama 3.3 70B) than the LLM judge (Gemini 2.5 Flash), providing an additional cost benefit.

Hypothesis	Grok-4 Coverage	Nearest Model	Nearest Model Coverage
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
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
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
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
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
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
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
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
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

Table 9: Generated hypotheses for Grok-4 using SAEs.

Hypothesis	Grok-4 Frequency	Nearest Model	Nearest Model Frequency
This response proactively engages the user by asking clarifying questions, offering further assistance, or anticipating alternative interpretations.	75.6	GPT-5	33.2
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
This response uses emojis, informal greetings/closings, or adopts a playful/humorous tone.	38.3	qwen3-235b-thinking	36.7
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
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
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
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
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
This response demonstrates the ability to perform real-time web browsing or generate live links.	2.0	qwen3-235b-thinking	6.5

Table 10: Generated hypotheses for Grok-4 using the LLM baseline.

B.2 Correlations

Distribution of Latent Pairs. We expect that generally, latent pairs with similar labels co-occur and latents with dissimilar labels do not co-occur. For latents with frequency between 0.01 and 0.9, we find every pair's NPMI and semantic similarity, and plot the histogram in Figure 10. Most latents are dissimilar and do not co-occur, and we highlight the "interesting" region where dissimilar latents have high co-occurrence.

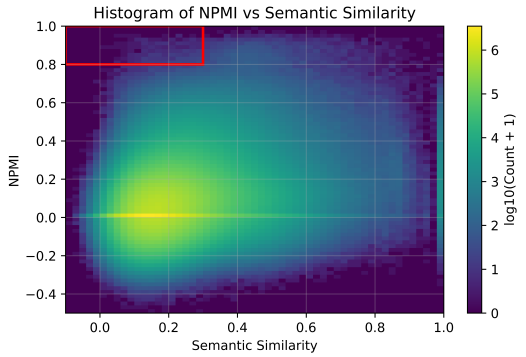

Figure 10: 2D histogram of latent pairs' NPMI and semantic similarities, from the ChatbotArena responses dataset.

Injected correlations baseline. Due to the context window limit, we pass in the dataset 2k samples (out of 10k) at a time. We ask another LLM to check the analysis for the injected correlations, which are not always found in every sample.

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

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""
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
```

ChatbotArena correlations.

Latent 1	Latent 2	NPMI	Sem. Sim.	$\mathbf{P(F1|F2)}$	$\mathbf{P(F2|F1)}$
Offensive request from the user	Narrative transitions and emotional beats in fiction writing	0.816	0.101	0.583	0.712
Names appearing in potentially harmful or offensive content	Narrative transitions and emotional beats in fiction writing	0.816	0.134	0.708	0.607
Names appearing in potentially harmful or offensive content	Story pacing and scene transitions in narrative generation	0.839	0.148	0.678	0.726
Temporal transitions in inappropriate or offensive content	First-person narrative perspective in storytelling	0.801	0.154	0.869	0.465

Table 12: Examples of "interesting" feature co-occurrences in ChatbotArena model responses.

ChatbotArena baseline. We pass all 5k responses into an LLM. Gemini 2.5 Flash returned 780 correlations, with many only occurring in 1 or 2 documents. Asking another LLM to check the analysis, we found that it did not include "offensive text + narrative text".

Gemini 2.5 Pro returned

```
1. Illicit Drugs and Nutritional Misinformation
2. Classic Literature and Modern Video Games
3. Physics Jargon and Pseudoscience
4. Artificial Intelligence Acronyms and Legal Degrees
5. Historical Events and Factual Nonsense
6. Cooking Recipes and Bizarre Ingredient Combinations
7. Video Game Fan Fiction and Mundane Plot Twists
8. Food Debates and Legal Formalism
9. Ancient Warfare and Magic/Superstition
10. Computer Science and Nonsensical Foreign Terms
11. Biology and Direct Factual Contradiction
12. Conspiracy Theories and Unexpected Justifications
13. Extreme Medical Trauma and Practical Life Advice
14. Academic Theory and Specific Racial/Social Concepts
```

B.3 Clustering

Failure to recover ground truth labels for sentiment and emotion clustering. We show example targeted clustering results below. We were unable to find a combination of queries and k that resulted in good alignment with ground truth labels.

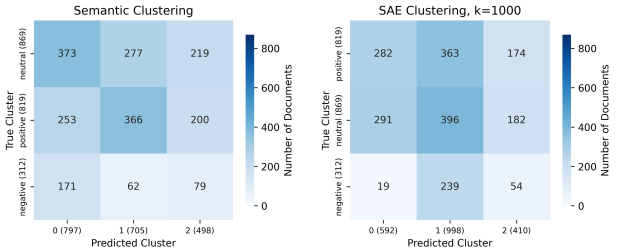

Figure 11: Twitter sentiment clustering results.

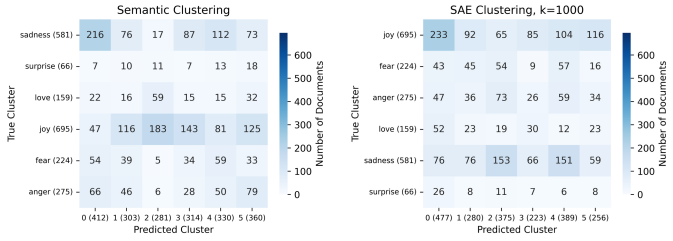

Figure 12: Twitter emotion clustering results.

Clustering prompts, responses and the Pile. We show the per-cluster accuracies for semantic and SAE clustering in Figure 13, and see that the SAE clusters have comparable per-cluster accuracies, with generally higher variance across clusters. This suggests the clusters are similarly valid.

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 to semantic cluster descriptions.

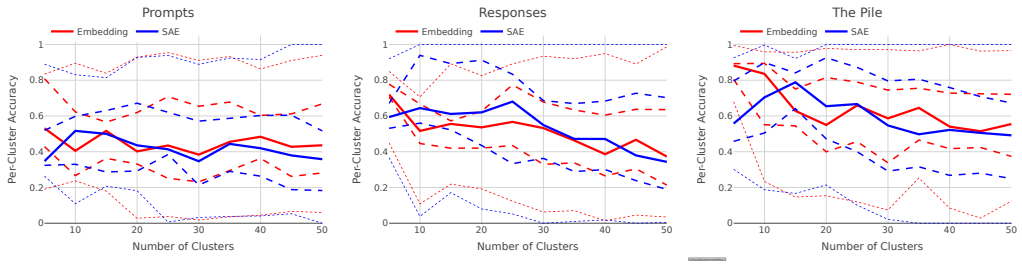

Figure 13: Per-cluster accuracies for different n_{clusters} for prompts, responses and the Pile. The solid lines are the median, dashed lines the interquartile range and dotted lines the range.

Semantic	**Acc.**	**SAE**	**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.

Semantic	**Acc.**	**SAE**	**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.

Semantic	**Acc.**	**SAE**	**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.

GSM8k: Identifying reasoning patterns. We show results from semantic clustering (Table 16) and SAE clustering with all latents (Table 17). The number of clusters is chosen based on human judgement of cluster quality, however, results are similar across similar cluster numbers.

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 16: Semantic 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 17: 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 18: SAE clustering using the top 500 features related to "step by step reasoning", $n_{\text{clusters}} = 3$.

B.4 Retrieval

Example queries. We show 5 selected queries for each dataset to illustrate the types of properties we can 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 19: Example queries across the six datasets.

Hyperparameter dependence. We plot the dependence of MAP and MP@50 on the number of phrases for query expansion (Figures 14-16), and on temperature for latent aggregation (Figure 17). 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 17 that a higher T is better for responses and the Pile, likely because the SAE was trained on chat data, thus it learnt many higher-quality latents for that distribution that can be aggregated for overall better performance.

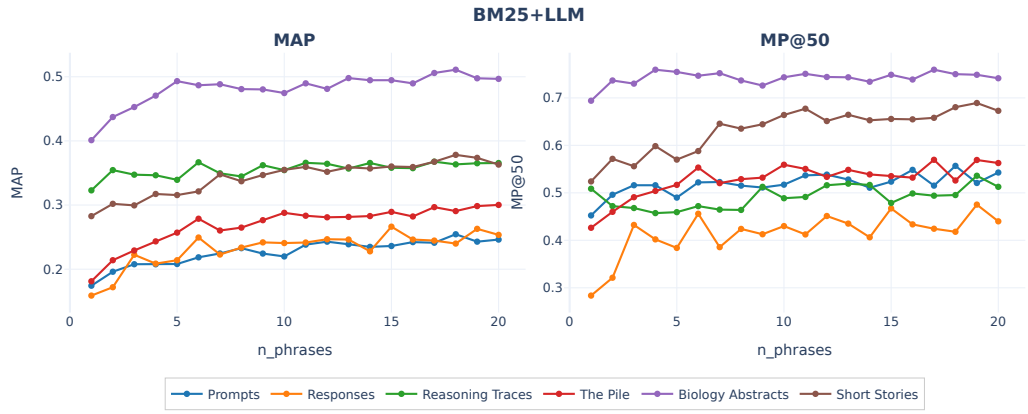

Figure 14: BM25+LLM

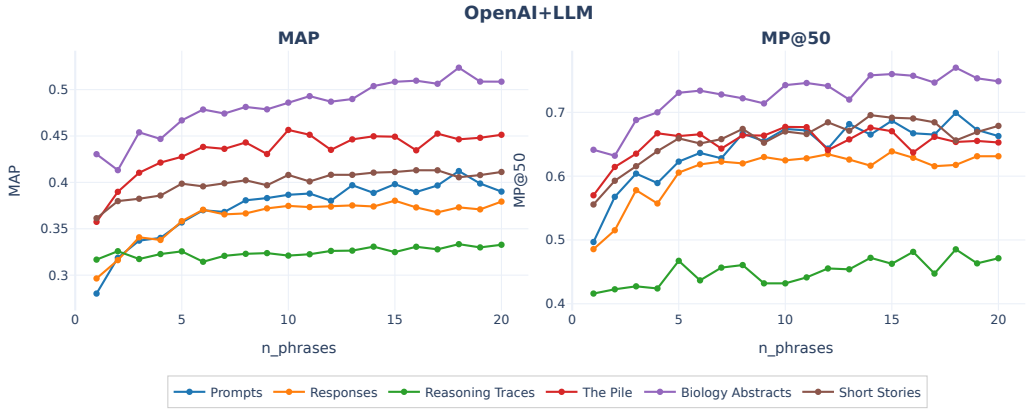

Figure 15: OpenAI+LLM

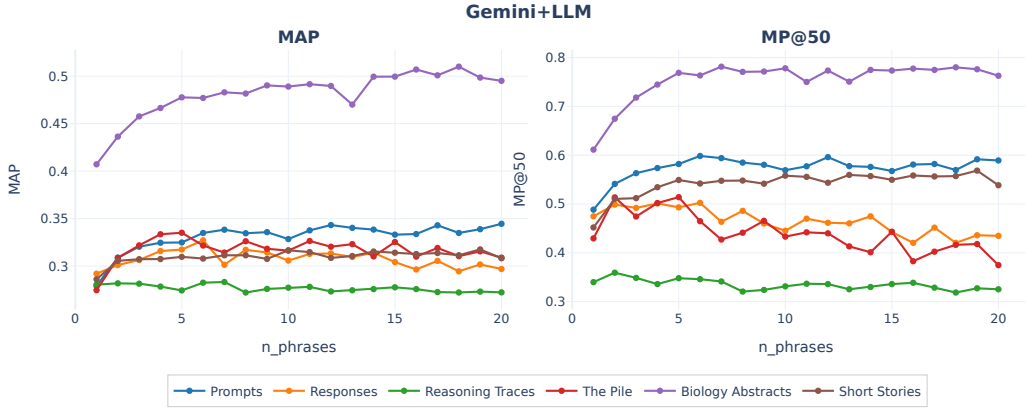

Figure 16: Gemini+LLM

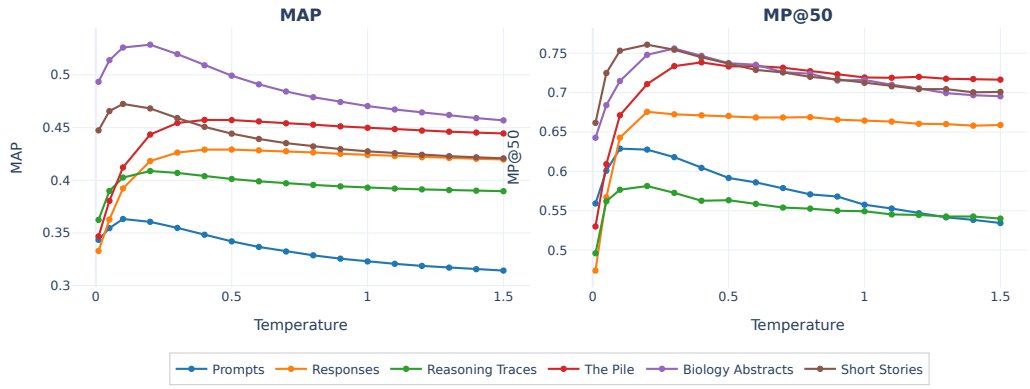

Figure 17: Performance of SAE method at different T used to aggregate features, for each dataset.

Metrics. The formulae for the metrics we report are:

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

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

Examples of well-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 20: Prompts Data: 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 21: Responses Data: 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 22: Reasoning Traces Data: 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 23: The Pile Data: 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 24: Biology Abstracts Data: 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 25: Short Stories Data: Top 3 Most Improved and Most Degraded Queries

Semantic embeddings rely on presence of key phrases, while SAEs do not. Generally, the semantic embedding with query expansion baseline retrieves texts containing the plausible phrases. For example, for 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.", the baseline retrieved samples tend to have the model stating it is giving its reasoning, while the SAE method does not necessarily rely on that.

OpenAI+LLM Top Results:

```
Okay, here is the step-by-step reasoning with a chain of thought:

1. Originally there were 2 apples in the bucket.
2. Then I saw 2 more apples in my hand. So now there are 2 apples in my hand.
...
```

```
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: Sure, what is the equation?
...
```

```
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.
```

```
...
```

SAE Top Results:

```
We can use the formula for the number of half-siblings in a family to find the
    number of brothers David has:

n = (s + t) / 2
...
```

```
| Thought Process |
| -- | He walks to the kitchen |
| -- | He takes the cup and puts the ball in it |
...
```

```
Possible answers:

1. Bobby has 3 brothers.
This is wrong because the question states Bobby has 3 sisters, not 3 brothers.
...
```

Ranking similarity. To quantify how different the rankings returned by the different retrieval methods are, we find the rank-biased overlap [70] 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.

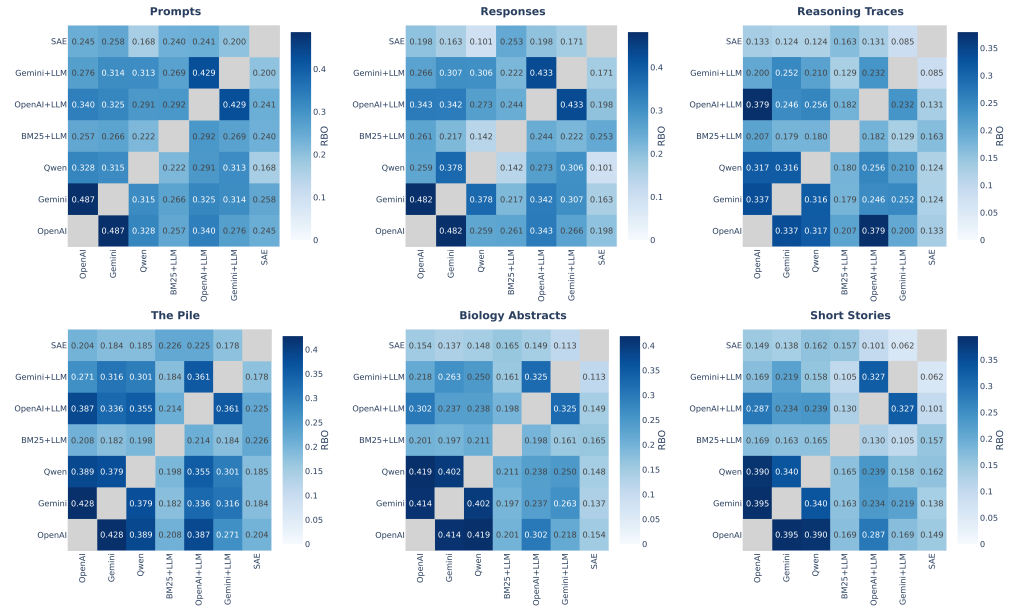

Figure 18: Ranking similarity among the relevant documents, using Rank-Biased Overlap (RBO) [70] with hyperparameter $p = 0.98$ since we are concerned about the top 50 results.

C LLM Judge Details

C.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 a 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 [1], we use a 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 a 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.

C.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
...
"""
```

To judge if the hypotheses are present in each text sample, we use

```
system_prompt = """
{prefix}
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,
    ...
}}
"""

prefix_dict = {
    "civilcomments": "You are given a comment from the internet.",
    "responses": "You are given an LLM-generated text."
}
```

```
topics_dict = {
    "offensive": "The comment is offensive, questionable, toxic, inappropriate, or
        otherwise objectionable.",
    "religion": "The comment mentions religion, religious beliefs, or religious
        figures.",
    "drugs": "The comment mentions drugs, drug use, or drug-related activities.",
    "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()])
```

For the LLM baseline, we use the following prompt:

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

Your task is to identify any **unexpected or unusual co-occurrences** of features **
    within** each text in the dataset.
A "feature" can be:
- Topic, subject, concept, idea, etc.
- Language, style, tone, sentiment, etc.
- Specific linguistic or syntactic patterns
- **Any** other identifiable property in the text

Find co-occurrences that seem interesting or unusual, that someone might want to
    know exists within some texts in the dataset.
Flag them even if they only occur in a few samples.

Return a list of all identified co-occurrences with a short explanation for each.
{"\n".join([f"DOC {i+1}\n{text}" for i, text in enumerate(sampled_texts)])}
"""
```

C.3 Clustering

We can get the help of an LLM for generating query phrases, 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.
"""
```

To generate cluster labels, we use the following:

```
system_prompt = """
You are an assistant for labelling 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.
"""
```

C.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 behaviour.

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}
"""
```

SAE relabeling. To relabel all SAE latents, we capture their activations on 50k texts from the Pile or 10k responses from LMSYS, and use the following prompt with $n_{\text{exanples}} = 20$:

```
SYSTEM = f"""
You are a meticulous assistant that helps label language model features.

You will be given a list of {n_examples} positive and {n_examples} negative example
    phrases.
Among the positive examples, some parts are highlighted between <<this>>. The
    highlighted text shows where the feature activates.
The negative examples are where the feature is NOT active.

You task is to describe what the feature activates on.
IMPORTANT NOTES:
1. The << >> markers show where the feature activated, but they are only a guide ---
     focus on the overall meaning of the sentence or phrase, not just the marked
    tokens.
2. Pay close attention to the surrounding context, especially the words before the
    marked tokens, as they often reveal the semantic or topical pattern the feature
     responds to.
3. Consider the broader idea, theme, or concept that connects the positive examples
    and is absent from the negative examples --- this is often more important than
    the literal form of the marked text.
4. Your description should capture the pattern that fits the positive examples and
    excludes the negative ones.
5. Be as descriptive and specific as possible, adding in nuances like "often concept
     X" if needed. Do not include example activations in your explanation.

Return your response as a NOUN PHRASE of what the feature represents, in the
    following format:
[EXPLANATION]: <noun phrase>
"""
```

D Dataset Details

D.1 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.
        ",
}
```

D.2 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.
"""
```